Volume 83, issue 2, March 2014

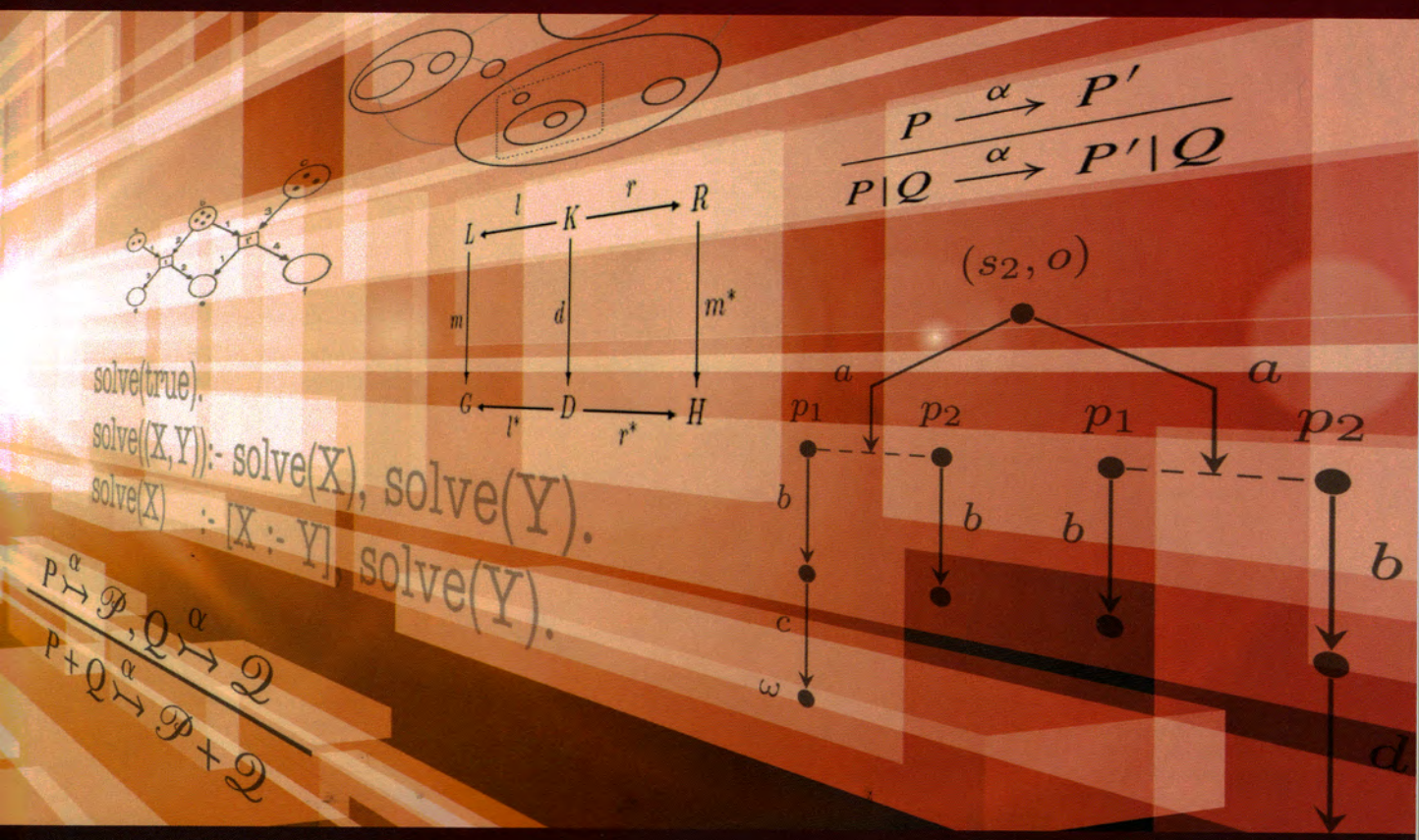
ISSN 2352-2208

ELSEVIER

JOURNAL OF

LOGICAL AND ALGEBRAIC METHODS IN PROGRAMMING

Special Issue: Festschrift in Honour of Gunther Schmidt on the Occasion of his 75th Birthday
Guest Editors: Rudolf Berghammer, Bernhard Möller, Michael Winter



Available online at www.sciencedirect.com

ScienceDirect

THE JOURNAL OF LOGICAL AND ALGEBRAIC METHODS IN PROGRAMMING

Volume 83, issue 2, March 2014

ELSEVIER
Amsterdam–Boston–London–New York–Oxford–Paris–Philadelphia–San Diego–St. Louis

The Journal of Logical and Algebraic Methods in Programming is indexed/abstracted in *Anbar Electronic Intelligence, ABI/Inform, Computing Reviews, Engineering Index, Engineering Information Abstracts, INSPEC Abstracts, Mathematical Reviews, Computer & Information Abstracts, ISI/CompuMath, ISI Current Contents, ISI Citation Index, ISI Current Contents/Engineering and Computing & Technology. Also covered in the abstract and citation database Scopus®. Full text available on ScienceDirect®*

Publication information: *The Journal of Logical and Algebraic Methods in Programming* (ISSN 2352-2208). For 2014, volume 83 (6 issues) is scheduled for publication. Subscription prices are available upon request from the Publisher or from the Elsevier Customer Service Department nearest you or from this journal's website (http://www.elsevier.com/locate/jlamp). Further information is available on this journal and other Elsevier products through Elsevier's website (http://www.elsevier.com). Subscriptions are accepted on a prepaid basis only and are entered on a calendar year basis. Issues are sent by standard mail (surface within Europe, air delivery outside Europe). Priority rates are available upon request. Claims for missing issues should be made within six months of the date of dispatch.

Orders, claims, and journal inquiries: please contact the Elsevier Customer Service Department nearest you:
St. Louis: Elsevier Customer Service Department, 3251 Riverport Lane, Maryland Heights, MO 63043, USA; phone: (877) 8397126 [toll free within the USA]; (+1) (314) 4478878 [outside the USA]; fax: (+1) (314) 4478077; e-mail: JournalCustomerService-usa@elsevier.com
Oxford: Elsevier Customer Service Department, The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK; phone: (+44) (1865) 843434; fax: (+44) (1865) 843970; e-mail: JournalsCustomerServiceEMEA@elsevier.com
Tokyo: Elsevier Customer Service Department, 4F Higashi-Azabu, 1-Chome Bldg, 1-9-15 Higashi-Azabu, Minato-ku, Tokyo 106-0044, Japan; phone: (+81) (3) 5561 5037; fax: (+81) (3) 5561 5047; e-mail: JournalsCustomerServiceJapan@elsevier.com
Singapore: Elsevier Customer Service Department, 3 Killiney Road, #08-01 Winsland House I, Singapore 239519; phone: (+65) 63490222; fax: (+65) 67331510; e-mail: JournalsCustomerServiceAPAC@elsevier.com

Advertising information: If you are interested in advertising or other commercial opportunities please e-mail Commercialsales@elsevier.com and your inquiry will be passed to the correct person who will respond to you within 48 hours.

Author inquiries: For inquiries relating to the submission of articles (including electronic submission) please visit this journal's homepage at http://www.elsevier.com/locate/jlamp. For detailed instructions on the preparation of electronic artwork, please visit http://www.elsevier.com/artworkinstructions. Contact details for questions arising after acceptance of an article, especially those relating to proofs, will be provided by the publisher. You can track accepted articles at http://www.elsevier.com/trackarticle. You can also check our Author FAQs at http://www.elsevier.com/authorFAQ and/or contact Customer Support via http://support.elsevier.com.

Funding body agreements and policies: Elsevier has established agreements and developed policies to allow authors whose articles appear in journals published by Elsevier, to comply with potential manuscript archiving requirements as specified as conditions of their grant awards. To learn more about existing agreements and policies please visit http://www.elsevier.com/fundingbodies

Special Issue: Festschrift in Honour of Gunther Schmidt on the Occasion of his 75th Birthday

GUEST EDITORS
Rudolf Berghammer
University of Kiel,
Germany

Bernhard Möller
University of Augsburg,
Germany

Michael Winter
Brock University,
St Catherines,
Canada

Printed by Henry Ling Ltd., The Dorset Press, Dorchester, UK

For a full and complete Guide for Authors, please go to: http://www.elsevier.com/locate/jlamp

THE JOURNAL OF
LOGICAL AND ALGEBRAIC METHODS IN PROGRAMMING

Volume 83, issue 2, March 2014

CONTENTS

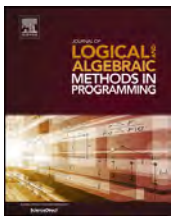
doi:10.1016/S2352-2208(14)00025-X

Journal of Logical and Algebraic Methods in Programming 83 (2014) 85–86

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp



Preface

Relation algebra was introduced for compacting first-order logic formulas, by eliminating certain quantifiers, into *point-free* form that is amenable to simple algebraic manipulation, i.e., (in)equational reasoning. But there are many other forms of point-free calculi, starting probably with the calculus of matrices and the algebra of sets, and continuing with lattice theory, categories and allegories, or semirings and their relatives, including Kleene Algebra and other structures with constructs for iteration.

Over the last 20 years, the series *RAMiCS — Relational and Algebraic Methods in Computer Science* of conferences has successfully brought together researchers on the theory and applications of these kinds of algebraic approaches. This year Gunther Schmidt, one of the founders and central figures of the RAMiCS conferences will turn 75. Without him the field would by far not be as advanced as it is today, both technically and concerning its international interconnection. This is most gratefully and respectfully acknowledged by the RAMiCS community, who therefore wants to honour him with a Festschrift in form of this current special issue of the Journal JLAP (which has already benevolently accompanied the RAMiCS efforts for many years).

The issue starts with a paper by A. Amstrong, G. Struth and T. Weber on *Programming and automating mathematics in the Tarski–Kleene hierarchy*. Next, R. Berghammer shows both theory and the RelView system at work in *Computing minimal extending sets by relation-algebraic modelling and development*. The following paper is by R. Berghammer, A. Rusinowska and H. de Swart and deals with a more application-oriented topic, namely *Spatial voting games, relation algebra and* RelView. Then the paper *Exploring modal worlds* by H.-H. Dang, R. Glück, B. Möller, P. Roocks and A. Zelend provides a round-trip through semiring-based modal algebra. The idea is to cover topics as varied as bisimulations, formal concept analysis, preference database queries, separation logic and an algebra of software modules in a unified formal manner. That paper is followed by a more theoretical article by J. Desharnais, A. Grinenko and B. Möller on *Relational style laws and constructs of linear algebra*. Next, I. Düntsch and E. Orlowska study *Discrete dualities for some algebras with relations*. A truly multi-author paper *Inference engine based on closure and join operators over Truth Table Binary Relations* on practical applications is presented by S. Elloumia, B. Boulifaa, A. Jaoua, M. Saleha, J. Al Otaibia and M. Frías. After that we are taken to the field of program semantics by W. Guttmann in his paper *Multirelations with infinite computations*. An unorthodox application of the algebraic concept of semirings to the description and analysis of routing protocols is presented by P. Höfner and A. McIver in *Hopscotch—reaching the target hop by hop*. The next contribution is by W. Kahl and deals with *Towards "mouldable code" via nested code graph transformations*. This is followed by another more theoretical paper by R. Maddux on *Arrow's Theorem for incomplete relations*. A combination of theory and application is given by J. Oliveira with *A relation-algebraic approach to the "Hoare logic" of functional dependencies*. Another more theoretic approach is presented by N. Tsumagari, H. Furusawa and Y. Kawahara in their article *Relations into algebras of probabilistic distributions*. At the end of the technical part of the issue, all the algebra shown before is supplemented by yet a different kind in *Relational properties of sequential composition of co-algebras* by M. Winter and P. Kempf.

The issue is rounded off by two non-technical articles. We are presented with *Gunther Schmidt's life as a mathematician and computer scientist* by R. Berghammer and M. Winter, while M. Müller and B. Möller write, with tongue in cheek, *On nothing* — something one often feels is the result of all efforts anyway.

We are most grateful to Jan Bergstra and John Tucker, the parting, and Rocco Nicola and Luca Aceto, the new, Editors-in-Chief of JLAP for hosting this fine special issue, which continues a series that was originated by Jan and John in 2006 and has since successfully reflected the progress of the RAMiCS conferences. Next we want to thank the authors for their interesting and profound contributions, and the referees for their careful and constructively critical evaluation of the submissions. Moreover, we gratefully acknowledge the help of Georg Struth in administering the reviews of the papers co-authored by any of the editors to warrant independence, and the collaboration of Inge Bethke and Alberto Lluch Lafuente concerning the technicalities of the issue.

Finally, want to deeply thank Gunther Schmidt for all his profound and stimulating contributions to the theory and applications of relations and algebraic structures in general. We will most cordially congratulate him at his 75th birthday

http://dx.doi.org/10.1016/j.jlap.2014.02.017

which will take place on November 28, 2014 — following a widespread superstition we refrain from doing so now. We hope that he will spend many more lively and fruitful years in good health and be as creative and prolific as ever!

Rudolf Berghammer
University of Kiel, Germany

Bernhard Möller
University of Augsburg, Germany

Michael Winter
Brock University, St Catharines, Canada

Available online 12 February 2014

Journal of Logical and Algebraic Methods in Programming 83 (2014) 87–102

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Programming and automating mathematics in the Tarski–Kleene hierarchy

Alasdair Armstrong [a], Georg Struth [a,*], Tjark Weber [b]

[a] Department of Computer Science, The University of Sheffield, UK
[b] Department of Information Technology, Uppsala University, Sweden

ARTICLE INFO

Article history:
Available online 12 February 2014

Dedicated to Professor Gunther Schmidt on the occasion of his 75th birthday

Keywords:
Formalised mathematics
Interactive theorem proving
Kleene algebra
Relation algebra

ABSTRACT

We present examples from a reference implementation of variants of Kleene algebras and Tarski's relation algebras in the theorem proving environment Isabelle/HOL. For Kleene algebras we show how models can be programmed, including sets of traces and paths, languages, binary relations, max-plus and min-plus algebras, matrices, formal power series. For relation algebras we discuss primarily proof automation in a comprehensive library and present an advanced formalisation example.

1. Introduction

The advancement of relational mathematics as a conceptual and methodological tool across the sciences has been the main impetus of Gunther Schmidt's research for many decades. His articles and books elaborate the fundamental role of the calculus of relations for modelling and reasoning about discrete systems. Having experienced as a young mathematician how computers revolutionised numerical and engineering mathematics, he set out to pioneer a similar approach within his own field of interest: long before model checkers became an industry standard and complex mathematical theorems could be verified by machines, he thought about computer software for programming relational mathematics and automating relational proofs.

The *Munich School*, under his influence, developed a wide range of relational tools, including the RelView system for manipulating finite relations [1] and formalisations of relation algebras in theorem proving environments; see Wolfram Kahl's article [2] for an insider's overview.

Today, relational approaches play a fundamental role in computer science—in program semantics, logics of programs, formal program development, databases and beyond. New applications in engineering or the social sciences are emerging.

In the spirit of the Munich School, this paper presents a series of examples in programming and automating mathematics. They are drawn from a reference formalisation that integrates Tarski's relation algebra with variants of Kleene algebras, which are closely related, in the theorem proving environment Isabelle/HOL. The formalisation of variants of Kleene algebras and relation algebras, including all algebras, models and theorems discussed in this paper, is available from the Archive of Formal Proofs [3,4]. We have implemented these algebras as a structured modular hierarchy using Isabelle's axiomatic type classes. Starting from (join-)semilattices, we have axiomatised variants of dioids with and without certain distributivity and unit axioms and expanded them by axioms for the Kleene star. Since many models of interest have a richer algebraic

* Corresponding author.
E-mail addresses: a.armstrong@sheffield.ac.uk (A. Armstrong), g.struth@sheffield.ac.uk (G. Struth), tjark.weber@it.uu.se (T. Weber).

http://dx.doi.org/10.1016/j.jlap.2014.02.001
2352-2208/

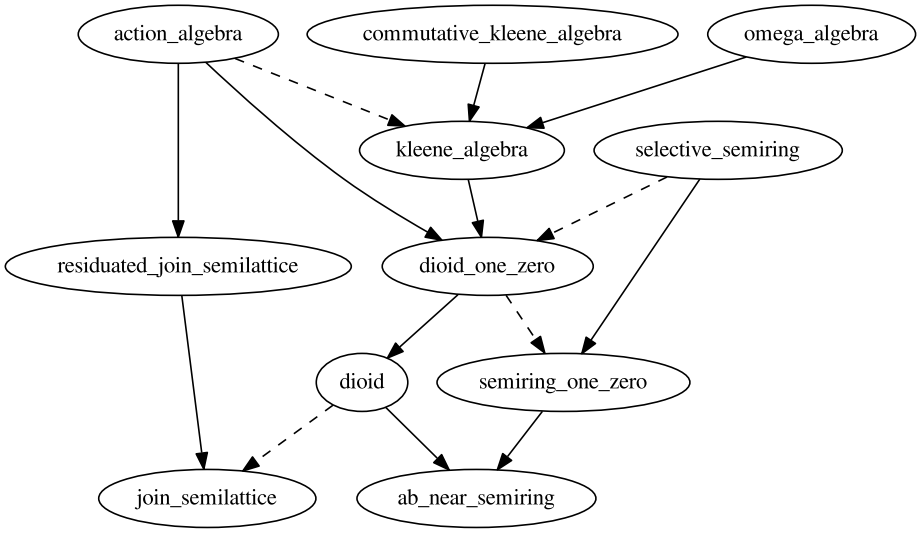


Fig. 1. Simplified Kleene algebra hierarchy. Solid arrows denote syntactic class extensions, while dashed arrows indicate subclass relationships established by proof.

structure, we have also included action algebras [5], which add operations of residuation. We have expanded Kleene algebras further to omega algebras with an operation of infinite iteration and, finally, to Tarski's relation algebras and relation algebras with a star or reflexive transitive closure operation. From this basis, further axiomatic variants can easily be implemented. The first steps towards this hierarchy have already been documented [6], so our review in this article can be brief.

Instead, for Kleene algebras, we discuss the programming approach for the most important models. These include power-sets over a monoid, (regular) languages, binary relations, sets of program traces in which state and action symbols alternate, sets of paths in graphs, max-plus and min-plus algebras. We also show that matrices over dioids form dioids (implementing the Kleene star is tedious), and that formal power series from free monoids into Kleene algebras form Kleene algebras. All of these models have important applications in computer science.

In the context of relation algebras, we explain how Isabelle's recent integration of external automated theorem provers supports the rapid semi-automatic development of a comprehensive library based on the standard text books by Schmidt and Ströhlein [7], Schmidt [8] and Maddux [9]. In fact, calculational proofs which eminent mathematicians found worth publishing some decades ago are often fully automatic with this approach. As an advanced formalisation example we explore the Munich axiomatisation of direct relational products [7]. In addition, we have programmed the binary relation and Boolean matrix model of relation algebra.

While automation is desirable for applications and useful for mathematical explorations, it is not the primary goal of our reference implementation. For Kleene algebras, we typically present readable proofs at text book granularity, using Isabelle's proof language Isar [10]. Apart from some idiosyncrasies our formalisation could serve as a comprehensive introduction to reasoning in Kleene algebras. A similar approach is planned for relation algebra.

The integration of relation algebra and Kleene algebras achieves a uniform approach in which all theorems about the Kleene star become automatically available for a relational reflexive transitive closure operation. This is mandatory for programming applications. By linking the algebras in the hierarchy with various models, the development of the latter is significantly streamlined and unified. With algebraic libraries, only a few axioms must be checked to make abstract theorems automatically available. Large parts of Isabelle's library for binary relations, for instance, become redundant relative to a modular algebraic development. In programming applications, the link between algebras and models couples control flow with data flow and supports seamless reasoning, for instance, about changes and updates in program stores or state spaces.

2. Algebraic hierarchy

Kleene algebras expand dioids (idempotent semirings) by an operation of Kleene star or finite iteration. Dioids, in turn, expand join-semilattices by an operation of multiplication. Tarski's relation algebras are dioids as well as Boolean algebras which share the join operation and the least element. In addition they axiomatise an operation of relational converse. In other words, the relational operations of complementation, meet and conversion are missing in Kleene algebras whereas the Kleene star is missing in relation algebra.

This section briefly overviews this theory hierarchy and serves as a road map for the formal development in the Archive which defines more than 40 related type classes. The main structures in the Kleene algebra hierarchy are shown in Fig. 1. Isabelle already contains a vast amount of mathematical structures and libraries, which any formalisation of new structures can and should use. Relation algebras, for instance, can be based on Isabelle's formalisation of Boolean algebras as well as on the variants of semirings formalised within the Kleene algebra hierarchy. These semirings, in turn, can be based on Isabelle's formalisation of abelian semigroups. By these integrations, all facts proved for the basis structures become

automatically available in all expansions. In a similar fashion, the binary relations models of Kleene algebra and relation algebra are based on Isabelle's extensive library for binary relations. As a general rule, algebraic structures should therefore be designed in a modular fashion in order to enhance reuse and expandability.

A *join-semilattice* $(L, +)$ consists of a set L with an associative, commutative and idempotent operation of join denoted by $+$. Join-semilattices are also posets with order relation defined by

$$x \leqslant y \longleftrightarrow x + y = y,$$

such that the supremum $x + y$ exists for all $x, y \in L$. It follows that addition is isotone, that is, $x \leqslant y \longrightarrow z + x \leqslant z + y$. A join-semilattice is *bounded* if it has an additive unit 0 satisfying $x + 0 = x$. This unit is the least element with respect to the order. Every bounded join-semilattice is therefore also a commutative monoid.

Adding a multiplicative semigroup structure yields variants of near semirings (or seminearrings). In general, a near semiring consists of an additive and a multiplicative semigroup which interact via one single distributivity law (left or right). We are only interested in near semirings in which addition is commutative and the right distributivity law holds. Formally, an *abelian near semiring* is a structure $(S, +, \cdot)$ such that $(S, +)$ is a commutative (abelian) semigroup, $(S, \cdot)$ a semigroup and the right distributivity axiom

$$(x + y) \cdot z = x \cdot z + y \cdot z$$

holds. A *semiring* is an abelian near semiring which also satisfies the left distributivity law

$$x \cdot (y + z) = x \cdot y + x \cdot z.$$

A *near dioid* is an abelian near semiring whose additive reduct is a join-semilattice. Hence every near dioid can be ordered, and multiplication is right isotone: $x \leqslant y \longrightarrow x \cdot z \leqslant y \cdot z$. A *pre-dioid* is a near dioid in which multiplication is left isotone:

$$x \leqslant y \longrightarrow z \cdot x \leqslant z \cdot y.$$

A *dioid* is a near dioid in which also the left distributivity law holds. Hence dioids are additively idempotent semirings. All variants of near semirings can be expanded by additive units 0 and multiplicative units 1. We have separately formalised 0 as a left annihilator $(0 \cdot x = 0)$ and an annihilator (also $x \cdot 0 = 0$), since right annihilation fails in interesting models. An element x, for instance, could represent an infinite computation, which should rather satisfy $x \cdot 0 = x$ than $x \cdot 0 = 0$. Finally, an abelian near semiring S (with or without units) is *selective* if either $x + y = x$ or $x + y = y$ holds for all $x, y \in S$. All selective abelian near semirings are near dioids with linear order $\leqslant$. The definition extends to semirings and dioids. Max-plus and min-plus algebras [11] are examples of selective semirings.

Next we have formalised the Kleene star over the family of near-dioids with 1. A *left near Kleene algebra* is a structure $(K, +, \cdot, ^*, 1)$ such that $(K, +, \cdot, 1)$ is a near dioid with 1 and the star satisfies the *left unfold axiom* and the *left induction axiom*

$$1 + x \cdot x^* \leqslant x^*, \qquad z + x \cdot y \leqslant y \longrightarrow x^* \cdot z \leqslant y.$$

This implies that x^* is the least (pre)fixpoint of the function $\lambda y.\, 1 + x \cdot y$. The same axioms are used to expand left pre-dioids and dioids with 1 to *left pre-Kleene algebras* and *left Kleene algebras* respectively. We have further expanded these structures by additive units 0. *Near Kleene algebras*, *pre-Kleene algebras* and *Kleene algebras* can be obtained by adding a *right induction axiom*

$$z + y \cdot x \leqslant y \longrightarrow z \cdot x^* \leqslant y$$

to the respective variant of left Kleene algebra. Finally, variants of *commutative Kleene algebras* are obtained by adding the axiom $x \cdot y = y \cdot x$.

Near semirings form the basis of process algebras; pre-Kleene algebras have been used for modelling games and probabilistic protocols; the right annihilation axiom has been omitted in the context of total program correctness.

(Left) Kleene algebras use Horn formulas to axiomatise the equational theory of regular expressions. Action algebras provide a finite equational axiomatisation in a larger signature, which is supported by many important models. A *residuated join-semilattice* is a structure $(L, +, \cdot, \leftarrow, \rightarrow)$ such that $(L, +)$ is a join-semilattice, $(S, \cdot)$ is a semigroup and *left* and *right residuation* are axiomatised by the Galois connections

$$x \cdot y \leqslant z \longleftrightarrow x \leqslant z \leftarrow y, \qquad x \cdot y \leqslant z \longleftrightarrow y \leqslant x \rightarrow z.$$

There exists an equivalent, though less elegant, equational axiomatisation. An *action algebra* is a residuated join-semilattice that is also a dioid, which is expanded by a star operation satisfying the two star axioms

$$1 + x + x^* \cdot x^* \leqslant x^*, \qquad 1 + x + y \cdot y \leqslant y \longrightarrow x^* \leqslant y.$$

These are inspired by the reflexive transitive closure operation, but those of Kleene algebra can equally be used. Thus, in particular, every action algebra is a Kleene algebra. In action algebras, the star can as well be axiomatised by the equations

$$1 + x + x^* \cdot x^* \leqslant x^*, \qquad x^* \leqslant (x + y)^*, \qquad (x \to x)^* \leqslant x \to x.$$

This yields the finite equational axiomatisation mentioned.

Finally, in the Kleene algebra hierarchy, an operation of infinite iteration can be added to Kleene algebras. A *left omega algebra* is a left Kleene algebra (with zero) expanded by an omega operation which satisfies the *unfold axiom* and the *coinduction axiom*

$$x^\omega \leqslant x \cdot x^\omega, \qquad y \leqslant z + x \cdot y \longrightarrow y \leqslant x^\omega + x^* \cdot z.$$

Omega algebras can be obtained by expanding Kleene algebras accordingly. We currently do not know any interesting properties that would hold in all omega algebras, but not in all left omega algebras.

Relation algebras form a different dioid expansion. A *relation algebra* is a structure $(R, +, \cdot, -, ;, ^\smile, 0, 1, 1')$ such that $(R, +, \cdot, -, 0, 1)$ is a Boolean algebra and $(R, +, ;, 0, 1')$ is a dioid. The operation of conversion is involutive, $x^{\smile\smile} = x^\smile$, additive, $(x + y)^\smile = x^\smile + y^\smile$, contravariant, $(x; y)^\smile = y^\smile; x^\smile$ and satisfies the residuation property $x^\smile; -(x; y) \leqslant -y$.

While axiomatic reasoning in dioids is trivial and in Boolean algebra it is rather simple, the interaction of the star or omega with the dioid operations and that of conversion with the Boolean and dioid operations adds a significant level of complexity and requires a certain level of experience.

Due to our focus on models of Kleene algebras and on relation algebras, most of the variants of semirings and Kleene algebras mentioned in this section are not essential to the remainder of this article, but nevertheless they play an important role in the modular design of the Tarski–Kleene hierarchy with a view on applications. Most of this article is based on dioids with additive and multiplicative units, Boolean algebras, Kleene algebras, omega algebras and relation algebras.

3. Formalising algebraic hierarchies in Isabelle/HOL

Isabelle's axiomatic type class infrastructure [12] is the standard tool for formalising algebraic hierarchies like that depicted in Section 2. Details can be found in the Archive proof documents [3,4]. Additional variants could be implemented easily from this basis. As an example, we have implemented abelian near semirings as

```
class ab_near_semiring = ab_semigroup_add + semigroup_mult +
  assumes right_distrib: "(x + y) · z = x · z + y · z"
```

expanding Isabelle's existing type classes for additive abelian semigroups and multiplicative semigroups, and adding the right distributivity axiom for the interaction between addition and multiplication. They can be expanded to abelian near semirings with a multiplicative unit 1, provided by class *one*, as follows:

```
class ab_near_semiring_one = ab_near_semiring + one +
  assumes mult_onel: "1 · x = x"
  and mult_oner: "x · 1 = x"
```

Simple relationships between algebras can be captured by subclass statements. The fact that selective near semirings are linear orders, for instance, is captured by the following statement and proof.

```
subclass (in selective_near_semiring) linorder
  ⟨proof⟩
```

The statement in brackets denotes that the linear order axioms need to be verified in the context of selective near semirings. The proof obligations are dictated by Isabelle. They are not shown in this article, which is indicated by writing ⟨*proof*⟩. This subclass statement makes all Isabelle theorems for linear orders available in the context of selective near semirings. We have carefully used expansions and subclassing to design our algebraic hierarchy and propagate theorems.

Our second example shows the implementation of relation algebra as an expansion of Isabelle's built-in axiomatic type class for Boolean algebra.

```
class relation_algebra = boolean_algebra +
  assumes comp_assoc: "(x ; y) ; z = x ; (y ; z)"
  and comp_unitr: "x ; 1' = x"
  and comp_distr: "(x + y ) ; z = x ; z + y  ; z"
  and conv_invol: "(x⌣)⌣ = x"
  and conv_add: "(x + y)⌣ = x⌣ + y⌣"
  and conv_contrav: "(x ; y)⌣ = y⌣ ; x⌣"
  and comp_res: "x⌣ ; -(x ; y) ≤ -y"
```

The proof that every relation algebra is a dioid now requires matching the signature of relation algebra with that of dioids, which in turn uses that of near-semirings. Concretely, the relation algebraic operation of $+$ matches the dioid operation $+$ and the relation algebraic operation $;$ matches the operation $\cdot$ of the dioid. This can be captured by a sublocale statement in Isabelle.

```
sublocale relation_algebra ⊆ dioid_one_zero "(op +)" "(op ;)" "1'" "0" "(op ⩽)" "(op <)"
  ⟨proof⟩
```

When writing a sublocale statement, Isabelle displays a list of operations to be matched. In this particular case these are the operations of addition, multiplication, one, zero, of the dioid and the associated operations of partial and strict order. The user then has to type the corresponding list of matching relation-algebraic operations. In the above list, all operations except relational composition have been overloaded. As with a subclass statement, Isabelle then dictates that the axioms of dioids with one and zero be proved in the context of relation algebra while matching the operations as indicated.

Our hierarchy also contains an axiomatic type class for relation algebras with a star operation. It is obtained by adding the star axioms of Kleene algebra to those of relation algebra. In the context of relation algebras, the star implements a reflexive transitive closure operation. Based on the formalised subclass relationships between relation algebras and dioids, the sublocale proof that every relation algebra with star is also a Kleene algebra is trivial: only the star axioms need to be verified and these are identical in both classes.

```
class relation_algebra_rtc = relation_algebra + star_op +
  assumes rtc_unfoldl: "1' +  x ; x⋆ ⩽ x⋆"
  and rtc_inductl: "z + x ; y ⩽ y ⟶ x⋆ ; z ⩽ y"
  and rtc_inductr: "z + y ; x ⩽ y ⟶ z ; x⋆ ⩽ y"

sublocale relation_algebra_rtc ⊆ kleene_algebra ⟨signature list⟩
  ⟨proof⟩
```

By our design of the Tarski–Kleene hierarchy with type classes and locales, theorems are propagated upwards in the hierarchy. All theorems of Kleene algebra, for instance, become automatically available in the context of relation algebras with star; all theorems of Boolean algebra become automatically available in the context of relation algebra.

Schmidt and Ströhlein's book, for instance, contains many examples of reflexive transitive closure theorems in relation algebra which are now inherited from Kleene algebra. One of their examples is the Church–Rosser theorem: they prove by induction on powers of x that the Church–Rosser property $(x+x^\smile)^* = x^*; x^{\smile *}$ follows from confluence $x^{\smile *}; x^* \leqslant x^*; x^{\smile *}$ in a relation algebra that is based on a complete Boolean algebra. Isabelle recognises this theorem as an instance of the more general implication

$$y^* \cdot x^* \leqslant x^* \cdot y^* \longrightarrow (x+y)^* = x^* \cdot y^*$$

which holds already in Kleene algebra—hence without an explicit induction.

First steps towards a similar implementation have already been undertaken by Wolfram Kahl [2] about a decade ago. The main differences are as follows.

First, Kahl has implemented a hierarchy of typed or heterogeneous algebras. While this is preferable for some modelling purposes, the proofs of many basic relational theorems can in fact be separated into one phase performing some trivial existence checks on objects and a second one performing equational reasoning in the untyped setting. In Isabelle, in addition, axiomatic type classes can no longer be used in the presence of typed relations; they must be replaced by locales. Definitions based on partial functions cause additional problems in Isabelle. Proofs become more tedious and much less automatic.

Second, our implementation relies heavily on a recent integration of automated theorem provers and counterexample generators in Isabelle, which were not available ten years ago. In the untyped setting, in particular, this typically allows proofs at textbook granularity or even fully automated proofs. Our reference implementation of Kleene algebras aims at readable textbook-style proofs instead of proof automation. The proof

```
lemma star_trans_eq: "x⋆ · x⋆ = x⋆"
proof (rule antisym) — this splits an equation into two inequalities
  have "x⋆ + x · x⋆ ⩽ x⋆"
    by (metis add_lub eq_refl star_1l)
  thus "x⋆ · x⋆ ⩽ x⋆"
    by (metis star_inductl)
  next show "x⋆ ⩽ x⋆ · x⋆"
    by (metis mult_isor mult_onel star_ref)
qed
```

which uses Isabelle's proof scripting language Isar, provides a simple example. Algebraists can easily follow the high-level proof structure, while individual proofs steps have been verified by Isabelle's automated provers. This turns our reference implementation almost into a compendium of algebraic reasoning in variants of Kleene algebras. A similar approach for relation algebras is planned for the future.

As a side effect, our proofs are robust to changes and easy to maintain. We have also used Isabelle's counterexample generators Nitpick and Quickcheck [13] for separating algebras. The star slide rule $x^* \cdot (y \cdot x^*)^* = (x^* \cdot y)^* \cdot x^*$ is refuted

by Nitpick by a six-element counterexample in left pre-Kleene algebras (it holds in left Kleene algebras). There are some interesting challenge problems for counterexample generators, for instance, to show that not all left Kleene algebras are Kleene algebras. Kozen's counterexample [14] involves transfinite induction.

4. Monoidal and language models of Kleene algebra

The following sections present our formalisation of the most important models of dioids and Kleene algebras. They are well known mathematically, but the functional programming approach to mathematics typical to proof assistants leads to some interesting insights. Most of these models, with the notable exception of binary relations and Boolean matrices, are not relevant to relation algebra.

This section discusses variants of language models. (Regular) languages are already available in Isabelle (cf. [15]), but the algebraic approach makes our implementation simpler and more generic. While theorems should generally be proved for the weakest axiom systems, models should be built for the strongest ones. With Isabelle's type classes, theorems are propagated down the subalgebra hierarchy; models are propagated upwards.

First we have shown that the powerset lifting of a monoid with suitably defined operations yields a Kleene algebra. Suppose that $(M, \cdot, 1)$ is a monoid (with elements of type α). Define the following operations on the powerset of M (type α *set* in Isabelle): the multiplicative unit is $1 = \{1\}$ (using overloading); the complex product is, using Isabelle's slightly idiosyncratic set builder notation,

$$X \cdot Y = \{x \cdot y \mid xy.\ x \in X \wedge y \in Y\}$$

for $X, Y \subseteq M$. A simple instantiation proof in Isabelle then establishes the following fact.

Theorem 1. *If $(M, \cdot, 1)$ is a monoid, then so is $(2^M, \cdot, 1)$.*

Instantiations link models with Isabelle's axiomatic type classes. With additive unit $0 = \{\}$ and addition $X + Y$ defined as $X \cup Y$ on powersets it takes little effort to show that $(2^M, +, \cdot, 0, 1)$ forms a dioid. The Kleene star on powersets can be defined as

$$X^* = \bigcup_{n \geqslant 0} X^n$$

using Isabelle's recursive definition of powers X^n for arbitrary monoids. This leads to the following instantiation result.

Theorem 2. *If $(M, \cdot, 1)$ is a monoid, then $(2^M, +, \cdot, ^*, 0, 1)$ is a Kleene algebra.*

The proof in Isabelle requires only a few auxiliary lemmas, notably the induction laws for powers

$$z + x \cdot y \leqslant y \longrightarrow x^n \cdot z \leqslant y, \qquad z + y \cdot x \leqslant y \longrightarrow z \cdot x^n \leqslant y$$

and the star continuity laws

$$X \cdot Y^* = \bigcup_{n \geqslant 0} X \cdot Y^n, \qquad X^* \cdot Y = \bigcup_{n \geqslant 0} X^n \cdot Y.$$

Only the two induction laws need user-guided induction; for the other laws induction is hidden in lemmas deep down in the Isabelle libraries.

Next we have shown that languages under the regular operations form Kleene algebras. Obviously, languages are sets of words, and words under concatenation and the empty word form monoids. Hence languages form Kleene algebras by Theorem 2. This simple argument by extension of mathematical structure is seamlessly supported by Isabelle. Here, words are usually programmed as lists—sets of which are, in fact, isomorphic to the free dioids—so it suffices to verify the simple algebraic fact that lists under concatenation form monoids.

instantiation `list :: (type) monoid_mult`
⟨*proof*⟩

interpretation `lan_kleene_algebra:`
`kleene_algebra "op +" "op ·" "1::α lan" "0" "op ⊆" "op ⊂" star`
⟨*proof*⟩

The instantiation statement, which captures this fact, formally links into Theorem 1 and makes the interpretation statement—languages under the regular operations form Kleene algebras—a trivial corollary of the more abstract Theorem 2 in Isabelle. The interpretation statement allows matching the Kleene algebra signature with the corresponding operations

on languages. As in the case of sublocale or subclass statements, Isabelle dictates the proof obligations. In this case, it must be checked that languages satisfy the star axioms of Kleene algebras; all other axioms being subsumed by Theorem 2.

Finally, we have shown that *regular* languages under the regular operations form Kleene algebras. Based on an existing data type for regular expressions [15] we have formalised the standard interpretation homomorphism from regular expressions to languages and introduced regular languages as a subtype of languages, as induced by this homomorphism. Isabelle's quotient package [16] supports lifting the regular operations from languages to regular languages, and transfers the facts needed to show that regular languages form Kleene algebras. Details can be found in the Archive proof document.

The two language models have been obtained by and large automatically from the abstract algebraic monoid model. Automation was supported by Isabelle's type class and locale mechanisms, by Isabelle's libraries, e.g., for numbers, monoids, powers, sets, and, last but not least, by the link with algebra. Instantiations and interpretations make all abstract theorems of Kleene algebra available in the language models. Many inductive proofs about the star in languages have thus been reduced to simple equational reasoning.

A decision procedure for regular expressions is available in Isabelle/HOL [15], but a formalised completeness proof is required to use it safely for equational reasoning in Kleene algebra. Currently we use this procedure only as an oracle. The completeness proof is based on rather tedious matrix encodings of automata constructions [17]; it has so far only been formalised in Coq [18].

5. Binary relation model

Due to Isabelle's excellent libraries for binary relations, it is fully automatic to show that binary relations of a given type form a dioid under the operations of union and relational composition, with the empty relation as additive identity and the diagonal relation as multiplicative identity.

```
interpretation rel_dioid: dioid_one_zero "op ∪" "op O" Id "{}" "op ⊆" "op ⊂"
  by (unfold_locales, auto)
```

In the interpretation proof, the unfold statement generates the proof obligations whereas Isabelle's built-in auto tactic discharges all of them at once. For an algebraic setting this is rather exceptional since tactics such as simp, auto or blast are usually not able to compete with state-of-the-art automated theorem provers which are called by Isabelle's sledgehammer tool.

As previously, the relational Kleene star is defined as a sum of powers: $R^* = \bigcup_{n \geqslant 0} R^n$. The interpretation proof that relations form Kleene algebras

```
interpretation rel_kleene_algebra: kleene_algebra "op ∪" "op O" Id "{}" "op ⊆" "op ⊂" rtrancl
  ⟨proof⟩
```

then follows precisely those for monoids and languages. It uses the same kind of continuity laws; only showing that R^* is equal to Isabelle's reflexive transitive closure operation is required in addition (Isabelle uses a specific relational power operation which is not linked properly with its monoidal one). In addition, the proof is once more modular relative to the dioid results in that only the star axioms need to be verified.

Despite the striking similarity to the language proofs, no powerset lifting is involved. One could, however, define a partial composition operation directly on ordered pairs, identify appropriate units and lift these operations to the powerset level of relations. This mathematically rather unconventional approach might allow us to further unify the programming of models in Isabelle. We discuss this in more detail below.

We have extended the relational model to omega algebras. The omega of a relation relates all elements in the relation's domain from which an infinite chain starts with all other elements [19]. It is most naturally defined coinductively.

```
coinductive_set omega :: "(α × α) set → (α × α) set" for R where
  "⟦(x,y) ∈ R; (y,z) ∈ omega R⟧ ⟹ (x,z) ∈ omega R"
```

Isabelle automatically derives a coinduction rule for the omega operation. We have proved a slightly simpler variant of this rule.

```
lemma omega_weak_coinduct:
  "P x z ⟹ (⋀x z. P x z ⟹ ∃y. (x,y) ∈ R ∧ P y z) ⟹ (x,z) ∈ omega R"
  by (metis omega.coinduct)
```

The unfold and coinduction axiom of omega algebra then follow by coinduction.

Finally we have verified the obvious: binary relations form relation algebras.

```
interpretation rel_relation_algebra: relation_algebra ⟨signature list⟩
  ⟨proof⟩
```

Once more, due to Isabelle's excellent libraries for binary relations, the proof is fully automatic. By this result a large amount of concrete reasoning with binary relations is captured algebraically. This convincingly demonstrates the unifying power of algebra. Much of explicit inductive reasoning about (reflexive) transitive closures of binary relations and the star in formal language theory has been subsumed by abstract equational reasoning, using the same algebraic laws for both models. A rationalisation of labour has thereby been achieved.

6. Trace and path models of Kleene algebra

Sets of traces form another interesting model of Kleene algebras. Intuitively, a trace represents an execution sequence of a labelled transition system or automaton in which state and action labels alternate, beginning and ending with a state label. For example, $p_1a_1p_2a_2\ldots p_{n-1}a_{n-1}p_n$ is a trace if the p_i are state labels and the a_i are action labels. *Trace fusion* merges two traces xp and qy into the trace xpy if $p = q$ and is undefined otherwise. This partial operation is lifted to a complex product on sets of traces as in the case of monoids. The Kleene star X^* of a set of traces can be defined again as a union of powers $\bigcup_{n\geqslant 0} X^n$. It can then be shown that sets of all traces built from a set of action and state labels under these operations together with set union, the empty set of traces as the additive unit and the set of all traces of length one (e.g. single state labels) as the multiplicative unit form Kleene algebras.

Traces are implemented as pairs of type $\pi \times (\alpha \times \pi)$ *list*, where π denotes the type of state labels and α the type of action labels. The fusion and complex product are formalised as follows.

```
definition "t_fusion x y ≡
  if last x = fst y then (fst x, snd x · snd y) else undefined"

definition "X · Y ≡ {t_fusion x y| x y. x ∈ X ∧ y ∈ Y ∧ last x = fst y}"
```

The operator · in the definition of the fusion product denotes list concatenation. We have shown by case analysis with respect to definedness that traces under fusion form a partial monoid. Due to Isabelle/HOL's limitations with partiality, we fully capture undefinedness only when lifting trace fusion to powersets; the last condition in the set comprehension guarantees that only well-defined trace products contribute. The proof of the interpretation statement

```
interpretation trace_kleene_algebra: kleene_algebra "op ∪" "op ·" t_one "{}" "op ⊆" "op ⊂" t_star
  ⟨proof⟩
```

follows those for languages and relations, but is more tedious and not fully subsumed by Theorem 1 due to partiality.

Path models are very similar to trace models, but only state labels are considered; action labels have been forgotten. These capture, for instance, sets of paths in a (di)graph or transition system. Our development follows essentially [19]. We have separately implemented models with and without the empty path.

Paths including the empty path have been implemented as lists. *Path fusion* is a partial operation similarly to trace fusion. In functional programming style this is written as follows.

```
fun p_fusion where
  "p_fusion [] _ = []"
| "p_fusion _ [] = []"
| "p_fusion ps (q # qs) = ps · qs"
```

For convenience, our implementation of this product is a total recursive function, and we can again establish a monoidal structure on paths. Undefined products are once more filtered out in the definition of complex product, but the filtering condition is more involved than for traces.

```
definition "p_filter ps qs ≡ (ps = [] ∧ qs = []) ∨ (ps ≠ [] ∧ qs ≠ [] ∧ last ps = hd qs)"

definition "X · Y ≡ {p_fusion ps qs| ps qs. ps ∈ X ∧ qs ∈ Y ∧ p_filter ps qs}"
```

The complexity of filtering shows up in the proofs of the multiplicative monoid laws at the powerset level. They require a case analysis over the structure of paths. From these facts it is then easy to prove that sets of paths form dioids. The Kleene star is again defined as a union of powers. The proof that sets of paths form Kleene algebras is similar to that for traces.

```
interpretation path_kleene_algebra: kleene_algebra "op ∪" "op ·" p_one "{}" "op ⊆" "op ⊂" p_star
  ⟨proof⟩
```

Alternatively, based on non-empty lists, we have built a path model without the empty path. This yields simpler definitions and proofs, but requires implementing a basic library for non-empty lists. The fusion product now becomes a primitive recursive function over paths. The filtering condition in the complex product is simply that the last element of the

first path and the first element of the second one must be equal. Verifying the dioid axioms is now almost automatic. The interpretation proofs follow the approach for the other models.

The similarity between the path and trace models is particularly apparent. They are all based on a powerset lifting of partial monoids. Moreover, languages, relations and paths can be obtained from traces by forgetting part of the structure. This can be formalised in terms of homomorphisms (forgetful functors) and Galois connections [19]. Path models are isomorphic to trace models with one single action symbol; language models are isomorphic to path models with one single state symbol (the fusion product is now total); relations are obtained by projecting on the first and the last element of a trace. On the one hand it is therefore certainly possible to derive all these models from one generic algebraic construction in Isabelle. On the other hand, however, it is not clear that the practical gain of this abstraction would justify the mathematical machinery involved and the effort in dealing with partiality.

7. Matrices

Matrices form an important model of Kleene algebras due to their relevance for completeness proofs [17]. They can be used for encoding computing systems via automata or transition systems. We provide two models; one for potentially infinite matrices and one for the more usual $m \times n$-matrices of fixed dimension. We mainly discuss the first model because it is simpler, but similar to the second. In the infinite case, matrices are functions from two index sets into some suitable algebra, which we assume to be a dioid. The only restriction is that, in the matrix product, summation ranges over a finite index set. A more general notion of product could easily be obtained, but it would leave the realm of semirings where only finite sums are available. It can then be shown that dioids are closed under matrix formation.

We have defined the type (α, β, γ) *infmatrix* as a synonym for functions of type $\alpha \to \beta \to \gamma$. The unit and zero matrix are formalised as follows.

definition *"ε i j ≡ (if i = j then 1 else 0)"*

definition *"δ i j ≡ 0"*

In the definition of ε, the equality comparison forces the two index types to be the same, which results in "square" matrices. The target type is usually constrained to be a dioid. Matrix addition is defined in the usual point-wise way.

definition *"(f ⊕ g) i j ≡ (f i j) + (g i j)"*

It is then straightforward to show that matrices over a dioid form a bounded join-semilattice with respect to addition. As already mentioned, matrix multiplication imposes a finiteness constraint on the index type over which the sum is taken.

definition *"(f ⊗ g) i j ≡ ∑k::β::finite {(f i k) · (g k j)}"*

In the dioid setting, addition is idempotent, and the finite sum $\sum_{\mathrm{k}}$ is actually a finite supremum. Its properties and preconditions are subtly different from the usual Isabelle setsum operation, and we have designed a specific library for finite suprema with around 30 facts. The finiteness constraint in the product is needed because infinite sums need not exist in a dioid. We could then verify the remaining dioid laws, some of which are rather involved. The most complex one is associativity of matrix multiplication, which relies on auxiliary lemmas for finite suprema and rearranging sums. The Isar proofs of these facts essentially translate manual proofs step by step. Given these individual algebraic laws, showing that dioids are closed under matrix formation is then automatic.

interpretation *matrix_dioid: dioid_one_zero "op ⊕" "op ⊗" ε δ "op ⩽" "op <"*
⟨*proof*⟩

We have also formalised the more standard case of $n \times m$-matrices over dioids. This is similar to a development in the Isabelle/HOL library for multivariate analysis, but we explicitly define type constructors for (square) matrices and use weaker assumptions on the element types in matrix operations. Formally,

datatype *(α,β,γ) matrix = Matrix "β atMost → γ atMost → α"*

datatype *(α,β) sqmatrix = SqMatrix "β atMost → β atMost → α"*

where α *atMost* is a finite type whose size is determined by α. We have then defined the usual operations of matrix addition and multiplication on finite (square) matrices, and proved that square matrices over a dioid form a dioid. The approach and proof effort is comparable to the infinite case.

Neither implementation so far includes the star. It is well known that matrices over a Kleene algebra form a Kleene algebra, and this result is useful for various reasons. Its absence is currently a main obstacle for safely using the Isabelle decision procedure for regular expressions for solving Kleene algebra identities. The definition of the star of a (square) matrix is recursive by splitting into appropriate sub-matrices (cf. [17]). Implementing this in Isabelle and verifying the laws

of Kleene algebra is rather tedious. An elegant existing implementation in Coq [18] shows the power of dependent types for this purpose.

It is well known that finite relation algebras with n elements are isomorphic to algebras of $n \times n$ Boolean matrices where, in addition to the dioid operations, a top matrix, the intersection of two matrices, complementation of a Boolean matrix and transposition as conversion need to be defined. We have implemented these concepts for infinite Boolean matrices (in the order mentioned) and proved that these matrices form a model of relation algebra.

definition *"τ i j ≡ True"*

definition *"(f ⊓ g) i j ≡ (f i j) · (g i j)"*

definition *"f^c ≡ (λi j. - f i j)"*

definition *"$f^\dagger$ ≡ (λi j. f j i)"*

interpretation *matrix_ra: relation_algebra* ⟨*signature list*⟩
⟨*proof*⟩

Only the verification of the residuation axiom—the last axiom in the relation algebra type class—required a certain amount of work in the matrix model. A similar construction for $n \times n$ Boolean matrices can be obtained along these lines.

8. Formal power series

Formal power series are functions from free monoids into dioids, Kleene algebras or, more generally, semirings. They are widely applied in language and automata theory, cf. Berstel and Reutenauer's book [20]. Our implementation generalises a formalisation of formal power series, as they are known from combinatorics and ring theory, by Chaieb in the Isabelle/HOL library [21].

We currently focus on Kleene algebras and leave the integration of non-idempotent semirings for future work. Although this may seem mathematically straightforward, this is not entirely trivial from a programming point of view. Dioids can use finite suprema whereas semirings require Isabelle's more general setsums for addition, which are defined for arbitrary commutative monoids. As usual, we program elements of free monoids as lists. Thus, the type of formal power series (with codomain β) is isomorphic to the type of functions from α *list* to β. In contrast, Chaieb's formal power series are functions from the natural numbers.

typedef *(α,β) fps = "{f::α list → β. True}"*
morphisms *$ Abs_fps* ⟨*proof*⟩

The function *Abs_fps* and its counterpart $ are coercions between the types of functions and formal power series.

The dioid operations on formal power series are defined as follows. The zero formal power series is the (isomorphic image of the) function $\lambda n.0$ that maps every monoid element to 0 in the dioid. The multiplicative unit of formal power series, denoted 1 by overloading, maps the monoidal unit (the empty list) to 1 in the dioid and every other element of the monoid to 0:

definition *"1 ≡ Abs_fps (λn. if n = [] then 1 else 0)"*

Addition of formal power series is defined point-wise.

definition *"f + g ≡ Abs_fps (λn. f $ n + g $ n)"*

The product of formal power series is the well known convolution or Cauchy product

$$(f \cdot g)\, n = \sum_{x \cdot y = n} (f\, x) \cdot (g\, y).$$

Programming this in Isabelle requires splitting a list representing an element of a free monoid into all possible prefix/suffix pairs and multiplying with respect to these splittings in the dioid.

definition *"f · g ≡ Abs_fps (λn. ∑{f $ x · g $ y| x y. n = x · y})"*

This summation over all splittings of n is captured by the following definition.

definition *"splitset n ≡ {(x, y)| x y. n = x · y}"*

This set can, of course, be defined more constructively by recursion. Both variants are helpful for proving algebraic properties.

The proof that formal power series form bounded join-semilattices requires little more than unfolding definitions. Establishing the multiplicative monoid properties, in contrast, is tedious, and needs a number of auxiliary lemmas and, once more, facts about finite suprema. The multiplicative left unit law is best proved by case analysis over the structure of lists using the recursive definition of splitset. For the right unit law, however, dual lists would be needed. We have therefore proved it in a more algebraic fashion from properties of finite suprema. Both proofs formalise a simple mathematical insight. Consider the left unit law

$$(1 \cdot f)\ xs = \sum_{ys \cdot zs = xs} (1\ ys) \cdot (f\ zs)$$

for a list xs. The power series 1 maps the empty list to 1 in the dioid, which yields $f\ xs$ in the product. All other lists ys are mapped to 0, which annihilates the product. Hence only $f\ xs$ survives the Cauchy product. The right unit law is dual. The proof of the associativity law relies on a generic lemma for rearranging the sums that occur, similar to the case of matrices. We have combined the proofs of the individual algebraic laws into an instantiation proof that formal power series into a dioid form a dioid.

```
instantiation fps :: (type,dioid_one_zero) dioid_one_zero
```
⟨*proof*⟩

Two approaches of formalising the star of formal power series can be found in the literature, but detailed proofs are usually omitted. First, it can be shown that for *proper* power series, which satisfy $f\ 1 = 0$, the star can be defined as the finite sum $f^*\ x = \sum_{0 \leqslant i \leqslant |x|} f^i\ x$. We have formalised the more interesting case of power series into Kleene algebras, where the star can be defined recursively as

$$f^*\ 1 = (f\ 1)^*, \qquad f^*\ x = f^*\ 1 \cdot \sum_{y \cdot z = x,\, y \neq 1} f\ y \cdot f^*\ z,$$

where $x \neq 1$. We have first defined the star on functions, where we can use Isabelle's package for recursive functions, and then lifted it to the type of formal power series.

```
fun star_fps_rep where
  "star_fps_rep f [] = (f [])⋆"
| "star_fps_rep f n = (f [])⋆ ·∑{f y · star_fps_rep f z |y z. n = y @ z ∧ y ≠ []}"

lift_definition star_fps :: "('a, 'b) fps ⇒ ('a, 'b) fps"
  is star_fps_rep ..
```

The verification of the star unfold and star induction axioms from these definitions is tedious. It uses case analyses on the structure of the free monoid and a few auxiliary lemmas. This yields the main result of this section.

```
instantiation fps :: (type,kleene_algebra) kleene_algebra
```
⟨*proof*⟩

9. Max-plus and min-plus algebras

Max-plus and min-plus algebras are important mathematical structures in their own right. A standard reference is Gondran and Minoux's book [11]. Applications include combinatorial optimisation, control theory, algorithm design or Internet routing. We have developed the basics of this approach; as far as we know, for the first time in Isabelle. All instances require extensions of number fields by elements $\pm\infty$, either in combination or separately. Separate extensions are currently not available in Isabelle, though they can easily be obtained.

We first discuss max-plus and min-plus algebras over $\mathbb{R}$. The carrier set of the max-plus algebra is the set $\mathbb{R} \cup \{-\infty\}$. The operation of addition is maximum, that of multiplication is addition, the additive unit is minus infinity and the multiplicative unit is zero.

We have defined a datatype for $\mathbb{R} \cup \{-\infty\}$ and extended the (recursive) functions of maximum and addition accordingly. After fixing the units, the instance proof that max-plus algebras over the reals form selective semirings is a straightforward case analysis over the data type; the proof that they also form dioids is then automatic from our abstract algebraic results.

The proofs for min-plus algebras are very similar. Now, the real numbers are extended by $+\infty$, multiplication is minimum, addition is multiplication, the additive unit is $+\infty$ and the multiplicative unit is 1.

None of these algebras can be expanded to Kleene algebras. In the max-plus algebra, for instance, the star should be an upper bound of $\{x^n \mid n \in \mathbb{N}\}$, but this set obviously has no upper bound in $\mathbb{R} \cup \{-\infty\}$. Such expansions work, however,

for restricted carrier sets. As an example we have formalised the min-plus algebra over $\mathbb{N} \cup \{+\infty\}$. In this case, $x^* = 1$ for all x. Verifying the laws of (commutative) Kleene algebra from this definition is trivial, based on Isabelle's library for numbers, monoids and semilattices. The combination of max/min-plus algebras with matrices is particularly interesting for algorithmic developments or modelling, for instance, discrete event systems or Petri nets.

10. Other models of Kleene algebra

We have implemented further models of dioids and Kleene algebras. It can, for instance, easily be shown that the Booleans under conjunction and disjunction and, more generally, bounded distributive lattices—distributive lattices with a least and a greatest element—form Kleene algebras. Here, multiplication is meet and the Kleene star maps each element to the greatest element of the lattice.

More interestingly, many of the models considered so far have a richer structure than that captured by Kleene algebra. In particular, residuation can be defined uniformly on all power set models, hence these and other models form action algebras. In the monoid model, residuation can be defined explicitly as

$$X \rightarrow Z = \bigcup\{Y \mid X \cdot Y \subseteq Z\}, \qquad Z \leftarrow Y = \bigcup\{X \mid X \cdot Y \subseteq Z\}.$$

This follows from general properties of Galois connections over complete lattices. To establish action algebras, we have first shown in an instance proof that $\rightarrow$ and $\leftarrow$ are upper adjoints in the Galois connection axioms of residuated lattices. The two proofs are duals and special cases of more abstract lattice theoretic proofs. From this basis, the instantiation proof for action algebras is purely algebraic; it only links the star axioms of Kleene algebras, for which monoidal models have already been built, with those of action algebras.

Residuation for languages, relations, paths and traces has been formalised in similar ways. The verification of the residuated semilattice axioms is straightforward; proofs of the star axioms of action algebra are similar to those for monoids. Finally, we have formalised residuation for the min-plus algebra over the extended natural numbers. Since multiplication is commutative there is only one residual: the operation of truncated subtraction defined by $x \rightarrow y = y - x$. Showing that this induces a residuated lattice is immediate, based on Isabelle's libraries for natural numbers. The extension to action algebra is equally simple.

11. Proof automation in relation algebra

Calculational reasoning in relation algebras is more difficult for humans than that in dioids or Boolean algebras, and perhaps even Kleene algebras. Proof automation is therefore typically less successful [22]. In this situation, a key to proof automation in applications is high-quality library design and modularisation. Domain specific knowledge is necessary to identify the right set of laws and prover specific knowledge to make these laws interact with built-in tactics and simplifiers.

When reading the following discussion, it might be helpful to consult the underlying Archive proof document [4] in parallel.

As a first measure to increase proof automation in relation algebra, we have implemented more than twenty useful laws of Boolean algebra which are currently missing in Isabelle's standard libraries. These include simplification rules, for instance

$$x \cdot y + x \cdot -y = x, \qquad (x + y) \cdot -x = y \cdot x, \qquad x + -x \cdot y = x + y,$$

isotonicity rules such as $x \leqslant y \longrightarrow w + x + z \leqslant w + y + z$, which are notoriously difficult to "find" for automated theorem provers, and Galois connections such as

$$x \cdot y = 0 \longleftrightarrow x \leqslant -y, \qquad x \cdot -y \leqslant z \longleftrightarrow x \leqslant y + z.$$

None of their proofs required any user interaction.

As a second measure, we have formalised Jónsson and Tarski's notion of *Boolean algebra with operators* [23,24] in Isabelle. Following Maddux [9] we restrict our attention to unary functions of type $B \rightarrow B$ where B is a Boolean algebra. Of particular relevance are pairs of *conjugate* functions, that is operators f and g which satisfy

$$(f\ x) \cdot y = 0 \longleftrightarrow x \cdot (g\ y) = 0.$$

We have proved the basic facts about conjugate functions, most of them from Jónsson and Tarski's articles and Maddux's book. We have shown that conjugation is a symmetric relation and that conjugates are uniquely defined. Second, we have proved that conjugates are adjoints in a Galois connection, that is,

$$f\ x \leqslant y \longleftrightarrow x \leqslant -g\ (-y).$$

It then follows easily that f and g are additive, isotone and strict, that is,

$$f\ (x + y) = f\ x + f\ y, \qquad x \leqslant y \longrightarrow f\ x \leqslant f\ y, \qquad f\ 0 = 0$$

and likewise for g. They also satisfy the cancellation laws

$$f\ (-g\ x) \leqslant -x, \qquad g\ (-f\ x) \leqslant -x.$$

All these proofs were again fully automatic. Next, we have proved modular laws for conjugate functions. After proving the auxiliary law $f\ (x \cdot -g\ y) \cdot y = 0$ and its symmetric counterpart we have derived the *modular law*

$$f\ x \cdot y = f(x \cdot g\ y) \cdot y$$

and its symmetric counterpart with a few user interactions. The modular laws of relation algebras are instances of these abstract laws.

An obvious link between Boolean algebras with operators and relation algebras are the conjugation relations between $\lambda y.\, x; y$ and $\lambda y.\, x^{\smile}; y$, as well as between $\lambda y.\, y; x$ and $\lambda y.\, y; x^{\smile}$. We have formalised this relationship in Isabelle. In total we have formalised more than fifty basic theorems of relation algebra. Only six proofs required user interaction. Among these are *Peirce's law*

$$x; y \cdot z^{\smile} = 0 \longleftrightarrow y; z \cdot x^{\smile} = 0,$$

one of Schröder's laws, which is similar, and the *Dedekind rule*

$$x; y \cdot z \leqslant \big(x \cdot z; y^{\smile}\big); \big(y \cdot x^{\smile}; z\big).$$

But all facts could be proved at the level of textbook granularity. The following proof is an example.

```
lemma dedekind: "x ; y · z ⩽ (x · z ; y⌣) ; (y · x⌣ ; z)"
proof -
  have "x ; y · z ⩽ (x · z ; y⌣) ; (y · ((x · z ; y⌣)⌣ ; z))"
    by (metis modular_2' modular_1_var)
  also have "... ⩽ (x · z ; y⌣) ; (y · x⌣ ; z)"
    by (metis conv_iso inf_le1 inf_mono mult_isol mult_isor order_refl)
  thus ?thesis
    by (metis calculation inf.commute order_trans)
qed
```

A key for increasing the automation of many of these proofs are the conjugation properties mentioned. In particular, they immediately imply the strictness or annihilation laws $x; 0 = 0$ and $0; x = 0$ of the relational dioid retract.

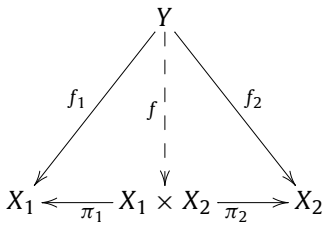
From these basic laws we have implemented special concepts, such as vectors, subidentities or tests, various kinds of functions, and a domain operation, following the books of Schmidt and Ströhlein, Schmidt and Maddux. While the basic properties of vectors and domain were proved automatically and without much user interaction, some properties of tests turned out to be surprisingly resilient, in particular the law $-(x; 1) \cdot 1' = -x \cdot 1'$, where x is a test. In the case of domain we verified little more than the axioms of domain semirings [25] in order to make all properties of this more abstract setting available in Isabelle.

We put particular emphasis on the development of a comprehensive library for functions. As usual we have implemented a partial function as a relation x satisfying $x^{\smile}; x \leqslant 1'$; whereas a total relation satisfies $1' \leqslant x; x^{\smile}$, an injective relation satisfies $x; x^{\smile} \leqslant 1'$, and a surjective relation satisfies $1' \leqslant x^{\smile}; x$. Moreover, a map or function is a total relation which is a partial function while a bijection is an injective and surjective function. We have proved most of the basic properties that can be found in the standard textbooks, and most of them automatically. For applications, we found the following properties very helpful. First, $(x \cdot z; y^{\smile}) = x; y \cdot z$, whenever y is a partial function. Second, $-(x; y) = x; -y$ whenever x is a function. Both required a moderate amount of user interaction. The following Isar proof of Theorem 4.2.2 (iii) from Schmidt and Ströhlein's book follows directly the textbook proof.

```
lemma ss_422iii: "p_fun_p y ⟹ (x · z ; y⌣) ; y = x ; y · z"
proof (rule antisym)
  assume "p_fun_p y"
  show "x ; y · z ⩽ (x · z ; y⌣) ; y"
    by (metis dedekind eq_refl  mult_subdistl order_trans)
  have "(x · z ; y⌣) ; y ⩽  x ; y · (z ; (y⌣ ; y))"
    by (metis mult_subdistr_var mult_assoc)
  also have "... ⩽ x ; y · z ; 1'"
    by (metis `p_fun_p y` inf_absorb2 inf_le1 le_infI le_infI2 mult_subdistl p_fun_p_def)
  finally show "(x · z ; y⌣) ; y ⩽ x ; y · z"
    by (metis mult.right_neutral)
qed
```

12. Formalising direct relational products

An important contribution of the Munich School consists in the axiomatisation of the concept of *direct product* by four simple identities in the calculus of relations. As far as we are aware, details have been published for the first time in Schmidt and Ströhlein's book.



It is obvious from the standard diagram for direct products that this requires a typed setting. The proofs in Schmidt and Ströhlein's book, however, suggest that the proofs demonstrating universality of the construction can be carried out safely without types. Our untyped formalisation in Isabelle directly follows their approach, but should be taken with a grain of salt since it could lead to false positives: theorems that we could prove may still fail in the presence of types; type safety has not been demonstrated formally within Isabelle. The projections π_1 and π_2, as relations, satisfy the axioms

$$\pi_1^\smile;\pi_1 = 1', \qquad \pi_2^\smile;\pi_2 = 1', \qquad \pi_1;\pi_1^\smile \cdot \pi_2;\pi_2^\smile = 1', \qquad \pi_1^\smile;\pi_2 = 1.$$

It follows that π_1 and π_2 are surjective functions. In Isabelle we have formalised direct products as follows, writing x and y instead of π_1 and π_2:

definition *"direct_product_p x y ≡*
(x⌣ ; x = 1′ ∧ y⌣ ; y = 1′ ∧ x ; x⌣ · y ; y⌣ = 1′ ∧ x⌣ ; y = 1)"

After some basic properties we have automatically proved two helpful technical lemmas about general functions using Lemma 4.2.2(iii).

lemma *dp_aux1:*
assumes *"p_fun_p z"*
and *"total_p w"*
and *"x⌣ ; z = 1"*
shows *"(w ; x⌣ · y ; z⌣) ; z = y"*
⟨*proof*⟩

lemma *dp_aux2:*
assumes *"p_fun_p z"*
and *"total_p w"*
and *"z⌣ ; x = 1"*
shows *"(w ; x⌣ · y ; z⌣) ; z = y"*
⟨*proof*⟩

Schmidt and Ströhlein's first main theorem about direct products states that these are monomorphic, that is, projections are, for given sets, characterised up to isomorphism. In fact, for two pairs of projections (w, x) and (y, z), this isomorphism is given explicitly by the following relation:

definition *"Φ ≡ (λw x y z. w ; y⌣ · x ; z⌣)"*

We have verified the isomorphism properties described in Schmidt and Ströhlein's proof automatically, using our two technical lemmas.

lemma *mono_dp_1:*
assumes *"direct_product_p w x"*
and *"direct_product_p y z"*
shows *"Φ w x y z ; y = w"*
⟨*proof*⟩

lemma *mono_dp_2:*
assumes *"direct_product_p w x"*
and *"direct_product_p y z"*
shows *"Φ w x y z ; z = x"*
⟨*proof*⟩

We have then formalised Schmidt and Ströhlein's proof that Φ is indeed a function, using the characterisation $\Phi; -1' = -\Phi$ which has been proved equivalent to the usual definition of a function in our function library. We provide a detailed proof, which follows Schmidt and Ströhlein step by step, as an example. It shows that, with support from Isabelle's automated theorem prover integration, well developed libraries and an appropriate level of abstraction, even quite advanced mathematical constructions and proofs can often be translated in a one-to-one fashion from paper and pencil proofs.

```
lemma Phi_map:
  assumes "direct_product_p w x"
  and "direct_product_p y z"
  shows "map_p (Φ w x y z)"
proof -
  have "Φ w x y z ; -(1') = Φ w x y z ; -(y ; y⌣ · z ; z⌣)"
    by (metis assms(2) direct_product_p_def)
  also have "... = Φ w x y z ; y ; -(y⌣) + Φ w x y z ; z ; -(z⌣)"
    by (metis compl_inf assms(2) ss43iii dp_map1 dp_map2 mult.assoc distrib_left)
 also have "... = w ; -(y⌣) + x ; -(z⌣)"
    by (metis (full_types) assms mono_dp_1 mono_dp_2)
 also have "... = -(w ; y⌣) + -(x ; z⌣)"
    by (metis assms(1) ss43iii dp_map1 dp_map2)
 also have "... = -(Φ w x y z)"
    by (metis compl_inf)
 finally show ?thesis
    by (metis map_prop)
qed
```

The proof that Φ is an injection is then fully automatic. Schmidt and Ströhlein write that $\Phi^\smile$ is a function as well "for reasons of symmetry". This symmetry is picked up by Isabelle and consequently the proof is fully automatic.

Finally we have constructed, for given functions f and g, a function F which makes the standard product diagram commute, and verified these commutation properties, as usual in category theory. This verification uses again our two technical lemmas and hence justifies their extraction. The proofs are fully automatic; projections are denoted by again by x and y.

```
definition "F ≡ (λ f x g y. f ; x⌣ · g ; y⌣)"

lemma f_proj:
  assumes "direct_product_p x y"
  and "map_p g"
  shows "F f x g y ; x = f"
  ⟨proof⟩

lemma g_proj:
  assumes "direct_product_p x y"
  and "map_p f"
  shows "F f x g y ; y = g"
  ⟨proof⟩
```

This completes Schmidt and Ströhlein's proof of universality of direct products. In addition we have verified universality of F more directly in the obvious category-theoretic way, assuming a second function G that makes the product diagram commute and showing, in a handful of interactions, that $G = F$.

In sum, all proofs in this section were automatic except for the verification that Φ is a map, where we have decided to follow Schmidt and Ströhlein's proof step by step. This case study demonstrates the quality of our relation algebra library and Isabelle's power in picking up the right lemmas and employing automated theorem proving technology. The fact that our development is based on untyped relations might raise objections from Munich. But, in fact, our proofs do not hide anything that Schmidt and Ströhlein themselves found worth mentioning. At least our proofs faithfully capture the case of a product $X \times X$, where types are not needed.

13. Discussion and conclusion

We have presented examples for programming and automating mathematics in the spirit of the Munich School of relation algebra. These are drawn from a reference formalisation for a hierarchy of Kleene algebras and relation algebras in Isabelle/HOL. The theory hierarchy implemented supports a wide range of computing applications and more advanced mathematical investigations. Most of the models presented are new in Isabelle, in particular the monoid, path, matrix, formal power series, min-plus and max-plus model, and the relational model of omega algebra. Interestingly, the omega operation

does not exist in the trace, path and language models presented in this paper [19]. While the construction of models usually required significant user interaction due to their inherently higher-order concepts, the degree of proof automation in calculational reasoning with Kleene and relation algebras observed was certainly encouraging.

Variants of Kleene algebras support and enrich each other. This is a main lesson learnt from our work on the Tarski–Kleene hierarchy. Kleene algebras, on the one hand, emphasise diversity by capturing different models of computational interest. By focusing on the star they describe one of the most ubiquitous and powerful operations of computing, which is absent in Tarski's axiomatisation of relation algebra. Relation algebra, on the other hand, specialises on one particular model and yields a more precise description of it. In applications such as program correctness and verification we expect that the two approaches will cooperate and complement each other in Isabelle and the tool will automatically set the scope by selecting the most useful hypotheses for proofs.

Relations are making their way from logics to mathematics and applied sciences; to quote from one of Gunther Schmidt's recent keynote speeches. Our work demonstrates how far state-of-the-art theorem proving environments such as Isabelle support this transition. On the one hand, the proof and programming power of these tools might exceed the expectations that computer scientists or mathematicians such as Gunther Schmidt had at the beginning of their careers. On the other hand, our formalisation also demonstrates some current shortcomings. Additional work is certainly needed to transform theorem provers into mainstream tools for programming and automating mathematics in relation algebra, Kleene algebras and beyond. However, to anyone working in the field of relational an algebraic methods in computer science, tools such as Isabelle or Coq are highly recommended.

Acknowledgements

The authors are grateful to Simon Foster, Victor Gomes and Walter Guttmann for contributions to the formalisation of the Tarski–Kleene hierarchy in Isabelle. Rudolf Berghammer has provided valuable information on the history of relational tool development by the Munich School—an account of which remains to be written. Hitoshi Furusawa and Yasuo Kawahara have helped to clarify the presentation of this paper through several discussions. Last, but not least, the second author is indebted to Gunther Schmidt for his continuous support and critical encouragement.

References

[1] R. Berghammer, F. Neumann, Relview – an OBDD-based computer algebra system for relations, in: V.G. Ganzha, E.W. Mayr, E.V. Vorozhtsov (Eds.), Computer Algebra in Scientific Computing, in: Lect. Notes Comput. Sci., vol. 3718, Springer, 2005, pp. 40–51.
[2] W. Kahl, Calculational relation-algebraic proofs in Isabelle/Isar, in: R. Berghammer, B. Möller, G. Struth (Eds.), Relational and Algebraic Methods in Computer Science, in: Lect. Notes Comput. Sci., vol. 3051, Springer, 2004, pp. 178–190.
[3] A. Armstrong, G. Struth, T. Weber, Kleene algebra, Archive of Formal Proofs, http://afp.sf.net/entries/Kleene_Algebra.shtml, 2013.
[4] A. Armstrong, S. Foster, G. Struth, T. Weber, Relation algebra, Archive of Formal Proofs, http://afp.sf.net/entries/Relation_Algebra.shtml, 2014.
[5] V.G. Pratt, Action logic and pure induction, in: J. van Eijck (Ed.), JELIA '90, in: Lect. Notes Comput. Sci., vol. 478, Springer, 1991, pp. 97–120.
[6] S. Foster, G. Struth, T. Weber, Automated engineering of relational and algebraic methods in Isabelle/HOL, in: H.C.M. de Swart (Ed.), RAMICS 2011, in: Lect. Notes Comput. Sci., vol. 6663, Springer, 2011, pp. 52–67.
[7] G. Schmidt, T. Ströhlein, Relationen und Graphen, Springer, 1987.
[8] G. Schmidt, Relational Mathematics, Cambridge University Press, 2011.
[9] R.D. Maddux, Relation Algebras, Elsevier, 2006.
[10] M. Wenzel, Isabelle/Isar—a versatile environment for human-readable formal proof documents, Ph.D. thesis, Technische Universität München, Germany, 2002.
[11] M. Gondran, M. Minoux, Graphs, Dioids and Semirings: New Models and Algorithms, Springer, 2010.
[12] F. Haftmann, M. Wenzel, Constructive type classes in Isabelle, in: T. Altenkirch, C. McBride (Eds.), TYPES 2006, in: Lect. Notes Comput. Sci., vol. 4502, Springer, 2007, pp. 160–174.
[13] J.C. Blanchette, L. Bulwahn, T. Nipkow, Automatic proof and disproof in Isabelle/HOL, in: C. Tinelli, V. Sofronie-Stokkermans (Eds.), FroCoS 2011, in: Lect. Notes Artif. Intell., vol. 6989, Springer, 2011, pp. 12–27.
[14] D. Kozen, On Kleene algebras and closed semirings, in: B. Rovan (Ed.), MFCS 1990, in: Lect. Notes Comput. Sci., vol. 452, Springer, 1990, pp. 26–47.
[15] A. Krauss, T. Nipkow, Proof pearl: Regular expression equivalence and relation algebra, J. Autom. Reason. 49 (2012) 95–106.
[16] C. Kaliszyk, C. Urban, Quotients revisited for Isabelle/HOL, in: W.C. Chu, W.E. Wong, M.J. Palakal, C.-C. Hung (Eds.), SAC 2011, ACM, 2011, pp. 1639–1644.
[17] D. Kozen, A completeness theorem for Kleene algebras and the algebra of regular events, Inf. Comput. 110 (1994) 366–390.
[18] T. Braibant, D. Pous, An efficient Coq tactic for deciding Kleene algebras, in: M. Kaufmann, L. Paulson (Eds.), ITP 2010, in: Lect. Notes Comput. Sci., vol. 6172, Springer, 2010, pp. 163–178.
[19] P. Höfner, G. Struth, Algebraic notions of nontermination: Omega and divergence in idempotent semirings, J. Log. Algebr. Program. 79 (2010) 794–811.
[20] J. Berstel, C. Reutenauer, Rational Series and Their Languages, Springer, 1988.
[21] A. Chaieb, Formal power series, J. Autom. Reason. 47 (2011) 291–318.
[22] P. Höfner, G. Struth, On automating the calculus of relations, in: A. Armando, P. Baumgartner, G. Dowek (Eds.), IJCAR 2008, in: Lect. Notes Comput. Sci., vol. 5195, Springer, 2008, pp. 50–66.
[23] B. Jónsson, A. Tarski, Boolean algebras with operators, part 1, Am. J. Math. 73 (1951) 891–939.
[24] B. Jónsson, A. Tarski, Boolean algebras with operators, part 2, Am. J. Math. 74 (1952) 127–162.
[25] J. Desharnais, G. Struth, Internal axioms for domain semirings, Sci. Comput. Program. 76 (2011) 181–203.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 103–119

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Computing minimal extending sets by relation-algebraic modeling and development

Rudolf Berghammer *

Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24098 Kiel, Germany

ARTICLE INFO

Article history:
Available online 10 February 2014

Keywords:
Social choice theory
Tournament
Minimal extending set
Relation algebra
Relation-algebraic modeling
RelView tool

ABSTRACT

In 2009 F. Brandt introduced minimal extending sets as a new tournament solution. Until now no efficient algorithm is known for their computation and, in fact, the NP-hardness of the corresponding decision problem has been proved quite recently by F. Brandt, P. Harrenstein and H.G. Seedig in a working paper. We develop a relation-algebraic specification of minimal extending sets. It is algorithmic and can be directly translated into the programming language of the ROBDD-based computer algebra system RelView. By this general and model-oriented approach we obtain almost the same efficiency as the specifically tailored program for minimal extending sets mentioned in another recent working paper of F. Brandt, A. Dau and H.G. Seedig. We also discuss an alternative approach that is based on testing the extending set property with relation-algebraic means, and a greedy strategy. Under favorable conditions it allows to solve much larger problem instances than our first solution and that of F. Brandt, A. Dau and H.G. Seedig.

1. Introduction

One of the most elementary questions in social choice theory is the aggregation of the preferences of certain individuals (voters, agents) in view of given alternatives (candidates) to a collective so-called *dominance relation*. For instance, in case of approval voting (see [9]) the individual preferences are the sets of alternatives the single voters approve and then collectively an alternative a dominates an alternative b if the number of voters which approve a is greater than the number of voters which approve b. Frequently it is assumed that there are no ties. Then the dominance relation is asymmetric (hence, irreflexive) and complete, i.e., the relation of a *tournament*. A tournament is acyclic iff it is a linear strict-order. Since finiteness is a general assumption in this context to ensure the existence of certain extremal sets and elements, acyclic tournaments possess exactly one winner that (strictly) dominates all other alternatives—the greatest element. But in case of cyclic tournaments it may happen that no alternative exists which dominates all other ones. To overcome this problem, a series of so-called *tournament solutions* has been proposed which define the sets of winners in such cases. For an overview see, for example, [20], and for computational issues see, for example, [10,19].

In [10,11] F. Brandt introduces minimal extending sets as a new tournament solution. Compared to other well-known tournament solutions its computational complexity has been an open problem for a long time. Quite recently it has been shown in the working paper [15] that deciding whether an alternative is contained in it is NP-hard. Even on small instances the computation of minimal extending sets seems to be rather difficult. In another working paper [14] F. Brandt, A. Dau

* Tel.: +49 431 7272; fax: +49 431 7613.
E-mail address: rub@informatik.uni-kiel.de.

http://dx.doi.org/10.1016/j.jlap.2014.02.002
2352-2208/

and H.G. Seedig report on their implementations of algorithms for the computation of the most important tournament solutions. Doing so, they also mention the cases where, despite the NP-hardness of the general problem, larger instances with specific properties can be treated successfully. Especially, this holds for the computation of the Banks set by means of the enumeration of all minimal feedback vertex sets via the elaborated algorithm of S. Gaspers and M. Mnich (see [18]). It is sufficiently fast on all tournaments with a not too large number of Banks trajectories (i.e., maximal transitive sets). But, although their program for the computation of minimal extending sets also bases on the algorithm of [18], the authors comment in [14] on it that it "already takes about 3 minutes on instances of 25 alternatives". In the original version of [14] that the author used when preparing the first version of the present paper they still write that "it is only feasible for instances of at most 20 alternatives".

In [7] we describe a simple computing technique for a series of tournament solutions. It rests upon relation algebra in the sense of [23,24] as the methodical tool, a goal-directed development of relation-algebraic specifications of tournament solutions from their formal logical descriptions, and the ROBDD-based computer algebra system RelView (see [4,21,22,27]) for the evaluation of the developed specifications and the visualization of the computed results. Because of the very positive results of our practical experiments, the above-mentioned difficulties when computing minimal extending sets, and since this tournament solution is not treated in [7], we have applied our technique to it and want to present the solutions and their performances in this paper.

Its remainder is organized as follows. In Section 2 we summarize the relation-algebraic preliminaries. The introduction of the minimal extending sets of a tournament and the development of an executable relation-algebraic specification from the formal logical description is done in Section 3. Section 4 shows how the result of Section 3 can be translated into RelView-code and presents results of our practical experiments with the tool. They demonstrate that, despite the very general and model-oriented approach, the use of ROBDDs to implement relations in RelView leads to a solution that is almost as efficient as the specifically tailored program mentioned in [14]. Section 5 describes an alternative method for solving the minimal extending sets problem that is based on a relation-algebraic model of truth values, the testing of the extending set property with relation-algebraic means and a greedy strategy. It assumes as pre-condition that the input has a unique minimal extending set. Our experiments together with those mentioned in [15] show that this seems to be true for all practical matters. In Section 6 we again present results of practical applications. They demonstrate that under favorable conditions the alternative approach allows to solve much larger problem instances than our first approach and the solution of F. Brandt, A. Dau and H.G. Seedig. The last Section 7 contains some concluding remarks.

2. Relation-algebraic preliminaries

In this section we recall the basics of heterogeneous relation algebra that are needed in the remainder of the paper. For more details, see [23,24] for example.

Given sets X and Y we write $R : X \leftrightarrow Y$ if R is a (typed, binary) relation with source X and target Y, i.e., a subset of the direct product $X \times Y$. If the sets of R's *type* $X \leftrightarrow Y$ are finite, then we may consider R as a Boolean matrix with $|X|$ rows and $|Y|$ columns. Since a Boolean matrix interpretation of relations is well suited for many purposes and also used by the RelView tool as the main possibility to depict them, in this paper we will frequently use matrix terminology and notation. Especially, we speak about the entries, rows and columns of a relation/matrix and write $R_{x,y}$ instead of $(x, y) \in R$ or $x\,R\,y$.

We assume the reader to be familiar with the five relation-algebraic basic operations, viz. R^T (*transposition*), $\overline{R}$ (*complement*), $R \cup S$ (*union*), $R \cap S$ (*intersection*) and $R; S$ (*composition*), the three special relations O (*empty relation*), L (*universal relation*) and I (*identity relation*), and the two relation-algebraic predicates $R \subseteq S$ (*inclusion*) and $R = S$ (*equality*). In case of the relations O, L and I we overload the symbols, i.e., avoid the binding of types to them. Furthermore, we assume that composition binds stronger than union and intersection.

For $R : X \leftrightarrow Y$ and $S : X \leftrightarrow Z$, by $syq(R, S) = \overline{R^\mathsf{T}; \overline{S}} \cap \overline{\overline{R^\mathsf{T}}; S}$ their *symmetric quotient* $syq(R, S) : Y \leftrightarrow Z$ is defined. We will only use its point-wise description saying that for all $y \in Y$ and $z \in Z$ it holds

$$syq(R, S)_{y,z} \quad \Longleftrightarrow \quad \forall x : R_{x,y} \leftrightarrow S_{x,z}, \tag{1}$$

where the variable x ranges over X. In logical formulae the ranges of the variables are usually expressed via set-memberships. When we will translate logical formulae into relation-algebraic expressions, some of such memberships will be important for reaching the desired result, some however not, like $x \in X$ in case of (1). To improve readability, in the logical formulae of this paper we will explicitly state only the important membership-relationships that are transformed into relation-algebraic constructions. Those which are not used within a calculation are mentioned in the surrounding text.

Vectors are a well-known relation-algebraic means to model subsets of a given set. For our applications it suffices to define vectors as specific relations with the target being the singleton set $\mathbf{1} = \{\perp\}$. In the Boolean matrix interpretation a vector is a Boolean column vector and, as in linear algebra, we prefer in this context lower case letters. Furthermore, we omit in case of a vector $r : X \leftrightarrow \mathbf{1}$ always the second subscript, i.e., write r_x instead of $r_{x,\perp}$. Then r *describes*, by definition, the subset Y of X if for all $x \in X$ it holds r_x iff $x \in Y$.

If $r : X \leftrightarrow \mathbf{1}$ is a vector and Y is the subset of the set X that the vector r describes, then $inj(r) : Y \leftrightarrow X$ denotes the *embedding relation* (in [24] called *natural injection*) of the set Y into the superset X, which is induced by r. In Boolean

matrix terminology this means that the relation $inj(r)$ is obtained from the identity relation $\mathsf{I} : X \leftrightarrow X$ by deleting all rows which do not correspond to an element of Y, and point-wisely this means that for all $y \in Y$ and $x \in X$ it holds

$$inj(r)_{y,x} \quad \Longleftrightarrow \quad y = x. \tag{2}$$

In conjunction with powersets 2^A we will frequently use *membership relations* $\mathsf{M} : A \leftrightarrow 2^A$. Point-wisely they are defined for all $x \in A$ and $X \in 2^A$ by the equivalence

$$\mathsf{M}_{x,X} \quad \Longleftrightarrow \quad x \in X. \tag{3}$$

If the $2^{|A|}$ single columns of the relation M are considered as $2^{|A|}$ vectors of type $A \leftrightarrow \mathbf{1}$, then each column precisely describes one set of the powerset 2^A in the above sense. A combination of M with embedding relations allows such a *column-wise enumeration* also for subsets $\mathfrak{A}$ of 2^A. Let $r : 2^A \leftrightarrow \mathbf{1}$ describe $\mathfrak{A}$ in the sense defined above. If we define $S = \mathsf{M}; inj(r)^\mathsf{T} : A \leftrightarrow \mathfrak{A}$, then for all $x \in A$ and $X \in \mathfrak{A}$ it holds $S_{x,X}$ iff $x \in X$. In the Boolean matrix model this means that S consists of those columns of M which correspond to 1-entries of r. With the help of M, furthermore, it is easy to specify the *superset* ordering on 2^A relation-algebraically. Namely, if we define the relation $\mathsf{S} : 2^A \leftrightarrow 2^A$ by $\mathsf{S} = \overline{\overline{\mathsf{M}^\mathsf{T}}; \mathsf{M}}$, then, by a simple calculation, for all sets $X, Y \in 2^A$ it can be shown that

$$\mathsf{S}_{X,Y} \quad \Longleftrightarrow \quad X \supseteq Y. \tag{4}$$

To model direct products $X \times Y$ relation-algebraically, the *projection relations* $\pi : X \times Y \leftrightarrow X$ and $\rho : X \times Y \leftrightarrow Y$ are the convenient means. They are the relational variants of the well-known projection functions and, hence, fulfill

$$\pi_{(a,b),x} \quad \Longleftrightarrow \quad a = x \qquad \rho_{(a,b),y} \quad \Longleftrightarrow \quad b = y, \tag{5}$$

for all $(a, b) \in X \times Y$, $x \in X$ and $y \in Y$. Projections enable us to specify two pairing operations. Assuming π and ρ as above, the *right pairing* (also called *fork*) of $R : Z \leftrightarrow X$ and $S : Z \leftrightarrow Y$ is defined as $[R, S] = R; \pi^\mathsf{T} \cap S; \rho^\mathsf{T}$. This leads to $Z \leftrightarrow X \times Y$ as type of $[R, S]$ and to the point-wise description

$$[R, S]_{z,(x,y)} \quad \Longleftrightarrow \quad R_{z,x} \wedge S_{z,y}, \tag{6}$$

for all $z \in Z$ and $(x, y) \in X \times Y$. The dual construction is the *left pairing* (or *strict join*) of $R : X \leftrightarrow Z$ and $S : Y \leftrightarrow Z$. It is defined as $[\![R, S]\!] = \pi; R \cap \rho; S$. In this case we get $X \times Y \leftrightarrow Z$ as its type and

$$[\![R, S]\!]_{(x,y),z} \quad \Longleftrightarrow \quad R_{x,z} \wedge S_{y,z}, \tag{7}$$

for all $z \in Z$ and $(x, y) \in X \times Y$, as its point-wise description. Based on the left pairing we may define for a relation $R : X \leftrightarrow Y$ the vector $vec(R) : X \times Y \leftrightarrow \mathbf{1}$ by $vec(R) = [\![R, \mathsf{I}]\!]; \mathsf{L}$, where $\mathsf{I} : Y \leftrightarrow Y$ and $\mathsf{L} : Y \leftrightarrow \mathbf{1}$. Using again a point-wise description we get for all $x \in X$ and $y \in Y$ that

$$R_{x,y} \quad \Longleftrightarrow \quad vec(R)_{(x,y)}. \tag{8}$$

Because of this property $vec(R)$ represents the relation R as a vector and, therefore, we call it the *vector model* of R.

3. A relation-algebraic specification of minimal extending sets

Within the framework of relation algebra, a *tournament* is not a directed graph, but a relation $D : A \leftrightarrow A$ on a non-empty and *finite* set A (in our case the set of *alternatives*) such that the properties $D \cap D^\mathsf{T} = \mathsf{O}$ and $\overline{\mathsf{I}} \subseteq D \cup D^\mathsf{T}$ hold. The equation specifies D as *asymmetric* and the inclusion as *complete*. In case of a relationship $D_{a,b}$ we say that alternative a *dominates* alternative b. The social choice literature frequently uses the symbol "$\succ$" for the dominance relation and $a \succ b$ for our relationship $D_{a,b}$. Due to this interpretation of dominance as "is greater than", for a set X of alternatives an alternative $a \in X$ is called *maximal* (or *undominated*) in X if there is no $b \in X$ such that $D_{b,a}$ holds. For sets, social choice theory uses the common order-theoretic definitions of maximality and minimality w.r.t. set inclusion. Given a subset $\mathfrak{A}$ of 2^A, a set $X \in \mathfrak{A}$ is called *maximal* in $\mathfrak{A}$ if $X \subseteq Y$ implies $X = Y$, for all $Y \in \mathfrak{A}$, and $X \in \mathfrak{A}$ is called *minimal* in $\mathfrak{A}$ if $Y \subseteq X$ implies $Y = X$, for all $Y \in \mathfrak{A}$.

Next we need the notion of a transitive set: $X \in 2^A$ is said to be *transitive* (w.r.t. the tournament D) if for all $a, b, c \in X$ from $D_{a,b}$ and $D_{b,c}$ it follows $D_{a,c}$. Hence, X is a transitive set iff the *restriction* $D_{|X} : X \leftrightarrow X$ of the tournament D to X is a transitive relation, that is, a linear strict-order because of the assumed asymmetry of D. Since A is finite, transitive sets possess exactly one maximal alternative that dominates all other alternatives. The *Banks set* $BA(D)$ of D consists of the maximal elements of the maximal transitive subsets of A. This tournament solution is introduced in [2], and [26] shows that deciding whether an alternative is contained in it is NP-complete.

In F. Brandt's papers [10,11] minimal extending sets are introduced via Banks sets. For the tournament $D : A \leftrightarrow A$ and a set $B \in 2^A$ it is specified that

$$B \text{ is an } \textit{extending set of } D \quad \Longleftrightarrow \quad \forall a : a \notin B \rightarrow a \notin BA(D_{|B \cup \{a\}}), \tag{9}$$

where a ranges over A and $D_{|B\cup\{a\}}$ denotes the restriction of the tournament D to the set $B\cup\{a\}$, and B is called a *minimal extending set* of D if it is a minimal element of the set of all extending sets of D w.r.t. set inclusion. Extending sets are also called *BA-stable*, since the right-hand side of (9) is an instance of the general notion of *stability* for the specific tournament solution *BA*; cf. [10,11] for details. To get a relation-algebraic specification of minimal extending sets it is advantageous to use the following immediate consequence of (9): $B\in 2^A$ is an extending set of D iff *there is no* $a\in A\setminus B$ *that is a maximal element of a maximal transitive subset of* $B\cup\{a\}$.

As the first step towards a relation-algebraic specification of minimal extending sets we develop a vector $t:2^A\leftrightarrow\mathbf{1}$ that describes the set of the transitive sets of the tournament D as a subset of 2^A. The following calculation, in which $X\in 2^A$ is assumed to be an arbitrary set and a, b and c range over A, is a slight modification of the calculation given in [7]. We only include it to make the paper self-contained. The calculation starts with a logical formalization of the transitivity of the set X. Besides the point-wise descriptions of the basic operations of relation algebra and some fundamental laws of predicate logic, it applies in the second transformation property (3), with $\mathsf{M}:A\leftrightarrow 2^A$, and also property (8), and in the third transformation it uses property (7).

$$
\begin{aligned}
& \forall a,b,c : a\in X\wedge b\in X\wedge c\in X\wedge D_{a,b}\wedge D_{b,c}\rightarrow D_{a,c}\\
\iff\ & \neg\exists a,b,c : a\in X\wedge b\in X\wedge c\in X\wedge D_{a,b}\wedge D_{b,c}\wedge\overline{D}_{a,c}\\
\iff\ & \neg\exists a,b : \mathsf{M}_{a,X}\wedge\mathsf{M}_{b,X}\wedge vec(D)_{(a,b)}\wedge\exists c : \mathsf{M}_{c,X}\wedge\overline{D}_{a,c}\wedge D_{b,c}\\
\iff\ & \neg\exists a,b : [\![\mathsf{M},\mathsf{M}]\!]_{(a,b),X}\wedge vec(D)_{(a,b)}\wedge\exists c : [\![\overline{D},D]\!]_{(a,b),c}\wedge\mathsf{M}_{c,X}\\
\iff\ & \neg\exists a,b : vec(D)_{(a,b)}\wedge[\![\mathsf{M},\mathsf{M}]\!]_{(a,b),X}\wedge\big([\![\overline{D},D]\!];\mathsf{M}\big)_{(a,b),X}\\
\iff\ & \neg\exists a,b : vec(D)^{\mathsf{T}}_{\bot,(a,b)}\wedge\big([\![\mathsf{M},\mathsf{M}]\!]\cap[\![\overline{D},D]\!];\mathsf{M}\big)_{(a,b),X}\\
\iff\ & \overline{vec(D)^{\mathsf{T}};\big([\![\mathsf{M},\mathsf{M}]\!]\cap[\![\overline{D},D]\!];\mathsf{M}\big)}_{\bot,X}\\
\iff\ & \overline{vec(D)^{\mathsf{T}};\big([\![\mathsf{M},\mathsf{M}]\!]\cap[\![\overline{D},D]\!];\mathsf{M}\big)}^{\mathsf{T}}_{X}.
\end{aligned}
$$

Because of the point-wise description of the equality of relations, from the last expression of this calculation we obtain

$$t=\overline{vec(D)^{\mathsf{T}};\big([\![\mathsf{M},\mathsf{M}]\!]\cap[\![\overline{D},D]\!];\mathsf{M}\big)}^{\mathsf{T}}:2^A\leftrightarrow\mathbf{1} \tag{10}$$

as relation-algebraic specification of the vector t, and this ends the first step.

Also in the remaining steps of our development the point-wise descriptions of the basic operations of relation algebra and the properties (1) to (8) of Section 2 are combined with logical reasoning. But we again will explicitly mention only the applications of the properties (1) to (8).

The next important notion used in the description of minimal extending sets is that of a maximal transitive subset. Therefore, it seems to be reasonable to consider in the second development step a relation $Q:2^A\leftrightarrow 2^A$ that relates $Y\in 2^A$ and $X\in 2^A$ iff the set X is a maximal transitive subset of the set Y. To obtain a relation-algebraic specification of Q we proceed as in the case of the vector t. In the starting point of the following calculation the relationship t_X is used to express the transitivity of the subset X of A. Its first transformation applies property (4), with $\mathsf{S}:2^A\leftrightarrow 2^A$, and in the third transformation a universal vector $\mathsf{L}:2^A\leftrightarrow\mathbf{1}$ is introduced for reasons of type adaptation. The range of the variable Z is 2^A.

$$
\begin{aligned}
& t_X\wedge X\subseteq Y\wedge\forall Z : t_Z\wedge Z\subseteq Y\wedge X\subseteq Z\rightarrow X=Z\\
\iff\ & t_X\wedge\mathsf{S}_{Y,X}\wedge\forall Z : t_Z\wedge\mathsf{S}_{Y,Z}\wedge\mathsf{S}_{Z,X}\rightarrow\mathsf{I}_{Z,X}\\
\iff\ & t_X\wedge\mathsf{S}_{Y,X}\wedge\neg\exists Z : t_Z\wedge\mathsf{S}_{Y,Z}\wedge\mathsf{S}_{Z,X}\wedge\overline{\mathsf{I}}_{Z,X}\\
\iff\ & \big(\mathsf{L};t^{\mathsf{T}}\big)_{Y,X}\wedge\mathsf{S}_{Y,X}\wedge\neg\exists Z : \big(\mathsf{L};t^{\mathsf{T}}\big)_{Y,Z}\wedge\mathsf{S}_{Y,Z}\wedge\mathsf{S}_{Z,X}\wedge\overline{\mathsf{I}}_{Z,X}\\
\iff\ & \big(\mathsf{L};t^{\mathsf{T}}\cap\mathsf{S}\big)_{Y,X}\wedge\neg\exists Z : \big(\mathsf{L};t^{\mathsf{T}}\cap\mathsf{S}\big)_{Y,Z}\wedge(\mathsf{S}\cap\overline{\mathsf{I}})_{Z,X}\\
\iff\ & \big(\mathsf{L};t^{\mathsf{T}}\cap\mathsf{S}\big)_{Y,X}\wedge\overline{\big(\mathsf{L};t^{\mathsf{T}}\cap\mathsf{S}\big);(\mathsf{S}\cap\overline{\mathsf{I}})}_{Y,X}\\
\iff\ & \big(\mathsf{L};t^{\mathsf{T}}\cap\mathsf{S}\cap\overline{\big(\mathsf{L};t^{\mathsf{T}}\cap\mathsf{S}\big);(\mathsf{S}\cap\overline{\mathsf{I}})}\big)_{Y,X}.
\end{aligned}
$$

Again, as an immediate consequence of the last expression, we get for the relation Q the relation-algebraic specification

$$Q=\mathsf{L};t^{\mathsf{T}}\cap\mathsf{S}\cap\overline{\big(\mathsf{L};t^{\mathsf{T}}\cap\mathsf{S}\big);(\mathsf{S}\cap\overline{\mathsf{I}})}:2^A\leftrightarrow 2^A. \tag{11}$$

In the third step of our development we attack the problem of specifying, for arbitrarily given $a\in A$ and $B\in 2^A$, the property that a is the maximal element of a maximal transitive subset of $B\cup\{a\}$. Originally, we aimed at the relation-algebraic specification of a relation of type $A\leftrightarrow 2^A$. But during the calculation we discovered that it is more simple (and, in fact, the resulting RelView-code also proved to be slightly more efficient than the code resulting from the original idea)

if we aim instead at the relation-algebraic specification of the vector model of this relation. Hence, in this step we consider a vector $r : A \times 2^A \leftrightarrow \mathbf{1}$ such that $r_{(a,B)}$ holds iff a is a maximal element of a maximal transitive subset of $B \cup \{a\}$. In the following four calculations of this step X and Y range over 2^A and b ranges over A. The starting point of the first calculation uses that $Q_{B\cup\{a\},X}$ specifies X as a maximal transitive subset of $B \cup \{a\}$. In the second transformation property (3) is applied, with $\mathsf{M} : A \leftrightarrow 2^A$.

$$\begin{aligned}
\exists X : Q_{B\cup\{a\},X} \wedge \neg\exists b : b \in X \wedge D_{b,a} &\iff \exists X : \big(\exists Y : B \cup \{a\} = Y \wedge Q_{Y,X}\big) \wedge \neg\exists b : b \in X \wedge D_{b,a} \\
&\iff \exists X : \big(\exists Y : B \cup \{a\} = Y \wedge Q_{Y,X}\big) \wedge \neg\exists b : \mathsf{M}_{b,X} \wedge D^{\mathsf{T}}_{a,b} \\
&\iff \exists X : \big(\exists Y : B \cup \{a\} = Y \wedge Q_{Y,X}\big) \wedge \overline{D^{\mathsf{T}}; \mathsf{M}}_{a,X}.
\end{aligned}$$

Now we split the development and consider first the left part of the outer conjunction of the last formula. Here we proceed as follows, using property (3) in the second transformation, with $\mathsf{M} : A \leftrightarrow 2^A$, property (6) in the fourth transformation, and property (1) in the fifth transformation. The type of the identity relation I the second transformation introduces is $A \leftrightarrow A$.

$$\begin{aligned}
\exists Y : B \cup \{a\} = Y \wedge Q_{Y,X} &\iff \exists Y : \big(\forall b : (b = a \vee b \in B) \leftrightarrow b \in Y\big) \wedge Q_{Y,X} \\
&\iff \exists Y : \big(\forall b : (\mathsf{I}_{b,a} \vee \mathsf{M}_{b,B}) \leftrightarrow \mathsf{M}_{b,Y}\big) \wedge Q_{Y,X} \\
&\iff \exists Y : \big(\forall b : \neg(\overline{\mathsf{I}}_{b,a} \wedge \overline{\mathsf{M}}_{b,B}) \leftrightarrow \mathsf{M}_{b,Y}\big) \wedge Q_{Y,X} \\
&\iff \exists Y : \big(\forall b : \overline{[\![\overline{\mathsf{I}}, \overline{\mathsf{M}}]\!]}_{b,(a,B)} \leftrightarrow \mathsf{M}_{b,Y}\big) \wedge Q_{Y,X} \\
&\iff \exists Y : syq\big(\overline{[\![\overline{\mathsf{I}}, \overline{\mathsf{M}}]\!]}, \mathsf{M}\big)_{(a,B),Y} \wedge Q_{Y,X} \\
&\iff \big(syq\big(\overline{[\![\overline{\mathsf{I}}, \overline{\mathsf{M}}]\!]}, \mathsf{M}\big); Q\big)_{(a,B),X}.
\end{aligned}$$

Due to the type of the relation-algebraic expression of the calculation's result, viz. $A \times 2^A \leftrightarrow 2^A$, we have to adapt the relation $\overline{D^{\mathsf{T}}; \mathsf{M}}$ of the right part of the outer conjunction in such a way that it gets the same type $A \times 2^A \leftrightarrow 2^A$, but the meaning with regard to the relationship between a and X does not change. This can be obtained via the projection relation $\pi : A \times 2^A \leftrightarrow A$. In the corresponding calculation given below, π is introduced in the second transformation, and the correctness of the transformation follows from the left property of (5).

$$\begin{aligned}
\overline{D^{\mathsf{T}}; \mathsf{M}}_{a,X} &\iff \exists b : a = b \wedge \overline{D^{\mathsf{T}}; \mathsf{M}}_{b,X} \\
&\iff \exists b : \pi_{(a,B),b} \wedge \overline{D^{\mathsf{T}}; \mathsf{M}}_{b,X} \\
&\iff \big(\pi; \overline{D^{\mathsf{T}}; \mathsf{M}}\big)_{(a,B),X}.
\end{aligned}$$

After these two auxiliary calculations we now can conclude the original calculation as given below, using the universal vector $\mathsf{L} : 2^A \leftrightarrow \mathbf{1}$ in the third transformation to obtain the pattern for composition.

$$\begin{aligned}
\exists X : Q_{B\cup\{a\},X} \wedge \neg\exists b : b \in X \wedge D_{b,a} &\iff \exists X : \big(\exists Y : B \cup \{a\} = Y \wedge Q_{Y,X}\big) \wedge \overline{D^{\mathsf{T}}; \mathsf{M}}_{a,X} \\
&\iff \exists X : \big(syq\big(\overline{[\![\overline{\mathsf{I}}, \overline{\mathsf{M}}]\!]}, \mathsf{M}\big); Q\big)_{(a,B),X} \wedge \big(\pi; \overline{D^{\mathsf{T}}; \mathsf{M}}\big)_{(a,B),X} \\
&\iff \exists X : \big(syq\big(\overline{[\![\overline{\mathsf{I}}, \overline{\mathsf{M}}]\!]}, \mathsf{M}\big); Q \cap \pi; \overline{D^{\mathsf{T}}; \mathsf{M}}\big)_{(a,B),X} \wedge \mathsf{L}_X \\
&\iff \Big(\big(syq\big(\overline{[\![\overline{\mathsf{I}}, \overline{\mathsf{M}}]\!]}, \mathsf{M}\big); Q \cap \pi; \overline{D^{\mathsf{T}}; \mathsf{M}}\big); \mathsf{L}\Big)_{(a,B)}.
\end{aligned}$$

And finally this leads to the relation-algebraic specification of the vector r we have searched for in this development step via the equation (and additional typing)

$$r = \big(syq\big(\overline{[\![\overline{\mathsf{I}}, \overline{\mathsf{M}}]\!]}, \mathsf{M}\big); Q \cap \pi; \overline{D^{\mathsf{T}}; \mathsf{M}}\big); \mathsf{L} : A \times 2^A \leftrightarrow \mathbf{1}. \tag{12}$$

By means of the vector r of (12) is now possible to specify in the fourth step relation-algebraically a further vector $es : 2^A \leftrightarrow \mathbf{1}$ that describes the subset of 2^A consisting of the extending sets of the tournament D. Below the corresponding calculation is presented. In this development $B \in 2^A$ is assumed as an arbitrary set, X ranges over 2^A, and a ranges over A. The starting point uses that $r_{(a,B)}$ holds iff a is a maximal element of a maximal transitive subset of $B \cup \{a\}$. In the second transformation the right property of (5) is applied, with $\rho : A \times 2^A \leftrightarrow 2^A$, and also property (3), with $\mathsf{M} : A \leftrightarrow 2^A$, and the third transformation of the calculation uses property (8).

$$\begin{aligned}
\neg\exists a : a \notin B \wedge r_{(a,B)} &\iff \neg\exists a, X : X = B \wedge a \notin X \wedge r_{(a,X)} \\
&\iff \neg\exists a, X : \rho_{(a,X),B} \wedge \overline{\mathsf{M}}_{a,X} \wedge r_{(a,X)} \\
&\iff \neg\exists a, X : \rho^{\mathsf{T}}_{B,(a,X)} \wedge vec(\overline{\mathsf{M}})_{(a,X)} \wedge r_{(a,X)}
\end{aligned}$$

$$\begin{aligned} &\iff && \neg\exists a, X : \rho^{\mathsf{T}}_{B,(a,X)} \wedge \left(vec(\overline{\mathsf{M}}) \cap r\right)_{(a,X)} \\ &\iff && \overline{\rho^{\mathsf{T}}; \left(vec(\overline{\mathsf{M}}) \cap r\right)_B}. \end{aligned}$$

If we apply some well-known relation-algebraic rules concerning transposition (see [23,24] for example), then we get as final relation-algebraic specification of this step that

$$es = \overline{\left(vec(\overline{\mathsf{M}}) \cap r\right)^{\mathsf{T}}; \rho}^{\mathsf{T}} : 2^A \leftrightarrow \mathbf{1}. \tag{13}$$

The reason for this version, instead of $r = \overline{\rho^{\mathsf{T}}; (vec(\overline{\mathsf{M}}) \cap r)}$, is caused by the RelView system. Namely, a ROBDD-implementation of relations implies that, compared with a simple Boolean matrix implementation, transposition becomes more costly. This is due to the fact that it requires to exchange the variables for the encoding of the elements of the source with the variables for the encoding of the elements of the target. But in case of $\mathbf{1}$ as source or target transposition only means to exchange source and target (in RelView: two numbers), the ROBDD of the relation remains unchanged. See [22] for details.

Now we are almost done. It only remains, as the last step, to compute from the vector es a further vector $mes : 2^A \leftrightarrow \mathbf{1}$ that describes (again as a subset of 2^A) the subset of the minimal sets of the set es describes. If we assume $B \in 2^A$ to be an arbitrary set, then we can calculate as given below, where X ranges over 2^A and the first transformation uses property (4), with $\mathsf{S} : 2^A \leftrightarrow 2^A$, and additionally introduces the identity relation $\mathsf{I} : 2^A \leftrightarrow 2^A$.

$$\begin{aligned} es_B \wedge \forall X : es_X \wedge X \subseteq B \rightarrow X = B \quad &\iff && es_B \wedge \neg\exists X : es_X \wedge \mathsf{S}_{B,X} \wedge \overline{\mathsf{I}}_{B,X} \\ &\iff && es_B \wedge \neg\exists X : (\mathsf{S} \cap \overline{\mathsf{I}})_{B,X} \wedge es_X \\ &\iff && es_B \wedge \overline{(\mathsf{S} \cap \overline{\mathsf{I}}); es_B} \\ &\iff && \left(es \cap \overline{(\mathsf{S} \cap \overline{\mathsf{I}}); es}\right)_B. \end{aligned}$$

Hence, the subset of 2^A that contains the minimal extending sets of the tournament D is described by the vector mes with the relation-algebraic specification

$$mes = es \cap \overline{(\mathsf{S} \cap \overline{\mathsf{I}}); es} : 2^A \leftrightarrow \mathbf{1}. \tag{14}$$

How to get the minimal elements of a vector (set), here es, w.r.t. an order relation, here S, already appears as a relational function in [23].

In [10,11] it is conjectured that every tournament admits a unique minimal extending set. Meanwhile this so-called ME-conjecture is disproved. Using the probabilistic method, in [12] the existence of a counterexample for a weakening of the so-called TEQ-conjecture of [25] is shown. It has about 10^{136} alternatives and is also a counterexample for the ME-conjecture. As a consequence, nowadays the set $ME(D)$ of *minimal extending sets winners* of the tournament $D : A \leftrightarrow A$ is defined as the union of all minimal extending sets of D. Suppose $a \in A$ to be an arbitrary alternative. With the help of the relation-algebraic specification (14) then we get the property

$$\begin{aligned} a \in ME(D) \quad &\iff && \exists X : mes_X \wedge a \in X \\ &\iff && \exists X : \mathsf{M}_{a,X} \wedge mes_X \\ &\iff && (\mathsf{M}; mes)_a, \end{aligned}$$

where X ranges over 2^A and the second transformation uses property (3), with $\mathsf{M} : A \leftrightarrow 2^A$. This result implies that the set $ME(D)$ is described as a subset of A by the vector win, which is relation-algebraically specified by

$$win = \mathsf{M}; mes : A \leftrightarrow \mathbf{1}. \tag{15}$$

It is easy to check whether D admits a unique minimal extending set. This is the case iff, in Boolean vector terminology, exactly one of the entries of the vector mes is 1 (and all other entries are 0). In terms of relation algebra this means that *surjectivity* $\mathsf{L} = \mathsf{L}; mes$ and *injectivity* $mes; mes^{\mathsf{T}} \subseteq \mathsf{I}$ of mes hold or, in other words, that this vector is a *point* in the sense of [23,24]. As a consequence, the uniqueness check for minimal extending sets reduces to the test of the point property of the vector mes.

4. Implementation and results of experiments

RelView is a specific purpose computer algebra system for the visualization and manipulation of relations and for relational prototyping and programming. It is written in the C programming language, uses ROBDDs (short for: reduced ordered binary decision diagrams) for the implementation of relations, and makes full use of the X-windows graphical user interface. Details can be found in [4,21,22] and on the system's homepage [27], and applications concerning social choice theory can be found, for instance, in [5–8]. Already at the end of the 1980s the development of RelView was initiated by G. Schmidt.

The first version of the tool is described in [1]. Since 1993 the working group "Computer aided program development" at the University of Kiel is responsible for the further development. RelView is available free of charge from its homepage [27].

The main purpose of the RelView tool is the evaluation of relation-algebraic expressions and the visualization of the computed results. All expressions are constructed from the relations of its workspace, using pre-defined operations and tests and user-defined functions and programs. A RelView-program is much like a function procedure in conventional programming languages, except that it only uses relations as data type and is not able to modify the tool's workspace (that is, its applications are *free of side effects*). It starts with a head line containing the program name and the list of formal parameters, which stand for relations. Then the declarations of the local domains, functions and variables follow, where declarations of direct product domains can be used to introduce projection relations. The third part of a RelView-program is the body, a while-program over relations. As such a program computes a value, finally, its last part consists of a return-clause, which is a relation-algebraic expression whose value after the execution of the body is the result of the program.

Below is the immediate translation of the relation-algebraic specifications (10) to (15) into a RelView-program for computing the vector $win : A \leftrightarrow \mathbf{1}$ of winners from the input tournament $D : A \leftrightarrow A$, where a RelView-program `vec` for the relational function *vec* is assumed to be at hand. In this program `^` denotes transposition, `-` complement, `|` union, `&` intersection, `*` composition, `[|...]` left pairing and `[...|]` right pairing. Via the pre-defined RelView-operations `I` and `On1` identity relations and empty vectors, respectively, are computed, where the types/sources of the results are determined by those of the arguments. The membership relation $\mathsf{M} : A \leftrightarrow 2^A$ is computed by means of the pre-defined RelView-operation `epsi`, with the base set A provided by the (vector) argument $\mathsf{O} : A \leftrightarrow \mathbf{1}$. Besides the tournament D the program needs the universal vector $\mathsf{L} : 2^A \leftrightarrow \mathbf{1}$ as a second argument. It provides in the declaration of the direct product domain `AxPA` for the direct product $A \times 2^A$ the relation $\mathsf{L};\mathsf{L}^\mathsf{T} : 2^A \leftrightarrow 2^A$ for the second component set 2^A. (In a RelView-program the declaration of a direct product domain P for $X \times Y$ is possible by `P = PROD(R,S)`, where the relations $R : X \leftrightarrow X$ and $S : Y \leftrightarrow Y$ are defined via the parameters of the program and/or relations of the system's workspace.) Furthermore, L is used in the expression $\mathsf{L};t^\mathsf{T} \cap \mathsf{S}$ of the specification (11) and in the specification (12) of r. Finally we introduce an auxiliary variable `Q1` to avoid the two-fold evaluation of the expression $\mathsf{L};t^\mathsf{T} \cap \mathsf{S}$, and apply the pre-defined RelView-operation `minsets` to compute the vector description *mes* of the set of minimal extending sets. The latter operation works directly on the ROBDDs and is much faster than the construction with the superset relation S that originally is used in the specification (14) (see also [8] for more details).

```
ME(D,L)
  DECL AxPA = PROD(D,L*L^);
       pi, rho, M, S, t, Q1, Q, r, es, mes
  BEG  pi = p-1(AxPA);
       rho = p-2(AxPA);
       M = epsi(On1(D));
       S = -(-M^*M);
       t = -(vec(D)^*([|M,M] & [|-D,D]*M))^;
       Q1 = L*t^ & S;
       Q = Q1 & -(Q1*(S & -I(S)));
       r = (syq(-[-I(D),-M|],M)*Q & pi*-(D^*M))*L;
       es = -((vec(-M) & r)^*rho)^;
       mes = minsets(es)
       RETURN M*mes
  END.
```

By means of some pre-defined operations the RelView system allows to generate random relations, also of specific size and density and with specific algebraic properties. See Appendix A for more details. We have used this feature to test the above RelView-program with regard to its running time on randomly generated tournaments $D : A \leftrightarrow A$, where $10 \leqslant |A| \leqslant 22$. For tournaments with more than 22 alternatives the RelView-program `ME` failed to work because of lack of memory. All experiments have been carried out with the newest version of RelView (Version 8.1, released September 2012) on an ordinary desktop computer (with an AMD Phenom II X4 955 processor and 8 GB 1600 MHz DDR3 RAM, running Linux).

In the following table we present the average running times (in seconds) of our first series of experiments with randomly generated tournaments $D : A \leftrightarrow A$. All average times for $10 \leqslant |A| \leqslant 20$ result from 30 experiments per size, that for $|A| = 21$ from 20 experiments, and that for $|A| = 22$ from 10 experiments (where in each case the times for the random generation of tournaments are included, but they are negligible). The latter number is rather rounded; for $|A| = 22$ the running times of the 10 experiments range from 5518 s to 6966 s.

$\|A\|$	10	11	12	13	14	15	16	17	18	19	20	21	22
time	0.03	0.13	0.33	0.75	2.35	5.20	14.1	34.5	89.6	258	745	2125	6000

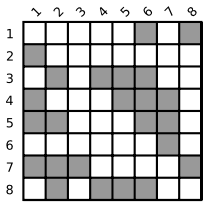


Fig. 1. A weak tournament on 8 alternatives ....

In a second series of experiments we have investigated whether the running times depend on the "distance" of the tournaments D to linear strict-orders, i.e., on the minimum number of pairs (a, b) from D that have to be replaced by (b, a) to get a linear strict-order. The following table shows the results (again in seconds) for $|A| = 12$, 14, 16, 18, 20 and 5 different distances, where the latter are not given as numbers but described by the percentages of pairs that have to be reversed to transform D into a linear strict-order. All average times for $|A| = 12$, 14, 16, 18 result from 30 experiments per percentage and size, and for $|A| = 20$ we performed 10 experiments per percentage.

percentage	10%	20%	30%	40%	50%
time $\|A\| = 12$	0.26	0.34	0.34	0.36	0.37
time $\|A\| = 14$	2.00	2.33	2.52	2.63	2.65
time $\|A\| = 16$	12.0	14.0	14.7	14.9	16.3
time $\|A\| = 18$	82.5	86.5	91.4	94.1	95.0
time $\|A\| = 20$	740	727	729	745	748

Hence, within the range we have tested smaller distances of tournaments to linear strict-orders seem to lead to a bit smaller running times of the RelView-program `ME`.

Finally we have been interested in the running times of the different parts of the RelView-program. As a result of the corresponding experiments we can say that the computation of the vector r from the relation Q is the most expensive step. In `ME` the execution of the corresponding eighth assignment of the body requires roughly half of the total running time. The other costly step is the computation of the relation Q from the vector t. Here in `ME` about one third of the total running time is needed for the execution of the corresponding sixth and seventh assignment.

Our relation-algebraic algorithm has been obtained without specific knowledge about the problem domain. We have only modeled the given problem in the language of relation algebra and then left the execution of the result to a clever implementation of this structure. Under this point of view we have applied a *declarative problem solving* approach. And, in fact, the RelView-program `ME` satisfies many of the characteristics of declarative programs. Our RelView-experiments show again that in specific situations declarative solutions can compete with specifically tailored imperative solutions.

Recall F. Brandt's former ME-conjecture that we have mentioned in Section 3 and that non-constructively has been disproved by a huge counterexample. Despite of a huge number of experiments we have performed with RelView for days, using a RelView-program that randomly generates tournaments, applies to each of them a modification of the above RelView-program `ME` (with `mes` as the return-clause instead of `M*mes`), and tests the point property of the obtained result, we have not found a counterexample with $|A| \leqslant 22$. However, further experiments with the tool have shown that many and also small counterexamples exist in the more general setting of (asymmetric but) non-complete dominance relations, so-called *weak tournaments*. Such dominance relations may occur if ties are allowed and the latter then are indicated by pairs of unrelated alternatives. In the following we present a small counterexample in case of weak tournaments, found by RelView in a split second. We use this example also to give an impression of some features of the tool.

Consider the set $A = \{1, 2, \ldots, 7, 8\}$ of 8 alternatives and the weak tournament $D : A \leftrightarrow A$ that is represented in RelView as the Boolean 8×8 matrix of Fig. 1, with the alternatives as row and as column labels. (Via a labeling mechanism in RelView it is possible to attach arbitrary identifiers as row and column labels for explanatory purposes.) In such a RelView-matrix a black square denotes a 1-entry and a white square denotes a 0-entry so that, for instance, alternative 1 dominates precisely the two alternatives 6 and 8, alternative 2 dominates only alternative 1 and alternative 3 dominates the four alternatives 2, 4, 5 and 6.

The leftmost of the three RelView-pictures of Fig. 2 shows a Boolean 8×33 matrix that column-wisely enumerates the 33 extending sets of D, in the sense introduced in Section 2. It can easily be checked that the first and the third column of this matrix describe the two minimal extending sets $\{4, 7, 8\}$ and $\{3, 4, 7\}$, respectively. These two columns are enumerated via the Boolean 8×2 matrix in the middle. From the definition of the set $ME(D)$ it follows that its vector description equals the union of the columns of this matrix. The resulting vector is shown as the rightmost one of the three RelView-pictures. In terms of relation algebra and using the embedding relations introduced in (2), the leftmost matrix depicts the relation $\mathsf{M}; inj(es)^{\mathsf{T}}$, the matrix in the middle depicts $\mathsf{M}; inj(mes)^{\mathsf{T}}$, and the vector depicts $\mathsf{M}; mes$ or, equivalently, $(\mathsf{M}; inj(mes)^{\mathsf{T}}); \mathsf{L}$, where the two vectors $es : 2^A \leftrightarrow \mathbf{1}$ and $mes : 2^A \leftrightarrow \mathbf{1}$ are as defined in the relation-algebraic specifications (13) and (14), respectively.

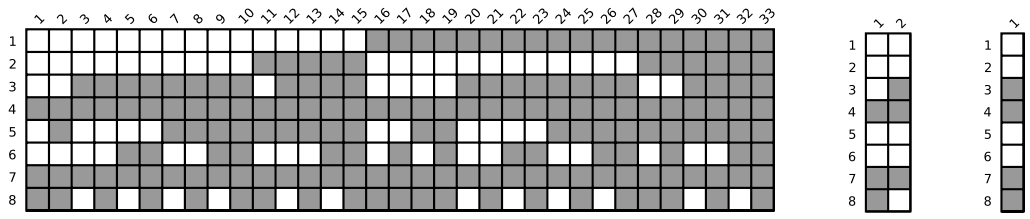


Fig. 2. ...and its extending sets, minimal extending sets, and minimal extending sets winners.

5. An alternative approach

In the following we first show how to recognize extending sets with relation-algebraic means. Using this result we then develop a greedy-like heuristic, the RelView-implementation of which under favorable conditions allows us to compute minimal extending sets for much larger tournaments than the RelView-program `ME` of Section 3 and the program mentioned in [14]. *It should be pointed out that the correctness of the heuristic is based on the pre-condition that the input tournament possesses a unique minimal extending set*. Because of our experiments and those mentioned in [15] we believe that this is true for all tournaments which appear in practical applications. If there is more than one minimal extending set, then the heuristic yields a subset of the minimal extending sets winners, i.e., an approximation of the winner set only.

As additional relation-algebraic construction we use a specification of *vector inclusion* by means of an expression with specific values only. For vectors $r, s : X \leftrightarrow \mathbf{1}$ the latter is given by $incl(r,s) = \overline{\mathsf{L}; (r \cap \bar{s})}$, where L has type $\mathbf{1} \leftrightarrow X$ (i.e., is a transposed universal vector). Due to the types of r, s and L the type of $incl(r,s)$ is $\mathbf{1} \leftrightarrow \mathbf{1}$. Furthermore, simple proofs show the following two facts.

$$incl(r,s) = \mathsf{L} \iff r \subseteq s \qquad incl(r,s) = \mathsf{O} \iff \neg(r \subseteq s). \tag{16}$$

For example, the left equivalence is shown by

$$incl(r,s) = \mathsf{L} \iff \overline{\mathsf{L}; (r \cap \bar{s})} = \mathsf{L} \iff \mathsf{L}; (r \cap \bar{s}) = \mathsf{O} \iff r \cap \bar{s} = \mathsf{O} \iff r \subseteq s,$$

where direction "$\Rightarrow$" of the third step of this calculation uses the so-called Tarski-rule (see [23,24]) in combination with the vector property of $r \cap \bar{s}$. Because of the two properties of (16), the relation-algebraic specification $incl(r,s)$ complies with the claim that it describes $r \subseteq s$ if $\mathsf{L} : \mathbf{1} \leftrightarrow \mathbf{1}$ stands for the truth value "true" and $\mathsf{O} : \mathbf{1} \leftrightarrow \mathbf{1}$ stands for the truth value "false". In the programming language of RelView the two relations $\mathsf{L} : \mathbf{1} \leftrightarrow \mathbf{1}$ and $\mathsf{O} : \mathbf{1} \leftrightarrow \mathbf{1}$ are available as `TRUE()` and `FALSE()`, respectively.

To recognize extending sets relation-algebraically, we assume $D : A \leftrightarrow A$ to be a tournament and consider an arbitrary set of alternatives. If we start with the definition of this set $B \in 2^A$ to be an extending set of D as given in (9), then we can calculate as given below, where a ranges over A.

$$\begin{aligned}
B \text{ extending set of } D &\iff \forall a : a \notin B \to a \notin BA(D_{|B \cup \{a\}}) \\
&\iff \forall a : a \in BA(D_{|B \cup \{a\}}) \to a \in B \\
&\iff \bigcup_a BA(D_{|B \cup \{a\}}) \subseteq B \\
&\iff BA(D_{|B}) \cup \bigcup_{a \notin B} BA(D_{|B \cup \{a\}}) \subseteq B \\
&\iff \bigcup_{a \notin B} BA(D_{|B \cup \{a\}}) \subseteq B.
\end{aligned}$$

Only direction "$\Rightarrow$" of the third transformation of the calculation requires an explanation: Assume $x \in \bigcup_a BA(D_{|B \cup \{a\}})$, that is, $x \in BA(D_{|B \cup \{a\}})$ for some $a \in A$. If $x = a$, then the assumption yields $x \in B$. If $x \neq a$, then $x \in BA(D_{|B \cup \{a\}})$ implies $x \in X$ for some (maximal transitive) subset X of $B \cup \{a\}$ and $x \neq a$ shows $x \in X \setminus \{a\} \subseteq B$. The correctness of the last transformation follows from $BA(D_{|B}) \subseteq B$.

Now we assume that the subset B of A is described by the vector $r : A \leftrightarrow \mathbf{1}$ and model/describe elements of A (considered as singleton subsets of A) by points of type $A \leftrightarrow \mathbf{1}$. If $a \in A$ is described by the point $p : A \leftrightarrow \mathbf{1}$, then a little reflection shows that $B \cup \{a\}$ is described as subset of A by $r \cup p : A \leftrightarrow \mathbf{1}$ and

$$D^{r,p} = inj(r \cup p); D; inj(r \cup p)^{\mathsf{T}} : \big(B \cup \{a\}\big) \leftrightarrow \big(B \cup \{a\}\big). \tag{17}$$

relation-algebraically specifies the restriction of the tournament D to the set $B \cup \{a\}$. If we, furthermore, assume a relation-algebraic specification *banks* for computing Banks sets to be at hand (as developed in [7,8]) such that the vector

$banks(D^{r,p}) : (B \cup \{a\}) \leftrightarrow \mathbf{1}$ describes the Banks set of $D^{r,p}$ as a subset of $B \cup \{a\}$, then the above calculation shows that r describes an extending set of D iff $isES(D,r) = \mathsf{L}$, where $isES(D,r)$ is relation-algebraically specified as

$$isES(D,r) = incl\Big(\bigcup_{p \subseteq \overline{r}} inj(r \cup p)^{\mathsf{T}}; banks\big(D^{r,p}\big), r\Big) : \mathbf{1} \leftrightarrow \mathbf{1}, \tag{18}$$

with p ranging over all points of type $A \leftrightarrow \mathbf{1}$. The composition of each vector $banks(D^{r,p})$ with the transposed embedding relation $inj(r \cup p)^{\mathsf{T}} : A \leftrightarrow (B \cup \{a\})$ from the left is necessary to describe the Banks set of each restricted tournament $D^{r,p}$ as a subset of A, and not as a subset of $B \cup \{a\}$ as $banks(D^{r,p})$ does.

Translated into the programming language of RelView we obtain for the two relation-algebraic specifications (17) and (18) the following code. In it `banks` computes the vector description $banks(D)$ of the Banks set of D (it is the RelView-version of the specification of [7,8]) and `isES` tests whether r describes an extending set of D. Compared with the RelView-program `ME` of Section 4, in the following two RelView-programs `banks` and `isES` six new pre-defined RelView-operations are used, viz. `incl` for testing inclusion, `empty` for testing emptiness, `inj` for the embedding relation, `point` for the (deterministic) selection of a point that is contained in a non-empty vector, `maxsets` as counterpart of `minsets` for the vector description of the set of maximal sets, and `dom` for composing the argument with a universal vector from the right.

```
banks(D)
  DECL M, t, BT
  BEG  M = epsi(On1(D));
       t = -(vec(D)^*([|M,M] & [|-D,D]*M))^;
       BT = M*inj(maxsets(t))^
       RETURN dom(BT & -(D^*BT))
  END.

isES(D,r)
  DECL union, w, p, Drp
  BEG  union = On1(D);
       w = -r;
       WHILE -empty(w) DO
         p = point(w);
         Drp = inj(r | p)*D*inj(r | p)^;
         union = union | inj(r | p)^*banks(Drp);
         w = w & -p OD
       RETURN incl(union,r)
  END.
```

In the RelView-program `banks` the second assignment computes the vector t of the relation-algebraic specification (10), the relation BT enumerates the maximal transitive subsets (the Banks trajectories) as columns, and the expression of the return-clause computes the vector of their maximal elements.

In [11] it is proven that the Banks set of a tournament D is an extending set of D and in [15] it is shown that each minimal extending set of D is a subset of the Banks set $BA(D)$. Experiments with the RelView tool have shown that in many cases $BA(D)$ is a good approximation for $ME(D)$. But there is room for improving the approximation by applying a simple heuristic. It follows the greedy strategy and looks as follows, if instead of sets of alternatives directly their vector descriptions are considered.

(a) Start with $v_1 = banks(D)$, that is, with the vector description of the Banks set of D.
(b) Construct a chain $v_1 \supset v_2 \supset \ldots$ of vectors of type $A \leftrightarrow \mathbf{1}$ such that each v_i describes an extending set of D and is obtained from its predecessor v_{i-1} by the removal of a point, i.e., as $v_i = v_{i-1} \cap \overline{p}$ with $p \subseteq v_{i-1}$ being a point.
(c) The chain terminates with a vector $v_n : A \leftrightarrow \mathbf{1}$ for which each removal of a point $v_n \cap \overline{p}$ describes a non-extending set of D.

Based on the RelView-program `isES`, the above heuristic (a)–(c) can easily be implemented in RelView via the subsequent RelView-program `approxME` that computes the terminal vector v_n of the above chain.

```
approxME(D)
  DECL v, w
  BEG  v = banks(D);
       w = nextVect(D,v);
       WHILE -empty(w) DO
```

```
        v = w;
        w = nextVect(D,v) OD
      RETURN v
  END.
```

In this program the auxiliary RelView-program `nextVect`, that we present below, yields during the construction of the chain $v_1 \supset v_2 \supset \ldots$ of vectors for a non-terminal vector $v \neq v_n$ the next vector $v \cap \overline{p}$, where $p \subseteq v$ is a point, and for the terminal vector v_n the empty vector $\mathsf{O} : A \leftrightarrow \mathbf{1}$. For obtaining `nextVect` we have applied a well-established programming technique with a Boolean variable, here `found`, that terminates the search for an object with a specific property as early as possible.

```
nextVect(D,v)
  DECL w, found, r, p
  BEG  w = v;
       found = FALSE();
       r = On1(v);
       WHILE -empty(w) & -found DO
         p = point(w);
         IF isES(D,v & -p) THEN found = TRUE();
                                r = v & -p
                           ELSE w = w & -p FI OD
       RETURN r
  END.
```

Notice that in the RelView-program `nextVect` we use that under our interpretation of the only two relations L and O of type $\mathbf{1} \leftrightarrow \mathbf{1}$ as truth values the two Boolean operations $\overline{\ }$ and $\cap$ on relations correspond to the two propositional connectives $\neg$ and $\wedge$, respectively. To report the end of the chain by the empty vector is possible since extending sets are always non-empty.

In general, the set of extending sets of a tournament is not upwards closed. As a consequence, there are tournaments $D : A \leftrightarrow A$ for which the result $v_n : A \leftrightarrow \mathbf{1}$ computed by the RelView-program `approxME` does not describe the subset $ME(D)$ of A but a proper superset X of $ME(D)$. Hence, it remains the task to check the extending set property for all proper subsets B of this superset X. Decisive for its relation-algebraic solution is the specification

$$psubsets(v) = syq\big(\mathsf{M}, inj(v)^{\mathsf{T}}; \mathsf{M}'\big); \mathsf{L} \cap \overline{syq(\mathsf{M}, v)} : 2^A \leftrightarrow \mathbf{1}, \tag{19}$$

where the vector $v : A \leftrightarrow \mathbf{1}$ is assumed to be given, $inj(v) : X \leftrightarrow A$ is the embedding relation induced by v (consequently v describes the subset X of A), $\mathsf{M} : A \leftrightarrow 2^A$ and $\mathsf{M}' : X \leftrightarrow 2^X$ are two membership relations, and the universal vector L has type $2^X \leftrightarrow \mathbf{1}$. In [3] it is shown that $syq(\mathsf{M}, v) : 2^A \leftrightarrow \mathbf{1}$ is a point. A little point-wise calculation shows that this point describes X as an element of the powerset 2^A, i.e., $syq(\mathsf{M}, v)_B$ iff $B = X$, for all sets $B \in 2^A$. As a consequence, for all sets $B \in 2^A$ we get that $\overline{syq(\mathsf{M}, v)}_B$ iff $B \neq X$. In combination with the following calculation (where the ranges of Y, a and x are 2^X, A and X, respectively) this shows that the vector $psubsets(v)$ of the relation-algebraic specification (19) describes the set of all proper subsets of X as a subset of 2^A. The calculation's second transformation uses property (1) and the fourth one uses property (2). Next property (3) is applied twice, with $\mathsf{M} : A \leftrightarrow 2^A$ and $\mathsf{M}' : X \leftrightarrow 2^X$. The correctness of the final transformation follows from the assumed range of Y, that is, from $Y \in 2^X$.

$$\begin{aligned}
\big(syq\big(\mathsf{M}, inj(v)^{\mathsf{T}}; \mathsf{M}'\big); \mathsf{L}\big)_B &\iff \exists Y : syq\big(\mathsf{M}, inj(v)^{\mathsf{T}}; \mathsf{M}'\big)_{B,Y} \wedge \mathsf{L}_Y \\
&\iff \exists Y : \big(\forall a : \mathsf{M}_{a,B} \leftrightarrow \big(inj(v)^{\mathsf{T}}; \mathsf{M}'\big)_{a,Y}\big) \\
&\iff \exists Y : \big(\forall a : \mathsf{M}_{a,B} \leftrightarrow \exists x : inj(v)_{x,a} \wedge \mathsf{M}'_{x,Y}\big) \\
&\iff \exists Y : \big(\forall a : \mathsf{M}_{a,B} \leftrightarrow \exists x : x = a \wedge \mathsf{M}'_{x,Y}\big) \\
&\iff \exists Y : (\forall a : a \in B \leftrightarrow a \in Y) \\
&\iff \exists Y : Y = B \\
&\iff B \subseteq X.
\end{aligned}$$

An immediate translation of the relation-algebraic specification (19) into the programming language of the RelView tool leads to the following code.

```
psubsets(v)
  DECL M, Inj, Mv
  BEG  M = epsi(v);
```

```
      Inj = inj(v);
      Mv = epsi(On1(Inj))
      RETURN dom(syq(M,Inj^*Mv)) & -syq(M,v)
  END.
```

Each point $p : 2^A \leftrightarrow \mathbf{1}$ and its corresponding vector M; $p : A \leftrightarrow \mathbf{1}$, where $\mathsf{M} : A \leftrightarrow 2^A$, describe the same set: the point as an element of 2^A and the vector as a subset of A. Using this fact and following the pattern of the above RelView-program `nextVect` it is rather simple to get with the help of the two RelView-programs `psubsets` and `isES` the following RelView-program `smallerES`. It computes, given a tournament $D : A \leftrightarrow A$ and the vector description $v : A \leftrightarrow \mathbf{1}$ of an extending set B of D, the vector description of a strictly smaller (w.r.t. inclusion) extending set X of D, if such a set $X \subset B$ exists, or the empty vector $\mathsf{O} : A \leftrightarrow \mathbf{1}$, otherwise.

```
smallerES(D,v)
  DECL M, s, found, r, p
  BEG  M = epsi(v);
       s = psubsets(v);
       found = FALSE();
       r = On1(v);
       WHILE -empty(s) & -found DO
         p = point(s);
         IF isES(D,M*p) THEN found = TRUE();
                             r = M*p
                        ELSE s = s & -p FI OD
       RETURN r
  END.
```

Now we are already done. It only remains as the last task to take the approximation v_n computed by the RelView-program `approxME` as a starting point and to compute then by means of a loop and calls of `smallerES` a strictly decreasing chain $v_n \supset v_{n+1} \supset \ldots$ of vectors of type $A \leftrightarrow \mathbf{1}$ until a call of `smallerES` yields the empty vector $\mathsf{O} : A \leftrightarrow \mathbf{1}$. Then the vector description of the (as unique assumed) minimal extending set is reached. Expressed in the programming language of RelView this procedure looks as follows.

```
heuristicME(D)
  DECL v, w
  BEG  v = approxME(D);
       w = smallerES(D,v);
       WHILE -empty(w) DO
         v = w;
         w = smallerES(D,v) OD
       RETURN v
  END.
```

Under favorable conditions the heuristic of this section, i.e., the RelView-program `heuristicME` together with its auxiliary programs `smallerES`, `psubsets`, `nextVect`, `approxME`, `isES` and `banks`, allows to solve much larger problem instances than the RelView-program `ME` of Section 3 and the program mentioned in [14].

6. Results of practical applications

In the first part of this section we present two applications of the RelView-programs of Section 5. The tournament of the second one stems from the social choice literature and shows the limitations of the approach of Section 5 as well as of the computational power of the RelView tool. We then compare the new solution with that of Section 3. For the experiments we have used the same RelView-version and the same desktop computer as mentioned in Section 4. Finally we report on further experiments concerning the search of small counterexamples for the former ME-conjecture.

If larger tournaments $D : A \leftrightarrow A$ are generated at random using a probability model with a uniform distribution, then the expected percentage of pairs that have to be reversed to transform D into a linear strict-order is 50% since for all $a, b \in A$ with $a \neq b$ the probability of $D_{a,b}$ equals the probability of $D_{b,a}$. As a theoretical result in this respect in [17] it is shown that in a probability model that assigns each tournament D on n alternatives the same probability $2^{\binom{n}{2}}$ the probability that a randomly chosen tournament D satisfies $|BA(D)| = n$ goes to 1 as n goes to infinite. But in practical applications, like sports tournaments, voting, and preference modeling, there are stronger and weaker teams, more and less popular candidates, and good and bad alternatives, respectively. As a consequence, the resulting tournaments have smaller distances to linear strict-orders and their Banks sets and minimal extending sets tend to be smaller, too. In such

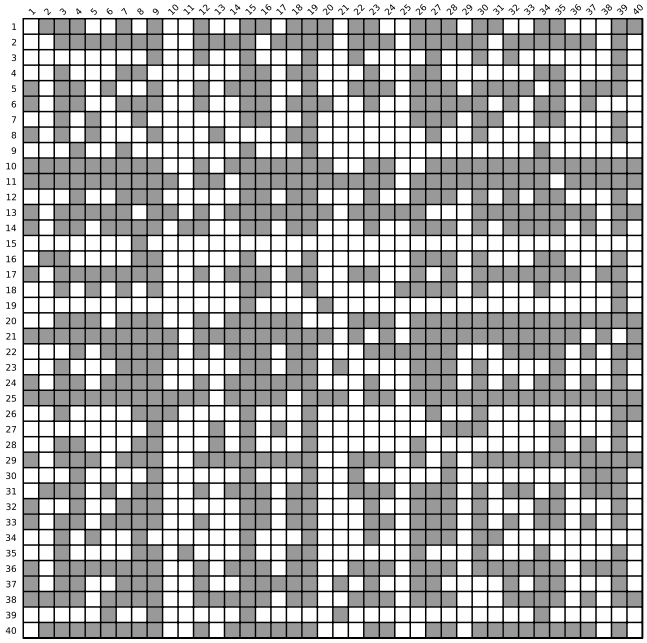


Fig. 3. A tournament on 40 alternatives ....

situations the RelView-programs of Section 5 allow the successful treatment of much larger tournaments than the "exact" RelView-program ME of Section 3.

To demonstrate such an application, in the picture of Fig. 3 a randomly generated tournament $D : A \leftrightarrow A$ on the set $A = \{1, 2, \ldots, 39, 40\}$ of 40 alternatives is depicted as a RelView-matrix. The expected percentage of pairs from D that have to be reversed to get a linear strict-order is 10%. For obtaining D a simple technique has been applied that allows to construct (by means of certain pre-defined RelView-operations) random tournaments with a user-specified distance to linear strict-orders. Details can be found in Appendix A.

The RelView-pictures of Fig. 4 visualize the computation of the minimal extending set of D. At the first position the vector description of the Banks set of D is shown. From it we get that the Banks set consists of 20 alternatives. Then follows a Boolean 40×18 matrix. It column-wisely enumerates, from left to right, the 18 vectors of the chain $banks(D) = v_1 \supset v_2 \supset \cdots \supset v_{18}$ of the heuristic given by the steps (a), (b) and (c) of Section 5. The columns of the Boolean 40×7 matrix at the third position correspond to the 7 proper subvectors of v_{18}. They describe the 7 proper subsets of the subset $\{11, 13, 25\}$ of A, i.e., of the subset that the vector v_{18} describes. According to RelView none of these subvectors describes

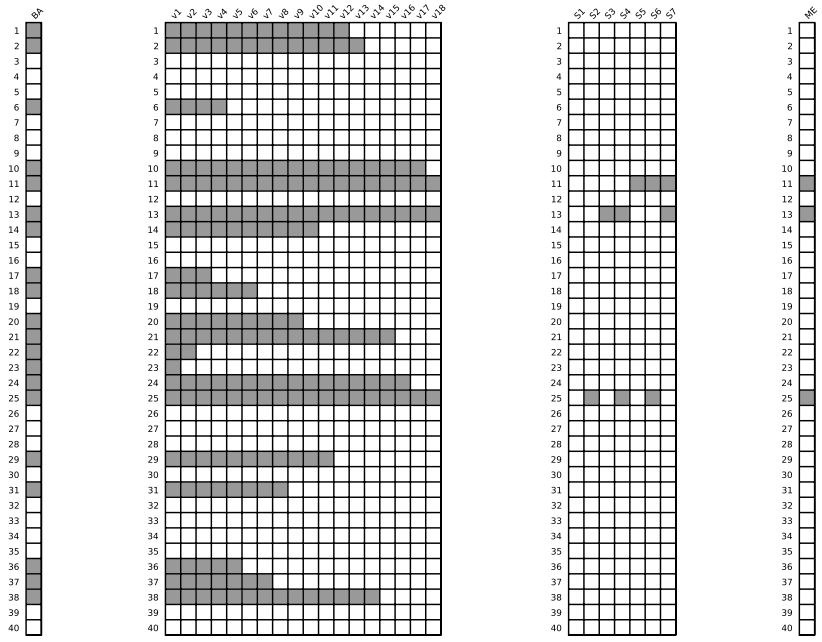


Fig. 4. ...and a visualization of the computation of its minimal extending set.

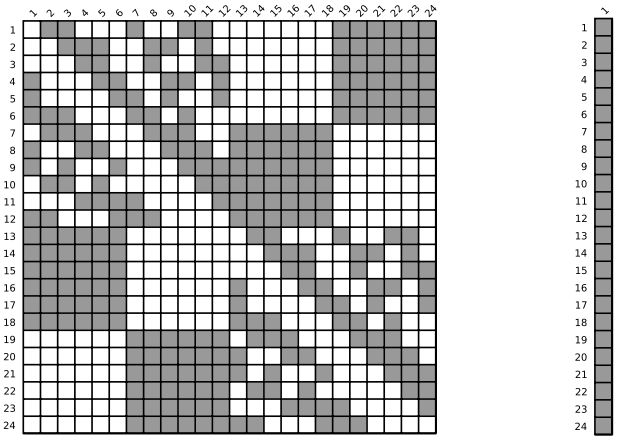


Fig. 5. A counterexample for the TEQ-conjecture and its minimal extending set.

an extending set of D. Hence, under our assumption the vector v_{18} describes the minimal extending set $\{11, 13, 25\}$ of D and its RelView-representation is shown as the last picture.

RelView required 16.87 s to compute the vector description of the minimal extending set of D via a call of the RelView-program `heuristicME`. A good portion of this running time was needed by the RelView-program `banks` to compute the vector description of the Banks set, viz. 10.96 s. For step (b), the construction of the chain $banks(D) = v_1 \supset v_2 \supset \cdots \supset v_{18}$, the tool required 5.75 s.

The larger the Banks set and the shorter the chain of step (b) of the heuristic of Section 5, the more vectors have to be tested when executing the main program `heuristicME`. An extreme example in this regard is the tournament $D : A \leftrightarrow A$ on the set $A = \{1, 2, \ldots, 23, 24\}$ the RelView-matrix of which is shown as the left one of the two pictures of Fig. 5. This tournament originates from [13] and is presently the smallest known counterexample for the former TEQ-conjecture of [25]. The universal vector to the right of the Boolean 24×24 matrix is the result computed by the RelView-program `approxME`. Hence, the chain of step (b) consists of the universal vector $\mathsf{L} : A \leftrightarrow \mathbf{1}$ only. A verification of $A = BA(D) = ME(D)$, i.e., *that the tournament D is not a counterexample for the former ME-conjecture although it is a counterexample for the former TEQ-conjecture*, by means of our heuristic requires to apply the RelView-program `isES` to all $2^{24} - 2 \approx 16.77 \cdot 10^6$ non-empty and proper subvectors of the universal vector $\mathsf{L} : A \leftrightarrow \mathbf{1}$ and to check whether each call yields $\mathsf{O} : \mathbf{1} \leftrightarrow \mathbf{1}$ as result.

Caused by the garbage collection mechanism that RelView presently uses, in case of a program execution with non-trivial ROBDDs and a huge number of iterations of loops it may happen that the execution fails because of lack of memory. For this reason we have not worked with the RelView-program `approxME` for showing $A = ME(D)$. Instead we have developed a variant of the RelView-program `smallerES` that tests, by means of calls of `isES`, whether there exists a subvector of $\mathsf{L} : A \leftrightarrow \mathbf{1}$ with a user-specified number k of 1-entries that describes an extending set of D. Applying the variant to all k from 1 to 23, after roughly 125 hours computing time we have been able to verify $A = ME(D)$. In the following table we present the running times of the variant for $10 \leqslant k \leqslant 20$. The second row shows the approximations of the numbers $\binom{24}{k}$ of subvectors (in thousands) that have to be tested for a fixed k, and in the third row the corresponding running times of the variant are given (in hours).

k	10	11	12	13	14	15	16	17	18	19	20
#subvectors	1961	2496	2704	2496	1961	1307	735.4	346.1	134.6	42.56	10.62
time	9.12	13.63	20.55	21.80	23.05	15.34	12.03	5.94	2.26	0.66	0.15

From the table we get that in the interval $10 \leqslant k \leqslant 20$ the average times for the test of a single subvector range from 0.016 s to 0.06 s. Notice that each such test requires $24 - k$ calls of the RelView-program `banks`. Altogether the verification of $A = ME(D)$ leads to approximately 198 million calls of `banks`, i.e., approximately 198 million solutions of an NP-hard problem via RelView.

For tournaments on at most 22 alternatives we also have compared the solution of Section 3 with the solution of Section 5. In case of small distances to linear strict-orders the second solution proved to be superior to the first one. This is due to the fact that for such tournaments usually the RelView-program `approxME` yields a vector with only a few 1-entries. To give an impression of the superiority we want to present results from a series of 50 randomly generated tournaments $D : A \leftrightarrow A$ with $|A| = 22$ and 10% as the expected percentage of pairs that have to be reversed to transform D into a linear strict-order: The minimum of the running times of the 50 calls of the RelView-program `heuristicME` was 0.03 s (with 1 1-entry in the result of `approxME`) and the maximum of the running times was 1.26 s (with 8 1-entries in the result of `approxME`). Compare these times with the 6000 s of Section 4. But the larger the distances become the more 1-entries are usually in the result of the RelView-program `approxME`, hence, the less the superiority is on average. For giving an impression only, we want to mention the results of two experiments with $|A| = 22$ and 30% as the expected percentage of

pairs that have to be reversed to get a linear strict-order. In the first case `approxME` computed a vector with 16 1-entries and this led to a total running time of 971.18 s for `heuristicME`. The number of 1-entries of the second experiment was 17. This additional 1-entry increased the total running time to 3221.27 s. If the expected percentage of pairs that have to be reversed goes to 50%, then usually the number of 1-entries in the result of `approxME` is very close to $|A|$ and, as a consequence, the solution of Section 3 is much more efficient than that of Section 5.

We conclude this section with some results concerning again the search of small counterexamples for the former ME-conjecture. As we have mentioned in Section 5, `point` is a deterministic RelView-operation. Each call of it and, as a consequence, also of the RelView-program `heuristicME` yields for the same argument the same result. In [22] it is shown how to generate random ROBDDs which implement permutation relations. With the help of the corresponding pre-defined RelView-operation `randomperm` it is easy to write a version `randompoint` of `point` that randomly selects a point from a non-empty vector. See again Appendix A. We have replaced in the RelView-programs `nextVect` and `smallerES` the calls of `point` by calls of `randompoint`. With the resulting code, that now non-deterministically computes a minimal extending set, we have searched for tournaments with more than one minimal extending set on larger sets of alternatives than the RelView-program `ME` allows. Despite of numerous experiments all our attempts were without success. For all tournaments each non-deterministic search returned the same vector. Due to this fact, the experiments we have mentioned in Section 4, and the results of experiments mentioned in [15] we believe that tournaments with more that one minimal extending set are extremely seldom and, hence, our second approach in all cases that appear in practice yields a correct result.

7. Conclusion

In the present paper we have developed an algorithmic relation-algebraic specification of minimal extending sets that could directly be translated into the programming language of the computer algebra system RelView. By experiments we have demonstrated that, despite of the very general and model-oriented approach, the resulting RelView-program is almost as efficient than the specifically developed program for the same task mentioned in [14]. This proves again the amazing gain of efficiency that can be reached by the use of ROBDDs for the implementation of relations, compared with other well-known implementations, like Boolean matrices, lists of pairs, and lists of successor or predecessor lists. We have also described an alternative heuristic approach. It assumes as pre-condition that there exists only one minimal extending set and allows, as we have shown, in certain situations to treat much larger instances successfully. Due to our numerous experiments and those of [15] we believe that the pre-condition holds in all practical applications.

Our model-oriented approach for getting solutions is very general and flexible. This allows to treat related problems with only small modifications. For instance, if, following a proposal of J. Duggan in [16] for defining Banks sets in case of weak tournaments, the set $B \in 2^A$ is defined as an extending set of the weak tournament $D : A \leftrightarrow A$ if there is no $a \in A \setminus B$ that is a maximal element of a maximal *transitive and complete* subset of $B \cup \{a\}$, where a set $X \in 2^A$ is complete w.r.t. D if for all $a, b \in X$ from $a \neq b$ it follows $D_{a,b}$ or $D_{b,a}$, then only part (10) of the entire relation-algebraic specification of Section 3 has to be modified to

$$t = \overline{vec(D)^{\mathsf{T}}; \left([\![\mathsf{M}, \mathsf{M}] \cap [\![\overline{D}, D]; \mathsf{M}\right)^{\mathsf{T}}} \cap \overline{\mathsf{L}; \left(\mathsf{M} \cap \mathsf{I} \cup \overline{D \cup D^{\mathsf{T}}}; \mathsf{M}\right)^{\mathsf{T}}} : 2^A \leftrightarrow \mathbf{1}, \tag{20}$$

since, as shown in [8], the second part of the intersection of the relation-algebraic specification (20) exactly describes the set of complete subsets of A as a subset of 2^A. The generality and flexibility of the approach and the concise form of the developed RelView-programs also has been an enormous help when solving the minimal extending set problem for the TEQ-counterexample of F. Brandt and H.G. Seedig.

A further advantage of our approach we want to emphasize is, that the developments of our algorithms are goal-oriented and formalized in such a way that errors are prevented and, in principle, an automatic verification of the results with regard to the logical starting points is possible. Presently we investigate the use of the automated theorem prover Prover9 (see [28]) in respect thereof.

Since the RelView-programs and -functions, which are obtained from the relation-algebraic specifications of the objects to be computed, are frequently formulated in a short and concise way and easy to alter in case of a slightly changed task, the tool is very suitable for prototyping specifications, the testing of hypotheses, experimentation with (known and new) concepts, exploration of ideas, generation of examples and counterexamples etc., while avoiding unnecessary overhead. Very positive in view of such applications are also the tool's possibilities for visualization and animation and for randomly generating relations. In the meantime systematic experiments are also an accepted means for doing research and obtaining new insights and results in formal sciences like mathematics and theoretical computer science. We believe that in this context tool support is indispensable and becomes increasingly important as one proceeds in investigations.

Acknowledgement

I want to express my gratitude to Gunther Schmidt for all I have learned from him since I became a member of his working group in 1979 and for his encouragement for the last 35 years. Furthermore, I want to thank Felix Brandt and Serge Gaspers for valuable hints during my investigations, Nikita Danilenko and Linda Haberland for reading a draft version of the paper, and the two unknown referees for their suggestions which helped to improve the paper.

Appendix A. Generation of random relations

RelView offers some possibilities to generate random relations, where in each case a probability model with a uniform distribution is used. We have already mentioned the pre-defined operation `randomperm`. It takes, for typing reasons only, a vector of type $X \leftrightarrow \mathbf{1}$ as input and yields a random permutation relation on X, i.e., a random relation $P : X \leftrightarrow X$ such that $P; P^{\mathsf{T}} = \mathsf{I}$ and $P^{\mathsf{T}}; P = \mathsf{I}$. A combination of `randomperm` and the pre-defined RelView-operation `point` for the deterministic selection of a point from a non-empty vector $v : X \leftrightarrow \mathbf{1}$ easily allows to implement a non-deterministic version `randompoint` of `point` that randomly selects the point from v as follows.

```
randompoint(v)
  DECL P, p
  BEG  P = randomperm(v);
       p = P^*point(P*v)
       RETURN p
  END.
```

With this RelView-program we have searched for tournaments with more than one minimal extending set as mentioned at the end of Section 6.

A further pre-defined RelView-operation for the generation of random relations is `randomcf`*dd*. Here *dd* stands for two digits between 00 and 99. The operation takes a relation of type $X \leftrightarrow X$ as input, again only to specify the type of the result, and yields, using matrix terminology, a Boolean matrix $S : X \leftrightarrow X$ such that for all entries below the main diagonal the probability for $S_{x,y}$ is 0.*dd* and all other entries of S are zeros. In terms of graph-theory, hence `randomcf`*dd* generates cycle-free directed random graphs.

In RelView a random tournament of type $A \leftrightarrow A$ can be obtained as follows. First, a random permutation relation $P : A \leftrightarrow A$ and a cycle-free random relation $S : A \leftrightarrow A$ are generated using the operations `randomperm` and `randomcf`*dd*. Using the pre-defined operations `trans` for transitive closures and `succ` for the successor relation on the carrier sets, next an upper-triangle relation $U : A \leftrightarrow A$ is computed, that is, a relation consisting of 0-entries below the main diagonal and of 1-entries otherwise. In the third step S and U are combined as $(S \cup U) \cap \overline{S}^{\mathsf{T}}$. This yields a random tournament such that precisely the pairs specified by the 1-entries below the main diagonal have to be reversed to get a linear strict-order. Finally, to be independent of the condition that precisely the pairs specified by the 1-entries below the main diagonal have to be reversed, the rows and columns of this tournament are permuted with the help of P. Here is a RelView-implementation of this procedure.

```
RandomTournament(v)
  DECL P, S, U
  BEG  P = randomperm(v);
       S = randomcf10(P);
       U = trans(succ(v))
       RETURN P^*((S | U) & -S^)*P
  END.
```

Due to the use of `randomcf10` in this RelView-program, the expected percentage of pairs of its result that have to be reversed to get a linear strict-order is 10%.

References

[1] H. Abold-Thalmann, R. Berghammer, G. Schmidt, Manipulation of concrete relations: The RelView-system, Bericht Nr. 8905, Fakultät für Informatik, Universität der Bundeswehr München, 1989.
[2] J. Banks, Sophisticated voting outcomes and agenda control, Soc. Choice Welf. 2 (1985) 295–306.
[3] R. Berghammer, G. Schmidt, H. Zierer, Symmetric quotients and domain constructions, Inf. Process. Lett. 33 (1989/90) 163–168.
[4] R. Berghammer, F. Neumann, RelView—An OBDD-based computer algebra system for relations, in: V.G. Gansha, E.W. Mayr, E. Vorozhtsov (Eds.), Computer Algebra in Scientific Computing, in: LNCS, vol. 3718, Springer, 2005, pp. 40–51.
[5] R. Berghammer, A. Rusinowska, H. de Swart, Applying relational algebra and RelView to coalition formation, Eur. J. Oper. Res. 175 (2007) 530–542.
[6] R. Berghammer, A. Rusinowska, H. de Swart, Applying relation algebra and RelView to measures in a social network, Eur. J. Oper. Res. 202 (2010) 182–198.
[7] R. Berghammer, A. Rusinowska, H. de Swart, Computing tournament solutions using relation algebra and RelView, Eur. J. Oper. Res. 226 (2013) 636–645.
[8] R. Berghammer, Computing and visualizing Banks sets of dominance relations using relation algebra and RelView, J. Log. Algebr. Program. 82 (2013) 123–136.
[9] S.J. Brams, P.C. Fishburn, Approval Voting, second ed., Springer, 2007.
[10] F. Brandt, Tournament solutions—Extensions of maximality and their applications to decision-making, Habilitationsschrift, Ludwig-Maximilians-Universität München, 2009.
[11] F. Brandt, Minimal stable sets in tournaments, J. Econ. Theory 146 (2011) 1481–1499.
[12] F. Brandt, M. Chudnovsky, I. Kim, G. Liu, S. Norin, A. Scott, P. Seymour, S. Thomasse, A counterexample to a conjecture of Schwartz, Soc. Choice Welf. 40 (2013) 739–743.

[13] F. Brandt, H.G. Seedig, A tournament of order 24 with two disjoint TEQ-retentive sets, arXiv:1302.5592 [cs.MA], 2013.
[14] F. Brandt, A. Dau, H.G. Seedig, Bounds of the disparity and separation of tournament solutions, Technische Universität München, 2013, Working paper, Version November 22, http://dss.in.tum.de/files/brandt-research/tsrel.pdf.
[15] F. Brandt, P. Harrenstein, H.G. Seedig, Minimal extending sets in tournaments, Technische Universität München, 2013, Working paper, Version October 16, http://dss.in.tum.de/files/brandt-research/me.pdf.
[16] J. Duggan, Uncovered sets, Wallis Institute of Political Economy, University of Rochester, 2011, Working Paper Nr. 63.
[17] M. Fey, Choosing from a large tournament, Soc. Choice Welf. 31 (2008) 301–309.
[18] S. Gaspers, M. Mnich, Feedback vertex sets in tournaments, J. Graph Theory 72 (2013) 72–89.
[19] O. Hudry, A survey on the complexity of tournament solutions, Math. Soc. Sci. 57 (2009) 292–303.
[20] J.-F. Laslier, Tournament Solutions and Majority Voting, Springer, 1997.
[21] B. Leoniuk, ROBDD-basierte Implementierung von Relationen und relationalen Operationen mit Anwendungen (ROBDD-based implementation of relational algebra with applications), Dissertation, Christian-Albrechts-Universität zu Kiel, 2001.
[22] U. Milanese, Zur Implementierung eines ROBDD-basierten Systems für die Manipulation und Visualisierung von Relationen (On the implementation of a ROBDD-based tool for the manipulation and visualization of relations), Dissertation, Christian-Albrechts-Universität zu Kiel, 2003.
[23] G. Schmidt, T. Ströhlein, Relations and Graphs, Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoretical Computer Science, Springer, 1993.
[24] G. Schmidt, Relational Mathematics, Encyclopedia of Mathematics and its Applications, vol. 132, Cambridge University Press, 2010.
[25] T. Schwartz, Cyclic tournaments and cooperative majority voting: A solution, Soc. Choice Welf. 7 (1990) 19–29.
[26] G. Woeginger, Banks winners in tournaments are hard to recognize, Soc. Choice Welf. 20 (2003) 523–528.
[27] RelView homepage: http://www.informatik.uni-kiel.de/~progsys/relview/.
[28] Prover9 homepage: http://www.prover9.org.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 120–134

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Spatial voting games, relation algebra and RelView

Rudolf Berghammer [a,*], Agnieszka Rusinowska [b], Harrie de Swart [c]

[a] Institut für Informatik, Christian-Albrechts-Universität Kiel, Olshausenstraße 40, 24098 Kiel, Germany
[b] Paris School of Economics - CNRS, Centre d'Economie de la Sorbonne, 106-112 Boulevard de l'Hopital, 75647 Paris Cedex 13, France
[c] Faculty of Philosophy, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

ARTICLE INFO

Article history:
Available online 12 February 2014

Keywords:
Relation algebra
RelView
Spatial voting game
Beating relation
Covering relation
Pareto set
Majority core
Uncovered set

ABSTRACT

We present a relation-algebraic approach to spatial voting games. We give relation-algebraic specifications of some important solution concepts of spatial voting games, such as the uncovered set, the majority core, the Pareto set, the win set, and the loss set. These specifications are relation-algebraic expressions and can be evaluated with the help of the BDD-based tool RelView after a simple translation into the tool's programming language. To give an impression of the tool's possibilities, we present some concrete applications.

1. Introduction

It was Gunther Schmidt who at the RelMiCS (Relational Methods in Computer Science) conference in Oisterwijk (2001) and at a COST TARSKI (Theory and Application of Relational Structures as Knowledge Instruments) workshop in Prague (2002) suggested to the authors of this paper that relation-algebraic methods and the software tool RelView may be used to solve problems from social choice theory and game theory. Somewhat surprised at first sight, we came together several times to see whether we could work out his suggestion. And, again somewhat to our surprise, his suggestion turned out to be very useful and has resulted in nearly ten papers in which we give logical specifications of solution concepts from social choice theory and game theory, translate them into relation-algebraic terms, which serve as input for the RelView software, which in its turn computes the results and makes them visible as Boolean matrices or directed graphs. The papers in question are, in order of appearance, Rusinowska et al. (2006, [31]), Berghammer et al. (2007, [5], and 2009, [6]), de Swart et al. (2009, [38]), Berghammer et al. (2010, [7]), Rusinowska et al. (2010, [32]), and Berghammer et al. (2011, [8], 2011, [9], and 2013, [10]). The [10] paper is meant as a tribute to Gunther Schmidt by giving another application of relation-algebraic methods, this time to compute solution concepts for spatial voting games.

Consider a situation where three parties have to decide on how to divide at most four million dollars over two issues, say education and health care. The alternatives are pairs $x = (x_1, x_2)$, where x_1 is the amount a party wants to spend for education and x_2 the amount the same party wants to spend for health care, under the restriction that $x_1 + x_2 \leqslant 4$. We restrict ourselves to finite sets of alternatives; for that reason we assume that x_1 and x_2 are natural numbers. Further we assume that each party i has an ideal point x_i^* and that party i prefers alternative x to alternative y iff x is closer to the ideal point x_i^* of i than y. By the distance between two alternatives we mean the Euclidean distance: party i (weakly) prefers x

* Corresponding author. Tel.: +49 (0)431 880 7272. Fax: +33 (0)431 880 7613
E-mail address: rub@informatik.uni-kiel.de (R. Berghammer).

http://dx.doi.org/10.1016/j.jlap.2014.02.003
2352-2208/

to y iff $\|x - x_i^*\| \leqslant \|y - x_i^*\|$, where $\|\cdot\|$ denotes the Euclidian norm. The social or common preference over the alternatives is determined by a voting rule, for instance, by majority rule, i.e., if more than half of the parties prefer alternative x to alternative y, then by definition society also prefers x to y. In order to ensure that there always is a majority, we assume that the number of parties is odd. A winning coalition is a set of parties that can enforce the social preference, i.e., if all parties in the coalition prefer x to y, then also society will prefer x to y. So, if the voting rule is majority rule and there are three parties, a winning coalition will consist of two or three parties.

More generally, a spatial voting game consists of a set of players (or parties, voters, agents, individuals, etc.), a set of winning coalitions (determined by a given voting rule), and a multidimensional (in practice, a two-dimensional) policy space, in which each player has an ideal point, which induces for each player a preference relation over the alternatives in the terms of the Euclidean distance.

There are several well-known solution concepts in the literature on spatial voting games and collective choice mechanisms, among which are the *Pareto set*, the *Majority core* and the *uncovered set*. In this paper we present logical characterizations of these solution concepts, translate them step-by-step into relation-algebraic expressions and use these expressions in extremely concise and short RelView programs to compute the solutions in particular cases. Because computing solutions of spatial voting games by hand is unfeasible even for relatively simple examples, it is important to develop software for these purposes. Because the RelView tool has been designed to deal with relations in a very efficient way, and because much of the theory of spatial voting games is in terms of relations, it is a natural step to apply RelView in order to compute solutions of spatial voting games. However, in order to be able to apply RelView the policy space has to be a finite discrete set, while in much of the literature the policy space consists of a convex subset of the Euclidian space $\mathbb{R}^2$.

A flexible tool for the geometric analysis of two-dimensional spatial voting games is provided, for example, by the *CyberSenate* software. It was developed by Joseph Godfrey of the WinSet Group, LLC (see http://www.winset.com/), and is used e.g. to compute several solutions of spatial voting games. In this software, configurations of ideal points can be created and modified by point and click methods (either generated by Monte Carlo routines or derived from empirical data). Then, median lines, Pareto sets, win sets, loss sets, uncovered set approximations, and other constructions can be quickly generated.

Until recently, not that much has been known concerning the location, size, and shape of the uncovered set, although important characteristics of spatial games depend on the location, size and the shape of this solution concept. Hartley and Kilgour (1987, [17]) establish the precise boundaries of the uncovered set for three voters and the two-dimensional Euclidean space.

A significant contribution to the spatial voting games' literature is the work by Bianco et al. (2004, [11]), where the authors present a simulation technique for estimating the uncovered set and for testing its empirical relevance in real world settings. They employ a grid-search computational algorithm for estimating the size, shape, and location of the uncovered set in abstract social choice situations, for any profile of Euclidean preferences on a two-dimensional space. They also apply it to the contemporary U.S. House. In particular, it is shown that the size, shape, and location of the uncovered set are very sensitive to the distribution of ideal points, and that the uncovered set has considerable explanatory power. The uncovered set is also not necessarily 'centrally' located. Their technique for estimating the uncovered set treats the policy space as a collection of discrete outcomes rather than as a continuous space, and hence it recovers only an approximation of the actual uncovered set. However, the authors show that their grid procedure estimate of the uncovered set converges to the interior of the uncovered set. If the uncovered set has a non-empty interior, or is a union of sets, each of which has a non-empty interior, then the uncovered set delineated by their grid procedure converges to the true uncovered set. The authors used data mainly from two-dimensional NOMINATE scores (Poole and Rosenthal, 2000, [30]), and some additional data from Gloseclose and Snyder (2000, [15]), and Jackman (2001, [18]).

Miller (2007, [26]) explores theoretical considerations which account for important features of the Bianco et al. (2004, [11]) results on the uncovered set, and presents theoretical insights of more general relevance to spatial voting theory. As other authors, he also focuses on two-dimensional majority-rule spatial voting games, and uses the *CyberSenate* software.

Works on the solutions of spatial voting game are usually restricted to two-dimensional spatial voting games. As stated in Bianco et al. (2004, [11]), '…Analyzing enactability using a two-dimensional model is a significant advance over many contemporary analyses, and we know of no well-cited analysis of the legislative process which utilizes a spatial model with more than two dimensions'. Empirical research indicates, furthermore, that two dimensions are often sufficient to represent some choice situations like legislator preferences in parliamentary context (see Poole and Rosenthal, 2000, [30]; Poole, 2005, [29]), or voter preferences over competing candidates or parties (see Budge et al., 1987, [12]).

2. Relation-algebraic preliminaries

In this section we recall the basics of (heterogeneous) relation algebra that are needed in the remainder of the paper. For more details, see Schmidt and Ströhlein (1993, [33]) and Schmidt (2010, [34]), for example.

Given two sets X and Y, we write $R : X \leftrightarrow Y$ if R is a (binary and typed) relation with source X and target Y, i.e., a subset of the Cartesian product $X \times Y$. If the sets of R's *type* $X \leftrightarrow Y$ are finite, then we may consider R as a Boolean matrix. Since such an interpretation is well suited for many purposes and also used by the RelView system as the main possibility to visualize relations, in this paper we frequently use matrix terminology and notation. In particular, we speak about the entries, rows and columns of a relation/matrix and write $R_{x,y}$ instead of $(x, y) \in R$ or xRy. We assume the

reader to be familiar with the basic operations on relations, viz. R^T (*transposition*), $\overline{R}$ (*complement*), $R \cup S$ (*union*), $R \cap S$ (*intersection*) and $R;S$ (*composition*), the predicates (tests) $R \subseteq S$ (*inclusion*) and $R = S$ (*equality*), and the special relations O (*empty relation*), L (*universal relation*) and I (*identity relation*). In case of O, L and I we overload the symbols, i.e., avoid the binding of types to them.

For $R : X \leftrightarrow Y$ and $S : X \leftrightarrow Z$, by $syq(R,S) = \overline{R^\mathsf{T}\,\overline{S}} \cap \overline{\overline{R^\mathsf{T}}\,S}$ their *symmetric quo tient* $syq(R,S) : Y \leftrightarrow Z$ is defined. In the present paper we only use its point-wise description, saying that for all $y \in Y$ and $z \in Z$ it holds that $syq(R,S)_{y,z}$ iff for all $x \in X$ the relationships $R_{x,y}$ and $S_{x,z}$ are equivalent.

In relation algebra *vectors* are a well-known means to model subsets of a given set X. Vectors are relations $r : X \leftrightarrow \mathbf{1}$ (we prefer in this context lower case letters) with a specific singleton set $\mathbf{1} = \{\perp\}$ as target. They can be considered as Boolean column vectors. To be consonant with the usual notation, we omit always the second subscript, i.e., write r_x instead of $r_{x,\perp}$. Then r *describes* the subset Y of X if for all $x \in X$ it holds r_x iff $x \in Y$. A point $p : X \leftrightarrow \mathbf{1}$ is a vector with precisely one 1-entry. Consequently, it describes a singleton subset $\{x\}$ of X and we then say that it describes the element x of X. If $r : X \leftrightarrow \mathbf{1}$ is a vector and Y the subset of X it describes, then $inj(r) : Y \leftrightarrow X$ denotes the *embedding relation* of Y into X. In Boolean matrix terminology this means that $inj(r)$ is obtained from $\mathsf{I} : X \leftrightarrow X$ by deleting all rows which do not correspond to an element of Y and point-wisely this means that for all $y \in Y$ and $x \in X$ it holds $inj(r)_{y,x}$ iff $y = x$.

In conjunction with powersets 2^N we use *membership relations* $\mathsf{M} : N \leftrightarrow 2^N$ and also *size comparison relations* $\mathsf{S} : 2^N \leftrightarrow 2^N$. Point-wisely they are defined for all $i \in N$ and $Y, Z \in 2^N$ as follows:

$$\mathsf{M}_{i,Y} \iff i \in Y \qquad \mathsf{S}_{Y,Z} \iff |Y| \leqslant |Z|$$

A combination of M with embedding relations allows a *column-wise enumeration* of an arbitrary subset $\mathfrak{S}$ of 2^N. Namely, if the vector $r : 2^N \leftrightarrow \mathbf{1}$ describes $\mathfrak{S}$ in the sense defined above and we define $S = \mathsf{M}; inj(r)^\mathsf{T}$, then we get $N \leftrightarrow \mathfrak{S}$ as type of S and that for all $i \in N$ and $Y \in \mathfrak{S}$ it holds $S_{i,Y}$ iff $i \in Y$. In the Boolean matrix model of relations this means that the sets of $\mathfrak{S}$ are precisely described by the columns of S, if the columns are considered as vectors of type $N \leftrightarrow \mathbf{1}$.

To model direct products $X \times Y$ of sets X and Y relation-algebraically, the *projection relations* $\pi : X \times Y \leftrightarrow X$ and $\rho : X \times Y \leftrightarrow Y$ are the convenient means. They are the relational variants of the well-known projection functions and, hence, fulfill for all pairs $u \in X \times Y$ and elements $x \in X$ and $y \in Y$ the following equivalences:

$$\pi_{u,x} \iff u_1 = x \qquad \rho_{u,y} \iff u_2 = y$$

Here u_1 denotes the first component of u and u_2 the second component. As a general assumption, in the remainder of the paper we always assume a pair u to be of the form $u = (u_1, u_2)$. Then $\hat{u}$ denotes the *transposed pair* (u_2, u_1). The projection relations enable us to specify the well-known pairing operation of functional programming relation-algebraically. The *pairing* of $R : Z \leftrightarrow X$ and $S : Z \leftrightarrow Y$ is defined as $[R,S] = R;\pi^\mathsf{T} \cap S;\rho^\mathsf{T} : Z \leftrightarrow X \times Y$, where π and ρ are as above. Point-wisely this definition says that

$$[R,S]_{z,u} \iff R_{z,u_1} \wedge S_{z,u_2}$$

for all $z \in Z$ and $u \in X \times Y$. Based on π and ρ, we are also able to establish a bijective correspondence between the relations of type $X \leftrightarrow Y$ and the vectors of type $X \times Y \leftrightarrow \mathbf{1}$. The transformation of a relation $R : X \leftrightarrow Y$ into its *corresponding vector* $vec(R) : X \times Y \leftrightarrow \mathbf{1}$ is given by $vec(R) = (\pi; R \cap \rho); \mathsf{L}$ and the step back from the vector $r : X \times Y \leftrightarrow \mathbf{1}$ to its *corresponding relation* $rel(r) : X \leftrightarrow Y$ by $rel(r) = \pi^\mathsf{T}; (\rho \cap r; \mathsf{L})$. Point-wisely this means that for all points $u \in X \times Y$ the following two properties hold:

$$vec(R)_u \iff R_{u_1,u_2} \qquad rel(r)_{u_1,u_2} \iff r_u$$

Relational algebra has a fixed and surprisingly small set of constants and operations which (in the case of finite carrier sets) can be implemented very efficiently. At Kiel University a computer system was developed for the visualization and manipulation of relations and for relational prototyping and programming, called RelView. The tool is written in the C programming language, uses reduced ordered binary decision diagrams for implementing relations, and makes full use of the X-windows graphical user interface. Details and applications can be found, for instance, in Behnke et al. (1999, [1]), Berghammer et al. (2002, [2], 2003, [3], and 2003, [4]).

The main purpose of RelView is the evaluation of relation-algebraic expressions. These are constructed from the relations of its workspace using pre-defined operations and tests, user-defined functions, and user-defined programs. A relational program is much like a function procedure in the programming languages Pascal or Modula 2, except that it only uses relations as data type. It starts with a head line containing the program name and the formal parameters. Then the declaration of the local relational domains, functions, and variables follows. Domain declarations can be used to introduce projection relations and pairings of relations in the case of Cartesian products, and injection relations and sums of relations in the case of disjoint unions, respectively. The third part of a program is the body, a while-program over relations. As a program computes a value, finally, its last part consists of a return-clause, which is a relation-algebraic expression whose value after the execution of the body is the result.

3. Spatial voting games and their solutions

A *simple game* is a pair $(N, \mathcal{W})$, where $N = \{1, 2, \ldots, n\}$ is a non-empty and finite set of players (parties, voters, agents, individuals, etc.) and $\mathcal{W}$ is a subset of the powerset 2^N of N. For instance, $\mathcal{W}$ may be the set of all subsets of N with cardinality larger than $\frac{n}{2}$ (in case the voting rule is majority rule); or with cardinality larger than $\frac{2}{3}n$ (if the used voting rule requires that more than two-thirds of the players have to vote with 'yes'). Any element of 2^N is called a *coalition*. A coalition $S \in 2^N$ with $S \in \mathcal{W}$ is called *winning*, while a coalition $S \in 2^N$ with $S \notin \mathcal{W}$ is called *losing*. The set $\mathcal{W}$ is assumed to satisfy the following properties: $\mathcal{W} \neq \emptyset$ and $\emptyset \notin \mathcal{W}$ (non-triviality) and that for all $S, T \in 2^N$ from $S \subseteq T$ and $S \in \mathcal{W}$ it follows $T \in \mathcal{W}$ (monotonicity). Then a *spatial voting game*, denoted by $(N, \mathcal{W}, X)$, is a simple game $(N, \mathcal{W})$ together with a finite and non-empty set X of points in an m-dimensional space $\mathbb{N}^m$, which is the set of policy positions that might be taken by a player. It is assumed that each player $i \in N$ has an *ideal point* $x_i^* \in X$ which represents that player's most preferred position in the policy space and which induces a (weak) preference relation $R^i : X \leftrightarrow X$ for player i in terms of the Euclidean distance from the ideal point:

$$R^i_{x,y} \quad \Longleftrightarrow \quad \|x - x_i^*\| \leqslant \|y - x_i^*\|$$

for each $x, y \in X$. Given a player $i \in N$, we define for each point $x \in X$ the subset $R^i(x) = \{y \in X \mid R^i_{y,x}\}$ of X and denote by P^i and I^i the strict preference relation, respectively the indifference relation of player i:

$$P^i_{x,y} \quad \Longleftrightarrow \quad \|x - x_i^*\| < \|y - x_i^*\| \qquad I^i_{x,y} \quad \Longleftrightarrow \quad \|x - x_i^*\| = \|y - x_i^*\|$$

All relations P^i are (strict) weak-order relations (i.e., complete and transitive) and specify the strict preferences of the players and all relations I^i are equivalence relations and specify the situations where the players are indifferent. From the definitions we get that for all players $i \in N$ the equations

$$I^i = \overline{P^i} \cap \overline{P^i}^{\mathsf{T}} \qquad R^i = P^i \cup I^i \qquad P^i = R^i \cap \overline{R^i}^{\mathsf{T}}$$

hold, such that it suffices to concentrate on one of those three types of relations. We work mainly with the *individual strict preference relations* P^i.

In the following we present a simple example for a spatial voting game with pairs as policy positions. It is considered throughout this paper.

Example 1. We consider again the situation mentioned in the introduction, where three parties have to decide on how to divide four million dollars over two issues, say education and health care, and social or common preference over the alternatives is determined by majority. Hence the set of winning coalitions consists in this example of all coalitions with at least two members. Furthermore, we assume that only an integer number of millions can be given to an issue, and that parties may also prefer not to spend all the money, but to save some money for later. The first party intends to give three millions to education and one million to health care, the second party intends to give one million to education and one million to health care and to save two millions for later, and the third party intends to give one million to education and three millions to health care.

If we model this situation by a spatial voting game $(N, \mathcal{W}, X)$, then we obtain the underlying simple game $(N, \mathcal{W})$ with three players and four winning coalitions as follows:

$$N = \{1, 2, 3\} \qquad \mathcal{W} = \big\{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\big\}$$

The task to divide the four million dollars over two issues is modeled by a two-dimensional policy space X which contains the points (i, j) such that $i \in \mathbb{N}$, $j \in \mathbb{N}$ and $i + j \leqslant 4$. Hence, we have:

$$X = \big\{(0,0), (1,0), (2,0), (3,0), (4,0), (0,1), (1,1), (2,1), (3,1), (0,2), (1,2), (2,2), (0,3), (1,3), (0,4)\big\}$$

Finally, the intentions of the three parties how to divide the money are modeled by the following three ideal points:

$$x_1^* = (3, 1) \qquad x_2^* = (1, 1) \qquad x_3^* = (1, 3)$$

To visualize this situation, in Fig. 1 we have drawn the points of X as an empty RelView-graph, where each of the 15 nodes corresponds to a point of X. The correspondences are given by the node's positions in the drawing as well as their attached labels. The three ideal points are emphasized by three specific drawings of the corresponding nodes. The node of x_1^* is drawn as a black square, that of x_2^* as a black circle, and that of x_3^* as a white square. By computing Euclidean distances, we get from these points at once the three individual strict preference relations P^1, P^2 and P^3 on X. How they are represented by the RelView tool as labeled 15×15 Boolean matrices is shown in the three pictures of Fig. 2. The row and column labels indicate that the rows and columns of the matrices correspond to the node numbers of the graph of Fig. 1. In such a matrix a black square means a 1-entry and a white square means a 0-entry. The picture on the left represents P^1, that in the middle P^2, and that on the right P^3.

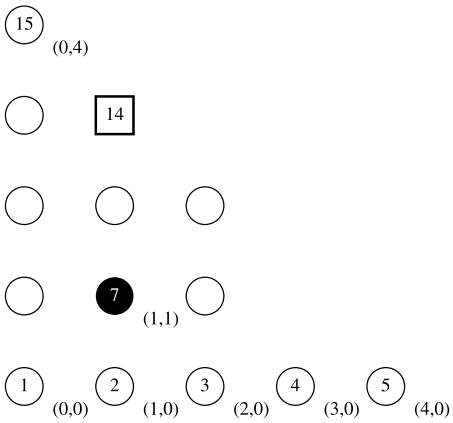


Fig. 1. Visualization of the policy positions.

The individual strict preference relations constitute weak-order relations. Informally this means that for each such relation P^i its Hasse-diagram is composed by a series of complete bipartite strict-order relations, one above another, and the different levels form the equivalence classes of the corresponding indifference relation I^i. We illustrate this by the RelView-picture of Fig. 3. It depicts the Hasse-diagram of the individual strict preference relation P^2 as a directed graph. This graph is composed by five complete bipartite directed graphs, where each of them depicts a complete bipartite strict-order relation. The node numbers of the directed graph correspond again to the node numbers of the graph of Fig. 1 and, hence, to the row and column numbers of the above RelView-matrix for P^2. The picture of Fig. 3 shows, e.g., that the ideal point $(1,1)$ of player 2 is the greatest element and forms the first level, the second level consists of the points $(0,1)$, $(1,2)$, $(1,0)$ and $(2,1)$, the third level consists of the points $(0,2)$, $(0,0)$, $(2,0)$ and $(2,2)$, the first and second level form (together with the arrows in between) a complete bipartite strict-order relation, and the same holds for the second and the third level (and the corresponding arrows). From the matrices of the relations P^1, P^2 and P^3 the levels and their arrangements may be obtained by the number of 1-entries in a row. For instance, in the picture of P^2 there is only one 0-entry in the row labeled with $(1,1)$.

After these preparations we are in a position to define some important derived relations, i.e., the beating and the covering relation, and solutions (sets of points) for majority as voting rule. In what follows it is always assumed that $(N, \mathcal{W}, X)$ is a spatial voting game with an odd number of players (such that there always is a majority preferring one alternative to another) and a policy space X with ideal points x_i^*, one for each player i, and that P^i is the individual strict preference relation of player $i \in N$. We start with the introduction of the *Pareto* set $PO(A)$ of a subset A of X. It is based on the individual strict preference relations $P^i : X \leftrightarrow X$.

Definition 1. For a subset A of X its *Pareto set* $PO(A)$ is defined as

$$PO(A) := \left\{x \in A \mid \neg\exists y : y \in A \wedge \forall i : P^i_{y,x}\right\}.$$

In the definition of the set $PO(A)$ the variable i ranges over the set N of players and the variable y ranges over the set X of points. Usually, in logical formulae the ranges of variables in quantifications are expressed via set-memberships. When translating formulae into relation-algebraic expressions some of such relationships are important for reaching the desired result, like $y \in A$ in Definition 1, some are however not, like that i is in N in the same definition. In the formulae of this paper we explicitely state only the important membership-relationships which are transformed into relation-algebraic constructions. Those which are not used within a calculation are mentioned in the surrounding text.

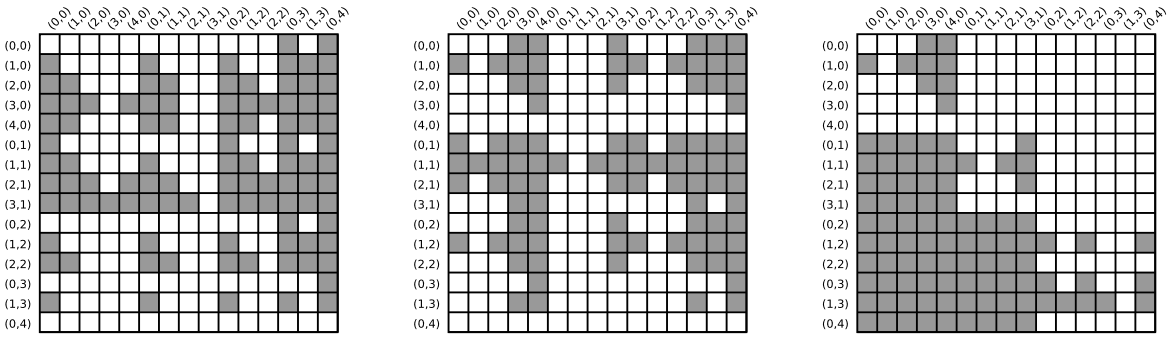


Fig. 2. Three individual strict preference relations.

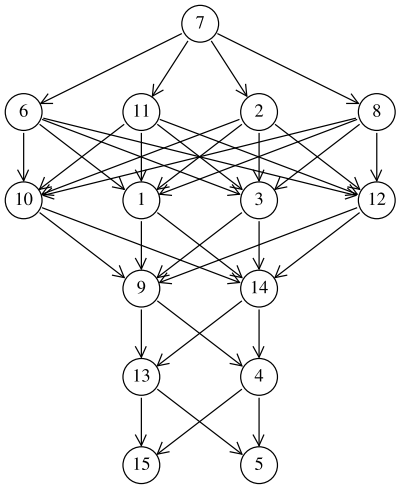


Fig. 3. The Hasse-diagram of an individual strict preference relation.

We say that a point $y \in X$ is unanimously preferred to a point $x \in X$ if y is closer than x to all ideal points. Using this terminology, the Pareto set of A is the set of all points $x \in A$ such that there is no point in A which is unanimously preferred to x. Next, we introduce the beating relation, the win set, and the loss set.

Definition 2. The *beating relation* $B : X \leftrightarrow X$ is defined by

$$B_{x,y} \quad \Longleftrightarrow \quad \left|\left\{i \in N \mid P^i_{x,y}\right\}\right| > \left|\left\{i \in N \mid P^i_{y,x}\right\}\right|$$

for all $x, y \in X$. Furthermore, the *win set* $W(x)$ and the *loss set* $W^{-1}(x)$ of the point $x \in X$ are defined by

$$W(x) := \{y \in X \mid B_{y,x}\} \quad \text{and} \quad W^{-1}(x) := \{y \in X \mid B_{x,y}\}$$

In case that $B_{x,y}$, we say that the point $x \in X$ beats (or dominates) the point $y \in X$ (by majority rule). Hence, x beats y if some majority of voters prefers x to y, i.e., if x is closer than y to more than half of the ideal points. Furthermore, the win set of x is the set of all points in X that beat x and the loss set of x is the set of all points in X that are beaten by x. The relation B is asymmetric.

Notice that majority rule (and hence the set $\mathcal{W}$ of winning coalitions) is implicitly used in the definition of the beating relation B, and hence further on also in the definitions of the majority core $MC(A)$, the covering relation C and the uncovered set $UC(A)$. One might define another beating relation $B' : X \leftrightarrow X$ as follows: For all $x, y \in X$ it holds $B'_{x,y}$ iff $\frac{2}{3}$ of the agents prefer x to y. Then the majority core, the covering relation, and the uncovered set relation would be different too.

Example 2. From the three individual strict preference relations P^1, P^2 and P^3 of Example 1 we get, for instance, for the beating relation B the following three relationships:

$$B_{(2,1),(2,2)} \qquad B_{(2,2),(1,1)} \qquad B_{(1,1),(2,1)}$$

The first relationship holds because the parties 1 and 2 prefer the point $(2,1)$ to the point $(2,2)$, the second one holds because the parties 1 and 3 prefer the point $(2,2)$ to the point $(1,1)$, and the third one holds because the parties 2 and 3 prefer the point $(1,1)$ to the point $(2,1)$. In other words, the relationships say that $(2,1) \in W((2,2))$, $(2,2) \in W(x^*_2)$ and $x^*_2 \in W((2,1))$.

Obviously, the x-column of the beating relation $B : X \leftrightarrow X$ describes the win set $W(x)$ of the point $x \in X$ in the sense of Section 2, and in the same way the loss set $W^{-1}(x)$ of x is described by the transpose of the x-row of B. In the next sections we show how to compute the beating relation from the individual strict preference relations using relation-algebra and RelView. Now, we introduce the majority core.

Definition 3. The *majority core* (or the set of *Condorcet winners*) $MC(A)$ of a subset A of X is defined as

$$MC(A) := \{x \in A \mid \forall y\colon\ y \in A \wedge y \neq x \rightarrow B_{x,y}\}$$

In words, the majority core of a set A is the set of all points in A that beat all other points in A.

The covering relation we introduce next is a subrelation of the beating relation. It allows to define the uncovered set as another solution concept. For more details on the theory of uncovered sets, see Miller (1980, [25]), McKelvey (1986, [21]), Shepsle and Weingast (1984, [36]) or Cox (1987, [14]).

Definition 4. The *covering relation* $C : X \leftrightarrow X$ is defined by

$$C_{y,x} \quad \Longleftrightarrow \quad B_{y,x} \wedge \forall z : B_{z,y} \rightarrow B_{z,x}$$

for all $x, y \in X$. For a subset A of X its *uncovered set* $UC(A)$ is defined as

$$UC(A) := \{x \in A \mid \neg \exists y\colon\ y \in A \wedge x \neq y \wedge C_{y,x}\}$$

Note that from the type $X \leftrightarrow X$ of the beating relation B it follows that the variable z in the formula defining the covering relation C ranges over the set X of points. We say that the point $x \in X$ is covered by the point $y \in X$ iff $C_{y,x}$. Hence, x is covered by y if y beats x and any point which beats y also beats x. The uncovered set of A consists of the points in A which are not covered by any other point in A. It is known that the uncovered set is a subset of the Pareto set (Miller, 1980, [25]; Shepsle and Weingast, 1984, [36]). Bianco et al. (2004, [11], p. 271) state the following known properties of the uncovered set in case $X \subseteq \mathbb{R}^m$ is a convex policy space:

1. The uncovered set is never empty (McKelvey, 1986, [21]).
2. If the majority core is not empty, then the uncovered set coincides with the majority core (Miller, 1980, [25]; McKelvey, 1986, [21]).
3. The uncovered set characterizes the set of feasible or enactable outcomes in many legislative and other majority rule decision-making decisions (Miller, 1980, [25]; Shepsle and Weingast, 1984, 1994, [36,37]; McKelvey, 1986, [21]; Ordeshook and Schwartz, 1987, [28]).
4. If B is any subset of A, then B is the uncovered set of A iff every point outside of B is covered by a point within B, and no point within B is covered by a point inside B. (Necessity proved in McKelvey, 1986, [21], sufficiency in Bianco et al., 2004, [11].)

Although it has been suggested that stable equilibria in multidimensional majority rule games are rare (see, for instance, McKelvey, 1976, 1979, [19,20]; Schofield, 1978, [35]; McKelvey and Schofield, 1987, [22]), if voters consider the consequences of their behavior, outcomes lie in the uncovered set (Miller, 1980, [25]; Shepsle and Weingast, 1984, [36]; McKelvey, 1986, [21]; Miller et al., 1989, [27]). In the uncovered sets, undominated policy points with respect to a cover relation are predicted. Uncovered set outcomes are not necessarily Condorcet winners, but if a Condorcet winner exists, then the uncovered set consists of that single outcome. Several works suggest that the uncovered set describes the set of enactable outcomes in many majority-rule decision making situations; see, e.g. Cox (1987, [14]), Shepsle and Weingast (1984, [36]), Calvert (1985, [13]), McKelvey (1986, [21]), Grofman et al. (1987, [16]).

4. Relation-algebraic description of solution concepts

In this section we translate the solution concepts of spatial voting games we have introduced in Section 3 into relation-algebraic expressions. We formulate the corresponding results in the prevalent mathematical theorem-proof-style to emphasize them and to enhance readability. But, in fact, we have obtained the relation-algebraic expressions by developing them formally from the original logical specifications. We regard this goal-oriented development of algorithms from formal logical specifications, that are correct by construction, as the first main advantage of our approach. As its second main advantage we regard its computer-support by means of RelView. All our relation-algebraic expressions immediately can be translated into short and concise RelView programs and, as a consequence, evaluated by the tool. Combining this with RelView's possibilities for visualization and animation also allows to experiment with the concepts in question while avoiding unnecessary overhead.

In what follows we assume in each case a spatial voting game $(N, \mathcal{W}, X)$ with an odd number of players $1, 2, \ldots, n$, a convex policy space X, and given individual strict preference relations $P^1, \ldots, P^n$. First, we give a relation-algebraic specification of the Pareto set $PO(A)$ of a subset A of the policy space X in the form of a vector.

Theorem 1. *Let $v^A : X \leftrightarrow \mathbf{1}$ be the vector that describes the subset A of X and let the relation $Q : X \leftrightarrow X$ be defined as $Q = P^1 \cap \cdots \cap P^n$. Then the vector*

$$po\big(Q\,, v^A\big) = v^A \cap \overline{Q^{\mathsf{T}}; v^A} : X \leftrightarrow \mathbf{1}$$

describes the Pareto set $PO(A)$ of A as a subset of X.

Proof. Using the definition of the auxiliary relation Q in the second step, we can calculate as given below (where the variable i ranges over the set N and the variable y ranges over the set X):

$$\begin{aligned}
x \in PO(A) &\iff x \in A \wedge \neg\exists y : y \in A \wedge \forall i : P^i_{y,x} \\
&\iff x \in A \wedge \neg\exists y : y \in A \wedge Q_{y,x} \\
&\iff v^A_x \wedge \neg\exists y : v^A_y \wedge Q_{y,x} \\
&\iff v^A_x \wedge \neg\exists y : Q^{\mathsf{T}}_{x,y} \wedge v^A_y \\
&\iff v^A_x \wedge \neg\big(Q^{\mathsf{T}}; v^A\big)_x \\
&\iff v^A_x \wedge \overline{Q^{\mathsf{T}}; v^A}_x \\
&\iff \big(v^A \cap \overline{Q^{\mathsf{T}}; v^A}\big)_x \\
&\iff po\big(Q, v^A\big)_x
\end{aligned}$$

Now the definition of the equality of relations shows the claim. □

For a translation of the remaining solution concepts into relation-algebraic expressions it is advantageous to combine in a preparatory step the individual strict preference relations $P^1, \ldots, P^n$ into one relation P, which is introduced now.

Definition 5. We say that the relation $P : N \leftrightarrow X^2$ *models* the spatial voting game $(N, \mathcal{W}, X)$ if $P_{i,u}$ is equivalent to $P^i_{u_1,u_2}$, for all $i \in N$ and $u \in X^2$.

If $i \in N$ is a player, then a little reflection shows that the transpose of the i-row of the relation $P : N \leftrightarrow X^2$ is a vector $v^i : X^2 \leftrightarrow \mathbf{1}$ such that for all pairs $u \in X^2$ it holds v^i_u iff $P_{i,u}$, that is, iff $P^i_{u_1.u_2}$. Hence, the transpose of the i-row of P is the corresponding vector $vec(P^i)$ of the strict preference relation P^i in the sense of Section 2. As a further consequence, the relation $P : N \leftrightarrow X^2$ modeling the spatial voting game $(N, \mathcal{W}, X)$ can be obtained from the individual strict preference relations $P^1, \ldots, P^n$ as follows: First, for each player $i \in N$ the relation P^i is transformed into the relation $vec(P^i)^{\mathsf{T}} : \mathbf{1} \leftrightarrow X^2$, i.e., into the transpose of the corresponding vector, and then P is formed by combining the transposed corresponding vectors row by row into a relation of type $N \leftrightarrow X^2$. The latter means that we have to form the relation-algebraic sum, that is, get P as

$$P = vec\big(P^1\big)^{\mathsf{T}} + \cdots + vec\big(P^n\big)^{\mathsf{T}} : N \leftrightarrow X^2$$

We do not go into details with regard to sums of relations and refer to Schmidt (2010, [34]), where a relation-algebraic specification via injection relations is given.

In order to specify the majority core, the win sets, and the loss sets relation-algebraically, we first have to do this with the beating relation $B : X \leftrightarrow X$. In what follows, we assume the projection relations $\pi, \rho : X^2 \leftrightarrow X$ of the direct product X^2 to be at hand as well as the membership relation $\mathsf{M} : N \leftrightarrow 2^N$ and the size comparison relation $\mathsf{S} : 2^N \leftrightarrow 2^N$. Each of these relations is available in RelView via a pre-defined operation and their BDD-implementations are rather small. See Leoniuk (2001, [23]) and Milanese (2003, [24]) for details. We start with the beating relation.

Theorem 2. *Suppose that the relation $P : N \leftrightarrow X^2$ models the spatial voting game $(N, \mathcal{W}, X)$. If we specify relations E and F by*

$$E = syq(P, \mathsf{M}) : X^2 \leftrightarrow 2^N \qquad F = syq\big(P; [\![\rho, \pi]\!], \mathsf{M}\big) : X^2 \leftrightarrow 2^N$$

then we get for the beating relation $B : X \leftrightarrow X$ the relation-algebraic specification (where $\mathsf{L} : 2^N \leftrightarrow \mathbf{1}$)

$$BeatRel(P) = rel\big(\big(E \cap F; \big(\mathsf{S} \cap \overline{\mathsf{S}^{\mathsf{T}}}\big)\big); \mathsf{L}\big) : X \leftrightarrow X$$

Proof. For an arbitrary pair $u \in X^2$ we prove in a preparatory step for all $Y \in 2^N$ the following equivalence (where the variable i ranges over the set N):

$$E_{u,Y} \iff syq(P, \mathsf{M})_{u,Y} \iff \forall i : P_{i,u} \leftrightarrow \mathsf{M}_{i,Y} \iff \forall i : P_{i,u} \leftrightarrow i \in Y \iff \{i \in N \mid P_{i,u}\} = Y$$

Using that the exchange relation $[\![\rho, \pi]\!] : X^2 \leftrightarrow X^2$ relates the given pair $u = (u_1, u_2)$ precisely with its transposition $\hat{u} = (u_2, u_1)$ because of

$$[\![\rho, \pi]\!]_{u,w} \iff \rho_{u,w_1} \wedge \pi_{u,w_2} \iff u_2 = w_1 \wedge u_1 = w_2 \iff w = \hat{u}$$

for all $w \in X^2$, in a rather similar way we can prove that for all $Z \in 2^N$ the following property holds:

$$F_{u,Z} \iff \{i \in N \mid P_{i,\hat{u}}\} = Z$$

By means of these two auxiliary results, we now can calculate as given below (where the two variables Y and Z range over the powerset 2^X):

$$\begin{aligned}
B_{u_1,u_2} &\iff \big|\{i \in N \mid P_{i,u}\}\big| > \big|\{i \in N \mid P_{i,\hat{u}}\}\big| \\
&\iff \exists Y, Z : \{i \in N \mid P_{i,u}\} = Y \wedge \{i \in N \mid P_{i,\hat{u}}\} = Z \wedge |Z| < |Y| \\
&\iff \exists Y : E_{u,Y} \wedge \exists Z : F_{u,Z} \wedge |Z| \leqslant |Y| \wedge \neg\big(|Y| \leqslant |Z|\big) \\
&\iff \exists Y : E_{u,Y} \wedge \exists Z : F_{u,Z} \wedge \mathsf{S}_{Z,Y} \wedge \neg \mathsf{S}_{Y,Z} \\
&\iff \exists Y : E_{u,Y} \wedge \big(F; (\mathsf{S} \cap \overline{\mathsf{S}^{\mathsf{T}}})\big)_{u,Y} \wedge \mathsf{L}_Y \\
&\iff \exists Y : \big(E \cap \big(F; (\mathsf{S} \cap \overline{\mathsf{S}^{\mathsf{T}}})\big)\big)_{u,Y} \wedge \mathsf{L}_Y \\
&\iff \big(\big(E \cap \big(F; (\mathsf{S} \cap \overline{\mathsf{S}^{\mathsf{T}}})\big)\big); \mathsf{L}\big)_u \\
&\iff \mathit{rel}\big(\big(\big(E \cap \big(F; (\mathsf{S} \cap \overline{\mathsf{S}^{\mathsf{T}}})\big)\big); \mathsf{L}\big)\big)_{u_1,u_2} \\
&\iff \mathit{BeatRel}(P)_{u_1,u_2}
\end{aligned}$$

Again the definition of the equality of relations shows the claim. □

The relation-algebraic specifications given in our Theorems 1 and 2 can be executed by means of the RelView tool after a straightforward translation into its programming language. The same is true for the relation-algebraic specifications we will present in the remainder of this section. We show the result of one such translation in the next section and also present there some results computed by the tool.

We have already mentioned that the x-column of the beating relation $B : X \leftrightarrow X$ describes the win set $W(x)$ of $x \in X$ and the transpose of the x-row of B describes the loss set $W^{-1}(x)$. In the next theorem these two facts are formulated with relation-algebraic means. Their proofs follow immediately from the definitions.

Theorem 3. *Let $p^x : X \leftrightarrow \mathbf{1}$ be the point which describes the element $x \in X$, the relation $P : N \leftrightarrow X^2$ model the spatial voting game $(N, \mathcal{W}, X)$, and $\mathit{BeatRel}(P) : X \leftrightarrow X$ be the relation-algebraic specification of the beating relation $B : X \leftrightarrow X$ of Theorem 2. Then the vector*

$$\mathit{win}\big(B, p^x\big) = \mathit{BeatRel}(P); p^x : X \leftrightarrow \mathbf{1}$$

describes the win set $W(x)$ of x and the vector

$$\mathit{loss}\big(B, p^x\big) = \mathit{BeatRel}(P)^{\mathsf{T}}; p^x : X \leftrightarrow \mathbf{1}$$

describes the loss set $W^{-1}(x)$ of x, both as subsets of X.

With a relation-algebraic specification of the beating relation $B : X \leftrightarrow X$ at hand, we also can specify the next solution concept, the majority core $MC(A)$ of a subset A of X, with relation-algebraic means.

Theorem 4. *Let again the vector $v^A : X \leftrightarrow \mathbf{1}$ describe the subset A of X, the relation $P : N \leftrightarrow X^2$ model the spatial voting game $(N, \mathcal{W}, X)$, and $\mathit{BeatRel}(P) : X \leftrightarrow X$ be the relation-algebraic specification of the beating relation $B : X \leftrightarrow X$ of Theorem 2. Then the vector*

$$\mathit{mc}\big(B, v^A\big) = v^A \cap \overline{\big(\overline{\mathit{BeatRel}(P)} \cap \overline{\mathsf{I}}\big); v^A} : X \leftrightarrow \mathbf{1},$$

describes the majority core $MC(A)$ of A as a subset of X.

Proof. For an arbitrarily given $x \in X$ we calculate as follows (where in the eighth step Theorem 2 is used and the variable y ranges over the set X):

$$\begin{aligned}
x \in MC(A) &\iff x \in A \wedge \forall y : y \in A \wedge y \neq x \rightarrow B_{x,y} \\
&\iff x \in A \wedge \neg\exists y : y \in A \wedge y \neq x \wedge \neg B_{x,y} \\
&\iff v^A_x \wedge \neg\exists y : v^A_y \wedge \overline{\mathsf{I}}_{x,y} \wedge \overline{B}_{x,y} \\
&\iff v^A_x \wedge \neg\exists y : \overline{B}_{x,y} \wedge \overline{\mathsf{I}}_{x,y} \wedge v^A_y
\end{aligned}$$

$$\begin{aligned}
&\iff && v_x^A \wedge \neg\exists y : (\overline{B} \cap \overline{\mathsf{I}})_{x,y} \wedge v_y^A \\
&\iff && v_x^A \wedge \neg\big((\overline{B} \cap \overline{\mathsf{I}}); v^A\big)_x \\
&\iff && \big(v^A \cap \overline{(\overline{B} \cap \overline{\mathsf{I}}); v^A}\big)_x \\
&\iff && \big(v^A \cap \overline{\big(\overline{BeatRel(P)} \cap \overline{\mathsf{I}}\big); v^A}\big)_x \\
&\iff && mc(B, v^A)_x
\end{aligned}$$

So, the desired result follows from the definition of the equality of relations. □

The definition of the last solution concept we want to treat, the uncovered set, is based on the covering relation. Hence, before attacking the uncovered set specification within the language of relation algebra we have to give a relation-algebraic specification of the covering relation. This can be done in terms of that of the beating relation as the following theorem shows.

Theorem 5. *Assume that the relation $P : N \leftrightarrow X^2$ models the spatial voting game $(N, \mathcal{W}, X)$. Then the covering relation $C : X \leftrightarrow X$ is relation-algebraically specified by*

$$CovRel(P) = BeatRel(P) \cap \overline{BeatRel(P)^{\mathsf{T}}; \overline{BeatRel(P)}} : X \leftrightarrow X,$$

where $BeatRel(P) : X \leftrightarrow X$ is the relation-algebraic specification of the beating relation $B : X \leftrightarrow X$ of Theorem 2.

Proof. If $x, y \in X$ are arbitrarily given points, then we are able to derive the following equivalence (where in the eight step Theorem 2 is used and the variable z ranges over the set X):

$$\begin{aligned}
C_{y,x} &\iff && B_{y,x} \wedge \forall z : B_{z,y} \rightarrow B_{z,x} \\
&\iff && B_{y,x} \wedge \neg\exists z : B_{z,y} \wedge \neg B_{z,x} \\
&\iff && B_{y,x} \wedge \neg\exists z : B_{z,y} \wedge \overline{B}_{z,x} \\
&\iff && B_{y,x} \wedge \neg\exists z : B^{\mathsf{T}}_{y,z} \wedge \overline{B}_{z,x} \\
&\iff && B_{y,x} \wedge \neg(B^{\mathsf{T}}; \overline{B})_{y,x} \\
&\iff && B_{y,x} \wedge \overline{B^{\mathsf{T}}; \overline{B}}_{y,x} \\
&\iff && \big(B \cap \overline{B^{\mathsf{T}}; \overline{B}}\big)_{y,x} \\
&\iff && \big(BeatRel(P) \cap \overline{BeatRel(P)^{\mathsf{T}}; \overline{BeatRel(P)}}\big)_{y,x} \\
&\iff && CovRel(P)_{y,x}
\end{aligned}$$

The definition of the equality of relations concludes the proof. □

If the relation-algebraic specification of this theorem is translated into RelView-code, then an auxiliary variable avoids the threefold call of the RelView-program for computing the beating relation. Having specified the covering relation within relation-algebraic means, we continue with the promised relation-algebraic specification of the uncovered set which is based on that of the covering relation of Theorem 5.

Theorem 6. *Let $v^A : X \leftrightarrow \mathbf{1}$ describe the subset A of X, the relation $P : N \leftrightarrow X^2$ model the spatial voting game $(N, \mathcal{W}, X)$, and $CovRel(P) : X \leftrightarrow X$ be the relation-algebraic specification of the covering relation $C : X \leftrightarrow X$ of Theorem 5. Then the vector*

$$uc\big(P, v^A\big) = v^A \cap \overline{\big(\overline{\mathsf{I}} \cap CovRel(P)\big)^{\mathsf{T}}; v^A} : X \leftrightarrow \mathbf{1}$$

describes the uncovered set $UC(A)$ of A as a subset of X.

Proof. For $x \in X$ being an arbitrary point we can calculate as follows (where in the sixth step Theorem 5 is used and the variable y ranges over the set A):

$$\begin{aligned}
x \in UC(A) &\iff && x \in A \wedge \neg\exists y : y \in A \wedge y \neq x \wedge C_{y,x} \\
&\iff && v_x^A \wedge \neg\exists y : (\overline{\mathsf{I}} \cap C)_{y,x} \wedge v_y^A \\
&\iff && v_x^A \wedge \neg\exists y : (\overline{\mathsf{I}} \cap C)^{\mathsf{T}}_{x,y} \wedge v_y^A
\end{aligned}$$

$$\begin{aligned}
&\iff && v^A_x \wedge \neg\big((\overline{\mathsf{I}} \cap C)^{\mathsf{T}}; v^A\big)_x \\
&\iff && v^A_x \wedge \overline{(\overline{\mathsf{I}} \cap C)^{\mathsf{T}}; v^A}_x \\
&\iff && \big(v^A \cap \overline{\big(\overline{\mathsf{I}} \cap \mathit{CovRel}(P)\big)^{\mathsf{T}}; v^A}\big)_x \\
&\iff && \mathit{uc}(P)_x
\end{aligned}$$

Again the definition of the equality of relations concludes the proof. □

So far, we have specified the solution concepts *PO*, *MC* and *UC* relation-algebraically in such a way that a vector, which describes a subset A of X, is mapped to a vector, which describes the solution $PO(A)$ (or $MC(A)$ and $UC(A)$, respectively) for this subset. In specific situations it may be of interest to know the solutions for more than one set, or even to know the solutions for all sets. The following last theorem of this section shows that our relation-algebraic specifications also solve these tasks without any modification, provided sets of sets are column-wisely enumerated in the sense of Section 2.

Theorem 7. *Assume that the relation $P : N \leftrightarrow X^2$ models the spatial voting game $(N, \mathcal{W}, X)$, the relation Q is defined as $Q = P^1 \cap \cdots \cap P^n$, and the relation $S : X \leftrightarrow \mathfrak{S}$ column-wisely enumerate the subset $\mathfrak{S}$ of 2^X. Then we have:*

1. *$po(Q, S) : X \leftrightarrow \mathfrak{S}$ column-wisely enumerates the set $\{PO(A) \mid A \in \mathfrak{S}\}$.*
2. *$mc(P, S) : X \leftrightarrow \mathfrak{S}$ column-wisely enumerates the set $\{MC(A) \mid A \in \mathfrak{S}\}$.*
3. *$uc(P, S) : X \leftrightarrow \mathfrak{S}$ column-wisely enumerates the set $\{UC(A) \mid A \in \mathfrak{S}\}$.*

Furthermore, the orders of all enumerations coincide.

Proof. To prove the first claim we assume arbitrary $A \in \mathfrak{S}$ and $x \in X$ and obtain by a generalization of the proof of Theorem 1 the following equivalence, such that the definition of the equality of relations shows the claim:

$$\begin{aligned}
x \in PO(A) &\iff x \in A \wedge \neg\exists y : y \in A \wedge \forall i : P^i_{y,x} \\
&\iff x \in A \wedge \neg\exists y : y \in A \wedge Q_{y,x} \\
&\iff S_{x,A} \wedge \neg\exists y : S_{y,A} \wedge Q_{y,x} \\
&\iff S_{x,A} \wedge \neg\exists y : Q^{\mathsf{T}}_{x,y} \wedge S_{y,A} \\
&\iff S_{x,A} \wedge \neg(Q^{\mathsf{T}}; S)_{x,A} \\
&\iff S_{x,A} \wedge \overline{Q^{\mathsf{T}}; S}_{x,A} \\
&\iff \big(S \cap \overline{Q^{\mathsf{T}}; S}\big)_{x,A} \\
&\iff po(Q, S)_{x,A}
\end{aligned}$$

In this calculation the equivalence of $x \in A$ and $S_{x,A}$ and of $y \in A$ and $S_{y,A}$ is a consequence of the column-wise enumeration of $\mathfrak{S}$ by S; see Section 2. In the same way the remaining claims can be shown. □

For example, if the relation $P : N \leftrightarrow X^2$ models the spatial voting game $(N, \mathcal{W}, X)$ and $\mathsf{M} : X \leftrightarrow 2^X$ is the membership relation induced by the set X, then the ith column of $mc(P, \mathsf{M})$ describes the majority core of the subset of X which is described by the ith column of M.

5. Applying RelView to some examples

All relation-algebraic specifications we have presented in Section 4 immediately can be transformed into the programming language of RelView. To give an impression how such programs look like, in Fig. 4 we present the program for computing the covering relation $C : X \leftrightarrow X$ from the relation $P : N \leftrightarrow X^2$ which models the given spatial voting game. In

```
CoveringRel(P)
  DECL B
  BEG B = BeatingRel(P)
      RETURN B & -(B^ * -B)
  END.
```

Fig. 4. A RelView-program for the covering relation.

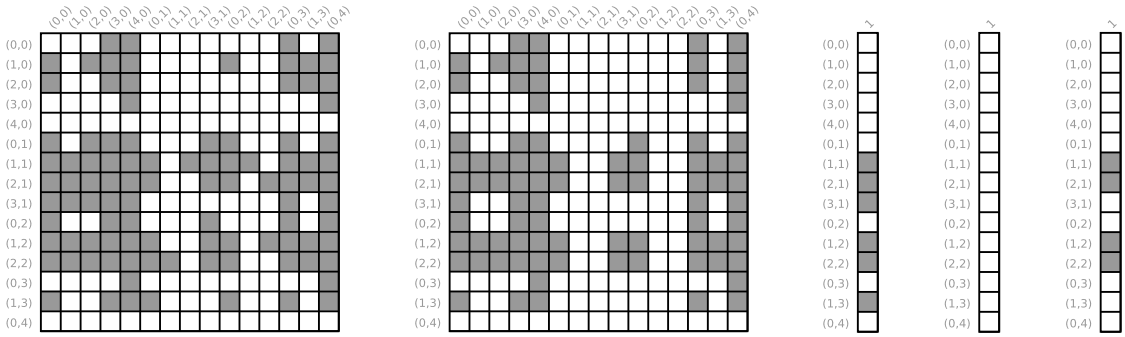


Fig. 5. Some results computed with the help of RelView.

the RelView-program `CovRel` the program `BeatRel` is assumed to compute the beating relation $B : X \leftrightarrow X$ following Theorem 2, the pre-defined infix operations & and $\star$ compute intersections and compositions, respectively, and the pre-defined operations – and ^ compute complements and transpositions, respectively. The correctness of the program of Fig. 4 and all RelView-programs we get from the theorems of Section 4 is guaranteed by the formal calculations of Section 4.

In the following example we consider again the spatial voting game of Example 1 with three players. Since the Boolean matrices for P^1, P^2 and P^3 have dimension 15×15, the Boolean matrix for the relation P which models this game has dimension 3×225 and, hence, is too large to be presented.

Example 3. The first and second of the five RelView-pictures of Fig. 5 show the beating relation B and the covering relation C of the spatial voting game of Example 1 and the remaining three pictures of the figure show, from left to right, the vectors which describe the Pareto set $PO(X)$, the majority core $MC(X)$, and the uncovered set $UC(X)$ for the set X of all points. Notice that the beating relation is not a so-called tournament relation since, for instance, the points $(1,0)$ and $(0,1)$ are unrelated. Precisely party 1 prefers $(1,0)$ to $(0,1)$ and precisely party 3 prefers $(0,1)$ to $(1,0)$ so that neither $(1,0)$ beats $(0,1)$ nor $(0,1)$ beats $(1,0)$. Because of the vector in the middle, for the set X of all points the majority core is empty, and from the labels of the remaining two vectors we obtain

$$PO(X) = \big\{(1,1),(2,1),(3,1),(1,2),(2,2),(1,3)\big\}$$

as the Pareto set of X and

$$UC(X) = \big\{(1,1),(2,1),(1,2),(2,2)\big\}$$

as the uncovered set of X.

In Schmidt (2010, [34]) it is shown that for a given order relation $R : Y \leftrightarrow Y$ and a vector $v : Y \leftrightarrow \mathbf{1}$ the vector

$$max(R, v) = v \cap \overline{(R \cap \dot{1}); v} : Y \leftrightarrow \mathbf{1}$$

describes the set of R-maximal elements of the set the vector v describes, and also that each membership relation $\mathsf{M} : Y \leftrightarrow 2^Y$ leads to a relation-algebraic specification of set inclusion on the powerset 2^Y via $\overline{\mathsf{M}^\mathsf{T}; \overline{\mathsf{M}}}$. We use these facts in the following example which treats the column-wise enumeration of solutions.

Example 4. From Example 3 we know already that in case of the spatial voting game of Example 1 the majority core of the set X of all points is empty. Therefore, it might be interesting to determine the maximal subsets of X which possess a non-empty majority core. To solve this task, we start with the membership relation $\mathsf{M} : X \leftrightarrow 2^X$. Since it column-wisely enumerates the powerset 2^X, Theorem 7 implies that the relation $mc(P, \mathsf{M}) : X \leftrightarrow 2^X$ column-wisely enumerates the set $\{MC(A) \mid A \in 2^X\}$. Next, we consider the vector $mc(P, \mathsf{M})^\mathsf{T}; \mathsf{L} : 2^X \leftrightarrow \mathbf{1}$, where $\mathsf{L} : X \leftrightarrow \mathbf{1}$. A little reflection shows that it describes the subset $\{A \in 2^X \mid MC(A) \neq \emptyset\}$ of the powerset 2^X. From the above remarks we can therefore conclude that the vector $max(\overline{\mathsf{M}^\mathsf{T}; \overline{\mathsf{M}}}, mc(P, \mathsf{M})^\mathsf{T}; \mathsf{L}) : 2^X \leftrightarrow \mathbf{1}$ describes the subset $\mathfrak{C}$ of 2^X that consists of the maximal sets of $\{A \in 2^X \mid MC(A) \neq \emptyset\}$. Finally, an application of the technique of Section 2 yields

$$S = \mathsf{M}; inj(max)\big(\overline{\mathsf{M}^\mathsf{T}; \overline{\mathsf{M}}}, mc(P, \mathsf{M})^\mathsf{T}; \mathsf{L}\big)^\mathsf{T} : X \leftrightarrow \mathfrak{C}$$

as column-wisely enumeration of the set $\mathfrak{C}$.

The leftmost of the four RelView-matrices of Fig. 6 shows the column-wise enumeration S of the set $\mathfrak{C}$ and the remaining three matrices of the figure are (from left to right) the RelView-representations of $po(Q, S)$, where Q is the intersection of the individual strict preference relations, $mc(P, S)$, and $uc(P, S)$, respectively. From the labels of the rows of the matrix of S we get that the four sets

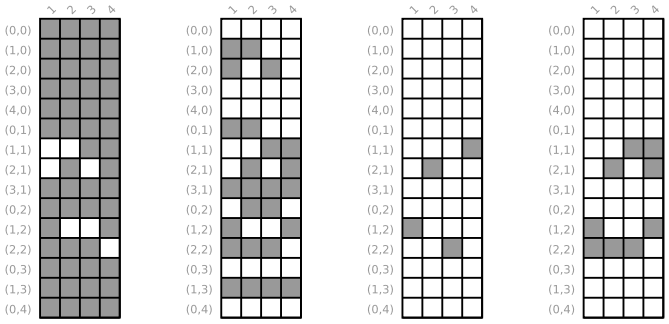


Fig. 6. Column-wise enumerations computed via RelView.

$$A_1 = X \setminus \{(1,1),(2,1)\} \qquad A_2 = X \setminus \{(1,1),(1,2)\}$$
$$A_3 = X \setminus \{(2,1),(1,2)\} \qquad A_4 = X \setminus \{(2,2)\}$$

possess non-empty majority cores and are maximal among all subsets of X with this property. Their majority cores are:

$$MC(A_1) = \{(1,2)\} \qquad MC(A_2) = \{(2,1)\}$$
$$MC(A_3) = \{(2,2)\} \qquad MC(A_4) = \{(1,1)\}$$

This follows from the labels of the rows of the matrix of $mc(P,S)$.

We conclude this section with a spatial voting game which is a slight modification of the spatial voting game we have considered so far.

Example 5. Assuming the same policy space $X = \{(i,j) \in \mathbb{N}^2 \mid i + j \leqslant 4\}$, we extend the spatial voting game of Example 1 by four additional parties 4 to 7 with the following ideal points:

$$x_4^* = (1,1) \qquad x_5^* = (1,3) \qquad x_6^* = (1,3) \qquad x_7^* = (1,3)$$

That is, the intention of party 4 is equal to that of party 2 and the intentions of the parties 5 to 7 are equal to the intention of party 3. Since common preference over the alternatives is again determined by majority, now all coalitions with at least four members win.

If we compute the beating relation B and the covering relation C for the new game, then the RelView tool yields the Boolean matrices at the first and second position of the series of five pictures of Fig. 7; the three vectors of the figure describe the three sets $PO(X)$, $MC(X)$ and $UC(X)$ for the set X of all points. As a consequence, the relation B equals the relation C and both are strict-order relations. Also the majority core $MC(X)$ and the uncovered set $UC(X)$ coincide and we have, due to the row labels of the two vectors of Fig. 7, the equality

$$MC(X) = UC(X) = \{(1,3)\}$$

Since through the addition of the four players no new individual strict preferences come into the play, the Pareto sets do not change.

6. Concluding remarks

This paper establishes once again that relation-algebraic methods in combination with the RelView tool enable us to compute efficiently many solution concepts from social choice theory and game theory. Even for simple examples the

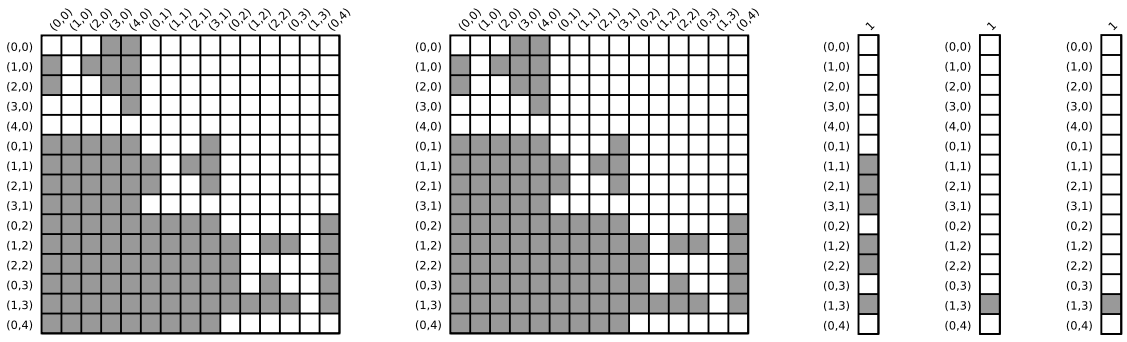


Fig. 7. Some further results computed via RelView.

computation of the different solutions cannot be done by hand in a feasible way. Since the relation-algebraic specifications immediately follow from the logical definitions, one may be sure that the corresponding RelView programs are correct. Although the RelView software is a general purpose tool and not tailored for a specific purpose, it is remarkably efficient, not in the least because of its implementation via ordered binary decision diagrams. Consequently, the tool can deal with practical examples from real life. Its visualization facilities in terms of Boolean matrices and/or directed graphs and the concise form of the programs make the tool extremely useful for experimentation and for educational purposes.

A drawback of the present version of RelView in regard to the problems we have considered in this paper is its restriction to relations as only datatype. This requires to compute the individual strict preference relations P^i either by hand or by a small program in a conventional programming language and to load them after that into the tool via its ASCII-file interface for relations. Because of methodological reasons we do not want to extend the tool's programming language by further datatypes. Nevertheless, to make RelView more easily applicable for problems, where in a first step some relations have to be computed using non-relational means, e.g., arithmetic operations, its newest version (Version 8.1, released September 2012 and available via [39]) allows to outsource such tasks into plug-ins for specific applications. There exist already such plug-ins, e.g., for computing the vector model of a weighted voting game (in the sense of Berghammer et al., 2011, [8]) from the game's weighted representation as input. For the near future we plan to extend RelView by a further plug-in which easily allows to insert spatial voting games and to compute the individual strict preference relations.

Acknowledgement

We thank Gunther Schmidt cordially for pointing out to us the potential of relation-algebraic methods and of RelView for the solution of many problems from social choice theory and game theory. Furthermore, we thank the unknown referees for their useful comments and suggestions, which helped to improve the paper.

References

[1] R. Behnke, R. Berghammer, T. Hoffmann, B. Leoniuk, P. Schneider, Applications of the RelView system, in: R. Berghammer, Y. Lakhnech (Eds.), Tool Support for System Specification, Development and Verification, Advances in Computing, Springer, 1999, pp. 33–47.
[2] R. Berghammer, B. Leoniuk, U. Milanese, Implementation of relational algebra using binary decision diagrams, in: H. de Swart (Ed.), Proceedings RelMiCS 2002, in: Lect. Notes Comput. Sci., vol. 2561, Springer, 2002, pp. 241–257.
[3] R. Berghammer, T. Hoffmann, B. Leoniuk, U. Milanese, Prototyping and programming with relations, Electron. Notes Theor. Comput. Sci. 44 (2003) 27–50.
[4] R. Berghammer, G. Schmidt, M. Winter, RelView and Rath – Two systems for dealing with relations, in: H. de Swart, et al. (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments I, in: Lect. Notes Comput. Sci., vol. 2929, Springer, 2003, pp. 1–16.
[5] R. Berghammer, A. Rusinowska, H. de Swart, Applying relation algebra and RelView to coalition formation, Eur. J. Oper. Res. 178 (2007) 530–542.
[6] R. Berghammer, A. Rusinowska, H. de Swart, An interdisciplinary approach to coalition formation, Eur. J. Oper. Res. 195 (2009) 487–496.
[7] R. Berghammer, A. Rusinowska, H. de Swart, Applying relation algebra and RelView to measures in a social network, Eur. J. Oper. Res. 202 (2010) 182–195.
[8] R. Berghammer, S. Bolus, A. Rusinowska, H. de Swart, A relation-algebraic approach to simple games, Eur. J. Oper. Res. 220 (2011) 636–645.
[9] R. Berghammer, A. Rusinowska, H. de Swart, Computations on simple games using RelView, in: V.P. Gerdt, et al. (Eds.), Proceedings CASC 2011, in: Lect. Notes Comput. Sci., vol. 6885, Springer, 2011, pp. 49–60.
[10] R. Berghammer, A. Rusinowska, H. de Swart, Computing tournament solutions using relation algebra and RelView, Eur. J. Oper. Res. 226 (2013) 636–645.
[11] W.T. Bianco, I. Jeliazkov, I. Sened, The uncovered set and the limits of legislative action, Polit. Anal. 12 (2004) 256–276.
[12] I. Budge, D. Robertson, D.J. Hearl (Eds.), Ideology, Strategy, and Party Change: Spatial Analyses of Post-War Election Programmes in 19 Democracies, Cambridge University Press, 1987.
[13] R. Calvert, Robustness of the multidimensional voting model: Candidate motivations, uncertainty, and convergence, Am. J. Polit. Sci. 29 (1985) 69–95.
[14] G.W. Cox, The uncovered set and the core, Am. J. Polit. Sci. 31 (1987) 408–422.
[15] T. Gloseclose, J.M. Snyder, Estimating party influence in congressional roll-call voting, Am. J. Polit. Sci. 44 (2000) 193–211.
[16] B. Grofman, G. Owen, N. Noviello, A. Glazer, Stability and centrality of legislative choice in the spatial context, Am. Polit. Sci. Rev. 8 (1987) 539–553.
[17] R. Hartley, M. Kilgour, The geometry of the uncovered set in the three-voter spatial model, Math. Soc. Sci. 14 (1987) 175–183.
[18] S. Jackman, Multidimensional analysis of roll call data via Bayesian simulation: Identification, estimation, inference, and model checking, Polit. Anal. 9 (2001) 227–241.
[19] R.D. McKelvey, Intransitivities in multidimensional voting models and some implications for agenda control, J. Econ. Theory 12 (1976) 472–482.
[20] R.D. McKelvey, General conditions for global intransitivities in formal voting models, Econometrica 47 (1979) 1085–1112.
[21] R.D. McKelvey, Covering, dominance, and institution free properties of social choice, Am. J. Polit. Sci. 30 (1986) 283–314.
[22] R.D. McKelvey, N. Schofield, Generalized symmetry conditions at a core, Econometrica 55 (1987) 923–933.
[23] B. Leoniuk, ROBDD-based implementation of relational algebra with applications, Dissertation, Universität Kiel, 2001 (in German).
[24] U. Milanese, On the implementation of a ROBDD-based tool for the manipulation and visualization of relations, Dissertation, Universität Kiel, 2003 (in German).
[25] N. Miller, A new solution set for tournaments and majority voting, Am. J. Polit. Sci. 24 (1980) 68–96.
[26] N. Miller, In search of the uncovered set, Polit. Anal. 15 (2007) 21–45.
[27] N. Miller, B. Grofman, S. Feld, The geometry of majority rule, J. Theor. Polit. 1 (1989) 379–406.
[28] P.C. Ordeshook, T. Schwartz, Agenda and the control of political outcomes, Am. Polit. Sci. Rev. 81 (1987) 179–199.
[29] K.T. Poole, Spatial Models of Parliamentary Voting, Cambridge University Press, 2005.
[30] K.T. Poole, H. Rosenthal, Congress: A Political-Economic History of Roll-Call Voting, Oxford University Press, 2000.
[31] A. Rusinowska, R. Berghammer, P. Ecklund, J.W. van der Rijt, M. Roubens, H. de Swart, Social software for coalition formation, in: H. de Swart, et al. (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments II, in: Lect. Notes Artif. Intelligence, vol. 4342, Springer, 2006, pp. 1–30.
[32] A. Rusinowska, R. Berghammer, H. de Swart, M. Grabisch, Social networks: prestige, centrality and influence, in: H. de Swart (Ed.), Proceedings RAMiCS 2011, in: Lect. Notes Comput. Sci., vol. 6663, Springer, 2011, pp. 21–38.

[33] G. Schmidt, T. Ströhlein, Relations and Graphs: Discrete Mathematics for Computer Scientists, EATCS Monogr. Theor. Comput. Sci., Springer, 1993.
[34] G. Schmidt, Relational Mathematics, Encycl. Math. Appl., vol. 132, Cambridge University Press, 2010.
[35] N. Schofield, Instability of simple dynamic games, Rev. Econ. Stud. 45 (1978) 575–594.
[36] K.A. Shepsle, B.R. Weingast, Uncovered sets and sophisticated voting outcomes with implications for agenda institutions, Am. J. Polit. Sci. 28 (1984) 49–74.
[37] K.A. Shepsle, B.R. Weingast, Positive theories of congressional institutions, Legis. Stud. Q. 19 (1994) 149–181.
[38] H. de Swart, R. Berghammer, A. Rusinowska, Computational social choice using relation algebra and RelView, in: R. Berghammer, et al. (Eds.), Proceedings RelMiCS/AKA 2009, in: Lect. Notes Comput. Sci., vol. 5827, Springer, 2009, pp. 13–28.
[39] RelView homepage: http://www.informatik.uni-kiel.de/~progsys/relview/.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 135–153

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Exploring modal worlds

Han-Hing Dang, Roland Glück, Bernhard Möller *, Patrick Roocks, Andreas Zelend

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany

ARTICLE INFO

Article history:
Available online 15 February 2014

Keywords:
Bisimulation
Formal concept analysis
Pareto front
Rectangles
Separation logic
Software product lines

ABSTRACT

Modal idempotent semirings cover a large set of different applications. The paper presents a small collection of these, ranging from algebraic logics for program correctness over bisimulation refinement, formal concept analysis, database preferences to feature oriented software development. We provide new results and/or views on these domains; the modal semiring setting allows a concise and unified treatment, while being more general than, e.g., standard relation algebra.

1. Introduction

Algebraic structures, such as *modal* idempotent semirings or Kleene algebras, offer a large variety of applications, while requiring only a small set of operators and axioms. Such algebras abstractly capture so-called Kripke structures, i.e., access relations over a set of worlds or states. In addition they provide the associated multi-modal operators box and diamond that allow reasoning, e.g., about possible actions of agents in a system or about state transitions in general. Particular instances of modal semirings are provided by the algebra of homogeneous binary relations and by abstract relation algebras.

This setting allows many general considerations and results, ranging from epistemic logics with knowledge and belief [1] to propositional dynamic Hoare logic and resource-based settings such as separation logic [2]. Moreover, many further applications are covered, like abstract reasoning about bisimulations for model refinement [3], formal concept analysis, simple and concise correctness proofs for the optimisation of database preference queries [4] or generally applicable models of module hierarchies in a feature oriented software development process [5].

In this paper, we take the readers on a short tour through several of these modal worlds and hope that they will enjoy the ride, maybe even feel some kind of explorer's excitement. We provide new results and/or views on the mentioned applications; the modal semiring setting allows a concise and unified treatment, while being more general than, e.g., standard relation algebra. Nevertheless the excellent relational papers and books by Gunther Schmidt [6,7] are gratefully and respectfully acknowledged as a constant source of inspiration (although at times the relational encoding requires some "decryption" to obtain smooth modal formulations). It is our pleasure to dedicate this paper to Gunther on the occasion of his 75th birthday!

The paper is organised as follows. In Section 2 we recapitulate the main definitions of modal idempotent semirings and provide some generally applicable laws. Section 3 extends an existing algebraic framework from autobisimulations to bisimulations between different relations. An algebraic treatment of formal concepts and rectangles is given in Section 4, while in Section 5 we set up a connection between rectangles and Pareto fronts in databases with preference queries. Section 6

* Corresponding author.
E-mail addresses: h.dang@informatik.uni-augsburg.de (H.-H. Dang), glueck@informatik.uni-augsburg.de (R. Glück), moeller@informatik.uni-augsburg.de (B. Möller), roocks@informatik.uni-augsburg.de (P. Roocks), zelend@informatik.uni-augsburg.de (A. Zelend).

http://dx.doi.org/10.1016/j.jlap.2014.02.004
2352-2208/

considers an abstract partial correctness approach to separation logic, and Section 7 provides an algebra of modules for structured documents in software product lines.

2. Basics of modal semirings

Idempotent semirings are a well-known concept for modelling choice and sequential composition by the algebraic operations $+$ and $\cdot$.

Definition 2.1. A *semiring* is a structure $(S, +, 0, \cdot, 1)$ with $0 \neq 1$ such that $+$ and $\cdot$ are associative binary operations on S with neutral elements 0 and 1 resp., $+$ is commutative, and $\cdot$ distributes both from left and right over $+$. Moreover, 0 is an annihilator of $\cdot$, i.e., $x \cdot 0 = 0 = 0 \cdot x$ holds for all $x \in S$.

The operations $+$ and $\cdot$ are also called *addition* and *multiplication*, resp. As usual, multiplication binds stronger than addition, so $x + y \cdot z$ stands for $x + (y \cdot z)$. Due to associativity we are free to omit superfluous parentheses.

A semiring is called *idempotent* if $x + x = x$ holds for all $x \in S$. In this case, the relation $\leqslant \subseteq S \times S$, defined by $x \leqslant y \Leftrightarrow_{df} x + y = y$, is a partial order on S, called the *natural order*. In particular, the supremum of two elements x and y with respect to the natural order is given by $x + y$, the least element is 0, and both addition and multiplication are isotone. The infimum of two elements x and y need not exist; if it does it is denoted by $x \sqcap y$. The element 0 is irreducible with respect to $+$, i.e., $x + y = 0 \Leftrightarrow x = 0 = y$ holds for all $x, y \in S$.

For an arbitrary set M, the structure $(\mathsf{Rel}(M), \cup, \emptyset, ;, \mathsf{id}(M))$ forms an idempotent semiring where $\mathsf{Rel}(M)$ denotes the set of all binary relations over M, ; denotes relational composition and $\mathsf{id}(M)$ the identity relation on M.

Definition 2.2. A semiring $(S, +, 0, \cdot, 1)$ is called *Boolean* if it is a distributive lattice with join $+$ and meet $\sqcap$ equipped with a *complement operation* $\bar{\cdot} : S \to S$ with the following properties for all $x, y \in S$:

$$x + \bar{x} = y + \bar{y} \quad \text{and} \quad x \sqcap \bar{x} = y \sqcap \bar{y}, \tag{1}$$

$$\overline{x + y} = \bar{x} \sqcap \bar{y} \quad \text{and} \quad \overline{x \sqcap y} = \bar{x} + \bar{y}. \tag{2}$$

In an idempotent Boolean semiring, the element $\top =_{df} \bar{0}$ is the greatest element with respect to the natural order. Moreover, $x + \bar{x} = \top$, $x \sqcap \bar{x} = 0$ and $\bar{\bar{x}} = x$ for all $x \in S$. The structure $(\mathsf{Rel}(M), \cup, \emptyset, ;, \mathsf{id}(M))$ becomes a Boolean semiring if we define the complement operation by $\overline{R} =_{df} (M \times M) \backslash R$ (where $\backslash$ denotes set theoretic difference). The greatest element is the universal relation $M \times M$.

In $\mathsf{Rel}(M)$ a subset $N \subseteq M$ can be characterised by the associated partial identity $\mathsf{id}(N)$. This is abstracted to general idempotent semirings by the notion of tests as axiomatised in [8].

Definition 2.3. An element p of an idempotent semiring is called a *test* if it has a *relative complement* $\neg p$ with the properties $p + \neg p = 1$ and $p \cdot \neg p = 0 = \neg p \cdot p$.

In an idempotent semiring $(S, +, 0, \cdot, 1)$ the set of tests is denoted by $\mathsf{test}(S)$. As a writing convention, elements of $\mathsf{test}(S)$ are denoted by p, q, r and variants thereof. On tests, multiplication coincides with the infimum, i.e., we have $p \sqcap q = p \cdot q$ for all $p, q \in \mathsf{test}(S)$. As a consequence of this fact, multiplication on tests is both idempotent and commutative. Moreover, on tests also addition distributes over multiplication, i.e., $p + q \cdot r = (p + q) \cdot (p + r)$ holds for all tests p, q and r. The structure $(\mathsf{test}(S), +, 0, \cdot, 1)$ is a Boolean semiring with $\neg$ as complement operation and greatest element 1. We set $p - q =_{df} p \cdot \neg q$.

Since all tests are $\leqslant 1$, multiplication with a test corresponds to restriction. If a stands for an abstract transition element, such as a relation, $p \cdot a$ restricts a to starting states that lie in the set p and $a \cdot q$ to ending states in q. In $(\mathsf{Rel}(M), \cup, \emptyset, ;, \mathsf{id}(M))$ the tests are exactly the subrelations of $\mathsf{id}(M)$. We will use that in Section 7.

A few further useful properties are collected in the following lemma.

Lemma 2.4.

1. *In a Boolean semiring $(S, +, 0, \cdot, 1)$ all elements $p \leqslant 1$ are tests with relative complement $\neg p = 1 \sqcap \bar{p}$.*
2. *In every idempotent semiring we have the following properties for all $a, b \in S$ such that $a \sqcap b$ exists, and all $p, q \in \mathsf{test}(S)$:*

$$p \cdot (a \sqcap b) = p \cdot a \sqcap b = p \cdot a \sqcap p \cdot b,$$

$$(a \sqcap b) \cdot p = a \cdot p \sqcap b = a \cdot p \sqcap b \cdot p.$$

The next concepts we introduce are the *domain* and *codomain* operations.

Definition 2.5. A *modal* semiring is an idempotent semiring $(S, +, 0, \cdot, 1)$ with two additional operations $\lceil\cdot : S \to \mathsf{test}(S)$ and $\cdot\rceil : S \to \mathsf{test}(S)$, fulfilling the following properties for all $x, y \in S$ and $p \in \mathsf{test}(S)$:

$$x \leqslant \lceil x \cdot x \qquad \text{and} \quad x \leqslant x \cdot x\rceil, \tag{d1/cd1}$$

$$\lceil(p \cdot x) \leqslant p \qquad \text{and} \quad (x \cdot p)\rceil \leqslant p, \tag{d2/cd2}$$

$$\lceil(x \cdot \lceil y) \leqslant \lceil(x \cdot y) \qquad \text{and} \quad (x\rceil \cdot y)\rceil \leqslant (x \cdot y)\rceil. \tag{locality}$$

The operator $\lceil\cdot$ is called the *domain*, $\cdot\rceil$ the *codomain* operator. If only (d1/cd1) and (d2/cd2) hold the operator is called a *predomain/precodomain* operator.

It can be shown that the inequations (d1/cd1) and (locality) strengthen to equations (see [9]). Moreover, both domain and codomain are fully strict, i.e., $\lceil x = 0 \Leftrightarrow x = 0$ and $x\rceil = 0 \Leftrightarrow x = 0$ hold for all $x \in S$. Domain and codomain operations on $\mathsf{test}(S)$ are simply the identity operations, i.e., for all $p \in \mathsf{test}(S)$ we have $\lceil p = p = p\rceil$. Both operations distribute over addition, i.e., $\lceil(x + y) = \lceil x + \lceil y$ and $(x + y)\rceil = x\rceil + y\rceil$ hold for all $x, y \in S$. As a consequence thereof, they are isotone with respect to the natural order, i.e., $x \leqslant y \Rightarrow \lceil x \leqslant \lceil y \wedge x\rceil \leqslant y\rceil$ holds for arbitrary $x, y \in S$. In the case of existence, domain and codomain are uniquely determined.

The domain and codomain operators on $(\mathsf{Rel}(M), \cup, \emptyset, ;, \mathsf{id}(M))$ are given by $\lceil R = \{(m, m) \mid \exists y : (m, y) \in R\}$ and $R\rceil = \{(m, m) \mid \exists y : (y, m) \in R\}$.

Based on domain and codomain we define the diamond and box operators for arbitrary $x \in S$ and $p \in \mathsf{test}(S)$ as follows:

Definition 2.6.

$$|x\rangle p =_{df} \lceil(x \cdot p), \qquad \langle x|p =_{df} (p \cdot x)\rceil, \tag{forward/backward diamond}$$

$$|x]p =_{df} \neg|x\rangle\neg p, \qquad [x|p =_{df} \neg\langle x|\neg p. \tag{forward/backward box}$$

For $R \in \mathsf{Rel}(M)$, forward diamond and backward diamond correspond to the preimage and image of a subset of M under R, resp. The forward box $|x]p$ models the set of all elements of M from where every transition under x leads inevitably into the subset corresponding to p. An analogous interpretation can be given for the backward box.

As an inheritance of domain and codomain, the diamond operators are isotone in both arguments and distribute in both arguments over addition. The box operators are antitone in the first argument and isotone in the second argument. This follows also from the fact that we have the following Galois connections [10] between the modal operators:

$$p \leqslant |a]q \quad \Leftrightarrow \quad \langle a|p \leqslant q, \qquad p \leqslant [a|q \quad \Leftrightarrow \quad |a\rangle p \leqslant q.$$

If locality holds, the operators also distribute over composition:

$$|a \cdot b\rangle p = |a\rangle|b\rangle p, \qquad \langle a \cdot b|p = \langle b|\langle a|p, \qquad |a \cdot b]p = |a]|b]p, \qquad [a \cdot b|p = [b|[a|p. \tag{3}$$

Finally, we have for element a and tests p, q with atomic p that

$$p \leqslant |a\rangle q \quad \Leftrightarrow \quad p \cdot a \cdot q \neq 0. \tag{4}$$

Here, a test p is atomic if $p \neq 0$ and for all tests q we have that $q \leqslant p \Rightarrow q = p$. Many further properties of modal operators can be found in [11].

3. Bisimulations

Bisimulations are a frequently used tool, not only in process algebra but also in model checking and control theory. In this section we show how to model them algebraically in modal semirings and prove some of their basic properties in a very simple calculational style.

In relational algebra a left and right total relation $B \subseteq X \times Y$ is called a *bisimulation* between two relations $R_1 \subseteq X \times X$ and $R_2 \subseteq Y \times Y$ if

$$B^\smile; R_1 \subseteq R_2; B^\smile \wedge B; R_2 \subseteq R_1; B.$$

In [3] this characterisation was used to reason about autobisimulations, i.e., bisimulations between a relation R and itself. Here we will derive a framework which allows reasoning about bisimulations between two different relations.

In our setting, left and right totality of a bisimulation b between two elements g_1 and g_2 can easily be modelled by the condition $\lceil b = \lceil g_1 + g_1\rceil \wedge b\rceil = \lceil g_2 + g_2\rceil$.

Given two functions $f_1, f_2 : \mathsf{test}(S) \to \mathsf{test}(S)$, we write $f_1 = f_2$ iff for all $p \in \mathsf{test}(S)$ the equality $f_1(p) = f_2(p)$ holds. Analogously we define the predicate $f_1 \leqslant f_2$ by $f_1(p) \leqslant f_2(p)$ for all $p \in \mathsf{test}(S)$. We say that an element $g_2 \in S$ is a *pseudoconverse* of an element $g_1 \in S$ if $|g_1\rangle = \langle g_2|$. In [3] it is shown that this requirement is equivalent to $\langle g_1| = |g_2\rangle$. These considerations lead to the following definition:

Definition 3.1. Let S be a modal semiring with locality. An element $b \in S$ is called a bisimulation between two elements $g_1 \in S$ and $g_2 \in S$ if the following conditions are fulfilled:

$$\ulcorner b = \ulcorner g_1 + g_1 \urcorner \wedge b \urcorner = \ulcorner g_2 + g_2 \urcorner, \tag{5}$$

$$\langle b||g_1\rangle \leqslant |g_2\rangle\langle b| \wedge |b\rangle|g_2\rangle \leqslant |g_1\rangle|b\rangle. \tag{6}$$

In this case, we write $g_1 \sim_b g_2$.

In the sequel we will show how some properties of bisimulations in a relational setting can be stated and proved in an algebraic manner based on Definition 3.1.

It is well known that the identity relation is a bisimulation between a relation and itself. Moreover, bisimulations are closed under taking the converse, relational composition and union. These properties are translated into the language of modal semirings in the following theorem.

Theorem 3.2. *In a modal semiring S, the following properties hold*:

1. *For every g, the test $\ulcorner g + g \urcorner$ is a bisimulation between g and itself.*
2. *Let b be a bisimulation between g_1 and g_2, and let $b^\smile$ be a pseudoconverse of b. Then $b^\smile$ is a bisimulation between g_2 and g_1.*
3. *Let b_{12} be a bisimulation between g_1 and g_2, and let b_{23} be a bisimulation between g_2 and g_3. Then $b_{12} \cdot b_{23}$ is a bisimulation between g_1 and g_3.*
4. *Let b and b' be bisimulations between g_1 and g_2. Then $b + b'$ is a bisimulation between g_1 and g_2.*

Proof.

1. $\ulcorner g + g \urcorner$ obviously fulfills Definition 3.1(5). For (6) we fix an arbitrary test p and reason as follows: by distributivity of $\langle\cdot|$, by $\ulcorner g, g \urcorner \in \mathsf{test}(S)$ and $\langle q|r = q \cdot r$, by $\ulcorner g, g \urcorner \in \mathsf{test}(S)$ again and $|q \cdot a\rangle r = q \cdot |a\rangle r$, by $\ulcorner g \cdot g = g$, and $|g \urcorner \cdot g\rangle p \leqslant |g\rangle p$ by $g \urcorner \leqslant 1$ and isotony of $|\cdot\rangle$,

   $$\langle \ulcorner g + g \urcorner||g\rangle p = \langle \ulcorner g||g\rangle p + \langle g \urcorner||g\rangle p = \ulcorner g \cdot |g\rangle p + g \urcorner \cdot |g\rangle p = |\ulcorner g \cdot g\rangle p + |g \urcorner \cdot g\rangle p = |g\rangle p + |g \urcorner \cdot g\rangle p = |g\rangle p.$$

   Analogously we obtain $|g\rangle\langle \ulcorner g + g \urcorner|p = |g\rangle p$, so we even have the equality $\langle \ulcorner g + g \urcorner||g\rangle p = |g\rangle\langle \ulcorner g + g \urcorner|p$. The equality $|\ulcorner g + g \urcorner\rangle|g\rangle p = |g\rangle|\ulcorner g + g \urcorner\rangle p$ can be shown symmetrically.
2. By properties of pseudoconverses and Definition 3.1(5), $\ulcorner(b^\smile) = b \urcorner = \ulcorner g_2 + g_2 \urcorner$ and symmetrically $(b^\smile) \urcorner = \ulcorner b = \ulcorner g_1 + g_1 \urcorner$. Moreover, by the definition of pseudoconverse, $g_1 \sim_b g_2$, Definition 3.1(6), and property of pseudoconverse:

   $$\langle b^\smile||g_2\rangle = |b\rangle|g_2\rangle \leqslant |g_1\rangle|b\rangle = |g_1\rangle\langle b^\smile|.$$

   In a similar manner we obtain $|b^\smile\rangle|g_1\rangle \leqslant |g_2\rangle|b^\smile\rangle$, hence $b^\smile$ is a bisimulation between g_2 and g_1.
3. First, by (3), $g_2 \sim_{b_{23}} g_3$, Definition 3.1(5), $g_1 \sim_{b_{12}} g_2$, Definition 3.1(5), (3), and $g_1 \sim_{b_{12}} g_2$, Definition 3.1(6):

   $$\ulcorner(b_{12} \cdot b_{23}) = \ulcorner(b_{12} \cdot \ulcorner b_{23}) = \ulcorner\big(b_{12} \cdot (\ulcorner g_2 + g_2 \urcorner)\big) = \ulcorner(b_{12} \cdot b_{12} \urcorner) = \ulcorner b_{12} = \ulcorner g_1 + g_1 \urcorner.$$

   The property $(b_{12} \cdot b_{23}) \urcorner = \ulcorner g_3 + g_3 \urcorner$ can be obtained symmetrically. Moreover, we have the following calculation (and a symmetric one for $\langle b_{12} \cdot b_{23}||g_1\rangle \leqslant |g_3\rangle\langle b_{12} \cdot b_{23}|$): by (3), $g_2 \sim_{b_{23}} g_3$, Definition 3.1(6), $g_1 \sim_{b_{12}} g_2$, Definition 3.1(6), and (3):

   $$|b_{12} \cdot b_{23}\rangle|g_3\rangle = |b_{12}\rangle|b_{23}\rangle|g_3\rangle \leqslant |b_{12}\rangle|g_2\rangle|b_{23}\rangle \leqslant |g_1\rangle|b_{12}\rangle|b_{23}\rangle = |g_1\rangle|b_{12} \cdot b_{23}\rangle.$$

4. First, by distributivity of domain over addition, the assumption and idempotence of addition we have $\ulcorner(b + b') = \ulcorner b + \ulcorner b' = \ulcorner g_1 + g_1 \urcorner$. An analogous calculation shows $(b + b') \urcorner = \ulcorner g_2 + g_2 \urcorner$. Moreover, we can argue as follows: by distributivity of diamond in its first argument, assumption, isotony of $+$, and distributivity of diamond in its first argument, and idempotence:

   $$\langle b + b'||g_1\rangle = \langle b||g_1\rangle + \langle b'||g_1\rangle \leqslant |g_2\rangle\langle b| + |g_2\rangle\langle b'| = |g_2\rangle\langle b + b'|.$$

   The property $|b + b'\rangle|g_2\rangle \leqslant |g_1\rangle|b + b'\rangle$ follows symmetrically. □

4. Concepts and rectangles

In this section we deal with formal concept analysis as pioneered by Ganter and Wille [12]. A formal concept defines a maximal association between certain objects and attributes. Applications include data mining, text mining, machine learning, knowledge management, the semantic web, software development and biology. We show that by using a modal semiring formulation many of the basic properties fall out of standard laws quite easily. Also we set up a connection with rectangles and pseudorectangles which are used in the actual computation of concepts.

4.1. The relational case

We start with a brief recapitulation of the basic notions.

Definition 4.1. A *context* is a triple (O, A, R) with a set O of *objects*, a set A of *attributes* and a relation $R \subseteq O \times A$ that associates objects with attributes.

Example 4.2. (See e.g. [13].) Let O be the set of natural numbers from 1 to 10 and $A = \{composite, even, odd, prime, square\}$ with the obvious association relation R. □

Definition 4.3. Over a context one defines two functions $F : \mathcal{P}(O) \to \mathcal{P}(A)$ and $G : \mathcal{P}(A) \to \mathcal{P}(O)$ by setting, for $X \subseteq O$ and $Y \subseteq A$,

$$F(X) =_{df} \{a \in A \mid \forall o \in X : o\, R\, a\},$$
$$G(Y) =_{df} \{o \in O \mid \forall a \in Y : o\, R\, a\}.$$

In words: $F(X)$ is the set of attributes shared by all objects in X, while $G(Y)$ is the set of objects that share all attributes in Y.

By the definitions of G, of the Cartesian product twice and of F we have

$$\begin{array}{lllll} X \subseteq G(Y) & \Leftrightarrow & (\forall o \in X : \forall a \in Y : o\, R\, a) & \Leftrightarrow & X \times Y \subseteq R \\ & \Leftrightarrow & (\forall a \in Y : \forall o \in X : o\, R\, a) & \Leftrightarrow & Y \subseteq F(X). \end{array} \tag{GC}$$

This means that F and G form a Galois connection [10] between the posets $(\mathcal{P}(O), \subseteq)$ and $(\mathcal{P}(A), \supseteq)$. The middle property in this calculation gives rise to the following notion.

Definition 4.4. X and Y define a *rectangle* of R iff $X \times Y \subseteq R$.

By the standard theory of Galois connections, one has $X \subseteq G(F(X))$ and $Y \subseteq F(G(Y))$ as well as $F(X) = F(G(F(X)))$ and $G(Y) = G(F(G(Y)))$. Hence, given a set X of objects, $G(F(X))$ is the greatest set of objects that have the same attributes as the objects in X, and similarly for $F(G(Y))$. This motivates the following notion.

Definition 4.5. A *concept* within the context (O, A, R) is a pair $C = (X, Y)$ with $X \subseteq O$, $Y \subseteq A$ and $X = G(Y)$ and $Y = F(X)$. X and Y are called the *extension* and the *intension* of C, resp.

Example 4.6. Let O, A and R be as in Example 4.2. Two examples of concepts are then

$$\big(\{3, 5, 7\}, \{odd, prime\}\big) \quad \text{and}$$
$$\big(\{1, 4, 9\}, \{square\}\big).$$

Informally, these concepts would be described as *the odd prime numbers* and *the squares* within the given context. □

By the standard theory of Galois connections, $X \subseteq O$ is the extension of a concept iff $X = G(F(X))$, which then is $(X, F(X))$. Symmetrically, $Y \subseteq A$ is the intension of a concept iff $Y = F(G(Y))$, which then is $(G(Y), Y)$. Moreover, $G \circ F \circ G = G$ and $F \circ G \circ F = F$; hence all images under F are intensions and all images under G are extensions. By (GC) the concepts $(X, F(X))$ and $(G(Y), Y)$ give rise to the rectangles $X \times F(X)$ and $G(Y) \times Y$, resp.

Finally, again by the Galois connection, F and G are universally conjunctive, i.e., preserve all existing infima. This can be exploited for a more efficient way of computing all concepts of a context: for $X \subseteq O$, $Y \subseteq A$ we have $F(X) = \bigcap_{o \in X} F(\{o\})$ and $G(Y) = \bigcap_{a \in Y} G(\{a\})$. For a concept (X, Y) therefore $Y = \bigcap_{o \in X} F(\{o\})$ and $X = \bigcap_{a \in Y} G(\{a\})$. By standard convention, for empty Y or X these intersections yield A or O, resp. Therefore it suffices to compute first all intensions $\{F(\{o\}) \mid o \in O\}$ or extensions $\{G(\{a\}) \mid a \in A\}$ and to obtain the others as intersections of these.

4.2. The general case and modalities

We now abstract from the relational case to a Boolean modal semiring.

First we observe that a rectangle $X \times Y$ can be relationally represented as $I_X ; \mathbb{T} ; I_Y$, where I_X, I_Y are the partial identity relations associated with X, Y and $\mathbb{T}$ is the universal relation $O \times A$. This leads to the following definition.

Definition 4.7. Let $p, q \leqslant 1$ be elements of a Boolean modal semiring S. Then the *rectangle* defined by p and q is

$$p \times q =_{df} p \cdot \top \cdot q,$$

where $\top =_{df} \bar{0}$ is the greatest element of S. For an arbitrary element a of S we say that $p \times q$ is a *rectangle* of a if $p \times q \leqslant a$.

Example 4.8. In the path semiring [9], a rectangle consists of all possible node sequences starting with a node in p and ending with a node in q. □

We list a few simple consequences of the definition.

Lemma 4.9.

1. $p \times q \sqcap a = p \cdot a \cdot q$.
2. $(p + q) \times r \leqslant a \Leftrightarrow p \times r \leqslant a \wedge q \times r \leqslant a$.
3. $p \times (q + r) \leqslant a \Leftrightarrow p \times q \leqslant a \wedge p \times r \leqslant a$.

Proof.

1. Immediate from the definition of rectangles and Lemma 2.4.2.
2. Immediate from distributivity and lattice algebra.
3. Immediate from distributivity and lattice algebra. □

This allows establishing a connection with program semantics. It is well known that the semantics of Hoare triples can be given algebraically as

$$\{p\}\, a\, \{q\} \quad \Leftrightarrow_{df} \quad p \cdot a \cdot \neg q \leqslant 0.$$

Using Lemma 4.9.1 this transforms as follows:

$$\{p\}\, a\, \{q\} \quad \Leftrightarrow \quad p \times \neg q \sqcap a \leqslant 0 \quad \Leftrightarrow \quad p \times \neg q \leqslant \bar{a}.$$

Informally, any transitions from p states to $\neg q$ states must lie outside a.

We have the following characterisation of the rectangles of a, which for the relational case is also given in [14], albeit without using the notion of modal operators.

Lemma 4.10. *The following properties are equivalent.*

1. $p \times q$ *is a rectangle of* a.
2. $p \leqslant |\bar{a}]\neg q$.
3. $q \leqslant [\bar{a}|\neg p$.
4. $|\bar{a}\rangle q \leqslant \neg p$.
5. $\langle \bar{a}|p \leqslant \neg q$.

Proof. We obtain by the definition of $\times$, shunting, Lemma 2.4, greatestness of $\top$ and the standard connection between box and restriction,

$$\begin{aligned} p \times q \leqslant a \quad &\Leftrightarrow \quad p \cdot \top \cdot q \leqslant a \quad \Leftrightarrow \quad p \cdot \top \cdot q \sqcap \bar{a} \leqslant 0 \\ &\Leftrightarrow \quad p \cdot \bar{a} \cdot q \sqcap \top \leqslant 0 \quad \Leftrightarrow \quad p \cdot \bar{a} \cdot q \leqslant 0 \quad \Leftrightarrow \quad p \leqslant |\bar{a}]\neg q. \end{aligned}$$

The remaining equivalences are standard for modal operators. □

Corollary 4.11. *For arbitrary tests* p, q *the following elements are rectangles of an element* a*:*

$$\big(|\bar{a}]\neg q\big) \times q, \qquad p \times \big([\bar{a}|\neg p\big).$$

Proof. Immediate from Lemma 4.10. □

The functions $f_a(p) =_{df} [\bar{a}|\neg p$ and $g_a(q) =_{df} |\bar{a}]\neg q$ are the abstract counterparts of F and G from Section 4.1. This motivates the following notion.

Definition 4.12. We call a pair (p, q) of tests a *concept* of a if $p = g_a(q)$ (equivalently, if $q = f_a(p)$).

4.3. Comparing rectangles

We will now set up a connection between concepts and maximal rectangles of an element. Let us therefore first investigate the order between rectangles.

It turns out that we need a special assumption about the domain/codomain operators defining the modal operators.

Definition 4.13. A modal semiring S satisfies the

1. *Tarski property* if for all $a \in S$ we have $a \neq 0 \Rightarrow \top \cdot a \cdot \top = \top$ (TAR);
2. *weak Tarski property* if for all $p \in \mathsf{test}(S)$ we have $p \neq 0 \Rightarrow \top \cdot p \cdot \top = \top$ (WT);
3. *weak Tarski domain property* if for all $p \in \mathsf{test}(S)$ we have $p \neq 0 \Rightarrow \lceil(\top \cdot p) = 1$ (WTD);
4. *weak Tarski codomain property* if for all $p \in \mathsf{test}(S)$ we have $p \neq 0 \Rightarrow (p \cdot \top)\rceil = 1$ (WTC).

It is clear that (TAR) implies (WT) and that (WT) implies both (WTD) and (WTC). The reverse implications do not hold. The path semiring is an example satisfying both (WTD) and (WTC) but not (WT). However, we have the following result.

Lemma 4.14. *In a modal semiring* (WTD) *and* (WTC) *are equivalent.*

Proof. We show (WTD) $\Rightarrow$ (WTC): by $(p \cdot \top)\rceil$ being the least right preserver of $p \cdot \top$, Boolean algebra, characterisation of box, $q < 1$ implies $\neg q \neq 0$ and hence by (WTD) we have $|\top]\, q = \neg\lceil(\top \cdot \neg q) = \neg 1 = 0$, and logic:

$$(p \cdot \top)\rceil = 1 \quad \Leftrightarrow \quad \forall q < 1 : p \cdot \top \cdot q < p \cdot \top \quad \Leftrightarrow \quad \forall q < 1 : p \cdot \top \cdot \neg q \neq 0$$
$$\Leftrightarrow \quad \forall q < 1 : p \not\leqslant |\top]\, q \quad \Leftrightarrow \quad \forall q < 1 : p \not\leqslant 0 \quad \Leftrightarrow \quad p \not\leqslant 0.$$

The reverse implication is symmetric. □

Because of this lemma we refer to both (WTD) and (WTC) uniformly as (WTM) ("M" standing for "modal").

Now we obtain the following representation of the order relation between non-empty rectangles.

Lemma 4.15. *Assume* (WTM). *If* $p, q, r, s \neq 0$ *then*

$$p \times q \leqslant r \times s \quad \Leftrightarrow \quad p \leqslant r \wedge q \leqslant s.$$

Proof. ($\Leftarrow$) Immediate from isotony of $\cdot$.

($\Rightarrow$) We have, by the definition of rectangles, the import/export property of domain and (WTM),

$$\lceil(p \times q) = \lceil(p \cdot \top \cdot q) = p \cdot \lceil(\top \cdot q) = p \cdot 1 = p.$$

Symmetrically, $(p \times q)\rceil = q$. Now the claim follows by isotony of domain and codomain. □

This enables a characterisation of maximal non-empty rectangles.

Lemma 4.16. *Assume* (WTM) *and* $p, q \neq 0$. *Then* $p \times q$ *is a maximal rectangle of* a *iff* (p, q) *is a concept of* a.

Proof. ($\Rightarrow$) By Corollary 4.11 and Lemma 4.10 $b =_{df} (|\bar{a}]\neg q) \times q$ is a rectangle of a with $p \leqslant |\bar{a}]\neg q$. Since $p \times q$ is assumed to be a rectangle of a, Lemma 4.10 and Lemma 4.15 yield $p \times q \leqslant b$. Now maximality of $p \times q$ and again Lemma 4.15 show $p = |\bar{a}]\neg q = g_a(q)$.

($\Leftarrow$) First, by the definition of a concept and Lemma 4.10, $p \times q$ is a rectangle of a. Let $r \times s \leqslant p \times q$ be another rectangle of a. Then by Lemmas 4.10 and 4.15 we have

$$p \leqslant |\bar{a}]\neg q \wedge r \leqslant |\bar{a}]\neg s \wedge p \leqslant r \wedge q \leqslant s.$$

By shunting we obtain $\neg s \leqslant \neg q$, and isotony of box in its second argument shows $r \leqslant |\bar{a}]\neg q$, so that $p \leqslant r \leqslant |\bar{a}]\neg q$. But since (p, q) is a concept we have $p = |\bar{a}]\neg q$ and hence also $p = r$. Symmetrically one obtains $q = s$. □

We conclude this section by another useful consequence of Lemma 4.10.

Lemma 4.17. *If* (WTM) *holds then*

$$q \neq 0 \wedge p \times q \leqslant a \quad \Rightarrow \quad p \leqslant |a\rangle q,$$
$$p \neq 0 \wedge p \times q \leqslant a \quad \Rightarrow \quad q \leqslant \langle a| p.$$

Proof. By $q \neq 0$, (WTM) and distributivity we have

$$1 = |\top\rangle q = |a + \bar{a}\rangle q = |a\rangle q + |\bar{a}\rangle q.$$

By this, distributivity and the shunted form $p \cdot |\bar{a}\rangle q \leqslant 0$ of Lemma 4.10.4,

$$p = p \cdot 1 = p \cdot |a\rangle q + p \cdot |\bar{a}\rangle q = p \cdot |a\rangle q,$$

which by $p \leqslant 1$ implies $p \leqslant |a\rangle q$. The second claim can be shown symmetrically. □

4.4. Pseudo-rectangles

It is a frequent task to find for a given element a coverage by formal concepts, i.e., by maximal rectangles. Since this is very expensive, in [15] the concept of a pseudo-rectangle is introduced, which can be determined in a much simpler way. Relationally, given a relation R, the pseudo-rectangle $\mathrm{P}(x, y, R)$ associated with a pair $(x, y) \in R$ is the union of all rectangles and hence of all maximal rectangles of R that contain (x, y). This is non-empty, since $\{(x, y)\}$ is a rectangle of R. We can give a general algebraic definition as follows.

Definition 4.18. We now assume that the underlying semiring S is a *quantale*, i.e., a complete lattice in which multiplication distributes over arbitrary suprema. Let $a \in S$ and $r \times s \leqslant a$ be a non-empty rectangle of a, i.e., assume $r, s \neq 0$. Then the set of a-rectangles covering $r \times s$ is

$$rect(r, s, a) =_{df} \{p \times q \mid r \times s \leqslant p \times q \leqslant a\},$$

and we call $\mathrm{P}(r, s, a) =_{df} \sum rect(r, s, a)$ the *pseudo-rectangle* induced by $r \times s$.

Lemma 4.19.

1. *If* (WTM) *holds then* $\mathrm{P}(r, s, a) \leqslant (|a\rangle s) \cdot a \cdot (\langle a|r)$.
2. *In the quantale of relations this strengthens to an equality provided* r, s *are atomic tests.*

Proof.

1. We show that $b =_{df} (|a\rangle s) \cdot a \cdot (\langle a|r)$ is an upper bound of $rect(r, s, a)$. Let $p \times q \in rect(r, s, a)$. By $r \times s \leqslant p \times q$ and Lemma 4.15 we obtain $r \leqslant p \wedge s \leqslant q$ so that by isotony also $r \times q \in rect(r, s, a)$ and $p \times s \in rect(r, s, a)$. Now, since $r \times s$ is non-empty, Lemma 4.17 implies $p \leqslant |a\rangle s \wedge q \leqslant \langle a|r$. Therefore, by $p \times q \leqslant a$, Lemma 4.9.1 and isotony,

$$p \times q = p \times q \sqcap a = p \cdot a \cdot q \leqslant |a\rangle s \cdot a \cdot \langle a|r = b.$$

2. We show that every pair in relation b lies also in every upper bound of $rect(r, s, a)$, so that b is indeed the least upper bound of $rect(r, s, a)$. Assume r, s to represent the elements u, v and that $(u, v) \in a$. For arbitrary pair (x, y) we obtain, by the definitions of b and relational composition as well as the definition of diamonds

$$\begin{aligned}(x, y) \in b &\Leftrightarrow x \in |a\rangle v \wedge y \in \langle a|u \wedge (x, y) \in a \\ &\Leftrightarrow (x, v) \in a \wedge (u, y) \in a \wedge (x, y) \in a.\end{aligned}$$

Together with $(u, v) \in a$ this implies that $c =_{df} \{u, x\} \times \{v, y\} \in rect(r, s, a)$ and hence $c \leqslant d$ for every upper bound d of $rect(r, s, a)$. By $(x, y) \in c$ this implies $(x, y) \in d$ as well. □

	a_1	a_2	a_3	a_4	a_5
o_1	1	1	0	0	0
o_2	1	1	0	0	1
o_3	1	1	1	1	0
o_4	1	0	0	0	1

(a) Relation a

	a_1	a_2	a_3	a_4	a_5
o_1	1	1	0	0	0
o_2	1	1	0	0	1
o_3	1	1	0	0	0
o_4	1	0	0	0	1

(b) Overapproximation $|a\rangle s \cdot a \cdot \langle a|r$

	a_1	a_2	a_3	a_4	a_5
o_1	1	1	0	0	0
o_2	1	1	0	0	0
o_3	1	1	0	0	0
o_4	0	0	0	0	0

(c) Pseudo-rectangle $\mathrm{P}(r, s, a)$

Fig. 1. Behaviour of pseudo-rectangles.

It is clear that computing $\mathrm{P}(r,s,a) \leqslant (|a\rangle s) \cdot a \cdot (\langle a|r)$ is less expensive than determining and summing up all maximal elements of $rect(r,s,a)$. The following example shows that the assumption of atomicity of r,s cannot be dropped from Part 2, not even in the relational case.

Example 4.20. Consider the relation a of Fig. 1(a). For $r =_{df} \{o_1,o_2\}$ and $s =_{df} \{a_1,a_2\}$ we obtain $|a\rangle s = \{o_1,o_2,o_3,o_4\}$ and $\langle a|r = \{a_1,a_2,a_5\}$. Hence $|a\rangle s \cdot a \cdot \langle a|r$ is the relation shown in Fig. 1(b). This is strictly larger than $\mathrm{P}(r,s,a)$ which is shown in Fig. 1(c). □

5. Pareto fronts and rectangles

A preference is a strict partial order on a given set, i.e., a special kind of a homogeneous binary relation. Queries in a database with a preference are supposed to return the maximal objects w.r.t. that preference, corresponding to optimal satisfaction in some sense of the user's wishes. By the so-called *Pareto operator* two preference orders a,b can be composed into the preference $a \otimes b$. The maxima set of $a \otimes b$ under a given set are *compromises* w.r.t. to the orders a,b in the following sense: In the maxima set there is no element which is strictly better in at least one of the preferences a,b while being not worse w.r.t. the other preference. Because of their characteristic shape, maxima sets of Pareto preferences are also called *Pareto fronts* or *skylines*. The Pareto principle is used to express that two (competing) objectives are equally important.

In the following we will derive a connection between these Pareto fronts and rectangles. This is based on our work in [4] and [16]. We recapitulate just some formal foundations while further definitions are introduced where needed.

Consider a set of type names. With each of these we associate some domain of data base tuples. A *type* is a set of type names. For type T let D_T be the cartesian product of the domains of the type names in T and let 1_T and $\top_T$ represent the identity and universal relations on D_T. A type assertion $a :: T^2$ is short for $a \leqslant 1_T \cdot a \cdot 1_T$. We view such elements as representations of preference relations on the domain D_T. An assertion $p :: T$ means that p is a test, representing a set of tuples, with $p \leqslant 1_T$.

The join operator $\bowtie$ (more precisely defined in the *Join Algebra* in [16]) acts quite similar to a Cartesian product. For elements $a :: T_a$ and $b :: T_b$ we have $a \bowtie b :: T_a \bowtie T_b$ where the latter is the union of T_a and T_b. Joins on the same type are equivalent to the intersection, i.e., $a_1 \bowtie a_2 = a_1 \sqcap a_2$ for $a_i :: T$. For the sake of readability we define $1_x =_{df} 1_{T_x}$ and $\top_x =_{df} \top_{T_x}$.

5.1. Idea

The set of elements which are maximal under a Pareto preference form a Pareto front when connected as shown in Fig. 2. This Pareto front subdivides the given domain into two areas: One describes the *dominance region* consisting of the maximal elements of the given dataset and the elements "dominated" by them. Elements in the other area are not dominated w.r.t the preference and the given dataset. We show that if there are N maximal elements the dominance region can be described by N rectangles and its complement by $N+1$ rectangles. Ordering these rectangles by size (or a weighted size with respect to a given probability distribution of elements) paves the way for a fast calculation on which side of the Pareto front a new element would be placed.

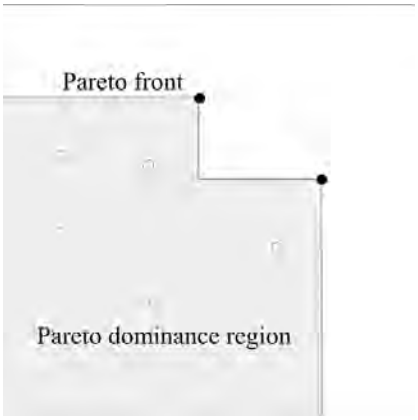


Fig. 2. The Pareto dominance region (gray) and the Pareto front (black line). The filled circles are the maxima and the unfilled circles are dominated objects.

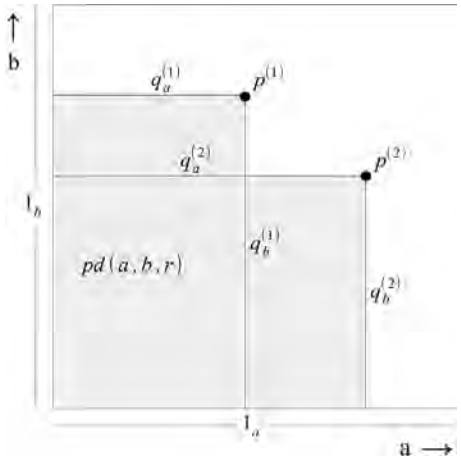


Fig. 3. Rectangular representation. The arrows indicate where elements are better w.r.t. the preferences a and b. The $p^{(i)}$ are points while the $q^{(i)}$ are areas.

5.2. Layered preferences

In this section we assume a Boolean modal semiring with converse $^\smile$ satisfying the axioms

$$p^\smile = p, \qquad (a+b)^\smile = a^\smile + b^\smile, \qquad (a\cdot b)^\smile = b^\smile \cdot a^\smile \quad \text{and} \quad \bar{a}^\smile = \overline{a^\smile}.$$

A *layered preference* (*or weak preorder*) is an irreflexive, transitive and negatively transitive element $a :: T_a^2$, i.e., satisfying, with $a' =_{df} \top_a \sqcap \bar{a}$,

$$1 \sqcap a = 0, \qquad a\cdot a \leqslant a, \qquad a'\cdot a' \leqslant a'.$$

For a layered preference a we define the *indifference* element $\mathsf{s}_a =_{df} a' \sqcap a'^\smile$. It is easy to show that s_a is an equivalence relation, i.e., reflexive, transitive and symmetric. Moreover, a is a linear strict-order, i.e., $a + a^\smile + \mathsf{s}_a = \top_a$.

5.3. Representing the Pareto front by rectangles

Let $a :: T_a^2, b :: T_b^2$ be layered preferences with disjoint types T_a, T_b and let $r :: T_a \bowtie T_b$ be a dataset. To ease reading we define the following notation for any test p and element x:

$$p_x = p \sqcap 1_x.$$

Thus p_x is the projection to the domain of x. With this we have $p = p_a \bowtie p_b$.

The maximal elements of the Pareto preference $a \otimes b$ in r are given by

$$(a\otimes b) \triangleright r =_{df} r - |a\otimes b\rangle r,$$

where the Pareto preference is defined by $a \otimes b =_{df} a \bowtie (b + \mathsf{s}_b) + (a + \mathsf{s}_a) \bowtie b$. Note that this coincidences with the *substitutable value* semantics of Pareto preferences defined in [16] (where the substitutability relations are $s_a = \mathsf{s}_a$ and $s_b = \mathsf{s}_b$). Note that if a is a total order then $\mathsf{s}_a = 1_a$. In general we have $1_x \leqslant \mathsf{s}_x$ for $x \in \{a, b\}$.

We define the *Pareto dominance region* by

$$pd(a,b,r) =_{df} \big|(a+\mathsf{s}_a)\bowtie(b+\mathsf{s}_b)\big\rangle\big((a\otimes b)\triangleright r\big).$$

Assume a maxima set of N atomic elements $p^{(1)}, \ldots, p^{(N)}$, i.e., that

$$(a\otimes b)\triangleright r = p^{(1)} + \cdots + p^{(N)}.$$

Then the Pareto dominance region is given by

$$pd(a,b,r) = q^{(1)} + \cdots + q^{(N)} \quad \text{with } q^{(i)} =_{df} \big|(a+\mathsf{s}_a)\bowtie(b+\mathsf{s}_b)\big\rangle p^{(i)}.$$

In Fig. 3 the $p^{(i)}$ and $q^{(i)}$ are shown.

Definition 5.1. A *joined rectangle* $r :: T_a \bowtie T_b$ is a test which can be written as a join, i.e., there are tests $r_a :: T_a$ and $r_b :: T_b$, such that $r = r_a \bowtie r_b$.

This coincidences with the usual definition of rectangles: Consider the following bijection, defined on atomic tests, where D_{T_a} and D_{T_b} are the domains of disjoint types T_a and T_b:

$$f : T_a \bowtie T_b \to D_{T_a} \times D_{T_b}, \qquad t_a \bowtie t_b \mapsto t_a \cdot \top \cdot t_b.$$

By defining $f(p) =_{df} \sum_{q\in \mathsf{At}(p)} f(q)$, where $\mathsf{At}(p)$ is the set of atomic tests $\leqslant p$, this mapping can be extended to arbitrary tests.

Corollary 5.2.

1. *Any element r, for which $f(r)$ is a rectangle in the sense of Definition 4.7, is a joined rectangle.*
2. *Atomic tests are joined rectangles.*

Thus, datasets in $T_a \bowtie T_b$ can considered as heterogeneous relations via f. Note that this interpretation coincidences with the concept of a "database relation": A dataset with n columns (attributes) can be considered as a heterogeneous n-ary relation between its attributes. From now on, we abbreviate "joined rectangle" to *rectangle*.

Since the $p^{(i)}$ are atomic and hence rectangles, also the $q^{(i)}$ are rectangles as justified by

$$\begin{aligned} q^{(i)} &= \big|(a+\mathsf{s}_a)\bowtie(b+\mathsf{s}_b)\big\rangle p^{(i)} = \big|(a+\mathsf{s}_a)\bowtie(b+\mathsf{s}_b)\big\rangle\big(p_a^{(i)}\bowtie p_b^{(i)}\big) \\ &= |a+\mathsf{s}_a\rangle p_a^{(i)} \bowtie |b+\mathsf{s}_b\rangle p_b^{(i)}. \end{aligned}$$

Therewith we have a representation of $pd(a,b,r)$ as a sum of N rectangles.

Next, we derive a representation of its complement with $(N+1)$ rectangles. We define the complement of the Pareto dominance region by

$$\overline{pd}(a,b,r) =_{df} \neg pd(a,b,r) = \mathsf{s}_a \bowtie \mathsf{s}_b - pd(a,b,r).$$

Note that "$\bowtie$" binds stronger than "$-$" and "$+$".

Our aim is to find a more compact representation of $\overline{pd}(a,b,r)$. To this end we first give the following lemma.

Lemma 5.3. *Let a, b be layered preferences. For the maxima set $(a\otimes b)\triangleright r = p^{(1)}+\cdots+p^{(N)}$, the $p^{(i)}$ can always be arranged such that for all $i\in\{1,\ldots,N-1\}$:*

1. (a) $p_a^{(i)}(\mathsf{s}_a+a)p_a^{(i+1)}$, (b) $p_b^{(i+1)}(\mathsf{s}_b+b)p_b^{(i)}$.
2. (c) $p^{(i)} \leqslant |\mathsf{s}_a+a\rangle p^{(i+1)}$, (d) $p^{(i+1)} \leqslant |\mathsf{s}_b+b\rangle p^{(i)}$.

Proof.

1. As a is a layered preference, the arrangement (a) is obviously possible. Next, we show that this implies (b). Since b is layered we have one of the following cases: (i) $p_b^{(i)}\ b\ p_b^{(i+1)}$, (ii) $p_b^{(i+1)}\ b\ p_b^{(i)}$, (iii) $p_b^{(i)}\ \mathsf{s}_b\ p_b^{(i+1)}$. Suppose case (i) $p_b^{(i)}\ b\ p_b^{(i+1)}$. Then $p^{(i)}\ ((\mathsf{s}_a+a)\bowtie b)\ p^{(i+1)}$ and hence $p^{(i)}\ (a\otimes b)\ p^{(i+1)}$, i.e., $p^{(i)}$ is dominated by $p^{(i+1)}$, a contradiction. Hence only the cases (ii) and (iii) are possible, which are compatible with (b).
2. Follows immediately from Part 1 and Eq. (4). □

Now we give a compact representation of $\overline{pd}(a,b,r)$:

Lemma 5.4. *The set $\overline{pd}(a,b,r)$ can be expressed as a sum of $(N+1)$ rectangles as follows (where $\neg$ binds stronger than $\bowtie$):*

$$\overline{pd}(a,b,r) = \neg q_a^{(1)} \bowtie \mathsf{s}_b + \neg q_a^{(2)} \bowtie \neg q_b^{(1)} + \cdots + \mathsf{s}_a \bowtie \neg q_b^{(N)}. \tag{7}$$

Proof. For convenience we set

$$q_a^{(0)} =_{df} \mathsf{s}_a, \qquad q_a^{(N+1)} =_{df} 0_a, \qquad q_b^{(0)} =_{df} 0_b, \qquad q_b^{(N+1)} =_{df} \mathsf{s}_b.$$

From Lemma 5.3 we conclude for $i\in\{1,\ldots,N-1\}$:

$$q_a^{(i+1)} = |a+\mathsf{s}_a\rangle p_a^{(i+1)} \leqslant |a+\mathsf{s}_a\rangle p_a^{(i)} = q_a^{(i)}.$$

Analogously $q_b^{(i)} \leqslant q_b^{(i+1)}$ and with the above conventions, $(q_a^{(i)})_{i=0,\ldots,N+1}$ is decreasing while $(q_b^{(i)})_{i=0,\ldots,N+1}$ is increasing in i. Due to this, the claim in Eq. (7) is equivalent to

$$\overline{pd}(a,b,r) = \sum_{i=0}^{N} \neg q_a^{(i)} \bowtie \neg q_b^{(i+1)}.$$

For this we show $pd(a,b,r)\cdot\overline{pd}(a,b,r) = 0$ and $pd(a,b,r)+\overline{pd}(a,b,r) = \mathsf{s}_a\bowtie\mathsf{s}_b$.

1. Remember that $pd(a,b,r) = \sum_{i=1}^{N} q^{(i)}$. We have to show that all summands in $pd(a,b,r) \cdot \overline{pd}(a,b,r)$ are 0. We conclude for all i, j:

$$q^{(i)} \cdot \left(\neg q_a^{(j)} \bowtie \neg q_a^{(j+1)}\right) = \underbrace{\left(q_a^{(i)} - q_a^{(j)}\right)}_{=:u_a} \bowtie \underbrace{\left(q_b^{(i)} - q_b^{(j+1)}\right)}_{=:u_b}.$$

   Now we have either $j \leqslant i-1$ which implies that $u_a = 0$, or we have $j \geqslant i$ which implies that $u_b = 0$. Hence all summands are 0.
2. We use the following decomposition of $\mathsf{s}_a \bowtie \mathsf{s}_b$:

$$\mathsf{s}_a \bowtie \mathsf{s}_b = \sum_{i,j=0}^{N} \underbrace{\left(q_a^{(i)} - q_a^{(i+1)}\right) \bowtie \left(q_b^{(j+1)} - q_b^{(j)}\right)}_{=:u_{i,j}}.$$

   Next, we show that any $u_{i,j}$ is contained either in $pd(a,b,r)$ or in $\overline{pd}(a,b,r)$. We distinguish two cases:
   (i) $j \leqslant i-1$: Because $q_b^{(i)}$ is increasing in i we have

$$u_{i,j} \leqslant q_a^{(i)} \bowtie q_b^{(j+1)} \leqslant q_a^{(i)} \bowtie q_b^{(i)} \leqslant pd(a,b,r).$$

   (ii) $j \geqslant i$: Because $\neg q_b^{(i)}$ is decreasing in i we have

$$u_{i,j} \leqslant \neg q_a^{(i+1)} \bowtie \neg q_b^{(j)} \leqslant \neg q_a^{(i+1)} \bowtie \neg q_b^{(i)} \leqslant \overline{pd}(a,b,r).$$

   Hence the sum of $pd(a,b,r)$ and $\overline{pd}(a,b,r)$ is the entire domain and the claim follows. □

Due to the above lemma we have a representation of $\overline{pd}(a,b,p)$ with $(N+1)$ rectangles.

5.4. Application

Assume a stream of *points*, i.e., atomic tests $t_a \bowtie t_b$. After receiving a point from the stream, the Pareto front has to be updated. We sketch how the above calculation will help to determine if a new point changes the Pareto front or not.

Assume a measure function $\mu : T_a \bowtie T_b \to [0,1]$ representing the probability in which area of $\mathsf{s}_a \bowtie \mathsf{s}_b$ points from the stream occur. An algorithm for deciding quickly if a new point is within pd or within $\overline{pd}$ should check the most probable rectangles w.r.t. μ first, i.e., we calculate $\mu(q^{(i)})$ for $i = 1, \ldots, N$ and $\mu(\neg q_a^{(i)} \bowtie \neg q_b^{(i+1)})$ for $i = 0, \ldots, N$. Then for a new point $t = t_a \bowtie t_b$ we check if $t \leqslant r^{(i)}$, where the sequence $(r^{(i)})$ enumerates the $2N+1$ rectangles of $\underline{pd}$ and $\overline{pd}$ in a μ-decreasing order. The algorithm terminates if $t \leqslant r^{(i)}$ is true which gives evidence whether t is in pd or $\overline{pd}$.

If a new point t is in $\overline{pd}$ and should be added to the dataset then the rectangles representing pd and $\overline{pd}$ have to be recalculated. Note that it suffices to recalculate only those rectangles $q^{(i)}$ where the index i belongs to the set

$$I = \left\{ i \in \{1, \ldots, N\} \mid p^{(i)} \leqslant |a \otimes b\rangle t \right\},$$

i.e., those rectangles are affected, where the corresponding points $p^{(i)}$ are dominated by t. For example, if $I = \{k\}$, then the three rectangles

$$q^{(k)}, \quad \neg q_a^{(k-1)} \bowtie \neg q_b^{(k)}, \quad \neg q_a^{(k)} \bowtie \neg q_b^{(k+1)}$$

have to be recalculated. Note that by transitivity of a and b and the definition of the Pareto preference, the set I is always an interval in $\mathbb{N}$, i.e., $\{l_1, l_1+1, \ldots, l_2\}$. If we have $|I| = k$, then $2k+1$ rectangles will be replaced by 3 new rectangles.

Such an algorithm for quickly deciding if a new point is dominated by the existing maxima set is of interest for an application where the current maxima set of a stream (generating constantly new points) should be always up-to-date in real time, i.e., the dominance test for new points is a time-critical task.

For real applications one might not have the probability measure μ available, but this can be roughly estimated by the points from the stream which are already known at a given time.

6. Separation logic, partial correctness and abortion

We now turn to reasoning about program resources in standard separation logic (SSL). SSL is an extension of Hoare logic and facilitates reasoning about concurrent programs and sharing in data structures. It enables, due to its popular *frame rule*, modular reasoning which allows scalable program proofs. It is well-known that propositional Hoare logic can be treated using modal Kleene algebras. In [2,17] a relation-algebraic treatment of SSL was presented. Starting from that approach we will show in the sequel that modal Kleene algebras can also be used to abstractly model separation logic both in a partial and a total correctness setting.

We first recapitulate the relational model of [17] to explain definitions of the algebra within that concrete model and to provide a better intuition. The relational model is built on a basic algebraic structure called *separation algebra* [18] that is used to abstractly capture resources of programs.

Definition 6.1. A *separation algebra* is a partial commutative monoid $(\Sigma, \bullet, u)$ where $\bullet$ is a commutative, associative and partial binary operation on Σ with unit u. An equation holds iff both sides are defined and equal, or both are undefined. The induced *combinability* relation # is given by

$$\sigma_0 \# \sigma_1 \quad \Leftrightarrow_{df} \quad \sigma_0 \bullet \sigma_1 \text{ is defined.}$$

Using this structure a *command* is a relation $C \subseteq \Sigma \times \Sigma$.

To model separation of commands we introduce as a next step special relations that enrich the setting with relations between pairs of states. By this all possible splits of a state w.r.t. # can be considered and thus an independent treatment of parts of a state is feasible.

Definition 6.2. Assume a separation algebra $(\Sigma, \bullet, u)$. The *split* relation $\lhd \subseteq \Sigma \times (\Sigma \times \Sigma)$ is given by

$$\sigma \lhd (\sigma_1, \sigma_2) \quad \Leftrightarrow_{df} \quad \sigma_1 \# \sigma_2 \wedge \sigma = \sigma_1 \bullet \sigma_2.$$

The *join* relation $\rhd$ is the converse of split, i.e., $\rhd = \lhd^\smile$ with $(\sigma_1, \sigma_2) \rhd \sigma \Leftrightarrow_{df} \sigma_1 \# \sigma_2 \wedge \sigma = \sigma_1 \bullet \sigma_2$. The *Cartesian product* $C \times D \subseteq (\Sigma \times \Sigma) \times (\Sigma \times \Sigma)$ of two commands C, D is defined by

$$(\sigma_1, \sigma_2)\,(C \times D)\,(\tau_1, \tau_2) \quad \Leftrightarrow_{df} \quad \sigma_1\, C\, \tau_1 \wedge \sigma_2\, D\, \tau_2.$$

Relation composition on Cartesian products is defined component-wise and hence we have the exchange law

$$(C_1 \times D_1)\,;\,(C_2 \times D_2) = C_1\,;\,C_2 \;\times\; D_1\,;\,D_2.$$

Finally, we can define $*$-composition on arbitrary commands C, D that allows their concurrent execution on combinable or disjoint parts of a state.

Definition 6.3. The *$*$-composition* of commands $C, D \subseteq \Sigma \times \Sigma$ is again a command defined by

$$C * D =_{df} \lhd\,;\,(C \times D)\,;\,\rhd.$$

By definition of the underlying separation algebra, also $*$ is associative, commutative and has unit $\{(u, u)\}$. We will abstract, in the following, relations to elements of a modal Kleene algebra S. In particular, we use pairs of elements $c, d \in S$ in $c * d = \lhd(c \times d)\rhd$ and omit $\cdot$ before and after the brackets for better readability. As an abstract counterpart of the above exchange law we state the additional axiom

$$(c_1 \times d_1) \cdot (c_2 \times d_2) = c_1 \cdot c_2 \;\times\; d_1 \cdot d_2. \tag{8}$$

Moreover, to characterise the interplay of $*$ with domain and codomain we assume validity of the following inequations for arbitrary c, d:

$$\ulcorner(c * d) \leqslant \ulcorner c * \ulcorner d \quad \text{and} \quad (c * d)\urcorner \leqslant c\urcorner * d\urcorner. \tag{9}$$

These laws are again abstract counterparts of valid relational variants. A proof in the relational model can be found in [17].

6.1. Characterising Hoare triples

With the given algebraic background we now start characterising Hoare triples of SSL with partial correctness semantics abstractly. In contrast to usual Hoare logic, the triples in SSL come with an extra safety condition which makes them resource-sensitive. The general idea of this is to distinguish program abortion, e.g., due to a lack of required resources, from non-termination.

Definition 6.4. A command starting from a state σ *aborts* iff $\sigma\ C\ \bot$ where $\bot \in \Sigma$ denotes a distinguished state [19]. For command C and assertions p, q the *SSL Hoare triple* $\{p\}\,C\,\{q\}$ for partial correctness holds iff for all states $\sigma \in p$ both

$$\neg(\sigma\ C\ \bot) \quad \text{and} \quad \sigma\ C\ \sigma' \quad \Rightarrow \quad \sigma' \in q \quad \text{hold.}$$

Conceptually a non-terminating program relationally coincides with $\emptyset$, i.e., considering a starting state σ no final state can be obtained due to non-termination. Program abortion is identified by $(\sigma, \bot) \in C$ which means that an execution starts from σ but eventually gets stuck.

Generally, in modal idempotent semirings the algebraic semantics of *Hoare triples* can be given by

$$\{p\}\, c\, \{q\} \quad \Leftrightarrow \quad p \leqslant |c]q \quad \Leftrightarrow \quad \langle c|p \leqslant q \quad \Leftrightarrow \quad p \cdot c \leqslant c \cdot q \quad \Leftrightarrow \quad p \cdot c \cdot \neg q \leqslant 0, \tag{10}$$

where assertions p, q can be realised using tests. This characterisation is too weak for SSL Hoare triples since the condition on program abortion is not considered. As before we start by modelling that condition in our presented relational approach and then turn to the abstract setting.

We introduce an extra command $\mathsf{abort} =_{df} \{(\bot, \bot)\}$ which is a test. To model basic commands in SSL we also define special commands D that respect abort, i.e., aborting executions in another command C will not be ruled out in $C\,;\,D$. Concrete examples are, e.g., the mutation, dereferencing or allocation commands in SSL [19].

Definition 6.5. A command c respects abort iff $\mathsf{abort} \cdot c = \mathsf{abort}$.

This corresponds to relations C that satisfy $\bot\, C\, \sigma \Rightarrow \sigma = \bot$. For such commands C the test $\bot$ is a left-annihilator. For treating $*$-compositions we need to extend the separation algebra operation $\bullet$ to also capture $\bot$ by

$$\sigma \bullet \tau = \bot \quad \Leftrightarrow \quad \sigma = \bot \vee \tau = \bot.$$

Hence $(\sigma, \tau) \rhd \bot \Leftrightarrow \sigma = \bot \vee \tau = \bot$. By this, we can infer useful laws in the relational model which we further abstract to the algebra.

Lemma 6.6. *For arbitrary c, d we have*

$$\begin{aligned} (c * d) \cdot \neg \mathsf{abort} &= (c \cdot \neg \mathsf{abort}) * (d \cdot \neg \mathsf{abort}), \\ (c * d) \cdot \mathsf{abort} &= (c \cdot \mathsf{abort}) * d + c * (d \cdot \mathsf{abort}) + (c \cdot \mathsf{abort}) * (d \cdot \mathsf{abort}), \\ c * \mathsf{abort} &= \mathsf{abort}. \end{aligned}$$

We assume these laws for the algebraic treatment in the following.

Lemma 6.7. abort*-respecting commands are closed under* $+, \cdot$ *and* $*$.

Now we are ready to characterise the safety condition in the Hoare triples of SSL, i.e., a state σ is safe w.r.t. a command C iff $\neg(\sigma\, C\, \bot)$ holds. In a point-free fashion, this can be formalised by $p \cdot \ulcorner(C \cdot \mathsf{abort}) \leqslant 0$. The test p includes at most those states from which C will not abort. Moreover the inequation is equivalent to $p \leqslant |C]\neg\, \mathsf{abort}$. Hence we can now give a pointfree characterisation of the set of safe states using the modal box operator. For better readability we abbreviate $\widehat{p} =_{df} p \cdot \neg\, \mathsf{abort}$ in the following. By this, we have $\widehat{1} = \neg\, \mathsf{abort}$ and the special case of Lemma 6.6: $\widehat{p * q} = \widehat{p} * \widehat{q}$.

Definition 6.8. By $\mathsf{safe}(c) =_{df} |c]\widehat{1}$, we characterise all *safe* states of an element c. We call a test p *safe for* c iff $p \leqslant \mathsf{safe}(c)$.

Note that the element $|c\rangle\widehat{1}$ would not be adequate to characterise safe states although it contains all starting states that will not end in $\bot$. To see this, consider as a simple example the relation $c = \{(\sigma, \bot), (\sigma, \tau)\}$. Clearly, we have $(\sigma, \sigma) \in |c\rangle\widehat{1}$, but still σ can lead to program abortion. Thus we need the box operator to state that all execution paths of c do not abort.

For a characterisation of SSL Hoare triples we use that $p \leqslant [c|\,\widehat{1}$ is equivalent to $|c\rangle p \leqslant \widehat{1}$. Hence, by a property of diamond we can immediately infer $\langle c|p \leqslant q \wedge \langle c|p \leqslant \widehat{1} \Leftrightarrow \langle c|p \leqslant \widehat{q}$. This form is still not fully adequate for our purposes due the asymmetry in excluding abort only in the assertion q. This asymmetry will falsify validity of the Hoare logic while inference rule. Thus, we define

Definition 6.9. A *partial correctness* Hoare triple in SSL is given by

$$\{p\}\, c\, \{q\} \quad \Leftrightarrow_{df} \quad \langle c|\widehat{p} \leqslant \widehat{q}.$$

Theorem 6.10. *All partial correctness inference rules of propositional Hoare logic remain valid under the partial correctness interpretation of Hoare triples with* abort.

A proof can be given analogously to the proof using the standard interpretation of Hoare triples without abortion in a modal Kleene algebra [9].

Another possibility for Definition 6.9 would be to use $\langle c|p \leqslant q \wedge q \leqslant \widehat{1}$ which implies the above condition and therefore is stronger. We will stay with the above definition since it is more compact and simpler to use.

One advantage of the above encoding of the Hoare triples is that they also imply that the test p involved always only characterises safe states.

Lemma 6.11. *$\{p\}\,c\,\{q\}$ implies $\widehat{p}$ is safe for c.*

Proof. We have, by Galois connection, isotony of box in its second argument, definition of safe(_),

$$\langle c|\widehat{p} \leqslant \widehat{q} \quad \Leftrightarrow \quad \widehat{p} \leqslant |c]\widehat{q} \quad \Rightarrow \quad \widehat{p} \leqslant |c]\widehat{1} \quad \Leftrightarrow \quad \widehat{p} \leqslant \mathsf{safe}(c). \qquad \square$$

This yields an abstract definition of the frame property [17] in a partial correctness setting. The frame property is an additional condition on commands in SSL to obtain validity of the prominent frame rule.

6.2. A simple proof for the frame rule

We start by giving an algebraic variant of the so-called *frame property*. Commands satisfying this property only depend on a certain set of resources, i.e., part of a state for a safe execution.

Definition 6.12. An element c has the *frame-property* iff

$$\big(\mathsf{safe}(c) \times 1\big)\ \rhd \cdot c \leqslant (c \times 1)\ \rhd .$$

In earlier work [2] this definition was used in combination with so-called *compensator* relations instead of the abstract identity relation 1. Such relations were used to model aliasing effects, e.g., on the set of shared variables. Like in [18], we follow a simplified approach that works with so-called *variable-as-resource* separation algebras. They do not require the usage of the above mentioned relations.

Using the above pointfree variant of the frame property with the modal encoding of Hoare triples, we can give an algebraic validity proof of the frame rule without requiring any further assumptions like safety monotonicity or any preservation property. We start by a localisation property that holds for elements c satisfying the frame property.

Lemma 6.13. *Assume c has the frame property. If $\widehat{p}$ is safe for c then*

$$\langle c|\widehat{p * r} \leqslant \big(\langle c|\widehat{p}\big) * \widehat{r}.$$

Proof. We have, by Lemma 6.6, definition of $*$ and backward diamond, Lemma 6.11 and (8), frame property, (8) and definition of $*$, (9) and since r is a test $\overline{r}^{\urcorner} = r$, definition of backward diamond,

$$\begin{aligned}\langle c|\widehat{p * r} = \langle c|(\widehat{p} * \widehat{r}) &= \big(\lhd(\widehat{p} \times \widehat{r})\ \rhd \cdot c\big)^{\urcorner} = \big(\lhd(\widehat{p} \times \widehat{r}) \cdot \big(\mathsf{safe}\,(c) \times 1\big)\ \rhd \cdot c\big)^{\urcorner} \leqslant \big(\lhd(\widehat{p} \times \widehat{r}) \cdot (c \times 1)\rhd\big)^{\urcorner} \\ &= \big((\widehat{p} \cdot c) * \widehat{r}\big)^{\urcorner} \leqslant (\widehat{p} \cdot c)^{\urcorner} * \widehat{r} = \big(\langle c|\widehat{p}\big) * \widehat{r}. \qquad \square\end{aligned}$$

Theorem 6.14. *If c has the frame property then the frame rule $\{p\}\,c\,\{q\} \Rightarrow \{p * r\}\,c\,\{q * r\}$ holds w.r.t. partial correctness.*

Proof. We have, by Lemma 6.13, assumption $\{p\}\,c\,\{q\}$, Lemma 6.6,

$$\langle c|\widehat{p * r} \leqslant \big(\langle c|\widehat{p}\big) * \widehat{r} \leqslant \widehat{q} * \widehat{r} = \widehat{q * r}. \qquad \square$$

Finally, we turn to the case of total correctness as in [2]. The semantics in this approach is: A state σ only belongs to the domain of a command iff there exists an execution starting from σ that does not abort and terminates in some final state τ. Hence, program abortion and non-termination are identified. As above the side conditions can be built into the definition of the SSL Hoare triples.

Definition 6.15. We define a *total correctness* SSL Hoare triple by

$$\{p\}\,c\,\{q\} \quad \Leftrightarrow_{df} \quad \langle c|p \leqslant q \wedge p \leqslant {}^{\ulcorner}c.$$

Theorem 6.16. *All total correctness inference rules of standard Hoare logic remain valid under the total correctness interpretation of Hoare triples.*

Proof. We only prove the termination condition of the sequential composition rule $\{p\}\,c\,\{r\} \wedge \{r\}\,d\,\{q\} \Rightarrow \{p\}\,c \cdot d\,\{q\}$: by p being a test, property of domain, $\{p\}\,c\,\{r\}$ and isotony of domain, $\{r\}\,d\,\{q\}$ with (10) and isotony of domain, and (locality):

$$p \leqslant \ulcorner c \quad \Rightarrow \quad p \leqslant p \cdot \ulcorner c \quad \Leftrightarrow \quad p \leqslant \ulcorner(p \cdot c) \quad \Rightarrow \quad p \leqslant \ulcorner(c \cdot r) \quad \Rightarrow \quad p \leqslant \ulcorner(c \cdot \ulcorner d) \quad \Leftrightarrow \quad p \leqslant \ulcorner(c \cdot d).$$

Validity proofs for the remaining inference rules can easily be obtained. □

Unfortunately, for a proof of the frame rule in the total correctness case one would have to additionally require $\ulcorner c * 1 \leqslant \ulcorner c$ as a point-free variant of a property called *termination monotonicity* [19] in the literature. Intuitively, if c terminates starting from a state σ it also will terminate starting from any possibly larger state $\sigma \bullet \tau$ assuming $\sigma \# \tau$. This property is in particular needed to calculate the termination condition in the consequence.

7. Overriding in domain modules

Feature Oriented Software Development (e.g. [5]) has been established in computer science as a general programming paradigm that provides formalisms, methods, languages, and tools for building maintainable, customisable, and extensible *software product lines* (*SPLs*) [20]. An SPL is a collection of programs that share a common part, e.g., functionality or code fragments. To encode an SPL, one can use *variation points* (*VPs*) in the source code. A VP is a location in a program whose content, called a *fragment*, can vary among different members of the SPL. A prominent example of an SPL is the Linux kernel, where `ifdef` directives are used as VPs. In [21] a *Structured Document Algebra (SDA)* is used to algebraically describe modules that include VPs and their composition. An SDA module is a collection of fragments named by VPs, and composition of SDA modules constructs programs.

In the next section we briefly recapitulate SDA.

7.1. Structured document algebra

VPs and fragments. Let V denote a set of VPs at which fragments may be inserted and $F(V)$ be the set of *fragments* which may, among other things, contain VPs from V. Elements of $F(V)$ are denoted by $f_1, f_2, \ldots$. There are two special elements, a default fragment $\square$ and an error $\lightning$. An error signals an attempt to assign two or more non-default fragments to the same VP within one module. The addition, or supremum operator $+$ on fragments obeys the following rules:

$$\square + x = x, \qquad \lightning + x = \lightning, \qquad x + y = y + x,$$
$$x + x = x, \qquad f_i + f_j = \lightning \quad (i \neq j),$$

where $x \in \{\square, f_i, \lightning\}$. This structure forms a flat lattice with least element $\square$ and greatest element $\lightning$. By standard lattice theory $+$ is also associative and has $\square$ as its neutral element.

Modules. A *module* is a partial function $m : V \rightharpoonup F(V)$ with finite domain. VP v is *assigned* in m if $v \in dom(m)$, otherwise *unassigned* or *external*. Every assigned VP $v \in dom(m)$ has at least the default value $\square$ assigned to it.

Module addition. The main goal of feature oriented programming is to construct programs step by step using reusable modules. In the algebra this is done by the module addition $+$. Addition fuses two modules while maintaining uniqueness (and signaling an error upon a conflict). Desirable properties for $+$ are commutativity and associativity. Since modules are partial functions, modules can be combined if they agree on VPs common to their domains.

For module addition, $+$ on fragments is lifted to partial functions:

$$(m+n)(v) =_{df} \begin{cases} m(v) & \text{if } v \in dom(m) - dom(n), \\ n(v) & \text{if } v \in dom(n) - dom(m), \\ m(v) + n(v) & \text{if } v \in dom(m) \cap dom(n), \\ \text{undefined} & \text{if } v \notin dom(m) \cup dom(n). \end{cases}$$

If in the third case $m(v) \neq n(v)$ and $m(v), n(v) \neq \square$ then $(m+n)(v) = \lightning$, thus signaling an error.

The set of modules forms a commutative monoid under $+$.

Subtraction. For modules m and n the *subtraction* $m - n$ is defined as:

$$(m-n)(v) =_{df} \begin{cases} m(v) & \text{if } v \in (dom(m) - dom(n)), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Subtraction satisfies the following laws:

$$\begin{array}{rcll} dom(m-n) & = & dom(m)-dom(n), & m-\emptyset & = & m, \\ \emptyset-n & = & \emptyset, & m-m & = & \emptyset, \\ (m+n)-p & = & (m-p)+(n-p), & m-n & \subseteq & m, \\ m-(n+p) & = & (m-n)-p, & m\subseteq n & \Rightarrow & m-n=\emptyset. \end{array}$$

Overriding. To allow *overriding*, an operation | can be defined in terms of subtraction and addition. Module m overrides n, written $m\,|\,n$:

$$m\,|\,n = m + (n-m)$$

This replaces all assignments in n for which m also provides a value. | is associative and idempotent with neutral element $\emptyset$.

Modules m and n are called *compatible* if for all $v \in dom(m)\cap dom(n)$ we have $m(v)=n(v)$. The following properties are equivalent:

$$m \text{ and } n \text{ are compatible}, \qquad m\,|\,n = m+n, \qquad m\,|\,n = n\,|\,m.$$

With this, we have the following laws:

$$m\,|\,(n+p) = (m\,|\,n)+(m\,|\,p), \qquad \text{(left distributivity)}$$

$$(m+n)\,|\,p = m\,|\,(n\,|\,p)$$
when m and n are compatible, (sequentialisation)

$$(m+n)\,|\,p = n\,|\,p$$
when m and n are compatible and $m\,|\,n=n$, (absorption)

$$m\,|\,(n+p) = n+(m\,|\,p)$$
when $dom(m)\cap dom(n)=\emptyset$. (localisation)

7.2. Abstracting from SDA

The set M of modules, i.e., partial maps $m : V \rightharpoonup F(V)$, and $+$ and $-$, defined as in Section 7.1, form an algebraic structure $SDA =_{df} (M,+,-,0)$ which satisfies the following laws for all $l,m,n \in M$:

1. $(M,+,0)$ is an idempotent and commutative monoid.
2. $(l-m)-n = l-(m+n)$.
3. $(l+m)-n = (l-n)+(m-l)$. (right distributivity)
4. $0-l=0$. (left annihilator)
5. $l-0=l$.

To reason about that structure with predomain theory we will use an algebraic *module* [22] (not to be confused with the above SDA modules). This is a triple $(R,M,:)$ where : is an operation $R\times M\to M$, called *scalar product*, R is a ring and M is an Abelian group. Since we do not have a ring and a group, we will use a commutative and idempotent monoid together with a Boolean algebra.

Definition 7.1. A *mono module* is an algebra $(B,M,:)$ where $(M,+,0)$ is an idempotent and commutative monoid and $(B,+,\cdot,0,1,\neg)$ is a Boolean algebra in which 0 and 1 are the least and greatest element and $\cdot$ and $+$ denote meet and join. Note that 0 and $+$ are overloaded, like in classical modules or vector spaces. The scalar product : is a mapping $B\times M\to M$ satisfying for all $p,q\in B$ and $a,b\in M$:

$$(p+q):a = p:a+q:a, \tag{11}$$

$$p:(a+b) = p:a+p:b, \tag{12}$$

$$0:a = 0, \tag{13}$$

$$(p\cdot q):a = p:(q:a), \tag{14}$$

$$1:a = a, \tag{15}$$

$$p:0 = 0. \tag{16}$$

Lemma 7.2. *Define, as for idempotent semirings,* $l\leqslant m \Leftrightarrow_{df} l+m=m$.

1. *Restriction* : *is isotone in both arguments.*
2. $p:a\leqslant a$.

Proof.

1. Follows from distributivity.
2. With $p \leqslant 1$, Part 1 and Eq. (15) we have: $p \leqslant 1 \Rightarrow p:a \leqslant 1:a = a$. □

Lemma 7.3. *In a mono module Lemma 2.4 holds. Lemma 2.4.2 has the form*

$$p:(a \sqcap b) = p:a \sqcap b = p:a \sqcap p:b. \tag{17}$$

Proof.

1. Since we use a Boolean algebra there is nothing to prove for Part 1.
2. We start with the first equation. We have to show that $p:(a \sqcap b)$ is the greatest lower bound of $p:a$ and b. Since $p \leqslant 1$ we have $p:(a \sqcap b) \leqslant b$ and by isotony $p:(a \sqcap b) \leqslant p:a$. Therefore $p:(a \sqcap b)$ is a lower bound of $p:a$ and b. Now let c also be a lower bound of $p:a$ and b. Then $c \leqslant a \sqcap b$ since c is a lower bound of a by transitivity and Lemma 7.2.2. Moreover, $\neg p:c \leqslant \neg p:(p:a) = (\neg p \cdot p:)a = 0:a = 0$ by Eqs. (14) and (13). Hence, by Eqs. (15) and (11) we have $c = (p + \neg p):c = p:c + \neg p:c = p:c$. Therefore $c = p:c \leqslant p:(a \sqcap b)$, i.e.,$p:(a \sqcap b)$ is indeed the greatest lower bound of $p:a$ and b. For the second equation we use idempotence of B, Eq. (14) and the first equation:

$$\begin{aligned} p:(a \sqcap b) &= (p \cdot p):(a \sqcap b) = p:(p:a \sqcap b) = p:(p:a \sqcap b) = p:(b \sqcap p:a) \\ &= p:b \sqcap p:a = p:a \sqcap p:b. \qquad \square \end{aligned}$$

Lemma 7.4. *Let M, N be sets. The algebra* RMM $=_{df} (\mathcal{P}(M), \mathcal{P}(M \times N), :)$*, where $:$ is restriction, i.e.,$p:a = \{(x, y) \mid x \in p \wedge (x, y) \in a\}$, forms a mono module.*

The proof is straightforward.
We now extend mono modules with the predomain operator $\ulcorner : M \to B$.

Definition 7.5. A *predomain mono module* $(B, M, :, \ulcorner)$ is a mono module where $\ulcorner : M \to B$ fulfills the analogues of (d1) and (d2), cf. Definition 2.5:

$$a \leqslant \ulcorner a:a, \qquad \ulcorner(p:a) \leqslant p.$$

Lemma 7.6. RMM *is a predomain mono module with* $\ulcorner a = \{x \mid (x, y) \in a\}$.

Proof.

(d1) Assume $(x, y) \in a$. Then $x \in \ulcorner a$ and therefore $(x, y) \in \ulcorner a:a$.
(d2) Assume $x \in \ulcorner(p:a)$. Then $x \in p$ and $(x, y) \in a$ for some $y \in N$. □

Having done this preliminary work, we can use an RMM over binary functional relations $R \subseteq M \times N$, i.e., $R^\smile; R \subseteq \mathsf{id}(M)$, to reason about SDA.

Lemma 7.7. *SDA's subtraction $m - l$ of modules is equivalent to $\neg\ulcorner l:m$ in the corresponding* RMM.

Proof. We conclude

$$\begin{aligned} (x, y) \in m - l \quad &\Leftrightarrow \quad \neg\exists z : (x, z) \in l \wedge (x, y) \in m \\ &\Leftrightarrow \quad x \in \neg\ulcorner l \wedge (x, y) \in m \quad \Leftrightarrow \quad (x, y) \in \neg\ulcorner l:m. \qquad \square \end{aligned}$$

Now it is easy to verify that the SDA laws from the beginning of Section 7.2 also hold in RMM. Note that the sides change, e.g., *right* distributivity becomes *left* distributivity. Further important properties of predomain also hold in a predomain module in analogous form.

Lemma 7.8. *Assume a predomain mono module $(B, M, :, \ulcorner)$. Then for all $p \in B$ and $a, b \in M$:*

1. $a = 0 \Leftrightarrow \ulcorner a = 0$.
2. $\ulcorner a \leqslant p \Leftrightarrow a \leqslant p:a$.
3. $p \leqslant \neg\ulcorner a \Leftrightarrow p:a \leqslant 0$.

4. $a \leqslant b \Rightarrow \ulcorner a \leqslant \ulcorner b$.
5. $\ulcorner(a+b) = \ulcorner a + \ulcorner b$.
6. $\ulcorner p : a \leqslant p \cdot \ulcorner a$.

The proofs are similar to the ones for a predomain IL-semiring.

SDA's overriding operator $m \mid n$ can also be defined in a predomain mono module: $b \mid a =_{df} b + \neg\ulcorner b : a$. In [23] this operator, embedded in a Kleene algebra, is used to update links in pointer structures.

Lemma 7.9. *Assume a predomain mono module* $(B, M, :, \ulcorner)$. *Then for all* $p \in B$ *and* $a, b \in M$:

1. $0 \mid a = a$,
2. $b \leqslant b \mid a$,
3. $b = \ulcorner b : (b \mid a)$,
4. $\ulcorner(b \mid a) = \ulcorner b + \ulcorner a$,
5. $c \mid (a+b) = c \mid a + c \mid b$.

8. Conclusion

Our short tour through the various modal worlds ends here. We hope that the reader enjoyed the discovery ride, both through the novel results and/or views on the mentioned fields of application. We hope to have demonstrated that an algebraic treatment with modal structures allows a simple and unified presentation. Moreover this survey is intended to contribute a basis and an incentive for further case studies and applications along these lines.

Acknowledgements

This research was partially funded by the German Research Foundation (DFG) projects MO 690/9-1 AlgSep — *Algebraic Calculi for Separation Logic* and MO 690/7-2 FeatureFoundation.

References

[1] B. Möller, Modal knowledge and game semirings, Comput. J. 56 (2013) 53–69.
[2] H.-H. Dang, P. Höfner, B. Möller, Algebraic separation logic, J. Log. Algebr. Program. 80 (2011) 221–247.
[3] R. Glück, B. Möller, M. Sintzoff, A semiring approach to equivalences, bisimulations and control, in: R. Berghammer, A. Jaoua, B. Möller (Eds.), Relations and Kleene Algebra in Computer Science, in: Lect. Notes Comput. Sci., vol. 5827, Springer, 2009, pp. 134–149.
[4] B. Möller, P. Roocks, M. Endres, An algebraic calculus of database preferences, in: J. Gibbons, P. Nogueira (Eds.), Mathematics of Program Construction, in: Lect. Notes Comput. Sci., vol. 7342, Springer, Berlin, Heidelberg, 2012, pp. 241–262.
[5] D. Batory, S. O'Malley, The design and implementation of hierarchical software systems with reusable components, ACM Trans. Softw. Eng. Methodol. 1 (1992) 355–398.
[6] G. Schmidt, T. Ströhlein, Relations and Graphs: Discrete Mathematics for Computer Scientists, Springer, 1993.
[7] G. Schmidt, Relational Mathematics, Encycl. Math. Appl., vol. 132, Cambridge University Press, 2011.
[8] E. Manes, D. Benson, The inverse semigroup of a sum-ordered semiring, Semigroup Forum 31 (1985) 129–152.
[9] J. Desharnais, B. Möller, G. Struth, Kleene algebra with domain, ACM Trans. Comput. Log. 7 (2006) 798–833.
[10] M. Erné, J. Koslowski, A. Melton, G. Strecker, A primer on galois connections, in: S. Andima, R. Kopperman, P. Ram Misra, M. Ellen Rudin, A.R. Todd (Eds.), Papers on General Topology and Its Applications, 7th Summer Conf. Wisconsin, in: Annals New York Acad. Sci., vol. 704, pp. 103–125.
[11] B. Möller, G. Struth, Algebras of modal operators and partial correctness, Theor. Comput. Sci. 351 (2006) 221–239.
[12] B. Ganter, R. Wille, Formal Concept Analysis – Mathematical Foundations, Springer, 1999.
[13] X. Liu, W. Hong, J. Song, T. Zhang, Using formal concept analysis to visualize relationships of syndromes in traditional Chinese medicine, in: D. Zhang, M. Sonka (Eds.), Medical Biometrics, in: Lect. Notes Comput. Sci., vol. 6165, Springer, 2010, pp. 315–324.
[14] G. Schmidt, Rectangles, fringes, and inverses, in: R. Berghammer, B. Möller, G. Struth (Eds.), Relations and Kleene Algebra in Computer Science, in: Lect. Notes Comput. Sci., vol. 4988, Springer, 2008, pp. 352–366.
[15] S.A. Ismail, A. Jaoua, Incremental pseudo rectangular organization of information relative to a domain, in: W. Kahl, T.G. Griffin (Eds.), Relational and Algebraic Methods in Computer Science, in: Lect. Notes Comput. Sci., vol. 7560, Springer, 2012, pp. 264–277.
[16] B. Möller, P. Roocks, An algebra of layered complex preferences, in: W. Kahl, T. Griffin (Eds.), Relational and Algebraic Methods in Computer Science, in: Lect. Notes Comput. Sci., vol. 7560, Springer, Berlin, Heidelberg, 2012, pp. 294–309.
[17] H.-H. Dang, B. Möller, Reverse exchange for concurrency and local reasoning, in: J. Gibbons, P. Nogueira (Eds.), MPC, in: Lect. Notes Comput. Sci., vol. 7342, Springer, 2012, pp. 177–197.
[18] C. Calcagno, P.-W. O'Hearn, H. Yang, Local action and abstract separation logic, in: Proc. of the 22nd Symposium on Logic in Computer Science, IEEE Press, 2007, pp. 366–378.
[19] H. Yang, P. O'Hearn, A semantic basis for local reasoning, in: M. Nielsen, U. Engberg (Eds.), Foundations of Software Science and Computation Structures, Proc. FOSSACS 2002, in: Lect. Notes Comput. Sci., vol. 2303, Springer, 2002, pp. 402–416.
[20] R. Lopez-Herrejon, D. Batory, A standard problem for evaluating product-line methodologies, in: J. Bosch (Ed.), GCSE '01: Generative and Component-Based Software Engineering, in: Lect. Notes Comput. Sci., vol. 2186, Springer, 2001, pp. 10–24.
[21] D. Batory, P. Höfner, B. Möller, A. Zelend, Features, modularity, and variation points, Technical Report CS-TR-13-14, The University of Texas at Austin, 2013.
[22] N. Jacobson, Basic Algebra, vols. I, II, Freeman, New York, 1985.
[23] T. Ehm, The Kleene algebra of nested pointer structures: theory and applications, Ph.D. thesis, Universität Augsburg, 2005.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 154–168

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Relational style laws and constructs of linear algebra

Jules Desharnais [a,*], Anastasiya Grinenko [a,**], Bernhard Möller [b,**]

[a] Département d'informatique et de génie logiciel, Université Laval, Québec, QC, Canada
[b] Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany

ARTICLE INFO

Article history:
Available online 12 February 2014

Keywords:
Linear algebra
Relation algebra
0–1 matrices
Column-sum operator
Row-sum operator
Direct sums
Direct products
Kronecker product

ABSTRACT

We present a few laws of linear algebra inspired by laws of relation algebra. The linear algebra laws are obtained from the relational ones by replacing union, intersection, composition and converse by the linear algebra operators of addition, Hadamard product, composition and transposition. Many of the modified expressions hold directly or with minor alterations.

We also define operators that sum up the content of rows and columns. These share many properties with the relational domain and codomain operators returning a subidentity corresponding to the domain and codomain of a relation. Finally, we use the linear algebra operators to write axioms defining direct sums and direct products and we show that there are other solutions in addition to the traditional – in the relational context – injection and projection relations.

1. Introduction

This paper presents a collection of laws of linear algebra that are similar to corresponding laws of relation algebra. The starting point is the observation that matrices with 0, 1 entries only are relations. Let Q and R be such matrices. Then their Hadamard product $Q \cdot R$, i.e., their entrywise arithmetic multiplication, is their intersection. The standard addition $Q + R$ and composition (multiplication) $Q\,R$ are not quite the union and relational composition, but they are not so far from that. Transpose R^{T} and conjugate transpose $R^{\dagger}$ are exactly the converse of R, where, for a matrix A with complex numbers as entries, $(A^{\dagger})_{i,j} = (A_{j,i})^{\dagger}$, with $(x + y\mathrm{i})^{\dagger} = x - y\mathrm{i}$. Our aim is to study what happens when the relational operators of a relational law are replaced by the linear algebra operators, and what happens when arbitrary matrices are used instead of relations.

Our purpose is to augment the repertoire of point-free laws of linear algebra, an endeavour in the spirit of the work of Macedo and Oliveira [1–3]. Some, if not most, of these laws are already known, but nevertheless we feel the "relational twist" is worth exploring.

Section 2 presents the notation and some basic laws. Section 3 introduces domain-like operators. Sections 4 and 5 are about direct sums and direct products; in both cases, the linear algebra setting yields additional solutions compared with the relational setting; these additional solutions are obtained by composing the relational solutions with unitary

* Principal corresponding author.
** Corresponding authors.
E-mail addresses: jules.desharnais@ift.ulaval.ca (J. Desharnais), anastasiya.grinenko.1@ulaval.ca (A. Grinenko), bernhard.moeller@informatik.uni-augsburg.de (B. Möller).

http://dx.doi.org/10.1016/j.jlap.2014.02.005
2352-2208/

transformations. We conclude in Section 6. We assume knowledge of the relational material that is used below, which can be found in [4,5]. There are numerous textbooks on linear algebra; see, e.g., [6].

The paper is an extension of [7], with additional results, proofs and examples, especially in Section 5 on direct products.

2. Basic laws

We consider finite matrices over the complex numbers. In the sequel, the term *relations* refers to matrices with 0, 1 entries only. Variables A, B, C denote arbitrary matrices, D a diagonal matrix, V a column vector and P, Q, R relations. Matrix *composition* is denoted by juxtaposition, as is customary in linear algebra. The other operators are arithmetic multiplication $\times$, matrix addition $+$, Hadamard product $\cdot$ (entrywise multiplication $(A \cdot B)_{i,j} = A_{i,j} \times B_{i,j}$), transpose $^{\mathsf{T}}$, conjugate transpose †, entrywise conjugation $A^{\ddagger}$ (i.e., $(A^{\ddagger})_{i,j} = (A_{i,j})^{\dagger}$), identity matrix $\mathbb{I}$ and zero matrix $\mathbf{0}$ ($\mathbf{0}_{i,j} = 0$ for all i, j). For relations, they are union $\cup$, intersection $\cap$, composition $;$, converse $^{\smile}$ and universal relation $\mathbb{T}$ ($\mathbb{T}_{i,j} = 1$ for all i, j). The size of a matrix with m rows and n columns is indicated by $m \leftrightarrow n$, occasionally as a subscript. The unary operators have precedence over the binary ones. The order of increasing precedence for the binary operators is $(+, \cup)$, $(\cdot, \cap)$, (composition, $;$). The ordering on relations is denoted by $\subseteq$ and the pointwise ordering on real matrices by $\leqslant$, i.e., $A \leqslant B \Leftrightarrow (\forall i, j \mid A_{i,j} \leqslant B_{i,j})$.

Some laws satisfied by these operators follow.

$$
\begin{array}{lllll}
\text{(a)} & A \cdot B = B \cdot A & & & \\
\text{(b)} & A^{\dagger} = A^{\mathsf{T}\ddagger} = A^{\ddagger\mathsf{T}} & A^{\mathsf{T}} = A^{\dagger\ddagger} = A^{\ddagger\dagger} & A^{\ddagger} = A^{\dagger\mathsf{T}} = A^{\mathsf{T}\dagger} \\
\text{(c)} & (A \cdot B)^{\mathsf{T}} = A^{\mathsf{T}} \cdot B^{\mathsf{T}} & (A \cdot B)^{\dagger} = A^{\dagger} \cdot B^{\dagger} & (A \cdot B)^{\ddagger} = A^{\ddagger} \cdot B^{\ddagger} \\
\text{(d)} & (AB)^{\mathsf{T}} = B^{\mathsf{T}} A^{\mathsf{T}} & (AB)^{\dagger} = B^{\dagger} A^{\dagger} & (AB)^{\ddagger} = A^{\ddagger} B^{\ddagger} \\
\text{(e)} & A^{\mathsf{T}\mathsf{T}} = A^{\dagger\dagger} = A & & \\
\text{(f)} & \mathbb{I}^{\mathsf{T}} = \mathbb{I} & (\mathbb{T}_{m \leftrightarrow n})^{\mathsf{T}} = \mathbb{T}_{n \leftrightarrow m} &
\end{array}
\tag{1}
$$

Using the Hadamard product we can characterise relations algebraically as the set of matrices A satisfying $A \cdot A = A$. For a relation R, $R^{\smile} = R^{\mathsf{T}} = R^{\dagger}$.

The universal relation $\mathbb{T}$ is the neutral element of the Hadamard product, i.e., $A \cdot \mathbb{T} = A$. As is customary in the relational setting, the same symbol $\mathbb{T}$ may denote matrices of different size (and similarly for $\mathbf{0}$ and $\mathbb{I}$).

Using matrix composition on relations rather than relational composition gives a more "quantitative" result. Indeed, $(Q\,R)_{i,j}$ is the number of paths from i to j by following Q and then R, rather than simply indicating whether there is a path or not. In particular, all entries of the matrix $\mathbb{T}_{l \leftrightarrow m} \mathbb{T}_{m \leftrightarrow n}$ are m, the size of the intermediate set (rows for the first matrix, columns for the second):

$$
\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{bmatrix}.
$$

Relations in combination with the Hadamard product can be used to impose "shapes" on arbitrary matrices. For an arbitrary matrix A and a relation R, we say that A has *shape* R iff $A \cdot R = A$. By the Hadamard characterisation of relations, every relation then has its own shape. For a further instance, if R is univalent, then $A \cdot R$ is a matrix with at most one non-zero entry in each row; thus, A has at most one non-zero entry in each row iff it has shape R for some univalent relation R. Instead of univalent relations, one may use equivalence relations, difunctional relations, symmetric relations, etc. to impose shapes. The shape of a matrix is not unique: if A has shape Q and $Q \subseteq R$, then A also has shape R.

We define a *subshape* relation $\sqsubseteq$ by

$$
A \sqsubseteq B \;\Leftrightarrow\; (\exists \text{ relation } R \mid A = B \cdot R). \tag{2}
$$

Intuitively, A results from B by replacing some entries in B by 0.

Proposition 1.

1. *Every matrix has shape* $\mathbb{T}$*, while only* $\mathbf{0}$ *has shape* $\mathbf{0}$*.*
2. *If* B *has shape* R *and* A *is arbitrary then* $A \cdot B$ *has shape* R *as well.*
3. *If* A *and* B *have shape* R *then* $A + B$ *has shape* R *as well.*
4. *The set of matrices of shape* R *forms an ideal in the ring of all matrices under* $+$ *and* $\cdot$*.*
5. *If* A *has shape* R *and* B *has shape* S *then* $A \cdot B$ *has shape* $R \cdot S$*.*
6. $\sqsubseteq$ *is a partial order.*
7. *Pointwise,* $\sqsubseteq$ *can be formulated as follows:* $A \sqsubseteq B \;\Leftrightarrow\; (\forall i, j \mid A_{i,j} = 0 \vee A_{i,j} = B_{i,j})$.

Proof.

1. Immediate from the definitions.
2. By associativity of $\cdot$ and the definition of shape, $(A \cdot B) \cdot R = A \cdot (B \cdot R) = A \cdot B$.

3. By distributivity of $\cdot$ over $+$ and the definition of shape, $(A+B)\cdot R = A\cdot R + B\cdot R = A + B$.
4. Immediate from Parts 2 and 3.
5. By associativity and commutativity of $\cdot$ and the definition of shape,

$$(A\cdot B)\cdot(R\cdot S) = (A\cdot R)\cdot(B\cdot S) = A\cdot B.$$

6. • Reflexivity: $A = A\cdot\mathbb{T}$.
   • Antisymmetry: Assume $A\sqsubseteq B$ and $B\sqsubseteq A$, say $A = B\cdot R$ and $B = A\cdot S$ for some relations R, S. Then $A = B\cdot R = A\cdot S\cdot R = B\cdot R\cdot S\cdot R = B\cdot R\cdot S = A\cdot S = B$.
   • Transitivity: Assume $A\sqsubseteq B$ and $B\sqsubseteq C$, say $A = B\cdot R$ and $B = C\cdot S$ for some relations R, S. Then $A = C\cdot S\cdot R$ and $S\cdot R$ is a relation as well, so that $A\sqsubseteq C$.
7. • ($\Rightarrow$) Assume $A = B\cdot R$. If $R_{i,j} = 0$ then $A_{i,j} = 0$ as well and we are done. Otherwise, $R_{i,j} = 1$ and hence $A_{i,j} = B_{i,j}$ as claimed.
   • ($\Leftarrow$) Define R by $R_{i,j} = 0$ if $A_{i,j} = 0$ and $R_{i,j} = 1$ otherwise. Then an easy calculation shows $A_{i,j} = B_{i,j}\times R_{i,j}$. □

A square matrix A is *diagonal* iff it has shape $\mathbb{I}$. A relation R is *univalent* iff $R^\dagger R$ is diagonal; the entry $(R^\dagger R)_{j,j}$ is the number of rows i such that iRj, which gives a measure of the degree of non-injectivity. For instance,

$$\begin{bmatrix}1&0&0\\1&0&0\\0&1&0\end{bmatrix}^\dagger\begin{bmatrix}1&0&0\\1&0&0\\0&1&0\end{bmatrix} = \begin{bmatrix}1&1&0\\0&0&1\\0&0&0\end{bmatrix}\begin{bmatrix}1&0&0\\1&0&0\\0&1&0\end{bmatrix} = \begin{bmatrix}2&0&0\\0&1&0\\0&0&0\end{bmatrix}.$$

If R is an equivalence relation, then RR has shape R. The entries in the "blocks" of RR contain the number of elements in the corresponding equivalence class. For example,

$$\begin{bmatrix}1&1&0\\1&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&1&0\\1&1&0\\0&0&1\end{bmatrix} = \begin{bmatrix}2&2&0\\2&2&0\\0&0&1\end{bmatrix}.$$

Let us say that matrix A is *unitarget* (*unisource*) iff it has shape R for some univalent (injective) relation R. A univalent (injective) relation is of course unitarget (unisource).

A diagonal matrix whose diagonal entries are all equal can represent a scalar. We thus say that a matrix D is a *scalar* iff $D = D\cdot\mathbb{I}$ and $D\mathbb{T} = \mathbb{T}D$ (equivalent definitions of scalars in relation algebra and Dedekind categories are given in [8–10]).

For instance,

$$\begin{bmatrix}3&0\\0&3\end{bmatrix} = \begin{bmatrix}3&0\\0&3\end{bmatrix}\cdot\begin{bmatrix}1&0\\0&1\end{bmatrix},$$

$$\begin{bmatrix}3&0\\0&3\end{bmatrix}\begin{bmatrix}1&1\\1&1\end{bmatrix} = \begin{bmatrix}3&3\\3&3\end{bmatrix} = \begin{bmatrix}1&1\\1&1\end{bmatrix}\begin{bmatrix}3&0\\0&3\end{bmatrix}.$$

Since $D\mathbb{T}$ and $\mathbb{T}D$ are then matrices whose entries are all equal, they could also be used to encode a scalar.

Various simple laws follow.

Proposition 2.

(a) *If D is diagonal, then $D\cdot A^\mathsf{T} = DA\cdot\mathbb{I}$.*
(b) *If D is diagonal, then $D = D^\mathsf{T}$.*
(c) *If D is diagonal, then $D = D\mathbb{T}\cdot\mathbb{I} = \mathbb{T}D\cdot\mathbb{I}$.*
(d) *If D is diagonal, then $A\cdot D = A^\mathsf{T}\cdot D$, with special cases $A\cdot\mathbb{I} = A^\mathsf{T}\cdot\mathbb{I}$, $A\mathbb{T}\cdot\mathbb{I} = \mathbb{T}A^\mathsf{T}\cdot\mathbb{I}$ and $A\mathbb{T}\cdot D = \mathbb{T}A^\mathsf{T}\cdot D$.*
(e) *$(A\mathbb{T}\cdot B)C = A\mathbb{T}\cdot BC$, with special case $(A\mathbb{T}\cdot\mathbb{I})C = A\mathbb{T}\cdot C$.*
$C(\mathbb{T}A\cdot B) = \mathbb{T}A\cdot CB$, with special case $C(\mathbb{T}A\cdot\mathbb{I}) = \mathbb{T}A\cdot C$.
(f) *$A(B\mathbb{T}\cdot C) = (A\cdot\mathbb{T}B^\mathsf{T})C$.*
(g) *If D_1 and D_2 are diagonal, then so is D_1D_2.*
(h) *If D_1 and D_2 are diagonal, then $D_1\cdot D_2 = D_1D_2$.*
(i) *If D is a scalar, then $DA = AD$ for all A.*
(j) *$\mathbb{T}A\mathbb{T}\cdot\mathbb{I}$ is a scalar. Note that $\mathbb{T}A\mathbb{T}$ is a matrix whose entries are all equal to the sum of the entries of A. Thus, $\mathbb{T}A\mathbb{T}\cdot\mathbb{I}$ is a scalar matrix representing the sum of the elements of A.*
(k) *$(A\cdot B)\mathbb{T} = (AB^\mathsf{T}\cdot\mathbb{I})\mathbb{T}$.*
$\mathbb{T}(A\cdot B) = \mathbb{T}(A^\mathsf{T}B\cdot\mathbb{I})$.
A consequence of these laws is $\mathbb{T}(A\cdot B)\mathbb{T} = \mathbb{T}(AB^\mathsf{T}\cdot\mathbb{I})\mathbb{T} = \mathbb{T}(A^\mathsf{T}B\cdot\mathbb{I})\mathbb{T}$, which says that the sum of the entries of $A\cdot B$ is the trace of AB^T and also the trace of $A^\mathsf{T}B$.
(l) *If R is a univalent relation, then*

$$R^\dagger(RA\cdot B) = A\cdot R^\dagger B \quad \textit{and} \quad \left(AR^\dagger\cdot B\right)R = A\cdot BR.$$

If R is an injective relation, then

$$R\big(R^\dagger A \cdot B\big) = A \cdot RB \quad \textit{and} \quad (AR \cdot B)R^\dagger = A \cdot BR^\dagger.$$

(m) *If A is unitarget, then $(A \cdot A)(B \cdot C) = AB \cdot AC$.*
If A is unisource, then $(B \cdot C)(A \cdot A) = BA \cdot CA$.
(n) *If R is a univalent relation, $R(B \cdot C) = RB \cdot RC$.*
If R is an injective relation, $(B \cdot C)R = BR \cdot CR$.
(o) *If Q is a univalent relation, then QR is a relation and $QR = Q\,;R$.*
If Q is an injective relation, then RQ is a relation and $RQ = R\,;Q$.
(p) *Let R be a relation and A be a real matrix. If $\mathbf{0} \leqslant A$, $\mathbf{0} \leqslant B$ or $A \leqslant \mathbf{0}$, $B \leqslant \mathbf{0}$, then $RA \cdot B \leqslant R(A \cdot R^\dagger B)$. If $\mathbf{0} \leqslant A$, $B \leqslant \mathbf{0}$ or $A \leqslant \mathbf{0}$, $\mathbf{0} \leqslant B$, then $R(A \cdot R^\dagger B) \leqslant RA \cdot B$. This is similar to the Dedekind rule for relations*: $R\,;P \cap Q \subseteq R\,;(P \cap R^\smile\,;Q)$.

Proof. When there are dual properties, we prove only the first one, the others following by symmetry.

(a)
$$\begin{aligned}
&(DA \cdot \mathbb{I})_{i,j}\\
=\ &(DA)_{i,j} \times \mathbb{I}_{i,j}\\
=\ &(\textstyle\sum k \mid D_{i,k} \times A_{k,j}) \times \mathbb{I}_{i,j}\\
=\ &\quad \langle \text{Since } D \text{ is diagonal, the terms with } k \neq i \text{ evaluate to } 0\rangle\\
&D_{i,i} \times A_{i,j} \times \mathbb{I}_{i,j}\\
=\ &\quad \langle \text{If } i \neq j, \text{ then } D_{i,i} \times A_{i,j} \times \mathbb{I}_{i,j} = 0 = D_{i,j} \times A_{j,i} \text{ since } \mathbb{I}_{i,j} = D_{i,j} = 0.\\
&\qquad \text{If } i = j, \text{ then } D_{i,i} \times A_{i,j} = D_{i,j} \times A_{j,i} \text{ and } \mathbb{I}_{i,j} = 1\rangle\\
&D_{i,j} \times A_{j,i}\\
=\ &D_{i,j} \times (A^\mathsf{T})_{i,j}\\
=\ &(D \cdot A^\mathsf{T})_{i,j}
\end{aligned}$$

(b)
$$\begin{aligned}
&D^\mathsf{T}\\
=\ &\quad \langle \text{By Part a with } D := \mathbb{I} \text{ and } A := D, \text{ and neutrality of } \mathbb{I} \text{ for composition, } \mathbb{I} \cdot D^\mathsf{T} = \mathbb{I}D \cdot \mathbb{I} = D\rangle\\
&(\mathbb{I} \cdot D^\mathsf{T})^\mathsf{T}\\
=\ &\quad \langle (1c, f, e)\rangle\\
&\mathbb{I} \cdot D\\
=\ &\quad \langle D \text{ is diagonal}\rangle\\
&D
\end{aligned}$$

(c)
$$\begin{aligned}
&D\mathbb{T} \cdot \mathbb{I}\\
=\ &\quad \langle \text{Part a with } A := \mathbb{T}\rangle\\
&D \cdot \mathbb{T}^\mathsf{T}\\
=\ &\quad \langle (1f)\rangle\\
&D \cdot \mathbb{T}\\
=\ &\quad \langle \text{Neutrality of } \mathbb{T} \text{ for } \cdot\rangle\\
&D
\end{aligned}$$

(d) We first prove the special case $A \cdot \mathbb{I} = A^\mathsf{T} \cdot \mathbb{I}$. By (1a), Part a with $D := \mathbb{I}$ and neutrality of $\mathbb{I}$ for composition, $A^\mathsf{T} \cdot \mathbb{I} = \mathbb{I} \cdot A^\mathsf{T} = \mathbb{I}A \cdot \mathbb{I} = A \cdot \mathbb{I}$. Using (1a) and the fact that D is diagonal, the general case $A \cdot D = A^\mathsf{T} \cdot D$ then follows from this result: $A \cdot D = A \cdot \mathbb{I} \cdot D = A^\mathsf{T} \cdot \mathbb{I} \cdot D = A^\mathsf{T} \cdot D$.

(e)
$$\begin{aligned}
&((A\mathbb{T} \cdot B)C)_{i,j}\\
=\ &(\textstyle\sum k \mid (A\mathbb{T})_{i,k} \times B_{i,k} \times C_{k,j})\\
=\ &\quad \langle (A\mathbb{T})_{i,k} = (A\mathbb{T})_{i,j} \text{ for all } k, \text{ and distributivity}\rangle\\
&(A\mathbb{T})_{i,j} \times (\textstyle\sum k \mid B_{i,k} \times C_{k,j})\\
=\ &(A\mathbb{T})_{i,j} \times (BC)_{i,j}\\
=\ &(A\mathbb{T} \cdot BC)_{i,j}
\end{aligned}$$

(f)
$$\begin{aligned}
&A(B\mathbb{T} \cdot C)\\
=\ &\quad \langle \text{Part e}\rangle\\
&A(B\mathbb{T} \cdot \mathbb{I})C\\
=\ &\quad \langle \text{Part d}\rangle\\
&A(\mathbb{T}B^\mathsf{T} \cdot \mathbb{I})C\\
=\ &\quad \langle \text{Part e and commutativity of } \cdot\ (1a)\rangle\\
&(A \cdot \mathbb{T}B^\mathsf{T})C
\end{aligned}$$

(g) $D_1 D_2 \cdot \mathbb{I}$
$= \quad \langle$By Parts c and e, $D_1 D_2 = (D_1 \mathbb{T} \cdot \mathbb{I}) D_2 = D_1 \mathbb{T} \cdot D_2 \rangle$
$D_1 \mathbb{T} \cdot D_2 \cdot \mathbb{I}$
$= \quad \langle D_2$ a diagonal$\rangle$
$D_1 \mathbb{T} \cdot D_2$
$= \quad \langle$As in the first step$\rangle$
$D_1 D_2$

(h) $D_1 \cdot D_2$
$= \quad \langle$Part b$\rangle$
$D_1 \cdot D_2^{\mathsf{T}}$
$= \quad \langle$Part a with $D := D_1$ and $A := D_2 \rangle$
$D_1 D_2 \cdot \mathbb{I}$
$= \quad \langle$Part g$\rangle$
$D_1 D_2$

(i) Using Parts c and e, the fact that $D\mathbb{T} = \mathbb{T}D$ because D is a scalar, and again Parts e and c, we have

$$DA = (D\mathbb{T} \cdot \mathbb{I})A = D\mathbb{T} \cdot A = \mathbb{T}D \cdot A = A(\mathbb{T}D \cdot \mathbb{I}) = AD.$$

(j) First, $(\mathbb{T}A\mathbb{T} \cdot \mathbb{I}) \cdot \mathbb{I} = \mathbb{T}A\mathbb{T} \cdot \mathbb{I}$. Second, $(\mathbb{T}A\mathbb{T} \cdot \mathbb{I})\mathbb{T} = \mathbb{T}A\mathbb{T} \cdot \mathbb{T} = \mathbb{T}(\mathbb{T}A\mathbb{T} \cdot \mathbb{I})$ by Part e.

(k) $((A \cdot B)\mathbb{T})_{i,j}$
$= (\sum k \mid (A \cdot B)_{i,k} \times \mathbb{T}_{k,j})$
$= \quad \langle \mathbb{T}_{k,j} = 1 \rangle$
$(\sum k \mid (A \cdot B)_{i,k})$
$= (\sum k \mid A_{i,k} \times B_{i,k})$
$= (\sum k \mid A_{i,k} \times (B^{\mathsf{T}})_{k,i})$
$= (AB^{\mathsf{T}})_{i,i}$
$= (\sum k \mid (AB^{\mathsf{T}} \cdot \mathbb{I})_{i,k})$
$= \quad \langle \mathbb{T}_{k,j} = 1 \rangle$
$(\sum k \mid (AB^{\mathsf{T}} \cdot \mathbb{I})_{i,k} \times \mathbb{T}_{k,j})$
$= ((AB^{\mathsf{T}} \cdot \mathbb{I})\mathbb{T})_{i,j}$

(l) Assume R is univalent.
$(R^{\dagger}(RA \cdot B))_{i,j}$
$= (\sum k \mid (R^{\dagger})_{i,k} \times (RA \cdot B)_{k,j})$
$= (\sum k \mid (R^{\dagger})_{i,k} \times (\sum l \mid R_{k,l} \times A_{l,j}) \times B_{k,j})$
$= \quad \langle l$ not free in $(R^{\dagger})_{i,k} \times B_{k,j}$, distributivity of $\times$ over $\sum \rangle$
$(\sum k, l \mid (R^{\dagger})_{i,k} \times R_{k,l} \times A_{l,j} \times B_{k,j})$
$= \quad \langle$Because R is univalent, $(R^{\dagger})_{i,k} \times R_{k,l} = R_{k,i} \times R_{k,l} = 0$ if $i \neq l$.
Otherwise, $(R^{\dagger})_{i,k} \times R_{k,l} = (R^{\dagger})_{i,k} \times R_{k,i} = (R^{\dagger})_{i,k}$, because R is a relation$\rangle$
$(\sum k \mid (R^{\dagger})_{i,k} \times A_{i,j} \times B_{k,j})$
$= A_{i,j} \times (\sum k \mid (R^{\dagger})_{i,k} \times B_{k,j})$
$= (A \cdot R^{\dagger} B)_{i,j}$

(m) Assume A is unitarget.
$((A \cdot A)(B \cdot C))_{i,j}$
$= (\sum k \mid (A \cdot A)_{i,k} \times (B \cdot C)_{k,j})$
$= (\sum k \mid A_{i,k} \times A_{i,k} \times B_{k,j} \times C_{k,j})$
$= \quad \langle$If $A_{i,k} = 0$ for all k, choose an arbitrary k_i; otherwise, let k_i be the unique k such that $A_{i,k} \neq 0 \rangle$
$A_{i,k_i} \times A_{i,k_i} \times B_{k_i,j} \times C_{k_i,j}$
$= (A_{i,k_i} \times B_{k_i,j}) \times (A_{i,k_i} \times C_{k_i,j})$
$= (\sum k \mid A_{i,k} \times B_{k,j}) \times (\sum k \mid A_{i,k} \times C_{k,j})$

$= (AB)_{i,j} \times (AC)_{i,j}$
$= (AB \cdot AC)_{i,j}$

(n) This follows from Part m and $R \cdot R = R$ since R is a relation.

(o) Assume Q is univalent. That $Q\,R$ is a relation follows from Part m and the hypothesis that Q and R are both relations:

$$Q\,R \cdot Q\,R = (Q \cdot Q)(R \cdot R) = Q\,R.$$

We now show that $Q\,R = Q\,;R$.

$(Q\,R)_{i,j}$
$= (\sum k \mid Q_{i,k} \times R_{k,j})$
$= \quad \langle$If $Q_{i,k} = 0$ for all k, choose an arbitrary k_i; otherwise, let k_i be the unique k such that $Q_{i,k} = 1\rangle$
$Q_{i,k_i} \times R_{k_i,j}$
$= \quad \langle$Using $\vee$ and $\wedge$ for the Boolean join and meet on $\{0,1\}\rangle$
$(\bigvee k \mid Q_{i,k} \wedge R_{k,j})$
$= (Q\,;R)_{i,j}$

(p) Assume $\mathbf{0} \leqslant A$, $\mathbf{0} \leqslant B$ or $A \leqslant \mathbf{0}$, $B \leqslant \mathbf{0}$.

$(R(A \cdot R^\dagger B))_{i,j}$
$= (\sum k \mid R_{i,k} \times (A \cdot R^\dagger B)_{k,j})$
$= (\sum k,l \mid R_{i,k} \times A_{k,j} \times (R^\dagger)_{k,l} \times B_{l,j})$
$\geqslant \quad \langle$By the assumption and because R is a relation, $R_{i,k} \times A_{k,j} \times (R^\dagger)_{k,l} \times B_{l,j} \geqslant 0$,
so only non-negative terms are dropped, and $R_{i,k} \times (R^\dagger)_{k,i} = R_{i,k}\rangle$
$(\sum k \mid R_{i,k} \times A_{k,j} \times B_{i,j})$
$= (RA \cdot B)_{i,j}$

When either $\mathbf{0} \leqslant A$, $B \leqslant \mathbf{0}$ or $A \leqslant \mathbf{0}$, $\mathbf{0} \leqslant B$, the proof is similar, except that this assumption reverses the inequality. □

If the linear operators in the laws of Proposition 2 are replaced by the corresponding relational ones (as described in the introduction), the relational laws that inspired them are easily recognised. We list them in the same order as in Proposition 2. In the case of dual properties, only the first one is given.

(a) If $t \subseteq \mathbb{I}$, then $t \cap R^\smile = t\,;R \cap \mathbb{I}$.
(b) If $t \subseteq \mathbb{I}$, then $t^\smile = t$.
(c) If $t \subseteq \mathbb{I}$, then $t = t\,;\mathbb{T} \cap \mathbb{I} = \mathbb{T}\,;t \cap \mathbb{I}$.
(d) If $t \subseteq \mathbb{I}$, then $R \cap t = R^\smile \cap t$ with special case $R\,;\mathbb{T} \cap t = \mathbb{T}\,;R^\smile \cap t$.
(e) $(P\,;\mathbb{T} \cap Q)\,;R = P\,;\mathbb{T} \cap Q\,;R$.
(f) $P\,;(Q\,;\mathbb{T} \cap R) = (P \cap \mathbb{T}\,;Q^\smile)\,;R$.
(g) If $s \subseteq \mathbb{I}$ and $t \subseteq \mathbb{I}$, then $s\,;t \subseteq \mathbb{I}$.
(h) If $s \subseteq \mathbb{I}$ and $t \subseteq \mathbb{I}$, then $s \cap t = s\,;t$.
(i) $\mathbf{0}\,;R = R\,;\mathbf{0}$ and $\mathbb{I}\,;R = R\,;\mathbb{I}$ ($\mathbf{0}$ and $\mathbb{I}$ are the only scalars).
(j) $\mathbb{T}\,;R\,;\mathbb{T} \cap \mathbb{I}$ is a scalar denoting the join of the elements of R.
(k) $(Q \cap R)\,;\mathbb{T} = (Q\,;R^\smile \cap \mathbb{I})\,;\mathbb{T}$.
(l) If R is univalent, $R^\smile\,;(R\,;P \cap Q) = P \cap R^\smile\,;Q$.
(m) If R is univalent, $(R \cap R)\,;(P \cap Q) = R\,;P \cap R\,;Q$.
(n) If R is univalent, $R\,;(P \cap Q) = R\,;P \cap R\,;Q$.
(o) The relational composition of relations is always a relation.
(p) Dedekind rule for relations: $R\,;P \cap Q \subseteq R\,;(P \cap R^\smile\,;Q)$.

3. Domain-like operators

As in relation algebra, the information content of a vector can be represented as a diagonal matrix. If vector V has type $n \leftrightarrow 1$, then the diagonal matrix $V\mathbb{T}_{1\leftrightarrow n} \cdot \mathbb{I}$ corresponds to V (its diagonal contains the same elements as V, in the same order). Given a diagonal matrix $D_{n\leftrightarrow n}$, the corresponding vector is $D\mathbb{T}_{n\leftrightarrow 1}$. A vector V of type $n \leftrightarrow 1$ is a *unit vector* iff $V^\dagger V = 1$ $(= \mathbb{T}_{1\leftrightarrow 1})$. Using the above correspondence between vectors and diagonal matrices and the fact that all entries of $\mathbb{T}A\mathbb{T}$ are equal to the sum of the elements of A, we say that a diagonal matrix D is a *unit diagonal matrix* iff $\mathbb{T}D^\dagger D\mathbb{T} = \mathbb{T}$, which is equivalent to $\mathbb{T}(D^\dagger \cdot D)\mathbb{T} = \mathbb{T}$ by Proposition 2(h).

A common operation in linear algebra is the multiplication of a matrix A by a vector V, giving the vector AV as a result. The dual operation $V^\dagger A$ is also frequent. In order to carry out the same operations at the level of diagonal matrices, we

introduce two operators, the *row-sum operator* $\vec{\Sigma}A$ which sums up the content of the rows of A and the *column-sum operator* $A^{\overleftarrow{\Sigma}}$ which sums up the content of the columns of A. They are defined by

$$\vec{\Sigma}A = A\mathbb{T}\cdot\mathbb{I}, \qquad A^{\overleftarrow{\Sigma}} = \mathbb{T}A\cdot\mathbb{I}. \tag{3}$$

A simple example explains how the operators work:

$$\vec{\Sigma}\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a+b & 0 \\ 0 & c+d \end{bmatrix}, \qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\overleftarrow{\Sigma}} = \begin{bmatrix} a+c & 0 \\ 0 & b+d \end{bmatrix}.$$

Note that the arrow over $\sum$ points towards its argument. It serves to disambiguate expressions like $A^{\vec{\Sigma}}B$ without additional parentheses.

Notice the similarity of these definitions with the relation algebraic definitions of the *domain* operator $\ulcorner R = R\,;\mathbb{T}\cap\mathbb{I}$ and *codomain* operator $R^{\urcorner} = \mathbb{T}\,;R\cap\mathbb{I}$, which encode the usual domain and codomain of a relation R as subidentity relations. Such domain and codomain operators have been investigated thoroughly in the more abstract setting of semirings and Kleene algebra [11,12]. It turns out that they share some properties with the row-sum and column-sum operators. There are some differences, though, as the following table shows.

	Linear algebra	Relation algebra	Name
(a)		$\ulcorner R\,;R = R$	
(b)		$\ulcorner R\,;\ulcorner R = \ulcorner R$	
(c)	$\vec{\Sigma}(AB) = \vec{\Sigma}(A^{\vec{\Sigma}}B)$	$\ulcorner(Q\,;R) = \ulcorner(Q\,;\ulcorner R)$	Locality
(d)	$\vec{\Sigma}A^{\vec{\Sigma}}B = \vec{\Sigma}A\cdot\vec{\Sigma}B$	$\ulcorner Q\,;\ulcorner R = \ulcorner Q\cap\ulcorner R$	
(e)	$\vec{\Sigma}(\vec{\Sigma}AB) = \vec{\Sigma}A^{\vec{\Sigma}}B$	$\ulcorner(\ulcorner Q\,;R) = \ulcorner Q\,;\ulcorner R$	Import–export
(f)	$\vec{\Sigma}(A+B) = \vec{\Sigma}A+\vec{\Sigma}B$	$\ulcorner(Q\cup R) = \ulcorner Q\cup\ulcorner R$	Distributivity
(g)	$\vec{\Sigma}\vec{\Sigma}A = \vec{\Sigma}A$	$\ulcorner\ulcorner R = \ulcorner R$	Idempotence
(h)	$\vec{\Sigma}(A^{\dagger}) = (A^{\overleftarrow{\Sigma}})^{\dagger}$	$\ulcorner(R^{\smile}) = R^{\urcorner} = R^{\urcorner\smile}$	
(i)	$\vec{\Sigma}(A^{\mathsf{T}}) = A^{\overleftarrow{\Sigma}}$	$\ulcorner(R^{\smile}) = R^{\urcorner}$	
(j)	$A\mathbb{T}\cdot B = \vec{\Sigma}AB$	$Q\,;\mathbb{T}\cap R = \ulcorner Q\,;R$	
(k)	D is diagonal $\Leftrightarrow \vec{\Sigma}D = D \Leftrightarrow D^{\overleftarrow{\Sigma}} = D$	$t\subseteq\mathbb{I} \Leftrightarrow \ulcorner t = t \Leftrightarrow t^{\urcorner} = t$	

(4)

We prove the less obvious laws.

1. *Proof of* (4c). By (3), Proposition 2(e) and neutrality of $\mathbb{T}$ for the Hadamard product,
$$\vec{\Sigma}\big(A\big(\vec{\Sigma}B\big)\big) = A(B\mathbb{T}\cdot\mathbb{I})\mathbb{T}\cdot\mathbb{I} = A(B\mathbb{T}\cdot\mathbb{T})\cdot\mathbb{I} = AB\mathbb{T}\cdot\mathbb{I} = \vec{\Sigma}(AB).$$
2. *Proof of* (4d). Since $\vec{\Sigma}A$ and $\vec{\Sigma}B$ are diagonal matrices, the result follows from Proposition 2(h).
3. *Proof of* (4e). By (3), Proposition 2(e), definition of the Hadamard product and (4d),
$$\vec{\Sigma}\big(\vec{\Sigma}AB\big) = (A\mathbb{T}\cdot\mathbb{I})B\mathbb{T}\cdot\mathbb{I} = A\mathbb{T}\cdot B\mathbb{T}\cdot\mathbb{I} = (A\mathbb{T}\cdot\mathbb{I})\cdot(B\mathbb{T}\cdot\mathbb{I}) = \vec{\Sigma}A\cdot\vec{\Sigma}B = \vec{\Sigma}A^{\vec{\Sigma}}B.$$
4. *Proof of* (4i). By (3) and Proposition 2(d), $\vec{\Sigma}(A^{\mathsf{T}}) = A^{\mathsf{T}}\mathbb{T}\cdot\mathbb{I} = \mathbb{T}A\cdot\mathbb{I} = A^{\overleftarrow{\Sigma}}$.
5. *Proof of* (4j). By (3) and Proposition 2(e), $A\mathbb{T}\cdot B = (A\mathbb{T}\cdot\mathbb{I})B = \vec{\Sigma}AB$.
6. *Proof of* (4k). This is direct by definition of diagonal matrices, Proposition 2(c) and (3). □

Unlike for relation algebra laws (4a) and (4b), $\vec{\Sigma}AA = A$ and $\vec{\Sigma}A^{\vec{\Sigma}}A = \vec{\Sigma}A$ do not hold, since, e.g.,

$$\vec{\Sigma}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \neq \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

and

$$\vec{\Sigma}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\vec{\Sigma}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \neq \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \vec{\Sigma}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

If t is a relational test (a subidentity), then forward and backward diamond modal operators can be defined by $|R\rangle t = \ulcorner(R\,;t)$ and $\langle R|t = (t\,;R)^{\urcorner}$ [13]. The corresponding linear algebra expressions are $\vec{\Sigma}(AD)$ and $(DA)^{\overleftarrow{\Sigma}}$, where D is a diagonal matrix. Both $\vec{\Sigma}(AD)$ and $(DA)^{\overleftarrow{\Sigma}}$ are diagonal matrices. If one views D as a description of the "content" or "amplitude" of a state, then $(DA)^{\overleftarrow{\Sigma}}$, for instance, is the content or amplitude of the state obtained from state D by transformation A.

Using (3), Proposition 2(e) and neutrality of $\mathbb{T}$ for the Hadamard product, we get $\vec{\Sigma}(AD)\mathbb{T} = AD\mathbb{T}$ and $\mathbb{T}(DA)^{\overleftarrow{\Sigma}} = \mathbb{T}DA$. This shows that the operation $\vec{\Sigma}(AD)$ involving diagonal matrices corresponds to the expression AV, involving vectors, and similarly for $(DA)^{\overleftarrow{\Sigma}}$ and $V^{\mathsf{T}}A$; in fact, this is just the same situation as for the domain and codomain operators of relation algebra.

A matrix A is unitary iff $A^{\dagger}A = AA^{\dagger} = \mathbb{I}$. If A is unitary and V is a unit vector, then AV is a unit vector. The corresponding property for diagonal matrices is that $\vec{\Sigma}(AD)$ is a unit diagonal matrix if A is unitary and D is a unit diagonal matrix. This is proved as follows.

$$
\begin{aligned}
& \mathbb{T}(\vec{\Sigma}(AD))^{\dagger}\vec{\Sigma}(AD)\mathbb{T} \\
= \quad & \langle (3) \rangle \\
& \mathbb{T}(AD\mathbb{T}\cdot\mathbb{I})^{\dagger}(AD\mathbb{T}\cdot\mathbb{I})\mathbb{T} \\
= \quad & \langle (1\text{c, d}), \text{ and } \mathbb{I} \text{ and } \mathbb{T} \text{ are relations} \rangle \\
& \mathbb{T}(\mathbb{T}D^{\dagger}A^{\dagger}\cdot\mathbb{I})(AD\mathbb{T}\cdot\mathbb{I})\mathbb{T} \\
= \quad & \langle \text{Proposition 2(e)} \rangle \\
& (\mathbb{T}D^{\dagger}A^{\dagger}\cdot\mathbb{T})(AD\mathbb{T}\cdot\mathbb{T}) \\
= \quad & \langle \text{Neutrality of } \mathbb{T} \text{ for the Hadamard product} \rangle \\
& \mathbb{T}D^{\dagger}A^{\dagger}AD\mathbb{T} \\
= \quad & \langle A \text{ is unitary} \rangle \\
& \mathbb{T}D^{\dagger}D\mathbb{T} \\
= \quad & \langle D \text{ is a unit diagonal matrix} \rangle \\
& \mathbb{T}. \qquad \square
\end{aligned}
$$

4. Direct sums

Relational *direct sums* are axiomatised as a pair (σ_1, σ_2) of injections satisfying the following axioms:

$$(a)\ \sigma_1 ; \sigma_1^{\smile} = \mathbb{I}, \qquad (b)\ \sigma_2 ; \sigma_2^{\smile} = \mathbb{I}, \qquad (c)\ \sigma_1 ; \sigma_2^{\smile} = \mathbf{0}, \qquad (d)\ \sigma_1^{\smile} ; \sigma_1 \cup \sigma_2^{\smile} ; \sigma_2 = \mathbb{I}. \tag{5}$$

Because σ_1, σ_2 are injective functions and because $\sigma_1^{\smile} ; \sigma_1$ and $\sigma_2^{\smile} ; \sigma_2$ are disjoint, the relational operators can be replaced by the linear ones, allowing other solutions in addition to the relational ones:

$$(a)\ \sigma_1\sigma_1^{\dagger} = \mathbb{I}, \qquad (b)\ \sigma_2\sigma_2^{\dagger} = \mathbb{I}, \qquad (c)\ \sigma_1\sigma_2^{\dagger} = \mathbf{0}, \qquad (d)\ \sigma_1^{\dagger}\sigma_1 + \sigma_2^{\dagger}\sigma_2 = \mathbb{I}. \tag{6}$$

As for relations, these direct sums allow one to build matrices by blocks (i.e., by combining smaller matrices). We refer to [2,3] for an extensive study of this construct.

Eqs. (5) define σ_1 and σ_2 up to isomorphism only. Other solutions can be obtained by suitable permutations of the rows and columns of the relations σ_1 and σ_2. With Eqs. (6), even more solutions are possible. If U, U_1 and U_2 are unitary, then $(U_1^{\dagger}\sigma_1 U, U_2^{\dagger}\sigma_2 U)$ is also a direct sum satisfying (6). This amounts to having a direct sum in a different orthonormal basis.

By (1d, e) and (6), for $i = 1, 2$,

$$\left(\sigma_i^{\dagger}\sigma_i\right)^{\dagger} = \sigma_i^{\dagger}\sigma_i \quad \text{and} \quad \sigma_i^{\dagger}\sigma_i\sigma_i^{\dagger}\sigma_i = \sigma_i^{\dagger}\sigma_i.$$

This means that $\sigma_i^{\dagger}\sigma_i$ is a projection, i.e., a matrix A satisfying $A^{\dagger} = A$ and $AA = A$, not to be confused with the projections making up the direct products of Section 5 (the subidentities of a relation algebra are projections in this sense). Projections in a Hilbert space can be ordered in such a way that the ordering is an orthomodular lattice, an algebraic structure that has been thoroughly investigated (see for instance [14]) after its relevance for quantum mechanics was revealed by Birkhoff and von Neumann [15].

5. Direct products

Relational *direct products* are axiomatised as a pair (π_1, π_2) of projections satisfying the following equations:

$$(a)\ \pi_1^{\smile} ; \pi_1 = \mathbb{I}, \qquad (b)\ \pi_2^{\smile} ; \pi_2 = \mathbb{I}, \qquad (c)\ \pi_1^{\smile} ; \pi_2 = \mathbb{T}, \qquad (d)\ \pi_1 ; \pi_1^{\smile} \cap \pi_2 ; \pi_2^{\smile} = \mathbb{I}. \tag{7}$$

These equations define π_1 and π_2 up to isomorphism. For example, the following relations π_1 of type $3 \times 2 \leftrightarrow 3$ and π_2 of type $3 \times 2 \leftrightarrow 2$ provide a solution:

$$\pi_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \pi_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{8}$$

If we use this solution in the linear algebra variant of (7), we see that $\pi_1^\dagger \pi_2 = \mathbb{T}$ and $\pi_1 \pi_1^\dagger \cdot \pi_2 \pi_2^\dagger = \mathbb{I}$ hold, but not $\pi_1^\dagger \pi_1 = \mathbb{I}$ and $\pi_2^\dagger \pi_2 = \mathbb{I}$, since

$$\pi_1^\dagger \pi_1 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \text{and} \quad \pi_2^\dagger \pi_2 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}.$$

But

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbb{T}_{3 \leftrightarrow 2} \mathbb{T}_{2 \leftrightarrow 3} \cdot \mathbb{I}$$

and

$$\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbb{T}_{2 \leftrightarrow 3} \mathbb{T}_{3 \leftrightarrow 2} \cdot \mathbb{I},$$

which leads to the appropriate laws for defining direct products with the linear algebra operators, where π_1 has type $m \times n \leftrightarrow m$ and π_2 type $m \times n \leftrightarrow n$:

$$\begin{aligned} &\text{(a)}\ \pi_1^\dagger \pi_1 = \mathbb{T}_{m \leftrightarrow n} \mathbb{T}_{n \leftrightarrow m} \cdot \mathbb{I}, \qquad &&\text{(c)}\ \pi_1^\dagger \pi_2 = \mathbb{T}, \\ &\text{(b)}\ \pi_2^\dagger \pi_2 = \mathbb{T}_{n \leftrightarrow m} \mathbb{T}_{m \leftrightarrow n} \cdot \mathbb{I}, \qquad &&\text{(d)}\ \pi_1 \pi_1^\dagger \cdot \pi_2 \pi_2^\dagger = \mathbb{I}. \end{aligned} \tag{9}$$

Then $\pi_1^\dagger \pi_1$ and $\pi_2^\dagger \pi_2$ are diagonal matrices whose entries in the diagonal are n and m, respectively.

We proceed in two steps. First, we show that the usual solutions of (7) in concrete relation algebras, like π_1 and π_2 above, are indeed solutions of (9). Then we use this result to introduce more general solutions.

We denote the concrete projections by ρ_1 and ρ_2. So, assume ρ_1 and ρ_2 are relations of type $m \times n \leftrightarrow m$ and $m \times n \leftrightarrow n$, respectively. Label the columns of ρ_1 and ρ_2 by the integers 1 to m and 1 to n, respectively, and the rows of ρ_1 and ρ_2 by ordered pairs of the form $\langle i, j \rangle$, with $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$. The order in which the row labels appear is arbitrary, but must be the same for ρ_1 and ρ_2. Define ρ_1 and ρ_2 by

$$\rho_{1\,\langle i,j\rangle,k} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise,} \end{cases} \qquad \rho_{2\,\langle i,j\rangle,k} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

This is of course the usual definition of projections in concrete relation algebras. We now show that they satisfy (9).

Proposition 3. *The projections ρ_1 and ρ_2 defined in (10) satisfy (9).*

Proof.

$$\begin{aligned} \text{(a)} \quad & (\rho_1^\dagger \rho_1)_{i,j} \\ = \ & (\textstyle\sum k, l \mid (\rho_1^\dagger)_{i,\langle k,l\rangle} \times \rho_{1\,\langle k,l\rangle,j}) \\ = \ & (\textstyle\sum k, l \mid \rho_{1\,\langle k,l\rangle,i} \times \rho_{1\,\langle k,l\rangle,j}) \\ = \ & \quad \langle \text{By (10), the terms with } k \neq i \text{ evaluate to } 0 \rangle \\ & (\textstyle\sum l \mid \rho_{1\,\langle i,l\rangle,i} \times \rho_{1\,\langle i,l\rangle,j}) \\ = \ & \quad \langle \text{By (10), } \rho_{1\,\langle i,l\rangle,i} = 1 \rangle \\ & (\textstyle\sum l \mid \rho_{1\,\langle i,l\rangle,j}) \\ = \ & \quad \langle (10) \text{ and } 1 \leqslant l \leqslant n \rangle \\ & \begin{cases} 0 & \text{if } i \neq j \\ n & \text{otherwise} \end{cases} \\ = \ & (\mathbb{T}_{m \leftrightarrow n} \mathbb{T}_{n \leftrightarrow m} \cdot \mathbb{I})_{i,j} \end{aligned}$$

(b) The proof of $\pi_2^\dagger \pi_2 = \mathbb{T}_{n\leftrightarrow m}\mathbb{T}_{m\leftrightarrow n} \cdot \mathbb{I}$ is similar.

(c)
$$\begin{aligned}
&(\rho_1^\dagger \rho_2)_{i,j}\\
=\ &(\sum k,l \mid \rho_{1\langle k,l\rangle,i} \times \rho_{2\langle k,l\rangle,j})\\
=\ &\quad \langle \text{By (10), the terms with } k\neq i \text{ or } l\neq j \text{ evaluate to } 0\rangle\\
&\rho_{1\langle i,j\rangle,i} \times \rho_{2\langle i,j\rangle,j}\\
=\ &\quad \langle (10)\rangle\\
&1\\
=\ &\mathbb{T}_{i,j}
\end{aligned}$$

(d)
$$\begin{aligned}
&(\rho_1\rho_1^\dagger \cdot \rho_2\rho_2^\dagger)_{\langle i,j\rangle,\langle k,l\rangle}\\
=\ &(\rho_1\rho_1^\dagger)_{\langle i,j\rangle,\langle k,l\rangle} \times (\rho_2\rho_2^\dagger)_{\langle i,j\rangle,\langle k,l\rangle}\\
=\ &(\sum s \mid \rho_{1\langle i,j\rangle,s} \times \rho_{1\langle k,l\rangle,s}) \times (\sum t \mid \rho_{2\langle i,j\rangle,t} \times \rho_{2\langle k,l\rangle,t})\\
=\ &\quad \langle \text{By (10), the terms with } s\neq i \text{ and those with } t\neq j \text{ evaluate to } 0\rangle\\
&\rho_{1\langle i,j\rangle,i} \times \rho_{1\langle k,l\rangle,i} \times \rho_{2\langle i,j\rangle,j} \times \rho_{2\langle k,l\rangle,j}\\
=\ &\quad \langle \text{By (10), } \rho_{1\langle i,j\rangle,i} = \rho_{2\langle i,j\rangle,j} = 1\rangle\\
&\rho_{1\langle k,l\rangle,i} \times \rho_{2\langle k,l\rangle,j}\\
=\ &\quad \langle (10)\rangle\\
&\begin{cases} 1 & \text{if } i=k \text{ and } j=l\\ 0 & \text{otherwise}\end{cases}\\
=\ &\mathbb{I}_{\langle i,j\rangle,\langle k,l\rangle}. \qquad \square
\end{aligned}$$

In relation algebra, direct products can be used to transform a relation R into a vector. This is called vectorisation. The vectorisation of a relation R is obtained by $\mathsf{vec}(R) = (\pi_1 \,;\, R \cap \pi_2)\,;\mathbb{T}$. With the relations π_1 and π_2 from (8), this works for arbitrary matrices A and the linear algebra operators. For instance, with $A = \begin{bmatrix} a & b\\ c & d\\ e & f\end{bmatrix}$,

$$\mathsf{vec}(A) = (\pi_1 A \cdot \pi_2)\mathbb{T}_{2\leftrightarrow 1} = \left(\begin{bmatrix} 1&0&0\\1&0&0\\0&1&0\\0&1&0\\0&0&1\\0&0&1\end{bmatrix}\begin{bmatrix} a & b\\ c & d\\ e & f\end{bmatrix} \cdot \begin{bmatrix} 1&0\\0&1\\1&0\\0&1\\1&0\\0&1\end{bmatrix}\right)\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix} a\\b\\c\\d\\e\\f\end{bmatrix}.$$

Note that since π_2 is a relation, then $\pi_2^\ddagger = \pi_2$, so that vectorisation could be defined by $\mathsf{vec}(A) = (\pi_1 A \cdot \pi_2^\ddagger)\mathbb{T}$ instead of $\mathsf{vec}(A) = (\pi_1 A \cdot \pi_2)\mathbb{T}$. The former expression will be used from now on, because it better suits the forthcoming generalisation (see in particular Proposition 6).

Expressing vectorisation with the linear operators generally works for the concrete relations ρ_1 of type $m \times n \leftrightarrow m$ and ρ_2 $m \times n \leftrightarrow n$ defined in (10), giving a vector $\mathsf{vec}(A)$ of type $m \times n \leftrightarrow 1$:

$$\begin{aligned}
&(\mathsf{vec}(A))_{\langle i,j\rangle,1}\\
=\ &((\rho_1 A \cdot \rho_2^\ddagger)\mathbb{T})_{\langle i,j\rangle,1}\\
=\ &((\rho_1 A \cdot \rho_2)\mathbb{T})_{\langle i,j\rangle,1}\\
=\ &(\sum k \mid (\rho_1 A \cdot \rho_2)_{\langle i,j\rangle,k} \times \mathbb{T}_{k,1})\\
=\ &\quad \langle \text{Definition of } \cdot \text{ and } \mathbb{T}_{k,1} = 1\rangle\\
&(\sum k \mid (\rho_1 A)_{\langle i,j\rangle,k} \times \rho_{2\langle i,j\rangle,k})\\
=\ &\quad \langle \text{By (10), the terms with } k\neq j \text{ evaluate to } 0\rangle\\
&(\rho_1 A)_{\langle i,j\rangle,j} \times \rho_{2\langle i,j\rangle,j}\\
=\ &\quad \langle \text{By (10), } \rho_{2\langle i,j\rangle,j} = 1 \text{ and definition of composition}\rangle\\
&(\sum k \mid \rho_{1\langle i,j\rangle,k} \times A_{k,j})\\
=\ &\quad \langle \text{By (10), the terms with } k\neq i \text{ evaluate to } 0\rangle\\
&\rho_{1\langle i,j\rangle,i} \times A_{i,j}\\
=\ &\quad \langle \text{By (10), } \rho_{1\langle i,j\rangle,i} = 1\rangle\\
&A_{i,j}.
\end{aligned}$$

Hence the entry in the row of $\mathsf{vec}(A)$ labelled by $\langle i, j\rangle$ is indeed the entry $A_{i,j}$ of matrix A.

Unvectorisation consists in retrieving the original matrix from its vectorised form. In the relational setting, unvectorisation is defined by

$$\mathsf{unvec}\big(\mathsf{vec}(A)\big) = \pi_1^{\smile} ; \big(\mathsf{vec}(A) ; \mathbb{T} \cap \pi_2\big)$$

and satisfies $\mathsf{unvec}(\mathsf{vec}(A)) = A$. This works as well for the relations π_1 and π_2 from (8), the linear operators and $\mathsf{vec}(A)$ as calculated above:

$$\begin{aligned}
&\pi_1^{\dagger}(\mathsf{vec}(A)\mathbb{T}_{1\leftrightarrow 2} \cdot \pi_2) \\
&= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \left(\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} [\,1 \quad 1\,] \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\
&= \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \\
&= A.
\end{aligned}$$

Compared to what happens with relations, there is the additional constraint that the $\mathbb{T}$ used for vectorisation must have one column and the one used for unvectorisation must have one row.

We show below that unvectorisation also works for more general solutions than the relations ρ_1 and ρ_2 defined in (10). Before getting there, we present a few more properties of the projection relations ρ_1 and ρ_2.

Proposition 4. *The projections ρ_1 and ρ_2 defined in (10) satisfy the following properties.*

(a) *For all l, $\rho_1\mathbb{T}_{m\leftrightarrow l} = \mathbb{T}_{m\times n\leftrightarrow l}$.*
For all l, $\rho_2\mathbb{T}_{n\leftrightarrow l} = \mathbb{T}_{m\times n\leftrightarrow l}$.

(b) $(A\rho_1^{\dagger} \cdot B\rho_2^{\dagger})(\rho_1 C \cdot \rho_2 D) = AC \cdot BD$.

(c) $(A\rho_1^{\dagger} \cdot B\rho_2^{\dagger})\rho_1 = A \cdot B\mathbb{T}$,
$(A\rho_1^{\dagger} \cdot B\rho_2^{\dagger})\rho_2 = A\mathbb{T} \cdot B$,
$\rho_1^{\dagger}(\rho_1 A \cdot \rho_2 B) = A \cdot \mathbb{T}B$,
$\rho_2^{\dagger}(\rho_1 A \cdot \rho_2 B) = \mathbb{T}A \cdot B$.

Proof.

(a) We prove the first assertion. The proof of the second one is similar. The typing of the two $\mathbb{T}$ is consistent with the typing $m \times n \leftrightarrow m$ of ρ_1.

$$\begin{aligned}
&(\rho_1\mathbb{T})_{\langle i,j\rangle,k} \\
&= (\textstyle\sum s \mid \rho_{1\,\langle i,j\rangle,s} \times \mathbb{T}_{s,k}) \\
&= \quad \langle \mathbb{T}_{s,k} = 1 \text{ and, by (10), the terms with } s \neq i \text{ evaluate to } 0\rangle \\
&\quad \rho_{1\,\langle i,j\rangle,i} \\
&= \quad \langle (10)\rangle \\
&\quad 1 \\
&= \mathbb{T}_{\langle i,j\rangle,k}
\end{aligned}$$

(b)
$$\begin{aligned}
&((A\rho_1^{\dagger} \cdot B\rho_2^{\dagger})(\rho_1 C \cdot \rho_2 D))_{i,j} \\
&= (\textstyle\sum k,l \mid (A\rho_1^{\dagger} \cdot B\rho_2^{\dagger})_{i,\langle k,l\rangle} \times (\rho_1 C \cdot \rho_2 D)_{\langle k,l\rangle,j}) \\
&= (\textstyle\sum k,l \mid (A\rho_1^{\dagger})_{i,\langle k,l\rangle} \times (B\rho_2^{\dagger})_{i,\langle k,l\rangle} \times (\rho_1 C)_{\langle k,l\rangle,j} \times (\rho_2 D)_{\langle k,l\rangle,j}) \\
&= (\textstyle\sum k,l \mid (\sum s \mid A_{i,s} \times (\rho_1^{\dagger})_{s,\langle k,l\rangle}) \times (\sum s \mid B_{i,s} \times (\rho_2^{\dagger})_{s,\langle k,l\rangle}) \\
&\qquad\qquad \times (\textstyle\sum s \mid \rho_{1\,\langle k,l\rangle,s} \times C_{s,j}) \times (\sum s \mid \rho_{2\,\langle k,l\rangle,s} \times D_{s,j})) \\
&= (\textstyle\sum k,l \mid (\sum s \mid A_{i,s} \times \rho_{1\,\langle k,l\rangle,s}) \times (\sum s \mid B_{i,s} \times \rho_{2\,\langle k,l\rangle,s}) \\
&\qquad\qquad \times (\textstyle\sum s \mid \rho_{1\,\langle k,l\rangle,s} \times C_{s,j}) \times (\sum s \mid \rho_{2\,\langle k,l\rangle,s} \times D_{s,j}))
\end{aligned}$$

$$
\begin{aligned}
&= && \langle (10) \rangle \\
&\quad (\sum k, l \mid A_{i,k} \times B_{i,l} \times C_{k,j} \times D_{l,j}) \\
&= && \langle k \text{ not free in } B_{i,l} \times D_{l,j}, l \text{ not free in } A_{i,k} \times C_{k,j} \text{ and distributivity of } \times \text{ over } \textstyle\sum \rangle \\
&\quad (\sum k \mid A_{i,k} \times C_{k,j}) \times (\sum l \mid B_{i,l} \times D_{l,j}) \\
&= (AC)_{i,j} \times (BD)_{i,j} \\
&= (AC \cdot BD)_{i,j}
\end{aligned}
$$

(c) We prove the first assertion only, the other properties following by duality. The result follows from Proposition 2(l), due to the univalence of ρ_1, and from (1d, e), Proposition 3(c) and $\mathbb{T}^\dagger = \mathbb{T}$.

$$\left(A\rho_1^\dagger \cdot B\rho_2^\dagger\right)\rho_1 = A \cdot B\rho_2^\dagger\rho_1 = A \cdot B\left(\rho_1^\dagger\rho_2\right)^\dagger = A \cdot B\mathbb{T}^\dagger = A \cdot B\mathbb{T}. \qquad \square$$

Now that we have some basic properties of the ρ_1 and ρ_2 projections, we ask the question whether there are more general solutions to (9). Trying to solve (9) even for the simplest non-trivial case of two projections of type $2 \times 2 \leftrightarrow 2$ is a hard task which we have not yet successfully completed. However, recall how in Section 4 unitary matrices U, U_1 and U_2 are used to obtain a new direct sum $(U_1^\dagger\sigma_1 U, U_2^\dagger\sigma_2 U)$ that is a solution of (6) from another solution (σ_1, σ_2). The next proposition shows that something similar can be accomplished with direct products, although only composition on the right by unitary matrices seems to yield something useful.

Proposition 5. *Let (π_1, π_2) be a solution of (9a, b, d), and U_1 and U_2 be unitary matrices. Then $(\pi_1 U_1, \pi_2 U_2)$ is a solution of (9a, b, d).*

Proof. The proof that $\pi_1 U_1$ satisfies (9a) follows from contravariance of † (1d), the hypothesis that π_1 satisfies (9a), Proposition 2(e) and unitarity of U_1:

$$(\pi_1 U_1)^\dagger \pi_1 U_1 = U_1^\dagger \pi_1^\dagger \pi_1 U_1 = U_1^\dagger (\mathbb{T}_{m \leftrightarrow n} \mathbb{T}_{n \leftrightarrow m} \cdot \mathbb{I}) U_1 = \mathbb{T}_{m \leftrightarrow n} \mathbb{T}_{n \leftrightarrow m} \cdot U_1^\dagger U_1 = \mathbb{T}_{m \leftrightarrow n} \mathbb{T}_{n \leftrightarrow m} \cdot \mathbb{I}.$$

The proof for (9b) is similar. Finally, (9d) follows from contravariance of † (1d), unitarity of U_1 and U_2, and the hypothesis that (π_1, π_2) satisfies (9d):

$$\pi_1 U_1 (\pi_1 U_1)^\dagger \cdot \pi_2 U_2 (\pi_2 U_2)^\dagger = \pi_1 U_1 U_1^\dagger \pi_1^\dagger \cdot \pi_2 U_2 U_2^\dagger \pi_2^\dagger = \pi_1 \pi_1^\dagger \cdot \pi_2 \pi_2^\dagger = \mathbb{I}.$$

A simple corollary of this proposition is that $(\rho_1 U_1, \rho_2 U_2)$ is a solution of (9a, b, d), where ρ_1 and ρ_2 are defined in (10).

Unfortunately, even if (π_1, π_2) is a solution of all of (9), $(\pi_1 U_1, \pi_2 U_2)$ generally fails to satisfy (9c). We illustrate this with the projections

$$\pi_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \qquad \pi_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{11}$$

which satisfy (9). Using the unitary matrices $U_1 = \left[\begin{smallmatrix} -1 & 0 \\ 0 & -1 \end{smallmatrix}\right]$ and $U_2 = \mathbb{I}$ yields $(\pi_1 U_1)^\dagger \pi_2 U_2 = \left[\begin{smallmatrix} -1 & -1 \\ -1 & -1 \end{smallmatrix}\right] \neq \mathbb{T}$. Using identical unitary matrices $U_1 = U_2 = \left[\begin{smallmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{smallmatrix}\right]$ yields $(\pi_1 U_1)^\dagger \pi_2 U_2 = \left[\begin{smallmatrix} 2 & 2 \\ 2 & 2 \end{smallmatrix}\right] \neq \mathbb{T}$.

Despite this drawback, it turns out that projections (π_1, π_2) that satisfy only (9a, b, d) have useful properties.

We first look at the case of vectorisation/unvectorisation. Using the polar representation of complex numbers, the general form of a unitary 2×2 matrix U is

$$U = \begin{bmatrix} re^{i\theta_1} & \sqrt{1-r^2}e^{i\theta_2} \\ -\sqrt{1-r^2}e^{i\theta_3} & re^{i\theta_4} \end{bmatrix},$$

where $0 \leqslant r \leqslant 1$, $i = \sqrt{-1}$, $0 \leqslant \theta_i < 2\pi$ for $i = 1, 2, 3, 4$, and $\theta_4 = \theta_2 + \theta_3 - \theta_1 \pmod{2\pi}$. With π_1 and π_2 as in (11), the generalised direct product is $(\pi_1 U_1, \pi_2 U_2)$. Using $U_1 = U_2 = U$ to make things simpler and $A = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, we find that

$$
\begin{aligned}
& \mathsf{vec}(A) \\
= \;& (\pi_1 U_1 A \cdot (\pi_2 U_2)^{\ddagger})\mathbb{T}_{2\leftrightarrow 1} \\
= \;& \quad \langle (1d) \text{ and } \pi_2 \text{ is a relation} \rangle \\
& (\pi_1 U_1 A \cdot \pi_2 U_2^{\ddagger})\mathbb{T}_{2\leftrightarrow 1} \\
= \;& a \begin{bmatrix} r^2 \\ -r\sqrt{1-r^2}\mathrm{e}^{\mathrm{i}(\theta_1-\theta_3)} \\ -r\sqrt{1-r^2}\mathrm{e}^{\mathrm{i}(\theta_3-\theta_1)} \\ 1-r^2 \end{bmatrix} + b \begin{bmatrix} r\sqrt{1-r^2}\mathrm{e}^{\mathrm{i}(\theta_1-\theta_2)} \\ r^2\mathrm{e}^{\mathrm{i}(\theta_1-\theta_4)} \\ -(1-r^2)\mathrm{e}^{\mathrm{i}(\theta_3-\theta_2)} \\ -r\sqrt{1-r^2}\mathrm{e}^{\mathrm{i}(\theta_3-\theta_4)} \end{bmatrix} \\
& + c \begin{bmatrix} r\sqrt{1-r^2}\mathrm{e}^{\mathrm{i}(\theta_2-\theta_1)} \\ -(1-r^2)\mathrm{e}^{\mathrm{i}(\theta_2-\theta_3)} \\ r^2\mathrm{e}^{\mathrm{i}(\theta_4-\theta_1)} \\ -r\sqrt{1-r^2}\mathrm{e}^{\mathrm{i}(\theta_4-\theta_3)} \end{bmatrix} + d \begin{bmatrix} 1-r^2 \\ r\sqrt{1-r^2}\mathrm{e}^{\mathrm{i}(\theta_2-\theta_4)} \\ r\sqrt{1-r^2}\mathrm{e}^{\mathrm{i}(\theta_4-\theta_2)} \\ r^2 \end{bmatrix}.
\end{aligned}
$$

This rather complex beast is a vector which contains a, b, c, d, but it does not have the simple form $\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ that we would obtain by using (π_1, π_2) instead of $(\pi_1 U_1, \pi_2 U_2)$. Nevertheless, no information is lost, and this vector can be unvectorised to retrieve A, as the following proposition shows. In this proposition, the definition of vec is as used above.

Proposition 6. *Let ρ_1 and ρ_2 be the projections defined in (10) and U_1 and U_2 be unitary matrices. Define $\pi_1 = \rho_1 U_1$, $\pi_2 = \rho_2 U_2$, $\mathsf{vec}(A) = (\pi_1 A \cdot \pi_2^{\ddagger})\mathbb{T}_{n\leftrightarrow 1}$ and $\mathsf{unvec}(V) = \pi_1^{\dagger}(V\mathbb{T}_{1\leftrightarrow n} \cdot \pi_2)$. Then $\mathsf{unvec}(\mathsf{vec}(A)) = A$.*

Proof.

$$
\begin{aligned}
& \mathsf{unvec}(\mathsf{vec}(A)) \\
= \;& \pi_1^{\dagger}((\pi_1 A \cdot \pi_2^{\ddagger})\mathbb{T}_{n\leftrightarrow 1}\mathbb{T}_{1\leftrightarrow n} \cdot \pi_2) \\
= \;& \quad \langle \mathbb{T}_{n\leftrightarrow 1}\mathbb{T}_{1\leftrightarrow n} = \mathbb{T}_{n\leftrightarrow n} \text{ (abbreviated to } \mathbb{T}) \text{ and definition of } \pi_1 \text{ and } \pi_2 \rangle \\
& (\rho_1 U_1)^{\dagger}((\rho_1 U_1 A \cdot (\rho_2 U_2)^{\ddagger})\mathbb{T} \cdot \rho_2 U_2) \\
= \;& \quad \langle (1c) \text{ and Proposition 2(k)} \rangle \\
& U_1^{\dagger}\rho_1^{\dagger}((\rho_1 U_1 A(\rho_2 U_2)^{\ddagger\mathsf{T}} \cdot \mathbb{I})\mathbb{T} \cdot \rho_2 U_2) \\
= \;& \quad \langle (1b, d)) \text{ and Proposition 2(e)} \rangle \\
& U_1^{\dagger}\rho_1^{\dagger}((\rho_1 U_1 A U_2^{\dagger}\rho_2^{\dagger} \cdot \mathbb{I})\mathbb{T} \cdot \mathbb{I})\rho_2 U_2 \\
= \;& \quad \langle \text{Proposition 2(c), using that } \rho_1 U_1 A U_2^{\dagger}\rho_2^{\dagger} \cdot \mathbb{I} \text{ is diagonal} \rangle \\
& U_1^{\dagger}\rho_1^{\dagger}(\rho_1 U_1 A U_2^{\dagger}\rho_2^{\dagger} \cdot \mathbb{I})\rho_2 U_2 \\
= \;& \quad \langle \text{Proposition 2(l), using that } \rho_1 \text{ and } \rho_2 \text{ are univalent} \rangle \\
& U_1^{\dagger}(U_1 A U_2^{\dagger} \cdot \rho_1^{\dagger}\rho_2)U_2 \\
= \;& \quad \langle \text{Proposition 3(c)} \rangle \\
& U_1^{\dagger}(U_1 A U_2^{\dagger} \cdot \mathbb{T})U_2 \\
= \;& \quad \langle \text{Neutrality of } \mathbb{T} \text{ for the Hadamard product} \rangle \\
& U_1^{\dagger}U_1 A U_2^{\dagger}U_2 \\
= \;& \quad \langle U_1 \text{ and } U_2 \text{ are unitary} \rangle \\
& A. \qquad \square
\end{aligned}
$$

Vectorisation and unvectorisation can be generalised in a different way, as shown in [3]. A matrix of type $l \times m \leftrightarrow n$ can be restructured as a matrix of type $l \leftrightarrow m \times n$ by vectorisation[1] and the process is reversed by unvectorisation. For instance, a matrix of type $6 \leftrightarrow 4$, i.e., $2 \times 3 \leftrightarrow 4$, can be restructured as a matrix of type $2 \leftrightarrow 3 \times 4$, i.e., $2 \leftrightarrow 12$. Vectorisation/unvectorisation as described above corresponds to the special case $m = 1$. In [3], the transformation strategy is point-free divide-and-conquer, whereby matrices are first split into blocks. By using suitable direct products, it should be possible to obtain this generalisation, but we have not yet worked out the details.

[1] We keep this name for the process, although the result need not be a vector.

Direct products can also be used for defining the Kronecker product. Given size-compatible projections π_1 and π_2, the *Kronecker product* $A \otimes B$ is defined by

$$A \otimes B = \pi_1 A \pi_1^\dagger \cdot \pi_2 B \pi_2^\dagger. \tag{12}$$

This is the standard Kronecker product of linear algebra when π_1 and π_2 are relations. For instance, with the π_1 and π_2 given in (8),

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \otimes \begin{bmatrix} j & k \\ l & m \end{bmatrix} = \begin{bmatrix} a\times j & a\times k & b\times j & b\times k & c\times j & c\times k \\ a\times l & a\times m & b\times l & b\times m & c\times l & c\times m \\ d\times j & d\times k & e\times j & e\times k & f\times j & f\times k \\ d\times l & d\times m & e\times l & e\times m & f\times l & f\times m \\ g\times j & g\times k & h\times j & h\times k & i\times j & i\times k \\ g\times l & g\times m & h\times l & h\times m & i\times l & i\times m \end{bmatrix}.$$

For more general direct products of the form $(\rho_1 U_1, \rho_2 U_2)$, the result is more complex (as in the case of $\mathsf{vec}(A)$). Nevertheless, the following proposition shows that the Kronecker product still has some of its usual properties.

Proposition 7. *Let ρ_1 and ρ_2 be the projections defined in (10) and U_1 and U_2 be unitary matrices. Define $\pi_1 = \rho_1 U_1$ and $\pi_2 = \rho_2 U_2$. Then,*

(a) $(A\pi_1^\dagger \cdot B\pi_2^\dagger)(\pi_1 C \cdot \pi_2 D) = AC \cdot BD$;
(b) $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$;
(c) *If A and B are invertible, then $A \otimes B$ is invertible, with inverse $A^{-1} \otimes B^{-1}$;*
(d) *If A and B are unitary, then $A \otimes B$ is unitary, with adjoint $A^\dagger \otimes B^\dagger$.*

Proof.

(a)
$$\begin{aligned} & (A\pi_1^\dagger \cdot B\pi_2^\dagger)(\pi_1 C \cdot \pi_2 D) \\ = & \quad \langle \text{Definition of } \pi_1 \text{ and } \pi_2, \text{ and (1d)} \rangle \\ & (AU_1^\dagger \rho_1^\dagger \cdot BU_2^\dagger \rho_2^\dagger)(\rho_1 U_1 C \cdot \rho_2 U_2 D) \\ = & \quad \langle \text{Proposition 4(b)} \rangle \\ & AU_1^\dagger U_1 C \cdot BU_2^\dagger U_2 D \\ = & \quad \langle U_1 \text{ and } U_2 \text{ are unitary} \rangle \\ & AC \cdot BD \end{aligned}$$

(b) This follows from the definition of $\otimes$ in (12) and Part a.
(c) This is direct from Part b.
(d) This is direct from Part b. □

6. Conclusion

We plan to continue the exploration of similar laws inspired by those of relation and Kleene algebra. In addition, we need to identify a small set of basic formulae and derive the others from them in a point-free way, in order to reduce the number of pointwise proofs of this paper. Using such basic laws as axioms in a theorem prover should help developing the theory. A referee made an interesting suggestion here. Introducing an operator of Hadamard inverse (entrywise arithmetic division by a matrix without 0 entries) and using block techniques such as those of [3], one can obtain a point-free proof of Proposition 2(e). We also intend to determine whether it is possible to find more general solutions to Eqs. (9a, b, d) defining direct products and to determine what are exactly the solutions to all four axioms (9a, b, c, d). Here too, proceeding by divide-and-conquer using blocks as in [3] may yield interesting insights. Finally, we plan to look at applications in the areas of quantum automata and program derivation.

Acknowledgements

This research was partially supported by NSERC (Natural Sciences and Engineering Research Council of Canada). We thank the referees for their comments and interesting suggestions, some of which are good hints for future research.

References

[1] J.N. Oliveira, Typed linear algebra for weighted (probabilistic) automata, in: N. Moreira, R. Reis (Eds.), Implementation and Application of Automata, in: Lect. Notes Comput. Sci., vol. 7381, Springer, 2012, pp. 52–65.

[2] H.D. Macedo, J.N. Oliveira, Matrices as arrows! A biproduct approach to typed linear algebra, in: C. Bolduc, J. Desharnais, B. Ktari (Eds.), Mathematics of Program Construction, in: Lect. Notes Comput. Sci., vol. 6120, Springer, 2010, pp. 271–287.
[3] H.D. Macedo, J.N. Oliveira, Typing linear algebra: A biproduct-oriented approach, Sci. Comput. Program. 78 (2013) 2160–2191.
[4] G. Schmidt, T. Ströhlein, Relations and Graphs, Springer, 1988.
[5] G. Schmidt, Relational Mathematics, Encycl. Math. Appl., vol. 132, Cambridge University Press, 2010.
[6] S. Roman, Advanced Linear Algebra, second ed., Grad. Texts Math., Springer, 2005.
[7] A. Grinenko, J. Desharnais, Some relational style laws of linear algebra, Student paper, in: 13th International Conference on Relational and Algebraic Methods in Computer Science (RAMiCS 13), Cambridge, UK, 2012, http://www.cl.cam.ac.uk/conference/ramics13/GrinenkoDesharnais.pdf.
[8] H. Furusawa, A representation theorem for relation algebras: Concepts of scalar relations and point relations, Bull. Inform. Cybern. 30 (1998) 109–119.
[9] Y. Kawahara, H. Furusawa, Crispness and representation theorem in Dedekind categories, Technical report DOI-TR 143, Department of Informatics, Kyushu University, 1997.
[10] M. Winter, A new algebraic approach to L-fuzzy relations convenient to study crispness, Inf. Sci. 139 (2001) 233–252.
[11] J. Desharnais, B. Möller, G. Struth, Kleene algebra with domain, ACM Trans. Comput. Log. 7 (2006) 798–833.
[12] J. Desharnais, G. Struth, Internal axioms for domain semirings, Sci. Comput. Program. 76 (2011) 181–203.
[13] J. Desharnais, B. Möller, G. Struth, Modal Kleene algebra and applications – a survey, J. Relat. Methods Comput. Sci. 1 (2004) 93–131.
[14] G. Kalmbach, Orthomodular Lattices, Academic Press, 1983.
[15] G. Birkhoff, J. von Neumann, The logic of quantum mechanics, Ann. Math. 37 (1936) 823–843.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 169–179

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Discrete dualities for some algebras with relations

Ivo Düntsch [a,*,1], Ewa Orłowska [b,2]

[a] Brock University, St. Catharines, Ontario, L2S 3A1, Canada
[b] National Institute of Telecommunications, Szachowa 1, 04-894 Warsaw, Poland

ARTICLE INFO

Article history:
Available online 5 March 2014

Dedicated to Gunther Schmidt on the occasion of his 75th birthday.

Keywords:
Discrete dualities
Proximities
Contact algebras
Syllogistic structures

ABSTRACT

In this paper we present a unifying discrete framework for various representation theorems in the field of spatial reasoning. We also show that the universal and existential quantifiers of restricted scope used in first order languages and represented as binary relations in the syllogistic algebras considered by Shepherdson (1956) [1] may be studied in this framework.

1. Introduction

In this paper we present discrete dualities between some classes of Boolean algebras or their reducts, additionally endowed with a binary relation, and the appropriate classes of relational systems (frames). We consider two types of algebras with relations: those studied in connection with spatial reasoning where the relations describe relationships between space regions, and those considered in theories of Aristotelian syllogistic presented in [1] where the relations represent statements of the form "All (resp. some) a are b".

For an overview of recent developments in the region based theory of space and related topics we invite the reader to consult [2] or [3].

By a *discrete duality* [4] we mean a system $\langle \mathsf{Alg}, \mathsf{Frm}, \mathfrak{Cm}, \mathfrak{Cf} \rangle$ where Alg is a class of algebras, Frm is a class of frames, $\mathfrak{Cm} : \mathsf{Frm} \to \mathsf{Alg}$ is a mapping assigning to every frame X in Frm its *complex algebra* $\mathfrak{Cm}(X)$ in such a way that $\mathfrak{Cm}(X)$ belongs to Alg, $\mathfrak{Cf} : \mathsf{Alg} \to \mathsf{Frm}$ is a mapping assigning to every algebra L in Alg its *canonical frame* $\mathfrak{Cf}(L)$ in such a way that $\mathfrak{Cf}(L)$ belongs to Frm, and the following two representation theorems hold:

1. Representation theorem for algebras: Every L in Alg is embeddable into $\mathfrak{Cm}(\mathfrak{Cf}(L))$.
2. Representation theorem for frames: Every X in Frm is embeddable into $\mathfrak{Cf}(\mathfrak{Cm}(X))$.

In the literature there are some topological representation theorems for algebras of spatial reasoning, see for example [5,6], but no corresponding abstract frames are considered, only the structures which – in terms of the notions used in

* Corresponding author.
E-mail addresses: duentsch@brocku.ca (I. Düntsch), orlowska@itl.waw.pl (E. Orłowska).

[1] Ivo Düntsch gratefully acknowledges support by the Natural Sciences and Engineering Research Council of Canada.
[2] Ewa Orłowska gratefully acknowledges partial support from the National Science Centre project DEC-2011/02/A/HS1/00395.

http://dx.doi.org/10.1016/j.jlap.2014.02.006
2352-2208/

the definition of discrete duality (dd-notions) – are counterparts to canonical frames endowed with a topology are introduced. In [7] a discrete representation theorem for Boolean proximity algebras is presented, but there is no a representation theorem for frames. In [1] a necessary and sufficient condition is given for obtaining a representation theorem for syllogistic algebras, saying that every such algebra is embeddable into the algebra which – in terms of dd-notions – is the complex algebra of a frame. In the present paper we develop discrete representations for some of those algebras, introduce the corresponding classes of frames, and prove representation theorems for them.

In Section 3 we present a discrete duality for Boolean algebras endowed with a proximity relation and their corresponding frames. Following [7], the canonical frames are constructed with prime filters – as usual in the Stone representation theorem for Boolean algebras.

In Section 4 a discrete duality for Boolean algebras with a contact relation and the appropriate frames is developed. In this case the canonical frames are constructed with clans as in [6].

In Section 5 a discrete representation theorem is presented for a syllogistic ∀-algebra with the relation representing the universal quantifier with a restricted scope as considered in [1].

In Section 6 a discrete representation theorem for ∀∃-frames is developed. The representation structure is built with an algebra with relations representing both the universal and existential quantifiers of restricted scope presented in [1].

2. Notation and first definitions

If R, S are binary relations on X, then $R;S = \{\langle a, c\rangle\colon\ (\exists b)[aRb \text{ and } bSc]\}$ is the *relational product of R and S*, and $R^{\smile} = \{\langle a, b\rangle\colon\ bRa\}$ is the *converse of R*.

A frame is a pair $\langle X, R\rangle$, where X is a nonempty set and R is a binary relation on X. Since the frames needed for establishing a discrete duality for Boolean algebras are just nonempty sets, in this paper we also allow for such a "degenerate" notion of frame. For a frame $\langle X, R\rangle$ we set $R(x) = \{y\colon\ xRy\}$, and define two operators on 2^X by

$$\langle R\rangle(A) = \big\{x \in X\colon\ (\exists y \in X)[xRy \wedge y \in A]\big\} = \big\{x \in X\colon\ R(x) \cap A \neq \emptyset\big\}, \tag{1}$$

$$[R](A) = \big\{x \in X\colon\ (\forall y \in X)[xRy \Rightarrow y \in A]\big\} = \big\{x \in X\colon\ R(x) \subseteq A\big\}. \tag{2}$$

The following properties of $\langle R\rangle$ and $[R]$ are well known, see [8]:

Lemma 2.1.

1. *$\langle R\rangle$ is a normal operator which distributes over $\cup$.*
2. *$\langle R\rangle$ and $[R]$ are dual to each other, i.e. $[R](A) = X \setminus \langle R\rangle(X \setminus A)$.*
3. *$\langle R\rangle$ is a closure operator if and only if $[R]$ is an interior operator if and only if R is reflexive and transitive.*

For the definition of closure and interior operator see [9].

If $\leqslant$ is a partial order on X, we call $x, y \in X$ *compatible*, if there is some z such that $x \leqslant z$ and $y \leqslant z$, otherwise, they are called *incompatible*. We can consider $\langle\leqslant\rangle$ and $[\leqslant]$ as topological operators of closure and interior. If $A \subseteq X$, we let $\uparrow A = \{y\colon\ (\exists x)[x \in A \text{ and } x \leqslant y]\}$. If $A = \{x\}$ we often just write $\uparrow x$ instead of $\uparrow\{x\}$. We define $\downarrow A$ analogously.

The *order topology $\tau_{\leqslant}$ on X generated by $\leqslant$* (also called the *Alexandrov topology*) has the sets of the form $[\leqslant](A)$ as a basis for the open sets. Since $[\leqslant]$ is an interior operator, $\tau_{\leqslant}$ is well defined; indeed, $\tau_{\leqslant}$ is closed under arbitrary intersections. Since for each $x \in X$, $\uparrow x$ is the smallest open set containing x, we see that the set $\{\uparrow x\colon\ x \in X\} \cup \{\emptyset\}$ also is a basis for $\tau_{\leqslant}$. $\mathrm{RegCl}(X, \tau_{\leqslant})$, the collection of regular closed sets in this topology, consists of sets of the form $\langle\leqslant\rangle[\leqslant](Y)$ for $Y \subseteq X$.

The operations on a Boolean algebra are denoted by $\wedge$ (meet), $\vee$ (join), $-$ (complement), 0 (minimum), and 1 (maximum); in particular, if R is a binary relation on X, then $-R$ is its complement in the Boolean set algebra with universe $2^{X\times X}$. We reserve the symbol $\neg$ for negation in syllogistic structures.

3. A discrete duality of proximity algebras and frames

In this section we show how the representation of proximity algebras of [7] fits into our framework. In this context, we call a structure $\langle X, R\rangle$ a *proximity frame*, if X is a nonempty set and R is a binary relation on X.

Suppose that B is a Boolean algebra. A binary relation δ on B is called a *proximity* if it satisfies the following for all $a, b, c \in B$:

Prox$_1$. $a\,\delta\,b$ implies $a \neq 0$ and $b \neq 0$.
Prox$_2$. $a\,\delta(b \vee c)$ if and only if $a\,\delta\,b$ or $a\,\delta\,c$.
Prox$_3$. $(a \vee b)\,\delta\,c$ if and only if $a\,\delta\,c$ or $b\,\delta\,c$.

The pair $\langle B, \delta\rangle$ is called a *Boolean proximity algebra* (BPA). These were introduced in [7] as an abstract version of proximity spaces [10]. Among others, the following axiomatic extensions of proximities have been studied:

A proximity δ is called a *contact relation* (or *Čech proximity*) if it satisfies

Prox_4. δ is symmetric.
Prox_5. If $a \wedge b \neq 0$ then $a\,\delta\,b$.

If δ is a contact relation on B, then the pair $\langle B, \delta\rangle$ is called a *Boolean contact algebra* (BCA).

The following is well known and easy to show:

Theorem 3.1. *If B is a subalgebra of the Boolean algebra of regular closed sets* $\text{RegCl}(X)$ *of some topological space X, then the relation δ on B defined by $a\,\delta\,b$ if and only if $a \cap b \neq \emptyset$ is a contact relation.*

BCAs of this form are called *standard models*. It is easy to see that the smallest contact relation on B is the *overlap relation* defined by aOb if and only if $a \wedge b \neq 0$.

δ is called *normal* (or a *Efremovič proximity*) if it satisfies

Prox_6. $a(-\delta)b$ implies there is some c such that $a(-\delta)c$ and $-c(-\delta)b$.

The reader is invited to consult the standard text [10] for more information on proximities.

The *canonical frame* $\mathfrak{Cf}(B, \delta)$ of a BPA $\langle B, \delta\rangle$ is the structure $\langle \text{Prim}(B), R_\delta\rangle$, where $\text{Prim}(B)$ is the set of prime filters of B, and, for $F, G \in \text{Prim}(B)$, $F R_\delta G$ if and only if $F \times G \subseteq \delta$.

Theorem 3.2. *(See [7].) Suppose that $\langle B, \delta\rangle$ is a BPA.*

1. *R_δ is symmetric if and only if δ is symmetric.*
2. *R_δ is reflexive if and only if δ satisfies* Prox_5.
3. *R_δ is transitive if and only if δ satisfies* Prox_6.

Conversely, if R is a binary relation on X, we define a binary relation δ_R on the powerset algebra of X by $A\,\delta_R\,B$ if and only if there are $a \in A$, $b \in B$ such that aRb (see [7,11]). The pair $\langle 2^X, \delta_R\rangle$ is called the *complex algebra* of $\langle X, R\rangle$, denoted by $\mathfrak{Cm}(X, R)$. It is easy to see that the complex algebra of $\langle X, R\rangle$ is a BPA. However, the full strength of Theorem 3.2 cannot be kept:

Theorem 3.3. *(See [7].) Suppose that $\langle X, R\rangle$ is a frame.*

1. *If R is symmetric, then δ_R is symmetric.*
2. *If R is reflexive, then δ_R satisfies* Prox_5.
3. *If R is transitive, then δ_R satisfies* Prox_6.

None of the implications can be reversed.

It was shown in [12] that a binary relation R on the set of prime filters of some Boolean algebra B is of the form R_δ for some contact relation δ on B if and only if R is reflexive, symmetric and closed in the product topology of the Stone space of B.

For a BPA $\langle B, \delta\rangle$ let $h : B \to 2^{\text{Prim}(B)}$ be the Stone map defined by $h(a) = \{F\colon\ a \in F\}$. The following theorem shows that

Theorem 3.4 *(Discrete representation theorem for BAPS). (See [7].) Each BPA can be embedded into the complex algebra of its canonical frame.*

Proof. It suffices to show that $a\,\delta\,b \iff h(a)\,\delta_{R_\delta}\,h(b)$. Assume $a\,\delta\,b$. We need to show that for some $F \in h(a)$ and for some $G \in h(b)$, $F \times G \subseteq \delta$. Consider filters $F' = {\uparrow}a$ and $G' = {\uparrow}b$. They are proper filters, so by the prime filter theorem for Boolean algebras there are prime filters F and G such that $F' \subseteq F$ and $G' \subseteq G$. Let $a', b' \in B$ satisfy $a \leqslant a'$ and $b \leqslant b'$. Then $a' \in F$ and $b' \in G$. By axiom Prox_2, $a\,\delta\,b$ implies $a\,\delta(b \vee c)$ for any $c \in B$. In particular, we have $a\,\delta(b \vee b')$. Since $b \vee b' = b'$, we have $a\,\delta\,b'$. By axiom Prox_3, $a\,\delta\,b'$ implies $(a \vee c)\,\delta\,b'$ for any $c \in B$. In particular, $(a \vee a')\,\delta\,b'$. Since $a \vee a' = a'$, we have $a'\,\delta\,b'$.

Conversely, assume $h(a)\,\delta_{R_\delta}\,h(b)$. Then for some $F \in h(a)$ and some $G \in h(b)$, $F \times G \subseteq \delta$. It follows that for every $a' \in F$ and for every $b' \in G$, $a'\,\delta\,b'$. Since $a \in F$ and $b \in G$, we have $a\,\delta\,b$. □

For a frame $\langle X, R\rangle$ let $k : X \to \mathfrak{Cf}\mathfrak{Cm}(X, R)$ be defined by $k(x) = F_x$, where F_x is the principal filter of 2^X generated by $\{x\}$. Clearly, k is injective.

Theorem 3.5 *(Discrete representation theorem for proximity frames). Each frame can be embedded into the canonical frame of a BPA.*

Proof. We show that for $x, y \in X$, then xRy if and only if $k(x) R_{\delta_R} k(y)$.

"$\Rightarrow$": Let xRy; then, $\{x\}\,\delta_R\,\{y\}$. Suppose that $A \in k(x)$ and $A' \in k(y)$, i.e. $x \in A$ and $y \in A'$. Now, $A\,\delta_R\,A'$ if and only if there are $s \in A$ and $t \in A'$ such that sRt, and we may choose $s = x$ and $t = y$.

"$\Leftarrow$": Let $k(x) R_{\delta_R} k(y)$. Then, $F_x \times F_y \subseteq \delta_R$ by definition of R_{δ_R} and thus, $\{x\}\,\delta_R\,\{y\}$. The definition of δ_R now implies xRy. □

4. A discrete duality for BCAs based on clans

In this section we exhibit a discrete duality for BCAs following the representation theorem presented in [6] on the basis of clans.

Suppose that $\langle B, \mathcal{C}\rangle$ is a BCA. A non-empty subset Γ of B is called a *clan* if for all $a, b \in L$,

CL1. If $a, b \in \Gamma$ then $a\,\mathcal{C}\,b$.
CL2. If $a \vee b \in \Gamma$ then $a \in \Gamma$ or $b \in \Gamma$.
CL3. If $a \in \Gamma$ and $a \leqslant b$, then $b \in \Gamma$.

Lemma 4.1.

1. ([6], *Fact* 3.3.) *Each prime filter is a clan, and each clan is the union of all prime filters it contains.*
2. ([6], *Proposition* 3.3.) *If $a\,\mathcal{C}\,b$, then there is a clan Γ such that $a, b \in \Gamma$.*

Indeed, given the canonical frame $\langle \mathrm{Prim}(B), R\rangle$ on prime filters, each union of a clique of R is a clan. Here, a clique of R is a nonempty subset M of $\mathrm{Prim}(B)$ with $M \times M \subseteq R$. Conversely, if Γ is a clan and $F, G \subseteq \Gamma$, $F, G \in \mathrm{Prim}(B)$, then $F \times G \subseteq \mathcal{C}$ by definition of a clan, and therefore $\langle F, G\rangle \in R$. It follows that $\{F \in \mathrm{Prim}(B)\colon\ F \subseteq \Gamma\}$ is a clique. It may be observed, however, that different cliques of R may result in the same clan.

A topological representation theorem for BCAs such that the representation algebras are built from sets of clans was proved by Dimov and Vakarelov in [6].

We will now develop a representation built on frames. A *BC frame* is a pair $\langle X, \geqslant\rangle$ where X is a set and $\geqslant$ is a partial ordering of X. The complex algebra of $\langle X, R\rangle$ is the structure $\langle B_X, \vee_X, \wedge_X, -_X, 0_X, 1_X, \mathcal{C}_X\rangle$, denoted by $\mathfrak{Cm}(X, R)$, where for all $Y, Z \subseteq X$

$$B_X = \big\{Y \subseteq X\colon\ Y = \langle\geqslant\rangle[\geqslant](Y)\big\},$$
$$Y \vee_X Z = Y \cup Z,$$
$$Y \wedge_X Z = \langle\geqslant\rangle[\geqslant](Y \cap Z),$$
$$-_X A = \langle\geqslant\rangle[\geqslant](X \setminus A),$$
$$0_X = \emptyset,$$
$$1_X = X,$$

and, if $Y, Z \in B_X$,

$$Y\,\mathcal{C}_X\,Z \iff Y \cap Z \neq \emptyset.$$

Theorem 4.2. *$\mathfrak{Cm}(X, R)$ is a standard BCA.*

Proof. The elements of B_X are exactly the regular closed sets of the topology $\tau_{\geqslant}$ on X induced by $\geqslant$. It is well known that the regular closed sets of any topological space form a complete Boolean algebra under the operations given above, see e.g. [13], Theorem 1.37. □

If $\langle B, \mathcal{C}\rangle$ is a BCA, its canonical frame $\mathfrak{cf}(B, \mathcal{C})$ is the structure $\langle X_B, \supseteq\rangle$, where $X_B = \mathrm{Clan}(B)$. We can now prove the representation theorem.

Theorem 4.3 *(Discrete representation theorem for BCAs). Each BCA can be embedded into the complex algebra of its canonical frame.*

Proof. Let $h : B \to 2^{X_B}$ be the Stone mapping $h(a) = \{\Gamma \in X_B\colon\ a \in \Gamma\}$.

1. h is injective: If $a, b \in B$ and without loss of generality $a \not\leqslant b$, there is some prime filter F such that $a \in F$ and $b \notin F$ by the Prime Ideal Theorem, see e.g. [14]. Since every prime filter is a clan, $F \in h(a)$ and $F \notin h(b)$.
2. $h(0) = \emptyset$ and $h(1) = X_B$: Since no prime filter, hence, no clan, contains 0 we have $h(0) = \emptyset$. Since each clan contains 1, we obtain $h(1) = X_B$.
3. $h(a) = \langle\supseteq\rangle[\supseteq](h(a))$: Note that for every $Y \subseteq X$

$$\Gamma \in \langle\supseteq\rangle[\supseteq](Y) \iff (\exists\Delta)\Big[\Gamma \supseteq \Delta \text{ and } \underbrace{(\forall\Delta')\big(\Delta \supseteq \Delta' \Rightarrow \Delta' \in Y\big)}_{\Delta \in [\supseteq](Y)}\Big]. \tag{3}$$

   "$\subseteq$": Let $a \in \Gamma$. By Lemma 4.1 there is some prime filter F such that $a \in F$, i.e. $F \in h(a)$, and $F \subseteq \Gamma$, i.e. $F \in [\supseteq](\{\Gamma\})$. Since F is a minimal clan, $\supseteq(\{F\}) = \{F\} \subseteq h(a)$, and it follows from (2) that $F \in [\supseteq](h(a))$. Altogether, this shows that $\Gamma \in \langle\supseteq\rangle[\supseteq](h(a))$.
   "$\supseteq$": Suppose that $\Gamma \in \langle\supseteq\rangle[\supseteq](h(a))$. By (1), there is some clan Δ such that $\Delta \subseteq \Gamma$ and $\Delta \in [\supseteq](h(a))$. Since $\supseteq$ is reflexive, we have, in particular, $\Delta \in h(a)$. Now, $\Delta \subseteq \Gamma$ implies $a \in \Gamma$.
4. $h(a \vee b) = h(a) \cup h(b)$: This follows immediately from CL2 and CL3.
5. $h(-a) = \langle\supseteq\rangle[\supseteq](X \setminus h(a))$: First, note that $X \setminus h(a) = [\supseteq](X \setminus h(a))$: If $a \notin \Gamma$ and $\Gamma \subseteq \Delta$, then $a \notin \Delta$.
   "$\subseteq$": Let $-a \in \Gamma$. Then, there is some prime filter F such that $-a \in F$ and $F \subseteq \Gamma$. Since F is proper, $a \notin F$, and therefore, $F \in X \setminus h(a)$. Now, $F \subseteq \Gamma$ implies $\Gamma \in \langle\supseteq\rangle(X \setminus h(a))$.
   "$\supseteq$": Let $\Gamma \in \langle\supseteq\rangle(X \setminus h(a))$. Then, there is some $\Delta \in X \setminus h(a)$ such that $\Gamma \supseteq \Delta$. Since $a \notin \Delta$ and $a \vee -a = 1 \in \Delta$, it follows from CL2 that $-a \in \Delta$. Now, $\Gamma \supseteq \Delta$ implies $-a \in \Gamma$.
   Since h preserves $0, 1, -$ and $\vee$, it also preserves $\wedge$.
6. $a \mathcal{C} b$ if and only if $h(a) \cap h(b) \neq \emptyset$:
   "$\Rightarrow$": Let $a \mathcal{C} b$. By Lemma 4.1 there is a clan Γ containing a and b. Hence, $\Gamma \in h(a) \cap h(b)$.
   "$\Leftarrow$": If $\Gamma \in h(a) \cap h(b)$, then, $a, b \in \Gamma$ and therefore, $a \mathcal{C} b$ by CL1.

This completes the proof. □

Theorem 4.4 *(Discrete representation theorem for contact frames). Each contact frame can be embedded into the canonical frame of its complex algebra.*

Proof. Define $k : X \to \mathrm{Clan}(B_X)$ by $k(x) = \{Y \in B_X\colon\ x \in Y\}$.

1. k is well defined: We need to show that $k(x)$ is a clan of B_X. Clearly, $k(x)$ is closed under $\subseteq$, and thus it satisfies CL3. Next, let $Y, Y' \in B_X$ such that $Y \cup Y' \in k(x)$. Then, $x \in Y$ or $x \in Y'$ and therefore, $Y \in k(x)$ or $Y' \in k(x)$. This shows CL2. Finally, let $Y, Y' \in k(x)$. Then, $x \in Y \cap Y'$, and therefore, $Y \mathcal{C}_{B_X} Y'$. This shows CL1.
2. k is injective: Let $x \in X$. Then, $[\geqslant](\{x\}) = \{x\}$ since $\geqslant$ is reflexive, and therefore $\langle\geqslant\rangle[\geqslant](\{x\}) = \langle\geqslant\rangle(\{x\}) = \{z \in X\colon\ z \geqslant x\}$. Suppose that $y \in X$ and $x \neq y$. Since $\geqslant$ is antisymmetric, we may suppose without loss of generality that $y \not\geqslant x$.
   Clearly, $\langle\geqslant\rangle(\{x\}) \in k(x)$. Assume that $\langle\geqslant\rangle(\{x\}) \in k(y)$. Then, $y \in \langle\geqslant\rangle(\{x\})$ which implies $y \geqslant x$, contradicting our hypothesis.
3. k preserves $\geqslant$: Let $x \geqslant y$. We need to show that $k(x) \supseteq k(y)$. Therefore, let $Y \in k(Y)$. Then, from
   (a) $Y = \langle\geqslant\rangle(Y) = \{t \in X\colon\ (\exists z \in X)[t \geqslant z \text{ and } z \in Y]\}$,
   (b) $y \in Y$, and
   (c) $x \geqslant y$,
   we obtain $x \in \langle\geqslant\rangle(Y) = Y$. Hence, $Y \in k(x)$.

This completes the proof. □

Let us briefly look at the topological properties of the two representations. Suppose that τ is the topology on $\mathrm{Clan}(B)$ of [6] which has the family $\{h(a)\colon\ a \in B\}$ as a closed basis, and $\tau_{\geqslant}$ the topology generated by the partial order $\geqslant$ in which $Y \subseteq \mathrm{Clan}(B)$ is open if and only if $Y = [\geqslant](Y)$. If $\Gamma \in \mathrm{Clan}(B)$, then the smallest open set in $\tau_{\geqslant}$ containing Γ is $\downarrow\Gamma$.

Since each $h(a)$ is (regular) closed in $\tau_{\geqslant}$, and $h(a)$ is a closed basis of τ we have $\tau \subseteq \tau_{\geqslant}$. The latter topology may be much larger than τ: Consider the case that B is the finite–cofinite subalgebra of the powerset 2^{ω} of natural numbers, and that $\mathcal{C}$ is the overlap relation on B. Then, each clan is a prime filter, and the ordering on $\mathrm{Clan}(B)$ is discrete. Therefore, so is the topology $\tau_{\geqslant}$, and each subset of ω is open in $\tau_{\geqslant}$. On the other hand, τ is the usual Stone topology on $\mathrm{Prim}(B)$, which, in this case is the one-point compactification of a countable discrete space, and B is isomorphic to $\mathrm{RegCl}(\omega, \tau)$.

While τ is semiregular – i.e. it has an open basis of regular open sets – this is not necessarily true for a topology $\tau_{\geqslant}$: Consider the ordering on $X = \{x, y, z\}$ shown in Fig. 1. There, $\downarrow x = \{x, z\} \subsetneq \{x, y, z\} = \langle\geqslant\rangle\downarrow x = [\geqslant]\langle\geqslant\rangle\downarrow x$.

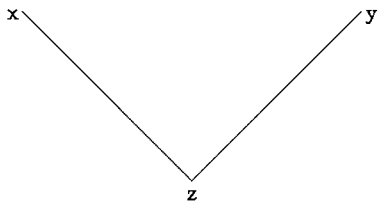


Fig. 1. An example.

5. A discrete representation of syllogistic ∀-algebras

In [1] an algebraic approach to the Aristotelian syllogistic is presented. The algebras proposed there may be seen as reducts of Boolean algebras endowed with some binary relations relevant for syllogistic. In the present section we consider one of those classes of algebras, namely, ∀-algebras, of the form $\langle L, \neg, \mathbf{\forall}\rangle$ with a binary relation which we denote with (a bold version of) the symbol of the universal quantifier for the following reason: In the abstract setting, the ∀-algebras are defined with the set of axioms which say that the relation $\mathbf{\forall}$ is reflexive and transitive, and determine its action on the negated elements. The axioms are chosen so that in case L is a set algebra and $\neg$ is the operator of set complementation, i.e. $\neg = -$, the relation $\mathbf{\forall}$ may be interpreted as set inclusion. From a logical perspective, sets correspond to unary predicates, and set inclusion $A \subseteq B$ is defined with the well known formula $(\forall z)[A(z) \Rightarrow B(z)]$. In this formula, the quantifier $\forall$ is a quantifier with a restricted scope.

Following [1], with some abuse of language we call a structure $\langle L, \neg, \mathbf{\forall}\rangle$ an ∀-*algebra*, if L is a nonempty set, $\neg$ is a unary operation on L and $\mathbf{\forall}$ a binary relation on L such that the following properties are fulfilled for all $a, b, c \in L$:

$\forall_1$. $a \mathbf{\forall} \neg\neg a$.
$\forall_2$. $\neg\neg a \mathbf{\forall} a$.
$\forall_3$. $a \mathbf{\forall} b$ and $b \mathbf{\forall} c$ imply $a \mathbf{\forall} c$.
$\forall_4$. $a \mathbf{\forall} b$ implies $\neg b \mathbf{\forall} \neg a$.
$\forall_5$. $a \mathbf{\forall} \neg a$ implies $a \mathbf{\forall} b$.

Note that $\forall_1$–$\forall_3$ imply that $\mathbf{\forall}$ is reflexive and transitive, and therefore, $[\mathbf{\forall}]$ is an interior operator. A primary example of an ∀-algebra is a structure $\mathfrak{Cm}(X) = \langle 2^X, -, \subseteq\rangle$, which we call the *complex algebra of* X.

If L is an ∀-algebra, $S \subseteq L$ is called $\mathbf{\forall}$-*closed*, if $S = [\mathbf{\forall}](S)$. A nonempty subset S of L is called an ∀-*set*, if it is $\mathbf{\forall}$-closed and satisfies

$$a, b \in S \quad \Rightarrow \quad a(-\mathbf{\forall})\neg b. \tag{4}$$

For later use we note the following easy fact:

Lemma 5.1. *The intersection of two* ∀*-sets is an* ∀*-set.*

Let X'_L be the collection of all ∀-sets. Observe that L is not an ∀-set, since L is nonempty and $\forall_1$ holds. Furthermore, X'_L may be empty, e.g. if $\mathbf{\forall}$ is the universal relation on L, in particular, if $|L| = 1$.

Example 1. Suppose that $\langle L, \neg, \mathbf{\forall}\rangle$ is an ∀-algebra, and $a, a' \notin L$, $a \neq a'$. Let $L' = L \cup \{a, a'\}$, and extend $\mathbf{\forall}$ and $\neg$ by

$$\mathbf{\forall}' = \mathbf{\forall} \cup \big(\{a\} \times L'\big) \cup \big(L' \times \big\{a'\big\}\big), \tag{5}$$

$$\neg' = \neg \cup \big\{\langle a, a'\rangle, \langle a', a\rangle\big\}. \tag{6}$$

Then, $\langle L', \mathbf{\forall}', \neg'\rangle$ is an ∀-algebra:

$\forall_1$: If $b \in L$, then $\neg' b = \neg b$ and $b \mathbf{\forall} \neg\neg b$. Otherwise, $a \mathbf{\forall}' \neg'\neg' a$, since $a = \neg'\neg' a$ and $a \mathbf{\forall}' a$. Similarly, $a' \mathbf{\forall}' \neg'\neg' a'$.

$\forall_2$: Similarly.

$\forall_3$: Let $b \mathbf{\forall}' c \mathbf{\forall}' d$. If $b = a$, then $b \mathbf{\forall}' d$ since $\{a\} \times L' \subseteq \mathbf{\forall}'$. If $b = a'$, then $c = d = a'$. Thus, let $b \notin \{a, a'\}$; then, $c \neq a$. If $c = a'$, then $a' \mathbf{\forall}' d$ implies $d = a'$, and therefore, $b \mathbf{\forall}' d$ since $L' \times \{a'\} \subseteq \mathbf{\forall}'$. Thus, let $\{b, c\} \cap \{a, a'\} = \emptyset$; then, $d \neq a$. If $d = a'$, then, as before, $b \mathbf{\forall}' d$. Otherwise, $b \mathbf{\forall} c \mathbf{\forall} d$, and $b \mathbf{\forall} d$ by $\forall_3$.

$\forall_4$ and $\forall_5$ follow immediately from the definition of $\mathbf{\forall}'$ and $\neg'$.

Observe that $\{a'\}$ is an ∀-set, and that no ∀-set contains a. □

The following lemma characterizes when an element of an ∀-algebra is contained in an ∀-set:

Lemma 5.2. *Let* $a \in L$*. Then,* $\mathbf{\forall}(a)$ *is an* ∀*-set if and only if* $\{a\} \times L \nsubseteq \mathbf{\forall}$*. Furthermore,* $\mathbf{\forall}(a)$ *is a subset of each* ∀*-set containing* a*.*

Proof. "⇒": Suppose that $\mathbf{\forall}(a)$ is an $\forall$-set. Then, $a(-\mathbf{\forall})\neg a$ which shows that $\{a\} \times L \not\subseteq \mathbf{\forall}$.

"⇐": Suppose that $\mathbf{\forall}(a)$ is not an $\forall$-set. Since $\mathbf{\forall}(a)$ is $\mathbf{\forall}$-closed, there are $b, c \in L$ such that $a \mathbf{\forall} b$, $a \mathbf{\forall} c$, and $b \mathbf{\forall} \neg c$. Then, $a \mathbf{\forall} \neg c$ by transitivity of $\mathbf{\forall}$, and therefore, $\neg\neg c \mathbf{\forall} \neg a$ by $\forall_4$. From $\forall_1$ we obtain $c \mathbf{\forall} \neg\neg c$, and, again by transitivity, we have $c \mathbf{\forall} \neg a$. Using $a \mathbf{\forall} c$ and again transitivity we obtain $a \mathbf{\forall} \neg a$, and therefore, $a \mathbf{\forall} d$ for all $d \in L$ by $\forall_5$.

If S is an $\forall$-set containing a, then the fact that S is $\mathbf{\forall}$-closed implies that $\mathbf{\forall}(a) \subseteq S$. □

Lemma 5.3. *$\mathbf{\forall}$ is the universal relation if and only if there is some $a \in L$ such that $(\{a\} \times L) \cup (\{\neg a\} \times L) \subseteq \mathbf{\forall}$.*

Proof. The ⇒ direction is obvious. Conversely, let $(\{a\} \times L) \cup (\{\neg a\} \times L) \subseteq \mathbf{\forall}$, and $b, c \in L$. Then, $a \mathbf{\forall} b$, $a \mathbf{\forall} \neg b$, and $\neg a \mathbf{\forall} c$ by the hypothesis. From $\forall_4$ we obtain $\neg\neg b \mathbf{\forall} \neg a$, and by (4) we have $b \mathbf{\forall} \neg\neg b$. Thus,

$$b \mathbf{\forall} \neg\neg b \mathbf{\forall} \neg a \mathbf{\forall} c,$$

and the claim follows from transitivity of $\mathbf{\forall}$. □

If X'_L is nonempty, then, clearly, X'_L is closed under union of chains, and thus, each element of X'_L is contained in a maximal element. Let X_L be the set of all maximal elements of X'_L which we call the *canonical frame of L* and denote it by $\mathfrak{Cf}(L)$. Our aim is to show under which conditions an $\forall$-algebra can be embedded into the complex algebra of its canonical frame. This will follow from a sequence of lemmas.

Lemma 5.4.

1. *If $S \in X'_L$, then there is no $a \in L$ such that $a \in S$ and $\neg a \in S$.*
2. *If $S \in X_L$, then $a \in S$ or $\neg a \in S$ for every $a \in L$.*

Proof. 1. Assume that $S \in X'_L$ and $a, \neg a \in S$. By (4), $a(-\mathbf{\forall})\neg\neg a$, contradicting $\forall_1$.

2. Let S be a maximal $\forall$-set, and assume there is some $a \in L$ such that $\neg a \notin S$. Let $S' = S \cup \mathbf{\forall}(a)$. If we can show that S' is an $\forall$-set, then $S = S'$ because of the maximality of S, and it follows that $a \in S$. Since $[\mathbf{\forall}]$ is an interior operator, we have $[\mathbf{\forall}](S') \subseteq S'$. Conversely, let $b \in S'$, and $b \mathbf{\forall} c$; we need to show that $c \in S'$. If $b \in S$, then $c \in S \subseteq S'$ since $[\mathbf{\forall}](S) = S$. Otherwise, $a \mathbf{\forall} b$ since $S' = S \cup \mathbf{\forall}(a)$ and $b \in S'$, and now the transitivity of $\mathbf{\forall}$ implies $a \mathbf{\forall} c$; hence, $c \in S'$.

Next, let $c, d \in S'$ and assume that $c \mathbf{\forall} \neg d$. Then, not both c, d can be in S. Suppose that $c \notin S$; then, $a \mathbf{\forall} c$. By transitivity of $\mathbf{\forall}$ we get $a \mathbf{\forall} \neg d$ which implies $\neg d \in S'$, a contradiction with 1. Similarly, if $d \notin S$ then $a \mathbf{\forall} \neg d$, a contradiction. Now since S' is an $\forall$-set and S is maximal, necessarily $S = S'$. It follows that $a \in S$ as required. □

Lemma 5.5. *Suppose that $h : L \to 2^{X_L}$ is defined by $h(a) = \{S \in X_L : a \in S\}$. Then, $[\subseteq](h(a)) = h(a)$, $a \mathbf{\forall} b$ if and only if $h(a) \subseteq h(b)$, and $h(\neg a) = X_L \setminus h(a)$.*

Proof. We first show that $[\subseteq](h(a)) = h(a)$. It is sufficient to consider $\supseteq$, since $\subseteq$ is a partial order: If $S \in h(a)$ and $S \subseteq T$, then clearly $a \in T$.

Next, let $a \mathbf{\forall} b$, and $S \in h(a)$, i.e. $a \in S$. Since $S \in X_L$ we see that, in particular, S is $\mathbf{\forall}$-closed, and therefore, $b \in S$ which implies $S \in h(b)$. Conversely, suppose that $a(-\mathbf{\forall})b$. Then, $\{a\} \times L \not\subseteq \mathbf{\forall}$, and therefore, $\mathbf{\forall}(a)$ is an $\forall$-set by Lemma 5.2. Assume that $\neg b \mathbf{\forall} \neg a$; then, by $\forall_4$, $\neg\neg a \mathbf{\forall} \neg\neg b$. Furthermore, $a \mathbf{\forall} \neg\neg a$ by $\forall_1$, and $\neg\neg b \mathbf{\forall} b$ by $\forall_2$, so that

$$a \mathbf{\forall} \neg\neg a \mathbf{\forall} \neg\neg b \mathbf{\forall} b. \tag{7}$$

Transitivity of $\mathbf{\forall}$ now implies $a \mathbf{\forall} b$, contradicting the assumption. Therefore, $\neg b(-\mathbf{\forall})\neg a$, and thus, $\mathbf{\forall}(\neg b)$ is an $\forall$-set as well. Now, by Lemma 5.1, $S = \mathbf{\forall}(a) \cap \mathbf{\forall}(\neg b)$ is an $\forall$-set, and so there is a maximal $\forall$-set S' containing S. By reflexivity of $\mathbf{\forall}$ we have $a, \neg b \in S'$. Hence, $S' \in h(a)$, and $b \notin S'$ by Lemma 5.4(1). It follows that $h(a) \not\subseteq h(b)$.

Finally, we show that $h(\neg a) = X_L \setminus h(a)$: Suppose that $S \in X_L$.

"⊆": Let $\neg a \in S$. Then, by Lemma 5.4(1), we have $a \notin S$, i.e. $S \notin h(a)$.

"⊇": Let $a \notin S$. Then, by Lemma 5.4(2), we have $\neg a \in S$. □

In general, h is not injective. We have, however,

Theorem 5.6. *h is injective if and only if $\mathbf{\forall}$ is a partial order.*

Proof. "⇒": Suppose that $h(a) = h(b)$ implies $a = b$. All we need to show is that $\mathbf{\forall}$ is antisymmetric. Suppose that there are $a, b \in L$ such that $a \mathbf{\forall} b$ and $b \mathbf{\forall} a$. If $S \in X_L$ and $a \in S$, then $a \mathbf{\forall} b$ and the fact that S is $\mathbf{\forall}$-closed imply that $b \in S$. It follows that $h(a) \subseteq h(b)$, and, similarly, $b \mathbf{\forall} a$ implies $h(b) \subseteq h(a)$. Injectivity of h now implies $a = b$.

"⇐": Suppose that $\mathbf{\forall}$ is antisymmetric and $a \neq b$. Then, $a(-\mathbf{\forall})b$ or $b(-\mathbf{\forall})a$, suppose without loss of generality the former. As shown in the proof of Lemma 5.5, $\neg b\,\mathbf{\forall}\neg a$ implies $a\,\mathbf{\forall}\,b$, and therefore, by contraposition, $a(-\mathbf{\forall})b$ implies $\neg b(-\mathbf{\forall})\neg a$. Thus, $\mathbf{\forall}(a)$ and $\mathbf{\forall}(\neg b)$ are $\forall$-sets by Lemma 5.2, and therefore, so is $S = \mathbf{\forall}(a) \cap \mathbf{\forall}(\neg b)$ by Lemma 5.1. It follows that any maximal $\forall$-set S' with $S \subseteq S'$ contains a and $\neg b$. By Lemma 5.4(1), $S' \in h(a)$ and $S' \notin h(b)$. □

The following is now obvious:

Corollary 5.7. *An $\forall$-algebra $\langle L, \neg, \mathbf{\forall}\rangle$ can be embedded into the complex algebra of its canonical frame if and only if $\mathbf{\forall}$ is a partial order.*

The $\mathbf{\forall}$-relation on the complex algebra of an $\forall$-frame is set inclusion which is a partial order. Thus, the next result comes as no surprise:

Theorem 5.8 *(Discrete representation theorem for $\forall$-frames). Every $\forall$-frame is embeddable into the canonical frame of its complex algebra.*

Proof. The complex algebra of X is the structure $\mathfrak{Cm}(X) = \langle 2^X, -, \subseteq\rangle$, and the elements of $\mathfrak{Cf}\mathfrak{Cm}(X)$ are the maximal $\forall_X$-sets, that is, those subsets K of 2^X maximal with the properties that

1. If $Y \in K$ and $Y \subseteq Y'$, then $Y' \in K$.
2. If $Y, Z \in K$, then $Y \cap Z \neq \emptyset$.

Let $k : X \to 2^{2^X}$ be defined by $k(x) = \{Y \subseteq X\colon\ x \in X\}$. Then, k is injective, and $k(x)$ satisfies (1) and (2) since it is a maximal filter of the set algebra 2^X. □

6. Discrete representations of $\forall\exists$-structures

In this section we consider the structures with two relations $\mathbf{\forall}$ and $\mathbf{\exists}$ interpreted as universal and existential quantifier, respectively, of restricted scope.

Following [1] by an $\forall\exists$-*algebra* we mean a structure $\langle L, \mathbf{\forall}, \mathbf{\exists}\rangle$ such that $L \neq \emptyset$, and $\mathbf{\forall}$, $\mathbf{\exists}$ are binary relations on L satisfying for all $a, b \in L$

$\forall\exists_1$ $\mathbf{\forall}$ is reflexive
$\forall\exists_2$ $\mathbf{\forall}$ is transitive
$\forall\exists_3$ $a\,\mathbf{\forall}\,b$ and $a\,\mathbf{\exists}\,c$ imply $c\,\mathbf{\exists}\,b$
$\forall\exists_4$ $a\,\mathbf{\exists}\,b$ implies $a\,\mathbf{\exists}\,a$
$\forall\exists_5$ $a\,\mathbf{\exists}\,a$ or $a\,\mathbf{\forall}\,b$.

Note that by $\forall\exists_1$ and $\forall\exists_3$, the relation $\mathbf{\exists}$ is symmetric. Although it is not natural to qualify a structure without operations as an algebra, we follow [1] in this respect because it enables us to formulate relationships between $\forall$-algebras and $\forall\exists$-structures.

An $\forall\exists$-*frame* is just a Boolean frame $X \neq \emptyset$.

The *complex algebra of* X is the structure $(2^X, \mathbf{\forall}_X, \mathbf{\exists}_X)$ such that for all $A, B \subseteq X$,

$$A\mathbf{\forall}_X B \quad \text{iff} \quad A \subseteq B,$$
$$A\mathbf{\exists}_X B \quad \text{iff} \quad A \cap B \neq \emptyset.$$

The *canonical frame of an* $\forall\exists$-*algebra* L is the set $X_L = \{A \subseteq L\colon\ A = [\mathbf{\forall}]A \text{ and } A \times A \subseteq \mathbf{\exists}\}$. The elements of X_L are referred to as $\forall\exists$-sets.

$\forall\exists$-algebras and $\forall$-algebras considered in Section 5 are related as presented in the following lemmas based on [1].

Lemma 6.1. *Given an $\forall$-algebra $\langle L, \neg, \mathbf{\forall}\rangle$, define a binary relation $\mathbf{\exists}$ on L by $a\,\mathbf{\exists}\,b$ if and only if $a(-\mathbf{\forall})\neg b$. Then $\langle L, \mathbf{\forall}, \mathbf{\exists}\rangle$ is an $\forall\exists$-algebra.*

Proof. If suffices to show that the axioms $\forall\exists_1, \ldots, \forall\exists_5$ are satisfied. Note that with the assumed definition of the relation $\mathbf{\exists}$, the axiom $\forall\exists_3$ has the form

$$a\,\mathbf{\forall}\,b \quad \text{and} \quad c\,\mathbf{\forall}\neg b \quad \Rightarrow \quad a\,\mathbf{\forall}\neg c.$$

To show that this form of $\forall\exists_3$ holds, assume $a\,\mathbf{\forall}\,b$ and $c\,\mathbf{\forall}\neg b$. By $\forall_4$ we get $\neg\neg b\,\mathbf{\forall}\neg c$ which together with $\forall_1$ gives $b\,\mathbf{\forall}\neg c$ by $\forall_3$. Applying $\forall_3$ again we obtain $a\,\mathbf{\forall}\neg c$ as required. $\forall\exists_4$ and $\forall\exists_5$ follow from $\forall_5$. □

Conversely, suppose that $\langle L, \forall, \exists\rangle$ is an $\forall\exists$-algebra. Let $f : L \to L'$ be a 1–1 map of L onto a set L' with $L \cap L' = \emptyset$, and let $L \cup L'$ be the (disjoint) union of L and its "copy" L'. We define an algebra $(L \cup L', \neg, \forall')$ by

$$\neg a = \begin{cases} f(a) & \text{if } a \in L \\ f^{-1}(a) & \text{if } a \in L' \end{cases}$$

$$a \forall' b \text{ if and only if } \begin{cases} a \forall b & \text{if } a, b \in L \\ f^{-1}(b) \forall f^{-1}(a) & \text{if } a, b \in L' \\ a(-\exists) f^{-1}(b) & \text{if } a \in L,\ b \in L' \\ \text{false} & \text{if } a \in L',\ b \in L \end{cases}$$

Lemma 6.2. *Let $\langle L, \forall, \exists\rangle$ be an $\forall\exists$-algebra. For all $a, b \in L$ the following conditions are equivalent:*

1. *$a \exists b$ holds in L.*
2. *$a \forall' \neg b$ does not hold in $L \cup L'$.*

Proof. Let $a, b, \in L$. Then $\neg b = f(b) \in L'$. By definition of $\forall'$, a $\forall' f(b)$ if and only if a $(-\exists) f^{-1}(f(b))$ and the latter holds if and only if a $(-\exists)b$. Hence the required equivalence holds. □

The following two lemmas provide some "extreme" examples of $\forall\exists$-algebras.

Lemma 6.3. *Let $\langle L, \forall, \exists\rangle$ be an $\forall\exists$-algebra.*

1. *If $\forall$ is the universal relation, then $\exists$ is universal or empty.*
2. *If $\exists \cap Id_L = \emptyset$, then $\forall$ is universal, where Id_L is the identity on L.*

Proof.

1. Assume that $\forall$ is universal and suppose that $\exists \neq \emptyset$, say $a \exists b$ holds. Since $\exists$ is symmetric, $b \exists a$. Let $c \in L$. Then $b \forall c$ and by $\forall_3$ we get $a \exists c$. It follows that $\{a\} \times L \subseteq \exists$. Furthermore, symmetry of $\exists$ and universality of $\forall$ imply that $\exists$ is universal.
2. If $\exists \cap Id_L = \emptyset$, then $a(-\exists)a$ for all $a \in L$. Hence by $\forall\exists_5$, for all $a, b \in L$ we have $a \forall b$.

This completes the proof. □

Lemma 6.4. *Let $\langle L, \forall, \exists\rangle$ be an $\forall\exists$-algebra. Then $\exists$ is reflexive if and only if $\forall \subseteq \exists$.*

Proof. If $Id_L \subseteq \exists$, then since $\exists$ is symmetric we get $\forall = Id_L ; \forall \subseteq \exists ; \forall = \exists^{\smile} ; \forall \subseteq \exists$ where the latter inclusion follows from $\forall\exists_3$. Conversely, if $\forall \subseteq \exists$, then $Id_L \subseteq \exists$ since $\forall$ is reflexive. □

Similarly as in Section 5 we show that a discrete representation theorem can be proved for the class of $\forall\exists$-algebras such that the relation $\forall$ is a partial order.

Lemma 6.5. *For all $a, b \in L$, $a \exists b$ implies $\forall(a) \times \forall(b) \subseteq \exists$.*

Proof. First, observe that $\forall^{\smile} ; \exists ; \forall \subseteq \exists$ because the symmetry of $\exists$ and axiom $\forall\exists_3$ imply that $\forall^{\smile} ; \exists \subseteq \exists$, and therefore using $\forall\exists_3$ again we obtain $\forall^{\smile} ; \exists ; \forall \subseteq \exists ; \forall \subseteq \exists$. Now assume $a \exists b$ and let $c \in \forall(a)$ and $d \in \forall(b)$. Thus $a \forall c$ and $b \forall d$. Then $(c, d) \in \forall^{\smile} ; \exists ; \forall \subseteq \exists$. □

Lemma 6.6. *Let $a \in L$ and $a \exists a$. Then $\forall(a)$ is the smallest $\forall\exists$-set containing a.*

Proof. Since $\forall$ is reflexive, we have $[\forall]\forall(a) \subseteq \forall(a)$. Conversely, assume that $b \in \forall(a)$ and $c \in \forall(b)$, that is $a \forall b$ and $b \forall c$. By transitivity of $\forall$ we have $a \forall c$ and hence $\forall(b) \subseteq \forall(a)$. It follows that $\forall(a) \subseteq [\forall]\forall(a)$. Next, by $a \exists a$ and Lemma 6.5, $\forall(a) \times \forall(a) \subseteq \exists$. Thus $\forall(a)$ is an $\forall\exists$-set. Clearly, any $\forall\exists$-set containing a must contain $\forall(a)$ as a subset. □

Let $h : L \to 2^{X_L}$ be defined by $h(a) = \{A \in X_L :\ a \in A\}$.

Lemma 6.7.

1. $a \mathbf{\forall} b$ *if and only if* $h(a) \subseteq h(b)$,
2. $a \mathbf{\exists} b$ *if and only if* $h(a) \cap h(b) \neq \emptyset$.

Proof.

1. Assume $a \mathbf{\forall} b$ and take $A \in X_L$ such that $a \in A$. Since $A \times A \subseteq \mathbf{\exists}$, $a \mathbf{\exists} a$. By Lemma 6.6 $\mathbf{\forall}(a) \subseteq A$ and hence $b \in A$ as required. Conversely, consider the following two cases. If $a(-\mathbf{\exists})a$, then by $\forall\exists_5$ we have $a \mathbf{\forall} b$. If $a \mathbf{\exists} a$, then by Lemma 6.6 we have $\mathbf{\forall}(a) \in h(a)$ and hence $\mathbf{\forall}(a) \in h(b)$ which implies $a \mathbf{\forall} b$.
2. Assume $a \mathbf{\exists} b$. By symmetry of $\mathbf{\exists}$ we have $b \mathbf{\exists} a$ and by $\forall\exists_4$ we get $a \mathbf{\exists} a$ and $b \mathbf{\exists} b$. By Lemma 6.6 $\mathbf{\forall}(a)$, $\mathbf{\forall}(b) \in X_L$. Consider $A = \mathbf{\forall}(a) \cup \mathbf{\forall}(b)$. We show that $A \in X_L$, that is $[\mathbf{\forall}]A = A$ and $A \times A \subseteq \mathbf{\exists}$. $[\mathbf{\forall}]A \subseteq A$ by reflexivity of $\mathbf{\forall}$. Conversely, $\mathbf{\forall}(a) \cup \mathbf{\forall}(b) = [\mathbf{\forall}]\mathbf{\forall}(a) \cup [\mathbf{\forall}]\mathbf{\forall}(b) \subseteq [\mathbf{\forall}](\mathbf{\forall}(a) \cup \mathbf{\forall}(b))$. Now, from $a \mathbf{\exists} a$, $b \mathbf{\exists} b$, and Lemma 6.5 we have $\mathbf{\forall}(a) \times \mathbf{\forall}(a) \subseteq \mathbf{\exists}$ and $\mathbf{\forall}(b) \times \mathbf{\forall}(b) \subseteq \mathbf{\exists}$. From $a \mathbf{\exists} b$ and Lemma 6.5 $\mathbf{\forall}(a) \times \mathbf{\forall}(b) \subseteq \mathbf{\exists}$. By symmetry of $\mathbf{\exists}$, $\mathbf{\forall}(b) \times \mathbf{\forall}(a) \subseteq \mathbf{\exists}$. □

Lemma 6.8. *The mapping h is injective if and only if $\mathbf{\forall}$ is a partial order.*

Proof. Assume that h is injective and let $a \mathbf{\forall} b$ and $b \mathbf{\forall} a$. Then $\mathbf{\forall}(a) = \mathbf{\forall}(b)$. If $A \in X_L$ and $a \in A$, then $\mathbf{\forall}(a) \subseteq A$ and hence $\mathbf{\forall}(b) \subseteq A$. Since $\mathbf{\forall}$ is reflexive, $b \in A$. Thus $h(a) \subseteq h(b)$. Similarly, we obtain $h(b) \subseteq h(a)$. By injectivity of h, $a = b$.

Conversely, assume that $\mathbf{\forall}$ is antisymmetric and let $a, b \in L$ with $a \neq b$. We may assume without loss of generality that $a(-\mathbf{\forall})b$. Then $b \notin \mathbf{\forall}(a)$, so $\mathbf{\forall}(a) \notin h(b)$. Clearly, $\mathbf{\forall}(a) \in h(a)$, and hence $h(a) \neq h(b)$. □

Lemmas 6.7 and 6.8 lead to the following theorem:

Theorem 6.9. *The $\forall\exists$-algebra $\langle L, \mathbf{\forall}, \mathbf{\exists}\rangle$ is embeddable into the complex algebra of its canonical frame if and only if the relation $\mathbf{\forall}$ is a partial order.*

Theorem 6.10 *(Discrete representation theorem for $\forall\exists$-frames).*
Every $\forall\exists$-frame is embeddable into the canonical frame of its complex algebra.

Proof. Consider $k : X \to 2^{2^X}$ defined as $k(x) = \{A \subseteq X\colon\ x \in A\}$. We show that k is well defined, that is $k(x) = [\mathbf{\forall}_X]k(x)$ and $k(x) \times k(x) \subseteq \mathbf{\exists}_X$. Indeed, if $x \in A$, then $x \in B$ for every $B \supseteq A$. Similarly, if $x \in A$ and $x \in B$ then clearly, $A \cap B \neq \emptyset$. Next, note that k is injective because if for every $A \subseteq X$, $x \in A$ if and only if $y \in A$, then considering $A = \{x\}$ we get $x = y$. □

7. Conclusion and outlook

In the first part of the paper we presented a discrete duality between Boolean algebras endowed with a proximity (respectively contact) relation and their corresponding frames. These developments extended some of the existing representation theorems for algebras of spatial reasoning to representation theorems for both algebras and frames such that the topology is not needed for the construction of representation structures into which the given structures are embeddable.

In the second part of the paper we considered some of the algebras related to Aristotelian syllogistic presented in [1]. We presented a discrete representation theorem for $\forall$-algebras with an antisymmetric relation $\forall$ and discrete representation theorems for $\forall\exists$-algebras with an antisymmetric $\mathbf{\forall}$ and for $\forall\exists$-frames.

There are various interesting issues for further work. The facts that in the complex algebras of $\forall\exists$-algebras the relation $\forall$ is the inclusion of sets and the relation $\exists$ is the overlap relation inspires a systematic study of relationships between syllogistic algebras and algebras of spatial reasoning. In the present paper we considered only two out of six classes of syllogistic algebras discussed in [1]. Discrete dualities for the remaining classes are also worth further study.

Acknowledgement

The authors gratefully acknowledge the helpful comments of the anonymous referees.

References

[1] J.C. Shepherdson, On the interpretation of Aristotelian syllogistic, J. Symb. Log. 21 (1956) 137–147.
[2] M. Aiello, J. van Benthem, I. Pratt-Hartmann (Eds.), Handbook of Spatial Logics, Springer-Verlag, Heidelberg, 2007.
[3] D. Vakarelov, Region-based theory of space: Algebras of regions, representation theory, and logics, in: D. Gabbay, S. Goncharov, M. Zakharyaschev (Eds.), Mathematical Problems from Applied Logic, vol. 2, Springer-Verlag, Heidelberg, 2007, pp. 267–348.
[4] E. Orłowska, I. Rewitzky, Discrete duality and its applications to reasoning with incomplete information, in: Proceedings of the International Conference on Rough Sets and Intelligent Systems Paradigms, in: Lecture Notes in Artificial Intelligence, vol. 4585, Springer, 2007, pp. 51–56.
[5] I. Düntsch, M. Winter, A representation theorem for Boolean contact algebras, Theor. Comput. Sci. (B) 347 (2005) 498–512.

[6] G. Dimov, D. Vakarelov, Contact algebras and region – based theory of space: A proximity approach – I, Fundam. Inform. 74 (2006) 209–249.
[7] I. Düntsch, D. Vakarelov, Region-based theory of discrete spaces: A proximity approach, Ann. Math. Artif. Intell. 49 (2007) 5–14.
[8] J. van Benthem, Correspondence theory, in: D. Gabbay, F. Guenthner (Eds.), Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic, Reidel, Dordrecht, 1984, pp. 167–247.
[9] G. Birkhoff, Lattice Theory, 2nd ed., Am. Math. Soc. Colloquium Publications, vol. 25, AMS, Providence, 1948.
[10] S.A. Naimpally, B.D. Warrack, Proximity Spaces, Cambridge University Press, Cambridge, 1970.
[11] A. Galton, The mereotopology of discrete space, in: C. Freksa, D. Mark (Eds.), Spatial Information Theory. Proceedings of the International Conference COSIT'99, in: Lecture Notes in Computer Science, vol. 1661, Springer-Verlag, Heidelberg, 1999, pp. 251–266.
[12] I. Düntsch, M. Winter, The lattice of contact relations on a Boolean algebra, in: R. Berghammer, B. Möller, G. Struth (Eds.), Proceedings of the 10th International Conference on Relational Methods in Computer Science and the 5th International Workshop on Applications of Kleene Algebra, in: Lecture Notes in Computer Science, vol. 4988, Springer-Verlag, Heidelberg, 2008, pp. 99–109.
[13] S. Koppelberg, General Theory of Boolean Algebras, Handbook on Boolean Algebras, vol. 1, North-Holland, 1989.
[14] R. Balbes, P. Dwinger, Distributive Lattices, University of Missouri Press, Columbia, 1974.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 180–193

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Inference engine based on closure and join operators over Truth Table Binary Relations

Samir Elloumi [a], Bilel Boulifa [a], Ali Jaoua [a,*], Mohammad Saleh [a], Jameela Al Otaibi [a], Marcelo F. Frias [b]

[a] Department of Computer Science, Qatar University, Qatar
[b] Department of Software Engineering, Instituto Tecnológico de Buenos Aires, CONICET, Argentina

ARTICLE INFO

Article history:
Available online 12 February 2014

Keywords:
Truth Table Binary Relation
Relation combining
Closure operators
Formal concept analysis
Inference engine
Cooperative conceptual reasoning

ABSTRACT

We propose a conceptual reasoning method for an inference engine. Starting from a knowledge base made of decision rules, we first map each rule to its corresponding Truth Table Binary Relation (TTBR), considered as a formal context. Objects in the domain of TTBR correspond to all possible rule interpretations (in terms of their truth value assignments), and elements in the range of TTBR correspond to the attributes. By using the 'natural join' operator in the 'ContextCombine' Algorithm, we combine all truth tables into a global relation which has the advantage of containing the complete knowledge of all deducible rules. By conceptual reasoning using closure operators, from the initial rules we obtain all possible conclusions with respect to the global relation. We may then check if expected goals are among these possible conclusions. We also provide an approximate solution for the exponential growth of the global relation, by proposing modular and cooperative conceptual reasoning. We finally present experimental results for two case studies and discuss the effectiveness of our approach.

1. Introduction

Automated reasoning has applications in domains such as automated theorem proving [1], software verification [2] and model finding [3]. Automated reasoning also plays an important role in expert systems [4,5], where techniques such as forward chaining, backward chaining, mixed or structural approaches are employed [6]. In this paper, for each decision rule involving some set of terms or attributes P, we create its truth table as a binary relation R over all involved attributes. We then combine all tables in a single one using our proposed 'ContextCombine' Algorithm. The latter explores all possibilities between the different assignments to generate a global solution in a new truth table. By using a Galois connection on the entire table thus obtained, we are able to infer all possible conclusions related with some input facts. Furthermore, we are able to regenerate all implications from this new context [7]. This procedure may produce tables whose size grows beyond reasonable limits. To cope with this limitation of the technique we propose a modular and cooperative reasoning approach that delays combination operations, by first reducing the different initial contexts associated with the different rules with respect to the facts initially submitted to the inference engine.

The article is organized as follows. In Section 2 we present the foundations of formal concept analysis. In Section 3 we propose *conceptual reasoning*, a reasoning method based on Galois connection. In Section 4 we propose a modular

* Corresponding author.
E-mail addresses: eloumis@qu.edu.qa (S. Elloumi), boulifa.bilel@gmail.com (B. Boulifa), jaoua@qu.edu.qa (A. Jaoua), Mohd.saleh@qu.edu.qa (M. Saleh), jameelaa@gmail.com (J. Al Otaibi), mfrias@itba.edu.ar (M.F. Frias).

http://dx.doi.org/10.1016/j.jlap.2014.02.007
2352-2208/

and cooperative reasoning approach which allows us to partially overcome the limitations of conceptual reasoning. Initial experimentation using this approach shows its efficiency in the two case studies (cf. Section 5.1 and Section 5.2) related to SAT/UNSAT problems and Medical Data, respectively. Finally, in Section 6, we draw some conclusions and present some proposals for further work.

2. Formal Concept Analysis and Relational Algebra

We first recall some basic notions from Formal Concept Analysis (FCA) [8,9] and Relational Algebra [10].

Definition 1. Let $\mathcal{O}$ and $\mathcal{P}$ be sets, called the set of objects and attributes, respectively. Let $\mathcal{R}$ be a relation on $\mathcal{O} \times \mathcal{P}$. For $o \in \mathcal{O}$ and $p \in \mathcal{P}$, $\mathcal{R}(o, p)$ holds if the object o has attribute p, denoted also by $(o, p) \in \mathcal{R}$. The triple $\mathcal{K} = (\mathcal{O}; \mathcal{P}; \mathcal{R})$ is called a *formal context*.

We may notice that the definition of a formal context is very similar to a relation where $\mathcal{O}$ (respectively, $\mathcal{P}$) represents the domain (respectively, the range) of the relation.

2.1. Galois connections and their properties

Definition 2. Let $(A, \leqslant_A)$ and $(B, \leqslant_B)$ be two partially ordered sets. Let $f : A \to B$ and $g : B \to A$ such that $\forall a \in A, b \in B$, $f(a) \leqslant_B b \iff g(b) \leqslant_A a$. Then, the pair (f, g) is called a *Galois connection*.

Proposition 3. *Let $\mathcal{O}$, $\mathcal{P}$ be two arbitrary sets. Let $\mathcal{R}$ be a relation on $\mathcal{O} \times \mathcal{P}$. Let $A \subseteq \mathcal{O}$ and $B \subseteq \mathcal{P}$ also be arbitrary. The pair of functions (f, g) with $f : 2^{\mathcal{O}} \to 2^{\mathcal{P}}$ and $g : 2^{\mathcal{P}} \to 2^{\mathcal{O}}$ defined by*

- $f(A) = \{p \in \mathcal{P} \mid \forall o \in A, (o, p) \in \mathcal{R}\}$,
- $g(B) = \{o \in \mathcal{O} \mid \forall p \in B, (o, p) \in \mathcal{R}\}$,

forms a Galois connection.

Let $A_1, A_2 \subseteq \mathcal{O}$ and $B_1, B_2 \subseteq \mathcal{P}$. It is well known [9] that a pair (f, g) forms a Galois connection if and only if the following properties are satisfied:

$$A_1 \subseteq A_2 \quad \Rightarrow \quad f(A_2) \subseteq f(A_1), \qquad B_1 \subseteq B_2 \quad \Rightarrow \quad g(B_2) \subseteq g(B_1),$$
$$A \subseteq (g \circ f)(A), \qquad B \subseteq (f \circ g)(B).$$

Definition 4. We call $(g \circ f)(A)$ the *closure* of A, and $(f \circ g)(B)$ the *closure* of B. The pair (A, B), where A is included in $\mathcal{O}$, B is included in $\mathcal{P}$, $f(A) = B$, and $g(B) = A$ is called a *formal concept* of context $\mathcal{K}$ with extent A and intent B. We also have $(g \circ f)(A) = A$ and $(f \circ g)(B) = B$.

2.2. Rule representation and reasoning

We are mainly concerned with knowledge base representation and automated reasoning. In this section we show how a truth table associated with a decision rule is represented as a formal context, and how different rules may be combined if there is no contradiction between them.

2.2.1. Rule representation

A decision rule of the form '*if A then B*' or equivalently '$A \to B$ is true', between attributes A and B, reflects the knowledge of a given domain in which the satisfaction of premise A implies the conclusion B. From a logical point of view, the expression '$A \to B$ is true' can be represented as a Truth Table (TT) where each row (or solution) is a truth value assignment ($1 = true, 0 = false$) for the attributes A and B. For instance, the assignment $(A = 0, B = 1)$ is a solution for '$A \to B$ is true'. At the same time, the later expression could be also represented as a Formal Context (FC) $\mathcal{K} = (\mathcal{O}; \mathcal{P}; \mathcal{R})$, where the set $\mathcal{O}$ is the set of possible solutions (truth-value assignments for A and B such that $A \to B$ is true), the set $\mathcal{P}$ is the set of attributes (or properties, in our case $\mathcal{P} = \{A, B\}$), and $\mathcal{R}(o, p)$ holds if the solution $o \in \mathcal{O}$ has the assignment *true* for the attribute $p \in \mathcal{P}$. In the remainder, the Truth Table Binary Relation $\mathcal{R}$ will be denoted by TTBR and the expression '$A \to B$ is true' will be abbreviated as '$A \to B$'.

Example 5. Let us consider the rule $A \to B$. In Table 1 we find a representation of a formal context $\mathcal{K} = (\mathcal{O}; \mathcal{P}; \mathcal{R})$, where $\mathcal{O} = \{s_1, s_2, s_3\}$, $\mathcal{P} = \{A, B\}$ and $\mathcal{R} = \{(s_1, A), (s_1, B), (s_2, B)\}$.

Table 1
$TTBR_1$ for rule $A \rightarrow B$.

	A	B
s_1	1	1
s_2	0	1
s_3	0	0

Table 2
$TTBR_2$ for rule *IF* $(B$ *OR* $D)$ *THEN* K.

	B	D	K
s'_1	0	1	1
s'_2	1	0	1
s'_3	0	0	0
s'_4	0	0	1
s'_5	1	1	1

Remark 6. Notice that $(f \circ g)(\{A\}) = \{A, B\}$, which means that the closure of set $\{A\}$ is $\{A, B\}$; which is interpreted as $A \rightarrow B$. We may also conclude that $A \rightarrow B$ because $g(\{A\}) = \{s_1\}$ is included in $g(\{B\}) = \{s_1, s_2\}$. According to Guigues et al. [7], the implication $A \rightarrow B$ holds if $g(\{A\}) \subseteq g(\{B\})$.

Remark 7. Formally, we get for arbitrary sets $\mathcal{O}$ of objects and $\mathcal{B}$ of attributes, and TTBR $\mathcal{R}$,

$$Y \in (f \circ g)(\mathcal{B}) \quad \Longleftrightarrow \quad \forall s \in \mathcal{O} : \big(\forall X \in \mathcal{B} : \mathcal{R}(s, X)\big) \rightarrow \mathcal{R}(s, Y).$$

In words: $(f \circ g)(\mathcal{B})$ consists of all those attributes Y which are logically implied by the attributes X in $\mathcal{B}$ with respect to $\mathcal{R}$.

In case of a decision rule having logical connectors AND, OR, NOT, we apply the same principle by enumerating all possible solutions and representing them as a TTBR. An example follows.

Example 8. Let us consider rule *IF* $(B$ *OR* $D)$ *THEN* K. In Table 2 we find a representation of a context $\mathcal{K} = (\mathcal{O}; \mathcal{P}; \mathcal{R})$ where:

- $\mathcal{O} = \{s'_1, s'_2, s'_3, s'_4, s'_5\}$,
- $\mathcal{P} = \{B, D, K\}$, and
- $\mathcal{R} = \{(s'_1, D), (s'_1, K), (s'_2, B), (s'_2, K), (s'_4, K), (s'_5, B), (s'_5, D), (s'_5, K)\}$.

2.2.2. Context combination

In order to represent the logic behind a set of decision rules we propose to combine the corresponding TTBRs into a single TTBR. Once we obtain the global TTBR we can start the reasoning process as explained in Section 3. The combining procedure is akin to the equi-join operator in relational database theory [11]. Combining two TTBRs, let's say $TTBR_1$ and $TTBR_2$, consists of processing case by case the consistency of any row in $TTBR_1$ with any other row in $TTBR_2$. If not found contradicting, the two checked rows are combined into the final TTBR. Otherwise, they will be discarded. So, by definition, the obtained combined relation is simultaneously consistent with $TTBR_1$ and $TTBR_2$. If two relations $TTBR'_1$ and $TTBR'_2$ are not sharing any attribute, then any row in $TTBR'_1$ is combined to any other row in $TTBR'_2$ in the target relation.

Example 9. If we consider a knowledge base composed of the rules presented in Examples 5 and 8, we can combine their associated TTBRs as shown in Table 3. Combining the two different truth tables consists of producing a new one while preserving the coherence between them (i.e., the same truth values assignment for the common attributes). Fig. 1 illustrates how $TTBR_1$ and $TTBR_2$ are combined in order to get $TTBR_3$ (shown in Table 3).

Algorithm 1 describes how to combine two TTBRs while preserving the coherent information that they share. In case of independent rules, i.e., rules with disjoint sets of attributes (see Example 10), all table's solutions are combined by computing their Cartesian product (lines 4–5). Otherwise, we retain only rows having the same truth values (designated by function 'SameSolution') for their common attributes (line 9). The addition of a new row in table $\mathcal{R}_3$ is done by procedure 'createNewObject' (line 10), that works as follows: let $TV(s, p, R)$ be the truth value assigned to attribute p in the solution s within relation R. $TV(s, p, R)$ is assigned 1 if $R(s, p) = 1$; otherwise $TV(s, p, R)$ is assigned 0. Hence, for all s_k in O_3, s_i in O_1, s_j in O_2, the following properties should hold:

- $TV(s_k, p, R_3) = TV(s_i, p, R_1) = TV(s_j, p, R_2)$ for all p in CA,
- $TV(s_k, p, R_3) = TV(s_i, p, R_1)$ for all p in P_1,
- $TV(s_k, p, R_3) = TV(s_i, p, R_2)$ for all p in P_2.

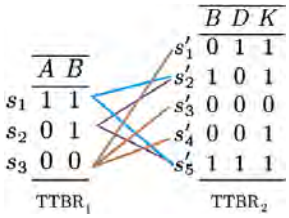


Fig. 1. How the combining is done.

Table 3
$TTBR_3$, result of combining $TTBR_1$ and $TTBR_2$.

	A	B	D	K
cs_1	0	0	1	1
cs_2	1	1	0	1
cs_3	0	1	0	1
cs_4	0	0	0	0
cs_5	0	0	0	1
cs_6	0	1	1	1
cs_7	1	1	1	1

Table 4
The combining of R_1 and R_2.

	A	B	C	D
cs_1	0	1	1	1
cs_2	0	0	1	1
cs_3	1	1	0	1
cs_4	1	1	0	0
cs_5	0	0	0	0
cs_6	1	1	1	1
cs_7	0	1	0	1
cs_8	0	1	0	0
cs_9	0	0	0	1

Algorithm 1: TwoContextsCombination.

Input: $\mathcal{K}_1 = (\mathcal{O}_1; \mathcal{P}_1; \mathcal{R}_1)$, $\mathcal{K}_2 = (\mathcal{O}_2; \mathcal{P}_2; \mathcal{R}_2)$: two TTBRs
Output: $\mathcal{K}_3 = (\mathcal{O}_3; \mathcal{P}_3; \mathcal{R}_3)$: combined TTBR

```
1  begin
2      P3 ← P1 ∪ P2 // new properties in the new context K3
3      CA ← P1 ∩ P2 // identify the common attributes.
4      if (CA = φ) then
5          R3 = CartesianProduct(R1, R2)
6      else
7          foreach Solution s_i ∈ O1 do
8              foreach Solution s_j ∈ O2 do
9                  if SameSolution(s_i, s_j, CA) then // same truth values for CA in s_i and s_j
10                     createNewObject(s_i, s_j, O3, R3) // we add new solution s_k to O3 such that:
11                     R3(s_k, p) = R1(s_i, p) = R2(s_j, p), ∀p ∈ CA
12                     R3(s_k, p) = R1(s_i, p) = ∀p ∈ P1
13                     R3(s_k, p) = R2(s_i, p) = ∀p ∈ P2
```

Example 10. Suppose that the knowledge base is composed of two independent decision rules: $R_1 = A \rightarrow B$ and $R_2 = C \rightarrow D$. Then, as a first step, we create two TTBRs equivalent to those in Table 1. Algorithm 1 combines the formal contexts associated with R_1 and R_2, and produces the TTBR depicted in Table 4.

The algorithm's worst-case running time complexity is in $O(|\mathcal{O}_1| \times |\mathcal{O}_2| \times |\text{CA}|)$, where $|S|$ denotes the cardinality of set S.

Once the global TTBR has been obtained, we can start the reasoning process using the closure operator. In fact, from Table 4, we calculate $(f \circ g)(\{A, C\}) = \{A, C, B, D\}$, which means that $(A \wedge C) \rightarrow (B \wedge D)$. We can also infer other rules, such as $(A \wedge D) \rightarrow B$, because $(f \circ g)(\{A, D\}) = \{A, B, D\}$.

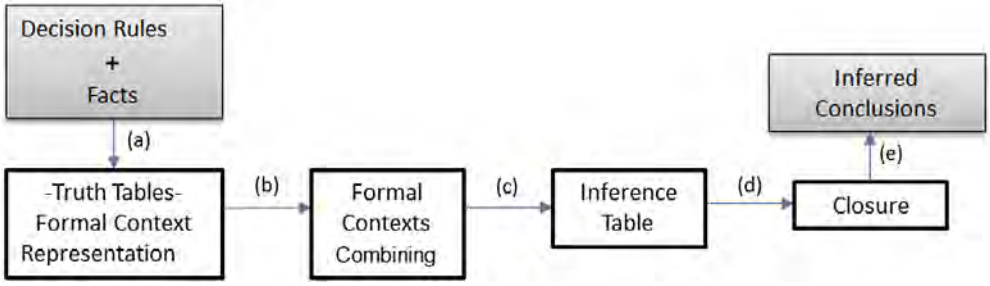


Fig. 2. A schematic description of conceptual reasoning.

Table 5
Example of knowledge base.

Id	Rule
1	IF A THEN B
2	IF C THEN D
3	IF M THEN E
4	IF K THEN F
5	IF G THEN H
6	IF I THEN J
7	IF B OR D THEN K
8	IF E THEN L
9	IF (F AND H) OR J THEN M
10	IF K AND L THEN N
11	IF M THEN O
12	IF N OR O THEN P

Also, from Table 3 we obtain $(f \circ g)(\{A\}) = \{A, B, K\}$ which indicates that A implies both B and K, which is coherent with respect to the considered rules.

3. Conceptual reasoning

In this section we detail our new approach based on conceptual reasoning as illustrated in Fig. 2.

3.1. General description

Starting from a set of rules in a knowledge based system, we generate, for each decision rule, a TTBR as indicated in Fig. 2(a). Then, combining the two contexts as per Algorithm 1 (cf. Fig. 2(b)), we obtain a total relation reflecting all possible cases (or solutions) which are equivalent to the initial set of decision rules (see Fig. 2(c)). The advantage of this method is that conflicts are detected during the combining step. In fact, if two ore more rules are in conflict we obtain an empty TTBR. In that case, the user is aware about the inconsistency of the knowledge base rules. After this step, starting from the initial fact A, and applying the Galois connection, the expression $(f \circ g)(\{A\}) - \{A\}$ gives us the set of all deduced attributes (see Fig. 2(d)–(e)). In this article we have also used, as an additional reasoning option, the Duquenne–Guigues set of implications extracted using the ConImp tool [12,13]. Hence, we could get all the deduced attributes from the implications without using the closure operator. The tool also allows the removal of redundancies from the initial relation.

Example 11. Let us consider the knowledge base depicted in Table 5, composed of 12 rules using 16 attributes: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P. The TTBR combining the 12 rules contains 258 rows. We present a reduced subset in Table 6.

To illustrate the case of possible inconsistency that we could detect by TTBR-combining, and for the sake of illustration, we consider rule $r_{13} = NOT(IF\ A\ THEN\ B)$ to be added to the knowledge base depicted in Table 5. The unique solution satisfying r_{13} is $s_1 = \{A = 1, B = 0\}$. Obviously, this rule is logically in conflict with rule $r_1 = (IF\ A\ THEN\ B)$ and the Global Context will be empty. It could be interesting for the user in such cases to revise the knowledge base and remove the possible inconsistency.

3.2. Reasoning

After producing the whole non-empty TTBR according to the knowledge base rules, we can start the reasoning step (see Fig. 2(d)–(e)). Reasoning, or deductive reasoning [14], means the inference of possible conclusions from valid facts. In

Table 6
Subset of the Global Context obtained from the knowledge base in Table 5.

	A	B	C	D	M	E	K	F	G	H	I	J	L	N	O	P
s_1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1
s_2	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1
s_3	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0
s_4	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	1
s_5	0	0	0	0	0	1	0	0	1	1	0	0	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s_{258}	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Algorithm 2: Deductive reasoning.

Input:
- $\mathcal{K} = (\mathcal{O}; \mathcal{P}; \mathcal{R})$: TTBR representing combined Truth Tables
- *DGI*: Duquenne–Guigues set of implications extracted from $\mathcal{K}$
- *Exp*: Boolean expression representing the initial facts.

Output:
- *Conc*: Derived conclusions

```
1  begin
2      if Exp ∈ DGI then
3          Conc ← ExtractDirectConclusion(Exp, DGI)
4      else
5          if AllConjunctive(Exp) then
6              //All attributes in Exp are connected with logical AND operator.
7              //Hence, we get new conclusions by Galois Connection.
8              Conc ← (f ∘ g)(cToSet(Exp))
9          else
10             NewAttrib ← EvaluateLogicExpression(Exp)
11             // NewAttrib is a new column to be added to K.
12             // ∀o ∈ O, R(o, NewAttrib) = 1 if Exp is true for o and 0 otherwise.
13             K ← AddAttrib(NewAttrib, K)
14             Conc ← (f ∘ g)({NewAttrib})
```

Algorithm 2 the valid facts are represented by parameter *Exp*, which may contain a single fact or a combination of facts obtained using logical connectors. The details are explained in the following paragraphs.

- If *Exp* represents a single fact (i.e., an attribute assumed to be true as input, let's say A): We can derive the possible conclusions by applying the closure operator. But we may instead, as indicated in lines 2–3, directly retrieve the Duquenne–Guigues set of conclusions using the ConImp tool [12,13]. An example of such rules is presented in Table 7. We observe all the possible conclusions that may be derived from the initial facts. For example, in row 3, we observe that properties D, K and F are derived from fact C.
- In case of an *Exp* representing a conjunction of attributes (line 5), let's say $(A \wedge B \wedge C \cdots)$: We compute the conclusions by applying the closure operator to the set representing *Exp* (line 8), i.e., $\{A, B, C, \ldots\}$. The function cToSet() (line 8) converts *Exp* to a set.
- In case of an *Exp* representing an arbitrary Boolean combination of attributes, let's say $((A \vee B) \wedge \neg C \cdots)$: We have to perform the following steps:
  1. Add a new attribute 'NewAttrib' reflecting *Exp* and compute its truth-value for all solutions in TTBR. This is done by procedures 'EvaluateLogicExpression' and 'NewAddAttrib' (lines 10–13).
  2. Derive possible conclusions by applying the closure operator to {*NewAddAttrib*} (line 14).

In the end, a propositional logic rule may be mapped to an expression and vice versa: $A \rightarrow B$ is equivalent to $(NOT\ B \vee A)$. So in our case expressions and rules are all processed in the same way. However, in most expert systems, a knowledge data base may be presented with a set of rules using or not negations. The extension we make in this article for handling negation (see Section 3.3) makes it possible to process any kind of Boolean expression.

The algorithm complexity is closely related to the size of the obtained TTBR, i.e., it depends on $|\mathcal{O}|$, $|Exp|$ (the number of attributes in *Exp*), as well as on the number *NDGI* of Duquenne–Guigues implications. The worst-case running time complexity is in $O(\max(NDGI, |\mathcal{O}|) \times |Exp|)$.

The characteristic of the reasoning process based on the Duquenne–Guigues set of implications, as well as on Galois connection, is to derive positive conclusions according to the considered TTBR. In reality, negated attributes could also be deduced by expanding the initial TTBR as follows: For each attribute A we add to the TTBR its negated version, i.e., $\neg A$. Obviously, for all $o \in \mathcal{O}$, the truth value $TV(o, A, R)$ equals $1 - TV(o, \neg A, R)$. Hence, we get an entire duplication of the

Table 7
Duquenne–Guigues set of implications discovered from the reduced TTBR.

Id	Implication
1	$A \rightarrow B\ K\ F$
2	$B \rightarrow K\ F$
3	$C \rightarrow D\ K\ F$
4	$D \rightarrow K\ F$
5	$E \rightarrow L$
6	$K \rightarrow F$
7	$G \rightarrow H$
8	$M \rightarrow E\ L\ O\ P$
9	$F\ H \rightarrow M\ E\ L\ O\ P$
10	$I \rightarrow M\ E\ J\ L\ O\ P$
11	$J \rightarrow M\ E\ L\ O\ P$
12	$K\ F\ L \rightarrow N\ P$
13	$N \rightarrow P$
14	$O \rightarrow P$

initial TTBR. In Section 3.3 we propose a new solution to avoid such duplication by extending the Galois connection. In the following we present the new definitions for the extended Galois connection, as well as its main properties and their proofs.

3.3. Extended Galois connection

In this subsection we propose a novel solution for the reasoning process, that consists of extending the Galois connection to derive both positive and negative conclusions. Let $\mathcal{K} = (\mathcal{O}; \mathcal{P}; \mathcal{R})$ be a formal context, $A \subseteq \mathcal{O}$ and $B \subseteq \mathcal{P}$. We extend the functions (f, g), defined in Section 2.1, by f_e and g_e respectively to process negative attributes, as shown:

- $f_e(A) = \{p \in \mathcal{P} \mid (o, p) \in \mathcal{R},\ \forall o \in A\} \cup \{\neg p \mid p \in \mathcal{P} \wedge (o, p) \notin \mathcal{R},\ \forall o \in A\}$,
- $g_e(B) = \{o \in \mathcal{O} \mid (o, p) \in \mathcal{R}$ if p positive and $(o, \neg p) \notin \mathcal{R}$ if p negative, $\forall p \in B\}$.

Example 12. Let us consider the TTBR from Table 1. We have the following results:

- $f_e(\{s_2, s_3\}) = \{\neg A\}$,
- $f_e(\{s_1\}) = \{A, B\}$,
- $f_e(\{s_1, s_2\}) = \{B\}$,
- $g_e(\{\neg A\}) = \{s_2, s_3\}$,
- $g_e(\{\neg B\}) = \{s_3\}$,
- $f_e(\{s_1, s_2, s_3\}) = \{\}$.

Lemma 13. *Let $\mathcal{K} = (\mathcal{O}; \mathcal{P}; \mathcal{R})$ be a formal context. Functions f_e and g_e form a Galois connection between the powersets of $\mathcal{O}$ and $\mathcal{P}$.*

Proof. Let $A \subseteq \mathcal{O}$, $A_1 \subseteq A_2 \subseteq \mathcal{O}$, $B \subseteq \mathcal{P}$, and $B_1 \subseteq B_2 \subseteq \mathcal{P}$. According to Proposition 3, we will prove that the following properties are satisfied:

(I) $A_1 \subseteq A_2 \Rightarrow f_e(A_2) \subseteq f_e(A_1)$, (II) $B_1 \subseteq B_2 \Rightarrow g_e(B_2) \subseteq g_e(B_1)$,
(III) $A \subseteq (g_e \circ f_e)(A)$, (IV) $B \subseteq (f_e \circ g_e)(B)$.

- (I) $A_1 \subseteq A_2 \Rightarrow f_e(A_2) \subseteq f_e(A_1)$:
  $f_e(A_2) = \{p \in \mathcal{P} \mid (o, p) \in \mathcal{R},\ \forall o \in A_2\} \cup \{\neg p \mid p \in \mathcal{P} \wedge (o, p) \notin \mathcal{R},\ \forall o \in A_2\}$. Since $A_1 \subseteq A_2$, all the conditions satisfied by the elements of A_2 are necessarily satisfied by all its members, in particular for all elements from A_1. Then,

$$\begin{aligned} f_e(A_2) &= \left\{p \in \mathcal{P} \;\middle|\; (o, p) \in \mathcal{R},\ \forall o \in A_2\right\} \cup \left\{\neg p \;\middle|\; p \in \mathcal{P} \wedge (o, p) \notin \mathcal{R},\ \forall o \in A_2\right\} \\ &\subseteq \left\{p \in \mathcal{P} \;\middle|\; (o, p) \in \mathcal{R},\ \forall o \in A_1\right\} \cup \left\{\neg p \;\middle|\; p \in \mathcal{P} \wedge (o, p) \notin \mathcal{R},\ \forall o \in A_1\right\} \\ &\subseteq f_e(A_1). \end{aligned}$$

- (II) $B_1 \subseteq B_2 \Rightarrow g_e(B_2) \subseteq g_e(B_1)$:

$$g_e(B_2) = \left\{o \in \mathcal{O} \;\middle|\; (o, p) \in \mathcal{R} \text{ if } p \text{ positive and } (o, \neg p) \notin \mathcal{R} \text{ if } p \text{ negative},\ \forall p \in B_2\right\}.$$

  Since the definition is preserved by subsets of B_2, the proof follows the steps of the previous case.

- (III) $A \subseteq (g_e \circ f_e)(A)$:
  By definition of f_e,

$$f_e(A) = \{p \in \mathcal{P} \mid (o, p) \in \mathcal{R},\ \forall o \in A\} \cup \{\neg p \mid p \in \mathcal{P} \wedge (o, p) \notin \mathcal{R},\ \forall o \in A\}. \tag{1}$$

  By definition of g_e,

$$(g_e \circ f_e)(A) = \{o \in \mathcal{O} \mid (o, p) \in \mathcal{R} \text{ if } p \text{ positive and } (o, \neg p) \notin \mathcal{R} \text{ if } p \text{ negative},\ \forall p \in f_e(A)\}. \tag{2}$$

  Let us assume there exists $o \in A$ such that $o \notin (g_e \circ f_e)(A)$, and let us look for a contradiction. Two possibilities arise, namely, there exists $p \in f_e(A)$ such that $(o, p) \notin \mathcal{R}$ with p positive, or there exists $p \in f_e(A)$ such that $(o, \neg p) \in \mathcal{R}$ with p negative. Let us analyze these two cases:
  - There exists $p \in f_e(A)$ such that $(o, p) \notin \mathcal{R}$ with p positive:
    Since p is a positive literal, according to (1), it must satisfy $(o, p) \in \mathcal{R}$, which leads to a contradiction.
  - There exists $p \in f_e(A)$ such that $(o, \neg p) \in \mathcal{R}$ with p a negative literal:
    Since p is negative, $p = \neg q$ for some attribute q. According to (1) $\neg q$ must belong to the term on the right-hand side of the union. Therefore, $(o, q) \notin \mathcal{R}$, which is to say that $(o, \neg p) \notin \mathcal{R}$, which leads to a contradiction.
- (IV) $B \subseteq (f_e \circ g_e)(B)$:
  By definition of g_e,

$$g_e(B) = \{o \in \mathcal{O} \mid \underbrace{(o, p) \in \mathcal{R} \text{ if } p \text{ positive}}_{LHS} \text{ and } \underbrace{(o, \neg p) \notin \mathcal{R} \text{ if } p \text{ negative}}_{RHS},\ \forall p \in B\}. \tag{3}$$

  By definition of f_e,

$$(f_e \circ g_e)(B) = \{p \in \mathcal{P} \mid (o, p) \in \mathcal{R},\ \forall o \in g_e(B)\} \cup \{\neg p \mid p \in \mathcal{P} \wedge (o, p) \notin \mathcal{R},\ \forall o \in g_e(B)\}. \tag{4}$$

  Let us assume there is $p \in B$ such that $p \notin (f_e \circ g_e)(B)$, and let us arrive at a contradiction. If p is positive, according to Eq. (4) there exists $o \in g_e(B)$ such that $(o_1, p) \notin \mathcal{R}$. But this contradicts term *LHS* in Eq. (3). Similarly, if p is negative, there exists $o_2 \in g_e(B)$ such that $(o, \neg p) \in \mathcal{R}$. But this now contradicts term *RHS* in Eq. (3). □

So far, conceptual reasoning considers the global TTBR to ensure the consistence of the knowledge base, as well as the derivation of any new positive or negative conclusion. However, a serious difficulty may arise when the size of the TTBR increases considerably. The truth table associated with N properties may contain up to 2^N solutions. To tame this exponential growth in size, we propose to delay the construction of the global TTBR by reducing the different TTBRs during the reasoning step. We detail this idea in the following section.

4. Cooperative and conceptual reasoning

Starting from the methods proposed in [15], we can make the reasoning effort modular and cooperative by:

1. combining enough tables but not exceeding some acceptable size threshold,
2. selecting only subtables respecting the initial facts,
3. combining the results using the ContextCombine operation and, finally,
4. using the Galois connection to get additional results.

Before making the combination of all the TTBRs associated with the different rules, we propose a preprocessing phase which allows us in many cases to reduce the cost of the construction of the global TTBR. In fact, it is possible to reduce the size of the different TTBRs by starting the reasoning and discovering possibly new facts. All rows with truth value 0 for those new facts are systematically eliminated. After such row eliminations, it is possible to derive new facts using the closure operator. As before, we iteratively reduce the TTBRs as much as possible, i.e., as long as we derive new facts. Only when no more TTBRs can be reduced, we perform the entire combining in order to obtain the global TTBR. Finally, we complete the inference step by selecting additional facts using Galois connection operators on the global TTBR.

Example 14. Let us consider the following rules and their corresponding TTBRs in Table 8:

- Rule 1: $A \rightarrow \neg B \wedge C$
- Rule 2: $B \rightarrow D$
- Rule 3: $C \rightarrow E$

A blind initial combination of T_1, T_2 and T_3, without considering initial facts yields a global table with 11 solutions. If instead we assume that we have the initial fact A, then we can reduce T_1 to the table RT_1 as indicated in Table 9 and

Table 8
TTBRs associated with rules 1, 2 and 3, respectively.

A	B	C
1	0	1
0	0	1
0	0	0
0	1	1
0	1	0

T_1

B	D
1	1
0	1
0	0

T_2

C	E
1	1
0	1
0	0

T_3

Table 9
Reduced TTBRs.

A	B	C
1	0	1

RT_1

B	D
0	1
0	0

RT_2

C	E
1	1

RT_3

Table 10
Combining of three tables.

A	B	C	D	E
1	0	1	0	1
1	0	1	1	1

Table 11
Results on the global TTBR.

File name	Time: mm:ss:ms	Global TTBR size
uf20-01.cnf	00:03:046	8 rows
uf20-02.cnf	00:05:390	29 rows
uf20-03.cnf	00:09:953	1 row
uf20-04.cnf	00:02:046	3 rows
uf20-05.cnf	00:23:765	2 rows
uf20-06.cnf	01:38:500	4 rows
uf20-07.cnf	00:02:843	23 rows
uf20-08.cnf	00:20:718	4 rows
uf20-09.cnf	00:02:109	1 rows
uf20-010.cnf	00:06:312	9 rows

conclude that $A \rightarrow (\neg B \wedge C)$. Consequently, we reduce table T_2 to RT_2 (i.e., we remove those rows in T_2 where $B = 1$). Similarly, we reduce table T_3 to RT_3 by removing those rows where $C = 0$. Combining now RT_1, RT_2 and RT_3 gives the TTBR from Table 10, which only has 2 rows. We could conclude now that $A \rightarrow (\neg B \wedge C \wedge E)$.

5. Experimental results

In order to study the efficacy of the proposed approach we have considered two case studies. The first one is related to the SAT problem [3], and the second one, provided in [16], relates to Medical Data Rules. In case study 1 we have considered two sub-cases: in the first one (satisfiable sets of rules) we have processed 10 files containing 91 rules and 20 attributes and we have obtained different TTBRs with reduced sizes, which are presented in Table 11. Also, we have presented the TTBR with 8 SAT-solutions for the first file and we have used the ConImp tool in order to derive a reduced set of implications. In the second sub-case (unsatisfiable sets of rules) we have considered 17 UNSAT problems for which the combined TTBRs were, as expected, empty. This result confirms the effectiveness of the approach for inconsistency detection between rules and the possible localization of the sub-set of rules responsible for this inconsistency. In case study 2 we have combined a set of rules related with medical data and we have obtained an improved version of the rules (i.e., the implications extracted from the combined TTBR). There is a concrete application for this: the reduced implications allow us to avoid some redundant medical tests required for diagnosis. The experiment were run in a Laptop Intel(R) Core(TM) i7-3630QM CPU @ 2.40 GHz (8 CPUs), 2.4 GHz, with 16 GB of RAM.

Table 12
TTBR representing the 91 rules.

	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
S_1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	1	1	1	0
S_2	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	1	1	1	0
S_3	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	0	1	1	1	0
S_4	1	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	1	1	1	0
S_5	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0
S_6	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	1	1	0
S_7	1	0	0	0	0	1	1	1	0	0	1	1	0	0	1	0	1	0	1	0
S_8	1	1	1	1	0	1	1	1	0	0	1	0	0	1	1	1	1	0	0	1

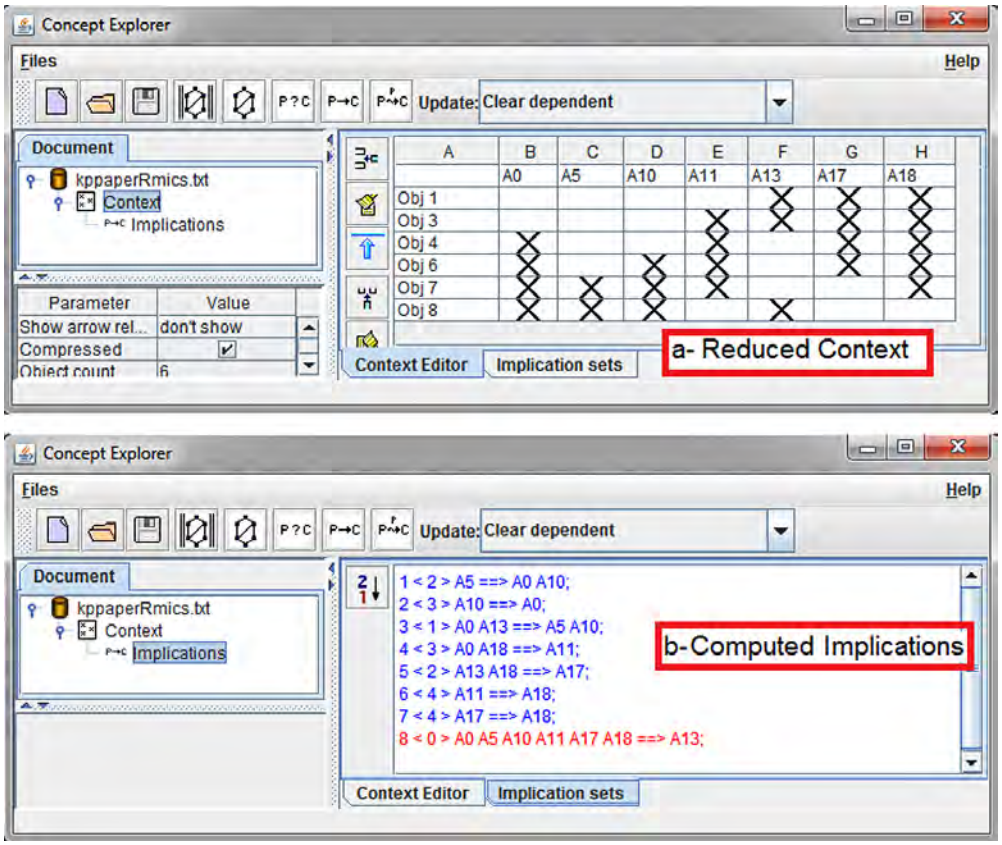


Fig. 3. Using ConImp [12,13] to generate the implications.

5.1. Case study (1): SAT/UnSAT problems

In order to illustrate the inconsistency detection capabilities of this work, as well as the reasoning aspects, we have made some experiments using some satisfiability benchmark problems accessible from http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html.

5.1.1. SAT problem

In Table 11 we summarize the experimental results for 10 files containing satisfiable sets of rules, indicating the requested time for the global combining as well as the size of the obtained TTBRs. As mentioned before, the 10 files have exactly the same size (91 rules and 20 attributes) but the required times for generating the TTBR vary from 2 seconds to less than 2 minutes. At the same time, we remark that the size of the obtained TTBRs is small (from 1 to 29 rows, but most cases have less than 5 rows). This could result in a significant advantage for the reasoning step.

In Table 12 we present the TTBR resulting from processing file uf20-01.cnf, which contains 91 rules and 20 attributes.

Starting from Table 12 we have extracted the set of Duquenne–Guigues implications as indicated in Fig. 3. In the upper screenshot we show the combined TTBR in a reduced format (a reduction option in the tool is used here to avoid redundancy and to eliminate empty columns). The lower screenshot shows the extracted set of implications.

Table 13
Attributes description.

Attribute	Description
trestbps	resting blood pressure (in mm Hg on admission to the hospital)
chol	serum cholesterol in mg/dl
fbs	fasting blood sugar >120 mg/dl
restecg	resting electrocardiogram hic results
thalach	maximum heart rate achieved
Exang	exercise induced angina
oldpeak	ST depression

Table 14
TTBR obtained according to the provided classification rules.

Id	'TMT'	'diagnosis'	'fbs'	'chol'	'restecg'	'trestbps'
1	0	0	0	0	0	0
2	0	0	0	0	0	1
3	0	0	1	0	0	0
4	0	0	1	0	0	1
5	0	1	0	0	0	0
6	0	1	0	0	0	1
7	0	1	0	0	1	0
8	0	1	0	0	1	1
9	0	1	0	1	0	0
10	0	1	0	1	0	1
11	0	1	0	1	1	0
12	0	1	0	1	1	1
13	0	1	1	0	0	0
14	0	1	1	0	0	1
15	0	1	1	0	1	0
16	0	1	1	0	1	1
17	0	1	1	1	0	0
18	0	1	1	1	0	1
19	0	1	1	1	1	0
20	0	1	1	1	1	1
21	1	1	0	0	0	0
22	1	1	0	0	0	1
23	1	1	0	0	1	0
24	1	1	0	0	1	1
25	1	1	0	1	0	0
26	1	1	0	1	0	1
27	1	1	0	1	1	0
28	1	1	0	1	1	1
29	1	1	1	0	0	0
30	1	1	1	0	0	1
31	1	1	1	0	1	0
32	1	1	1	0	1	1
33	1	1	1	1	0	0
34	1	1	1	1	0	1
35	1	1	1	1	1	0
36	1	1	1	1	1	1

5.1.2. Inconsistency detection

In Table 15 we present the results obtained for 17 files containing unsatisfiable sets of rules. The first column indicates the file name as well as the number of terms and clauses respectively. For instance, the first file 'aim-50-2_0-no-4.cnf' (50t 100c) indicates a problem containing 100 clauses (i.e., rules) and 50 terms (i.e., attributes). The second column indicates the time required to combine all TTBRs until an empty one is obtained. The format is hh:mm:ss:ms. Finally, the third column indicates the subset of rules as well as the intermediate ones responsible for the inconsistency (the ones in conflict). Computation of the global TTBR is made progressively, a subset at a time, producing the creation of new intermediate TTBRs. The last ones were identified by incrementing the total rules number. For instance in Table 15, row 1, we have initially 100 rules and 35 intermediate ones. A conflict is detected between rules 130 and 135. The initial subset of rules responsible for the inconsistency could also be located and provided to the final user.

5.2. Case study (2) on medical data

In this second case study, we experimented on a set of medical data rules provided in [16], related to a heart diseases dataset available at the UCI site [17]. Each attribute value is obtained by a medical examination test. The considered at-

Table 15
Inconsistency detection.

File name	Time	Inconsistency found (TTBR is empty) between rules
aim-50-2_0-no-4.cnf (50t 100c)	00:08:09:707	[130 135]
aim-100-1_6-no-1.cnf (100t 160c)	00:05:30:929	[137 130 142 143 146 151 154 155 98 111 107 148 147 149 159]
aim-100-1_6-no-2.cnf (100t 160c)	00:06:21:345	[85 86 102 98 99 106 126 125 154 127 120 133 128 108 117]
aim-100-1_6-no-3.cnf (100t 160c)	00:06:45:994	[81 97 86 96 99 91 148 113 122 101 98 105 141 125 112 111 118 114 139 117 107 106 109 144 121 119 120 124 147 129 130 145]
aim-100-1_6-no-4.cnf (100t 160c)	00:06:05:651	[192 197]
aim-100-2_0-no-1.cnf (100t 200c)	00:04:23:339	[116 117 112 141 151 170 118 163 168 108 130 155 156 137 181 158 119 159 115 152 157 164]
aim-100-2_0-no-2.cnf (100t 200c)	00:04:26:489	[242 248 230 239]
aim-100-2_0-no-3.cnf (100t 200c)	00:04:30:413	[209 234 217 218 221 222]
aim-100-2_0-no-4.cnf (100t 200c)	00:07:34:329	[205 238 242 230]
aim-200-1_6-no-1.cnf (200t 320c)	00:05:54:376	[229 297 255 278 203 244 254 294 209 197 256 230 235 187 275 287 273]
aim-200-1_6-no-2.cnf (200t 320c)	00:04:10:881	[187 221 172 132 156 114 181 208 183 186 166 139 101 247 162 111 146 93 155 153 171 199 161 175 182]
aim-200-1_6-no-3.cnf (200t 320c)	00:06:00:062	[248 275 247 242 255 236 195 238 252 230 229 258 271 261 263 274 281]
aim-200-1_6-no-4.cnf (200t 320c)	00:05:36:480	[184 185 141 222 229 217 218 219 176 212 232 214 115 237 223 171 146]
aim-200-2_0-no-1.cnf (200t 400c)	00:06:21:636	[277 278 287 283 275 245 219 215 276 267 280 205 186 292 291 285 305 248 279 386 317]
aim-200-2_0-no-2.cnf (200t 400c)	00:03:34:642	[227 228 238 237 231 148 250 271 325 275 254 270 272 278 279 240 283 399 265 290 299 226]
aim-200-2_0-no-3.cnf (200t 400c)	00:03:04:376	[253 254 228 251 280 236 222 283 272 200 206 285 290 295 292 289]
aim-200-2_0-no-4.cnf (200t 400c)	00:02:58:511	[271 309 345 368 366 365 344 308 353 383 318 340 372 287]

tributes are described in Table 13. Attributes 'thalach', 'Exang', 'oldpeak' and 'slope' are the result of performing the medical test 'TMT'. For our case study we consider the same exact rules presented in [16], namely,

1. if (TMT = '+ve') and (other attributes are '+ve' or 've') then diagnosis is '+ve'.
2. if (TMT = '−ve') and (fbs = '+ve') and (chol = '+ve') and (restecg = '+ve') then diagnosis is '+ve'.
3. if (TMT = '−ve') and (fbs = '−ve') and (chol = '+ve') and (restecg = '+ve') then diagnosis is '+ve'.
4. if (restecg = '+ve') and (TMT = '−ve') and (chol = '+ve') then diagnosis is '+ve'.
5. if (restecg = '−ve') and (TMT = '−ve') and (chol = '+ve') then diagnosis is '+ve'.
6. if (trestbps = '+ve') and ((chol = '+ve') or (restecg = '+ve')) and (TMT = '−ve') then diagnosis is '+ve'.
7. if (trestbps = '−ve') and ((chol = '+ve') or (restecg = '+ve')) and (TMT = '−ve') then diagnosis is '+ve'.

As described in [16] there exists some redundancy within the classification rules. For example, according to classification rules 2 and 3, if TMT gives negative results and if chol and restecg give positive results, then fbs is redundant. Also, according to classification rules 4 and 5, if TMT gives negative results, then if chol gives positive results, then restecg is redundant. Detecting such redundancies helps in deciding the number of medical tests required for diagnosis. The above results can be summarized as [16]:

- Perform TMT test.
- If result is '+ve' then diagnosis is '+ve'.
- If result is '−ve' then perform chol test.
- If chol test gives '+ve' as result, then don't perform trestbps test.

In our case, with Conceptual Reasoning, we have encoded the 7 rules in different TTBRs and have obtained a global TTBR containing 36 solutions as shown in Table 14. Also, we have extracted the following set of implications:

- If TMT result is '+ve' then diagnosis is '+ve'.
- If chol test gives '+ve', then diagnosis is '+ve'.
- If restecg result is '+ve', then diagnosis is '+ve'.

We remark that besides the fact that we were able to determine that there is no inconsistency between the different rules (because the global TTBR is non-empty), the rules we obtained provide important additional information for the restecg test. In fact, a positive result implies '+ve' for diagnosis, which is not mentioned in [16]. This specific case shows to what extent the global conceptual analysis of a set of rules is better than a restrictive analysis on a subset of rules. Furthermore, the global TTBR could be very interesting for experts to possibly infer new relationships between medical tests and diagnoses. Finally, the Cooperative and Conceptual Reasoning alternative provides exactly the same result as the Global Context which is useful in case of a larger rules set.

6. Conclusions and further work

We have shown that it is possible to combine truth tables represented as formal contexts, and from these, using a Galois connection, to make forward reasoning without managing conflicts caused by the forward chaining method. We may even reduce the reasoning context (this has already been proved in [18,7]) to speed up the reasoning step, if we use only positive attributes (attributes whose value is 1) as given facts, and seek to obtain goals without negations, which is usual. Cooperative conceptual reasoning is also a solution to speed up the reasoning step. By only calculating partial combining/ of initial tables, upon submission of a subset of facts, the system extracts only those rows satisfying the initial facts. Then, by combining all these sub-tables, we obtain a global table on which we apply the Galois connection to get the conclusions. This method constitutes a new forward reasoning method based on minimal contexts which are equivalent to the initial knowledge base set of rules. It shows efficiency in possible initial conflicts resolutions as well as knowledge expressiveness and reasoning.

Acknowledgements

This publication was made possible by a grant from the Qatar National Research Fund NPRP 04-1109-1-174. Its contents are solely the responsibility of the authors and do not necessarily reflect the official views of the QNRF. We thank the anonymous reviewers for their valuable comments which raised up the quality of the paper.

References

[1] A. Robinson, A. Voronkov, Handbook of Automated Reasoning, vols. I & II, Elsevier/MIT Press, 2001.
[2] S. Owre, J. Rushby, N. Shankar, A prototype verification system, in: Proceedings of the 11th International Conference on Automated Deduction, in: Lecture Notes in Artificial Intelligence, vol. 607, Springer, 1992, pp. 748–752.

[3] C.P. Gomes, H. Kautz, A. Sabharwal, B. Selman, Satisfiability solvers, in: Frank Van Harmelen, Vladimir Lifschitz, Bruce Porter (Eds.), Handbook of Knowledge Representation, in: Foundations of Artificial Intelligence, vol. 3, Elsevier, 2001.
[4] J.-L. Laurière, M. Vialatte, Snark: A language to represent declarative knowledge and an inference engine which uses heuristics, in: IFIP Congress, 1986, pp. 811–816.
[5] K.C. Lee, H.R. Cho, J.-S. Kim, An expert system using an extended and–or graph, Knowl.-Based Syst. 21 (2008).
[6] Z. Wu, G. Eadon, S. Das, E.I. Chong, V. Kolovski, M. Annamalai, J. Srinivasan, Implementing an inference engine for RDFS/OWL constructs and user-defined rules in oracle, in: ICDE 2008, 2008, pp. 1239–1248.
[7] J.L. Guigues, V. Duquenne, Familles minimales d'implications informatives résultant d'un tableau de données binaires, Math. Sci. Hum. 95 (1986) 5–18.
[8] A. Jaoua, S. Elloumi, Galois connection: Formal concepts and Galois lattice in real relations: Application in a real classifier, J. Syst. Softw. 60 (2002) 149–163.
[9] B. Ganter, R. Wille, Formal Concept Analysis: Mathematic Foundations, Springer-Verlag, 1999.
[10] G. Schmidt, Relational Mathematics, Encyclopedia of Mathematics and Its Applications, vol. 132, Cambridge University Press, 2011.
[11] E. Codd, A relational model of data for large shared data banks, Commun. ACM 13 (1970).
[12] S.A. Yevtushenko, System of data analysis "concept explorer", in: Proceedings of the 7th National Conference on Artificial Intelligence KII-2000, Russia, 1974, pp. 127–134.
[13] P. Burmeister, Formal concept analysis with ConImp: Introduction to the basic features, Technical Report, Technische Hochschule, 1998.
[14] E. Mendelson, The Propositional Calculus, 4th ed., Chapman and Hall, 1997, pp. 12–44.
[15] I. Nafkha, A. Jaoua, Cooperative conceptual retrieval for heterogeneous information, in: ICDIM 2006, 2006.
[16] A. Gupta, N. Kumar, V. Bhatnagar, Analysis of medical data using data mining and formal concept analysis, World Acad. Sci., Eng. Technol. 11 (2007).
[17] C. Merz, P. Murthy, UCI repository of machine learning database, ftp://ftp.ics.uci.edu/pub/machine-learning-databases/, 1996.
[18] S. Elloumi, J. Jaam, A. Hasnah, A. Jaoua, I. Nafkha, A multi-level conceptual data reduction approach based on the Lukasiewicz implication, Inf. Sci. 163 (2004) 253–262.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 194–211

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Multirelations with infinite computations

Walter Guttmann

Department of Computer Science and Software Engineering, University of Canterbury, New Zealand

ARTICLE INFO

Article history:
Available online 12 February 2014

Keywords:
Approximation order
Median
Multirelations
Program semantics
Relations
RelView
Sequential computations

ABSTRACT

Multirelations model computations with both angelic and demonic non-determinism. We extend multirelations to represent finite and infinite computations independently. We derive an approximation order for multirelations assuming only that the endless loop is its least element and that the lattice operations are isotone. We use relations, relation algebra and RelView for representing and calculating with multirelations and for finding the approximation order.

1. Introduction

Computation models that support both angelic and demonic choice are useful for modelling contracts between agents, interaction, games and protocols [1,20]. Finite computations in such models can be represented by multirelations [22–24]. However, multirelations have no means for representing computations that fail to terminate independently of finite computations; this is similar to relational computation models such as those of [17]. The first goal of this paper is to extend multirelations to represent finite and infinite computations independently.

The basic idea to achieve this is the same as for relations, namely to extend the state space. The main challenge is to provide a suitable approximation order, which is needed to define the semantics of recursion. In some relational models, the Egli–Milner order is used for approximation. But already in the relational setting it is not clear which approximation order to use when more precise models are considered; these require variations of the Egli–Milner order [11,14]. It is even less obvious how to generalise the Egli–Milner order to multirelations. Thus the second goal of this paper is to find means of defining approximation which are independent of particular computation models.

We apply relational methods to achieve the two goals. First, we express the basic definitions, operations and properties of multirelations in terms of relations. This involves transforming logical expressions to relational formulas and algebraic calculations with these formulas. Second, we derive relational programs for investigating the approximation order in RelView [3]. From particular instances of approximation we distill a relational definition that is suitable for multirelations; its properties are again shown by applying algebra.

The main contributions of this paper are as follows:

- We introduce strict multirelations in Section 5.2 as a reaction to the failure of having an approximation order for all up-closed multirelations.

E-mail address: walter.guttmann@canterbury.ac.nz.

http://dx.doi.org/10.1016/j.jlap.2014.02.008
2352-2208/

- We give an approximation order for strict, up-closed multirelations, which is expressed by Theorem 15 in terms of relations. It turns out to be the same as an order in a previous study of pointed distributive lattices [10]. We thus obtain many useful properties including representations of least fixpoints.
- In Theorem 18 we show that our approximation order is the $\subseteq$-least partial order which has the endless loop as least element and isotone lattice operations. This is a new characterisation with very weak assumptions expected to hold in many computation models.
- We give definitions, operations and properties of multirelations, multirelations with a special state, the endless loop, up-closed multirelations and strict multirelations in terms of relations. We implement programs using such multirelations in RelView.

Section 2 recalls heterogeneous relations. Section 3 expresses multirelations in terms of relations and shows how they represent computations. Section 4 extends the state space to represent infinite computations. Section 5 applies RelView to derive an approximation order, shows that it is suitable for multirelations, expresses it in terms of relations, characterises it based on weak assumptions, and instantiates previous results for the representation of fixpoints.

Isotone predicate transformers are isomorphic to up-closed multirelations [22,1,23,16]. Many basic properties of multirelations shown in Section 3 are known from existing research on these models; here we give a relational development. Sections 4 and 5 extend these models by representing finite and infinite computations independently. For example, a nondeterministic choice between the endless loop and the program that does not change the state will be different from both of these programs. Existing approaches that make this distinction do not support both angelic and demonic choice [18,21, 15,8].

2. Relations

In this section we recall basic definitions, operations and properties of heterogeneous relations [28,27,26]; see also [4,25,2].

2.1. Basic operations

Given sets A and B, a relation R of type $A \leftrightarrow B$ is a subset of the Cartesian product $A \times B$; we write $R : A \leftrightarrow B$ and abbreviate $(x, y) \in R$ by R_{xy}. The relations of type $A \leftrightarrow B$ form a complete Boolean algebra with union $\cup$, arbitrary union $\bigcup$, intersection $\cap$, arbitrary intersection $\bigcap$, complement $\overline{\ }$ and partial order $\subseteq$ bounded by the empty relation $\mathsf{O} = \emptyset$ and the universal relation $\mathsf{T} = A \times B$. The composition of two relations $Q : A \leftrightarrow B$ and $R : B \leftrightarrow C$ is the relation $Q\,R : A \leftrightarrow C$ defined by $(Q\,R)_{xz} \Leftrightarrow \exists y \in B : Q_{xy} \wedge R_{yz}$. The identity relation $\mathsf{I} : A \leftrightarrow A$ is defined by $\mathsf{I}_{xy} \Leftrightarrow x = y$. The converse of a relation $R : A \leftrightarrow B$ is the relation $R^\smile : B \leftrightarrow A$ defined by $R^\smile{}_{xy} \Leftrightarrow R_{yx}$. Union, intersection and composition are defined only if the types of the involved relations match as indicated above; we tacitly assume this in the remainder of this paper. Composition has higher precedence than union and intersection.

Union, intersection, composition and converse are $\subseteq$-isotone; complement is $\subseteq$-antitone. We furthermore use the following properties:

- Composition distributes over $\bigcup$ and converse distributes over $\bigcup$, $\bigcap$ and $\overline{\ }$.
- $(Q\,R)^\smile = R^\smile Q^\smile$ and $R^{\smile\smile} = R$ and $\mathsf{O}^\smile = \mathsf{O}$ and $\mathsf{I}^\smile = \mathsf{I}$ and $\mathsf{T}^\smile = \mathsf{T}$.
- $\mathsf{O}R = \mathsf{O}$ and $R\mathsf{O} = \mathsf{O}$ and $\mathsf{I}R = R = R\mathsf{I}$ and $\mathsf{T}\mathsf{T} = \mathsf{T}$.
- $P\,Q \subseteq R \Leftrightarrow P^\smile \overline{R} \subseteq \overline{Q} \Leftrightarrow \overline{R}\,Q^\smile \subseteq \overline{P}$; these are the Schröder equivalences.
- $\mathsf{T}R\mathsf{T} = \mathsf{T}$ if $R \neq \mathsf{O}$; this is the Tarski rule.
- $R = R^\smile$ if $R \subseteq \mathsf{I}$.

A relation R is total if $R\mathsf{T} = \mathsf{T}$, univalent if $R^\smile R \subseteq \mathsf{I}$, surjective if $\mathsf{T}R = \mathsf{T}$, injective if $RR^\smile \subseteq \mathsf{I}$, a mapping if R is total and univalent, bijective if R is injective and surjective, a vector if $R\mathsf{T} = R$, reflexive if $\mathsf{I} \subseteq R$, transitive if $RR \subseteq R$, antisymmetric if $R \cap R^\smile \subseteq \mathsf{I}$, a preorder if R is reflexive and transitive, and a partial order if R is an antisymmetric preorder. A vector $R : A \leftrightarrow B$ represents a set which contains those elements of A that are related by R to every element of B. We use the following properties:

- $(\bigcap_{i\in I} Q_i)R = \bigcap_{i\in I}(Q_i R)$ if R is injective and $\overline{Q\,R} = \overline{Q}\,R$ if R is bijective.
- $P\,Q \subseteq R \Leftrightarrow Q \subseteq P^\smile R$ and $Q\,P^\smile \subseteq R \Leftrightarrow Q \subseteq RP$ if P is bijective.
- $(P\mathsf{T} \cap Q)R = P\mathsf{T} \cap Q\,R$ and $Q\,(R \cap \mathsf{T}P) = Q\,R \cap \mathsf{T}P$ and $(Q \cap \mathsf{T}P^\smile)R = Q\,(P\mathsf{T} \cap R)$.
- Vectors are closed under $\bigcup$, $\bigcap$ and $\overline{\ }$.

2.2. Residuals and symmetric quotient

The right residual is $Q \backslash R = \overline{Q^\smile \overline{R}}$, so $(Q \backslash R)_{xy} \Leftrightarrow (\forall z\colon\ Q_{zx} \Rightarrow R_{zy})$. The left residual is $Q / R = \overline{\overline{Q} R^\smile}$, so $(Q / R)_{xy} \Leftrightarrow (\forall z\colon\ R_{yz} \Rightarrow Q_{xz})$. The symmetric quotient is $(Q \div R) = (Q \backslash R) \cap (Q^\smile / R^\smile)$, whence $(Q \div R)_{xy} \Leftrightarrow (\forall z\colon\ Q_{zx} \Leftrightarrow R_{zy})$. We use the following properties of these constructions:

(1) $\backslash$ is $\subseteq$-antitone in its first argument and $\subseteq$-isotone in its second argument.
(2) $/$ is $\subseteq$-isotone in its first argument and $\subseteq$-antitone in its second argument.
(3) $Q\,(Q \backslash R) \subseteq R$.
(4) $Q\,P \subseteq R \Leftrightarrow P \subseteq Q \backslash R$.
(5) $P\,Q \subseteq R \Leftrightarrow P \subseteq R / Q$.
(6) $Q \backslash (P \cap R) = (Q \backslash P) \cap (Q \backslash R)$.
(7) $\mathsf{I} \backslash R = R$.
(8) $R / \mathsf{I} = R$.
(9) $R \backslash \mathsf{T} = \mathsf{T}$.
(10) $(Q \backslash R) P = Q \backslash (R P)$ if P is bijective.
(11) $(Q / R) P = Q / (P^\smile R)$ if P is bijective.
(12) $(Q \div R)^\smile = (R \div Q)$.
(13) $(\overline{R} \div \overline{Q}) = (R \div Q)$.
(14) $(P \div Q)(Q \div R) = (P \div R) \cap \mathsf{T}(Q \div R)$.
(15) $(Q \div \mathsf{O}) = Q \backslash \mathsf{O}$.

2.3. Power sets

The power set of A is $2^A = \{B \mid B \subseteq A\}$. The membership relation $\mathsf{E} : A \leftrightarrow 2^A$ is defined by $\mathsf{E}_{xY} \Leftrightarrow x \in Y$. The subset relation $\mathsf{S} : 2^A \leftrightarrow 2^A$ is $\mathsf{S} = \mathsf{E} \backslash \mathsf{E}$, whence $\mathsf{S}_{XY} \Leftrightarrow X \subseteq Y$. The complement relation $\mathsf{C} : 2^A \leftrightarrow 2^A$ is $\mathsf{C} = (\mathsf{E} \div \overline{\mathsf{E}})$, whence $\mathsf{C}_{XY} \Leftrightarrow X = \overline{Y}$. We use the following properties from the literature:

(16) $(R \div \mathsf{E})$ is a mapping.
(17) $(\mathsf{E} \div R)$ is bijective.
(18) $\mathsf{E}(\mathsf{E} \div R) = R$.
(19) S is a partial order.
(20) $\mathsf{E}\mathsf{S} = \mathsf{E}$.
(21) C is a bijective mapping.
(22) $\mathsf{C}\mathsf{C} = \mathsf{I}$.
(23) $\mathsf{C}^\smile = \mathsf{C}$.
(24) $\mathsf{E}\mathsf{C} = \overline{\mathsf{E}}$.

To these, we add the following properties.

Theorem 1.

1. E *is total.*
2. $\mathsf{E} / \mathsf{E} = \mathsf{I}$.
3. $(Q \div \mathsf{E})(\mathsf{E} \div R) = (Q \div R)$.
4. $\mathsf{S}(\mathsf{E} \div R) = \mathsf{E} \backslash R$.
5. $\mathsf{E}(\mathsf{E} \backslash R) = R$.
6. $\mathsf{C}\mathsf{S} = \mathsf{S}^\smile \mathsf{C}$.

Proof.

1. $\mathsf{T} = (\mathsf{T} \div \mathsf{E})\mathsf{T} \subseteq (\mathsf{T} \backslash \mathsf{E})\mathsf{T} \subseteq (\mathsf{I} \backslash \mathsf{E})\mathsf{T} = \mathsf{E}\mathsf{T}$ using (16), (1) and (7).
2. $\mathsf{I} \subseteq \mathsf{E} / \mathsf{E} = \mathsf{E} / (\mathsf{I} \backslash \mathsf{E}) \subseteq \mathsf{E} / (\mathsf{I} \div \mathsf{E}) = \mathsf{E} / (\mathsf{E} \div \mathsf{I})^\smile = (\mathsf{E} / \mathsf{I})(\mathsf{E} \div \mathsf{I}) = \mathsf{E}(\mathsf{E} \div \mathsf{I}) = \mathsf{I}$ using (5), (7), (2), (12), (11), (17), (8) and (18).
3. $(Q \div \mathsf{E})(\mathsf{E} \div R) = (Q \div R) \cap \mathsf{T}(\mathsf{E} \div R) = (Q \div R)$ using (14) and (17).
4. $\mathsf{S}(\mathsf{E} \div R) = (\mathsf{E} \backslash \mathsf{E})(\mathsf{E} \div R) = \mathsf{E} \backslash (\mathsf{E}(\mathsf{E} \div R)) = \mathsf{E} \backslash R$ using (10), (17) and (18).
5. $\mathsf{E}(\mathsf{E} \backslash R) = \mathsf{E}\mathsf{S}(\mathsf{E} \div R) = \mathsf{E}(\mathsf{E} \div R) = R$ using the preceding claim, (20) and (18).
6. By a Schröder equivalence, (20) implies $\overline{\mathsf{E}}\mathsf{S}^\smile \subseteq \overline{\mathsf{E}}$. Thus $\mathsf{E}\mathsf{C}\mathsf{S}^\smile \mathsf{C} = \overline{\mathsf{E}}\mathsf{S}^\smile \mathsf{C} \subseteq \overline{\mathsf{E}}\mathsf{C} = \mathsf{E}\mathsf{C}\mathsf{C} = \mathsf{E}\mathsf{I} = \mathsf{E}$ using (24) and (22). Hence $\mathsf{C}\mathsf{S}^\smile \mathsf{C} \subseteq \mathsf{E} \backslash \mathsf{E} = \mathsf{S}$ using (4) and therefore $\mathsf{S}^\smile \mathsf{C} \subseteq \mathsf{C}^\smile \mathsf{S} = \mathsf{C}\mathsf{S}$ using (21) and (23). Thus also $\mathsf{C}\mathsf{S} = \mathsf{C}^\smile \mathsf{S} = (\mathsf{S}^\smile \mathsf{C})^\smile \subseteq (\mathsf{C}\mathsf{S})^\smile = \mathsf{S}^\smile \mathsf{C}^\smile = \mathsf{S}^\smile \mathsf{C}$ using (23). □

2.4. Cartesian product

The projections $p_1 : A \times B \leftrightarrow A$ and $p_2 : A \times B \leftrightarrow B$ are defined by $p_{1(x,y)z} \Leftrightarrow x = z$ and $p_{2(x,y)z} \Leftrightarrow y = z$. The tupling of relations $Q : A \leftrightarrow B$ and $R : A \leftrightarrow C$ is $[Q, R] : A \leftrightarrow B \times C$ defined by $[Q, R] = Q\,p_1{}^\smile \cap R p_2{}^\smile$ using projections p_1, p_2 of appropriate type; hence $[Q, R]_{x(y,z)} \Leftrightarrow Q_{xy} \wedge R_{xz}$. The parallel product of relations $Q : A \leftrightarrow B$ and $R : C \leftrightarrow D$ is $Q \parallel R : A \times C \leftrightarrow B \times D$ defined by $Q \parallel R = p_1 Q r_1{}^\smile \cap p_2 R r_2{}^\smile$ using projections p_1, p_2 and r_1, r_2 of appropriate types; hence $(Q \parallel R)_{(w,x)(y,z)} \Leftrightarrow Q_{wy} \wedge R_{xz}$.

3. Multirelations

In this section we recall basic definitions, operations and properties of multirelations and express them in terms of relations.

A *multirelation* [22,23] is a relation of type $A \leftrightarrow 2^B$. It maps an element of A to a set of subsets of B. Union, intersection and complement apply to multirelations as to relations. Particular multirelations are O, T and E.

3.1. Composition

Multirelational composition of $Q : A \leftrightarrow 2^B$ and $R : B \leftrightarrow 2^C$ is the multirelation $Q \mathbin{;} R : A \leftrightarrow 2^C$ given by

$$(Q \mathbin{;} R)_{xZ} \quad \Leftrightarrow \quad \exists Y \subseteq B : Q_{xY} \wedge \forall y \in Y : R_{yZ}$$

The following result expresses multirelational composition in terms of relations using a right residual; see also [26].

Theorem 2. $Q \mathbin{;} R = Q\,(\mathsf{E}\backslash R)$.

Proof.

$$\begin{array}{ll} & (Q \mathbin{;} R)_{xZ} \\ \Leftrightarrow & \exists Y \subseteq B : Q_{xY} \wedge \forall y \in Y : R_{yZ} \\ \Leftrightarrow & \exists Y \subseteq B : Q_{xY} \wedge \forall y \in B : \mathsf{E}_{yY} \Rightarrow R_{yZ} \\ \Leftrightarrow & \exists Y \subseteq B : Q_{xY} \wedge (\mathsf{E}\backslash R)_{YZ} \\ \Leftrightarrow & \big(Q\,(\mathsf{E}\backslash R)\big)_{xZ} \qquad \square \end{array}$$

The following result gives properties of multirelational composition.

Theorem 3.

1. *The operation* ; *is* $\subseteq$*-isotone.*
2. $Q \mathbin{;} \mathsf{T} = Q\,\mathsf{T}$.
3. $Q \mathbin{;} R = Q$ *if* Q *is a vector.*
4. $\mathsf{O} \mathbin{;} R = \mathsf{O}$.
5. $\mathsf{E} \mathbin{;} R = R$.
6. $\mathsf{T} \mathbin{;} R = \mathsf{T}$.
7. $(\bigcup_{i \in I} Q_i) \mathbin{;} R = \bigcup_{i \in I} (Q_i \mathbin{;} R)$.

Proof.

1. This follows from $Q \mathbin{;} R = Q\,(\mathsf{E}\backslash R)$ using (1) and that relational composition is $\subseteq$-isotone.
2. $Q \mathbin{;} \mathsf{T} = Q\,(\mathsf{E}\backslash\mathsf{T}) = Q\,\mathsf{T}$ using (9).
3. $Q\,(\mathsf{E} \div \mathsf{O})^\smile \subseteq Q\,\mathsf{T} = Q$ since Q is a vector. Hence

$$Q \subseteq Q\,(\mathsf{E} \div \mathsf{O}) = Q\,(\mathsf{E}\backslash\mathsf{O}) = Q \mathbin{;} \mathsf{O} \subseteq Q \mathbin{;} R \subseteq Q \mathbin{;} \mathsf{T} = Q\,\mathsf{T} = Q$$

   using (17), (15), the first claim and the second claim.
4. This follows from the preceding claim since O is a vector.
5. $\mathsf{E} \mathbin{;} R = \mathsf{E}(\mathsf{E}\backslash R) = R$ using Theorem 1.5.
6. This follows from the third claim since T is a vector.
7. $(\bigcup_{i \in I} Q_i) \mathbin{;} R = (\bigcup_{i \in I} Q_i)(\mathsf{E}\backslash R) = \bigcup_{i \in I} (Q_i (\mathsf{E}\backslash R)) = \bigcup_{i \in I} (Q_i \mathbin{;} R)$. $\square$

3.2. Dual

The *dual* of a multirelation $R : A \leftrightarrow 2^B$ is the multirelation $R^d : A \leftrightarrow 2^B$ given by

$$R^d{}_{xY} \quad \Leftrightarrow \quad \neg R_{x\overline{Y}}$$

where $\overline{Y}$ is the complement of Y relative to B. The following result expresses the dual in terms of relations using the complement relation C.

Theorem 4. *$R^d = \overline{R}\mathsf{C} = \overline{R\mathsf{C}}$.*

Proof.

$$\begin{aligned} & R^d{}_{xY} \\ \Leftrightarrow \quad & \neg R_{x\overline{Y}} \\ \Leftrightarrow \quad & \exists Z \subseteq B : Z = \overline{Y} \wedge \neg R_{xZ} \\ \Leftrightarrow \quad & \exists Z \subseteq B : \mathsf{C}_{ZY} \wedge \overline{R}_{xZ} \\ \Leftrightarrow \quad & (\overline{R}\mathsf{C})_{xY} \end{aligned}$$

Finally, $\overline{R}\mathsf{C} = \overline{R\mathsf{C}}$ using (21). □

The following result gives properties of dual multirelations.

Theorem 5.

1. *The operation $\cdot^d$ is $\subseteq$-antitone.*
2. *$R^d = \overline{R}$ if R is a vector.*
3. *$\mathsf{O}^d = \mathsf{T}$.*
4. *$\mathsf{E}^d = \mathsf{E}$.*
5. *$\mathsf{T}^d = \mathsf{O}$.*
6. *$R^{d^d} = R$.*
7. *$(\bigcup_{i\in I} R_i)^d = \bigcap_{i\in I} {R_i}^d$.*
8. *$(\bigcap_{i\in I} R_i)^d = \bigcup_{i\in I} {R_i}^d$.*

Proof.

1. This follows from $R^d = \overline{R}\mathsf{C}$ since relational composition is $\subseteq$-isotone and complement is $\subseteq$-antitone.
2. $R^d = \overline{R\mathsf{C}} = \overline{R\mathsf{T}\mathsf{C}} = \overline{R\mathsf{T}} = \overline{R}$ using that R is a vector and (21).
3. This follows from the preceding claim since O is a vector.
4. $\mathsf{E}^d = \overline{\mathsf{E}\mathsf{C}} = \overline{\mathsf{E}} = \mathsf{E}$ using (24).
5. This follows from the third claim since T is a vector.
6. $R^{d^d} = \overline{R^d}\mathsf{C} = R\mathsf{C}\mathsf{C} = R\mathsf{I} = R$ using (22).
7. $(\bigcup_{i\in I} R_i)^d = \overline{(\bigcup_{i\in I} R_i)\mathsf{C}} = \overline{\bigcup_{i\in I}(R_i\mathsf{C})} = \bigcap_{i\in I} \overline{R_i\mathsf{C}} = \bigcap_{i\in I} {R_i}^d$.
8. $(\bigcap_{i\in I} R_i)^d = (\bigcap_{i\in I} {R_i}^{d^d})^d = (\bigcup_{i\in I} {R_i}^d)^{d^d} = \bigcup_{i\in I} {R_i}^d$ using the preceding two claims. □

3.3. Up-closed multirelations

A multirelation $R : A \leftrightarrow 2^B$ is *up-closed* if

$$R_{xY} \wedge Y \subseteq Z \quad \Rightarrow \quad R_{xZ}$$

for each $x \in A$ and $Y, Z \subseteq B$. This means that if an element of A is related to a set Y, it must be related to all supersets of Y. The following result expresses this property in terms of relations using the subset relation S; see also [26].

Theorem 6. *R is up-closed if and only if $R = R\mathsf{S}$.*

Proof.

$$\begin{aligned}
& R : A \leftrightarrow 2^B \text{ is up-closed} \\
\Leftrightarrow\ & \forall x \in A : \forall Y, Z \subseteq B : R_{xY} \wedge Y \subseteq Z \Rightarrow R_{xZ} \\
\Leftrightarrow\ & \forall x \in A : \forall Y, Z \subseteq B : R_{xY} \wedge \mathsf{S}_{YZ} \Rightarrow R_{xZ} \\
\Leftrightarrow\ & \forall x \in A : \forall Z \subseteq B : (\exists Y \subseteq B : R_{xY} \wedge \mathsf{S}_{YZ}) \Rightarrow R_{xZ} \\
\Leftrightarrow\ & \forall x \in A : \forall Z \subseteq B : (R\mathsf{S})_{xZ} \Rightarrow R_{xZ} \\
\Leftrightarrow\ & R\mathsf{S} \subseteq R
\end{aligned}$$

This is equivalent to $R\mathsf{S} = R$ since $R = R\mathsf{I} \subseteq R\mathsf{S}$ using (19). □

The following result shows that the membership relation is an identity for the composition of up-closed multirelations. It also shows that multirelational operations preserve the property of being up-closed.

Theorem 7.

1. $R\ ;\ \mathsf{E} = R$ *if and only if* R *is up-closed.*
2. $P\ ;\ (Q\ ;\ R) = (P\ ;\ Q)\ ;\ R$ *if* Q *is up-closed.*
3. $(Q\ ;\ R)^d = Q^d\ ;\ R^d$ *if* Q *is up-closed.*
4. *Every vector is up-closed.*
5. O, E *and* T *are up-closed.*
6. $Q\ ;\ R$ *and* PR *and* R^d *are up-closed if* R *is up-closed.*
7. $\bigcup_{i\in I} R_i$ *and* $\bigcap_{i\in I} R_i$ *are up-closed if* R_i *is up-closed for each* $i \in I$.
8. $(\bigcap_{i\in I} Q_i)\ ;\ R = \bigcap_{i\in I}(Q_i\ ;\ R)$ *if* Q_i *is up-closed for each* $i \in I$.

Proof.

1. $R\ ;\ \mathsf{E} = R(\mathsf{E}\backslash\mathsf{E}) = R\mathsf{S} = R$ if R is up-closed. From $R\ ;\ \mathsf{E} = R$ we obtain $R\mathsf{S} = R(\mathsf{E}\backslash\mathsf{E}) = R\ ;\ \mathsf{E} = R$.
2. $\mathsf{E}(\mathsf{E}\backslash Q)(\mathsf{E}\backslash R) \subseteq Q(\mathsf{E}\backslash R)$ using (3), whence $(\mathsf{E}\backslash Q)(\mathsf{E}\backslash R) \subseteq \mathsf{E}\backslash(Q(\mathsf{E}\backslash R))$ using (4). Moreover

$$\mathsf{E}\backslash\big(Q(\mathsf{E}\backslash R)\big) = \mathsf{E}\backslash\big(Q\mathsf{S}(\mathsf{E}\div R)\big) = \mathsf{E}\backslash\big(Q(\mathsf{E}\div R)\big) = (\mathsf{E}\backslash Q)(\mathsf{E}\div R) \subseteq (\mathsf{E}\backslash Q)(\mathsf{E}\backslash R)$$

using Theorem 1.4, that Q is up-closed, (10) and (17). Thus $\mathsf{E}\backslash(Q(\mathsf{E}\backslash R)) = (\mathsf{E}\backslash Q)(\mathsf{E}\backslash R)$, whence

$$P\ ;\ (Q\ ;\ R) = P\ ;\ \big(Q(\mathsf{E}\backslash R)\big) = P\big(\mathsf{E}\backslash\big(Q(\mathsf{E}\backslash R)\big)\big) = P(\mathsf{E}\backslash Q)(\mathsf{E}\backslash R) = \big(P(\mathsf{E}\backslash Q)\big)\ ;\ R = (P\ ;\ Q)\ ;\ R$$

3. $Q\mathsf{S} = Q$ implies $\overline{Q}\mathsf{S}^\smile \subseteq \overline{Q} = \overline{Q}\mathsf{I} = \overline{Q}\mathsf{I}^\smile \subseteq \overline{Q}\mathsf{S}^\smile$ using a Schröder equivalence and (19), whence $\overline{Q}\mathsf{S}^\smile = \overline{Q}$. Therefore

$$\begin{aligned}
(Q\ ;\ R)^d &= \overline{Q\ ;\ R\mathsf{C}} = \overline{Q(\mathsf{E}\backslash R)}\mathsf{C} = \overline{Q\mathsf{S}(\mathsf{E}\div R)}\mathsf{C} = \overline{Q(\mathsf{E}\div R)}\mathsf{C} = \overline{Q}(\mathsf{E}\div R)\mathsf{C} = \overline{Q}(\bar{\mathsf{E}}\div\bar{R})\mathsf{C} \\
&= \overline{Q}(\bar{\mathsf{E}}\div\mathsf{E})(\mathsf{E}\div\bar{R})\mathsf{C} = \overline{Q}\mathsf{C}^\smile(\mathsf{E}\div\bar{R})\mathsf{C} = \overline{Q}\mathsf{C}(\mathsf{E}\div\bar{R})\mathsf{C} = \overline{Q}\mathsf{S}^\smile\mathsf{C}(\mathsf{E}\div\bar{R})\mathsf{C} = \overline{Q}\mathsf{C}\mathsf{S}(\mathsf{E}\div\bar{R})\mathsf{C} \\
&= \overline{Q}\mathsf{C}(\mathsf{E}\backslash\bar{R})\mathsf{C} = Q^d(\mathsf{E}\backslash\bar{R})\mathsf{C} = Q^d\big(\mathsf{E}\backslash(\bar{R}\mathsf{C})\big) = Q^d(\mathsf{E}\backslash R^d) = Q^d\ ;\ R^d
\end{aligned}$$

using Theorem 1.4, that Q is up-closed, (17), (13), Theorem 1.3, (12), (23), Theorems 1.6 and 1.4, (10) and (21).
4. Let R be a vector. Then $R = R\mathsf{I} \subseteq R\mathsf{S} \subseteq R\mathsf{T} = R$ using (19), whence $R = R\mathsf{S}$.
5. This follows from the first claim and Theorems 3.4, 3.5 and 3.6.
6. $(Q\ ;\ R)\ ;\ \mathsf{E} = Q\ ;\ (R\ ;\ \mathsf{E}) = Q\ ;\ R$ using the first two claims since R is up-closed. Clearly $PR\mathsf{S} = PR$ if R is up-closed. Moreover $R^d\ ;\ \mathsf{E} = R^d\ ;\ \mathsf{E}^d = (R\ ;\ \mathsf{E})^d = R^d$ using Theorem 5.4, the third claim and the first claim since R is up-closed.
7. $(\bigcup_{i\in I} R_i)\mathsf{S} = \bigcup_{i\in I}(R_i\mathsf{S}) = \bigcup_{i\in I} R_i$ using that R_i is up-closed for each $i \in I$. Moreover $(\bigcap_{i\in I} R_i)\mathsf{S} \subseteq \bigcap_{i\in I}(R_i\mathsf{S}) = \bigcap_{i\in I} R_i = (\bigcap_{i\in I} R_i)\mathsf{I} \subseteq (\bigcap_{i\in I} R_i)\mathsf{S}$ using that R_i is up-closed for each $i \in I$ and (19).
8. The preceding claim implies

$$\begin{aligned}
\big(\textstyle\bigcap_{i\in I} Q_i\big)\ ;\ R &= \big(\textstyle\bigcap_{i\in I} Q_i\big)(\mathsf{E}\backslash R) = \big(\textstyle\bigcap_{i\in I} Q_i\big)\mathsf{S}(\mathsf{E}\div R) = \big(\textstyle\bigcap_{i\in I} Q_i\big)(\mathsf{E}\div R) \\
&= \textstyle\bigcap_{i\in I}\big(Q_i(\mathsf{E}\div R)\big) = \bigcap_{i\in I}\big(Q_i\mathsf{S}(\mathsf{E}\div R)\big) = \bigcap_{i\in I}\big(Q_i(\mathsf{E}\backslash R)\big) = \bigcap_{i\in I}(Q_i\ ;\ R)
\end{aligned}$$

using Theorem 1.4 and (17). □

3.4. Computations

In the remainder of this paper we look at multirelations of type $A \leftrightarrow 2^B$ where $A = B$. Such multirelations model computations that involve two kinds of non-determinism associated with players in a game [22] and sometimes called angelic and demonic [1,24]. The underlying intuition is that there are two players, the 'angel' who makes a choice and the 'demon' who subsequently makes a choice based on the angel's selection. The following description is taken from [13].

Consider a multirelation $R : A \leftrightarrow 2^A$, a state $x \in A$ and the set of subsets $Ys = \{Y \subseteq A \mid R_{xY}\}$ to which x is related. The outer set structure of Ys represents angelic choice: the angel chooses a set $Y \in Ys$. The inner set structure of Ys represents demonic choice: the demon subsequently chooses an element $y \in Y$ which is the next state.

For example, let $A = \{0, 1, 2, 3, 4\}$ and let $R : A \leftrightarrow 2^A$ be given by

$$\begin{array}{lcl} 0 & \mapsto & \big\{\{1,2\},\{1,3,4\}\big\} \\ 1 & \mapsto & \big\{\{1\},\{2\}\big\} \\ 2 & \mapsto & \big\{\{1,2\}\big\} \\ 3 & \mapsto & \{\emptyset\} \\ 4 & \mapsto & \emptyset \end{array}$$

It describes the following computation. In state 0 an angelic choice between two sets $\{1, 2\}$ and $\{1, 3, 4\}$ is made. If the angel chooses $\{1, 3, 4\}$ the demon chooses which of 1, 3 and 4 is the next state. If the angel chooses $\{1, 2\}$ the demon chooses one of 1 and 2 as the next state. State 1 has a purely angelic choice between states 1 and 2, because each inner set is a singleton set in which the demon's choice is fixed. State 2 has a purely demonic choice between states 1 and 2, because the outer set is a singleton set in which the angel's choice is fixed. In state 3 the computation fails to progress since the demon cannot choose from the empty set. In the game interpretation this means that the angel wins; in terms of refinement this means that any specification is satisfied. In state 4 the computation fails to progress since the angel cannot choose from the empty set. In the game interpretation this means that the demon wins; in terms of refinement this means that no specification is satisfied.

The multirelation R is not up-closed, but can be extended to an up-closed multirelation $Q = R\mathsf{S}$ by adding the required supersets. Adding a superset Z of a set Y to which x is related does not change the computational interpretation. This just increases the angelic choice by options which are not interesting for the angel because they subsequently allow more choices for the demon. For example, in Q the state 0 is related to $\{1, 2, 3\}$, but angelic choice prefers $\{1, 2\}$ so as to restrict demonic choice as much as possible.

In the remainder of this paper we focus on up-closed multirelations. They feature a nice interplay between the outer and the inner set structures. Forming the union of up-closed multirelations simultaneously increases angelic choice and decreases demonic choice, while intersection simultaneously decreases angelic choice and increases demonic choice. Union and intersection thus provide angelic and demonic choice, respectively, at the level of computations.

Angelic choice and demonic choice are duals of each other. In an alternative computational interpretation the outer set structure describes demonic choice and the inner set structure describes angelic choice [7]. Then union represents demonic choice and intersection represents angelic choice.

4. Infinite computations

Multirelations have no particular means for representing infinite computations. The situation is akin to relational computation models like those of [17]. A special multirelation such as O or T could be interpreted as an infinite computation. However, the properties $\mathsf{O} \cap R = \mathsf{O}$ and $\mathsf{O} \cup R = R = \mathsf{T} \cap R$ and $\mathsf{T} \cup R = \mathsf{T}$ would imply that finite and infinite computations starting in the same state cannot be represented independently.

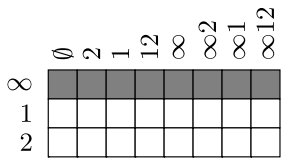
In this section, we extend the multirelational model to represent infinite computations independently of finite computations. The general idea is the same as for relations, namely to extend the state space by a special element ∞ that represents the outcome of an infinite computation. We therefore consider up-closed multirelations of type $A \leftrightarrow 2^A$ where $\infty \in A$. Note that we do not add traces – finite or infinite – but maintain that multirelations ignore the intermediate states of a computation.

4.1. A special state

State spaces extended this way are typically structured by a partial order, for example, the flat order with ∞ as least element. We use an alternative approach to distinguish ∞ by considering the special multirelation $\mathsf{N} : A \leftrightarrow 2^A$ given by

$$\mathsf{N}_{xY} \Leftrightarrow x = \infty$$

For $A = \{\infty, 1, 2\}$ its matrix is



Theorem 7.4 since N is a vector.

2. N is surjective as $\mathsf{TN} = \mathsf{TNT} = \mathsf{T}$ using that N is a vector and the Tarski rule with $\mathsf{N} \neq \mathsf{O}$.
3. Using the preceding claim, $\mathsf{T} \subseteq \mathsf{N}^\smile\mathsf{N}$ is equivalent to $\mathsf{NT} \subseteq \mathsf{N}$, which holds since N is a vector.
4. $\mathsf{TN}^\smile\mathsf{T} = \mathsf{T}$ using the Tarski rule since $\mathsf{N} \neq \mathsf{O}$ implies $\mathsf{N}^\smile \neq \mathsf{O}$.
5. $\mathsf{N} \cap R \subseteq \mathsf{N}$ and $\mathsf{N} \cap R = \mathsf{I}(\mathsf{N} \cap R) \subseteq \mathsf{T}(\mathsf{N} \cap R)$. Moreover $\mathsf{N} \cap \mathsf{T}(\mathsf{N} \cap R) \subseteq \mathsf{N}$ and

$$\mathsf{N} \cap \mathsf{T}(\mathsf{N} \cap R) = \mathsf{N} \cap \mathsf{T}(\mathsf{NT} \cap R) = \mathsf{N} \cap \mathsf{TN}^\smile R = \mathsf{NT} \cap \mathsf{TN}^\smile R = \mathsf{NTN}^\smile R = \mathsf{NN}^\smile R \subseteq \mathsf{I}R = R$$

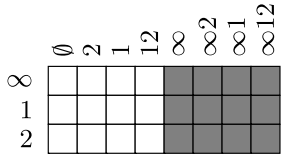
   using that N is a vector and injective.
6. This follows from Theorem 3.3 since N is a vector. □

4.2. The endless loop

Based on N we define the multirelation $\mathsf{L} : A \leftrightarrow 2^A$ which represents the endless loop. It relates each element of A to every subset of A that contains ∞ and is given by

$$\mathsf{L} = \mathsf{T}(\mathsf{N} \cap \mathsf{E})$$

A simple calculation shows $\mathsf{L}_{xY} \Leftrightarrow \infty \in Y$. For $A = \{\infty, 1, 2\}$ its matrix is



The following result gives properties of L.

Theorem 9.

1. $\mathsf{L} = \mathsf{TN}^\smile\mathsf{E}$.
2. $\mathsf{L}^\smile\mathsf{T} = \mathsf{E}^\smile\mathsf{N}$.
3. L *is total.*
4. L *is up-closed.*
5. $\mathsf{L}^\smile$ *is a vector.*
6. $\mathsf{TL} \subseteq \mathsf{E} \backslash \mathsf{L}$.
7. $\mathsf{L}(\mathsf{E} \backslash \mathsf{N}) = \mathsf{T}$.
8. $\mathsf{N} = \overline{(\bar{\mathsf{E}} \cap \mathsf{L})\mathsf{T}}$.

9. $\mathsf{N} \cap \mathsf{E} = \mathsf{N} \cap \mathsf{L}$.
10. $\mathsf{L}^d = \mathsf{L}$.
11. $\mathsf{L} ; R = \mathsf{T}\mathsf{N}^\smile R = \mathsf{T}(\mathsf{N} \cap R)$.
12. $\mathsf{L} \subseteq \mathsf{L} ; R$ *if and only if* $\mathsf{N} \cap \mathsf{E} \subseteq R$.
13. $\mathsf{L} = \mathsf{L} ; R$ *if and only if* $\mathsf{N} \cap \mathsf{E} = \mathsf{N} \cap R$.
14. $\mathsf{L} = \mathsf{L} ; \mathsf{L} = \mathsf{L} ; \mathsf{E}$.

Proof.

1. $\mathsf{L} = \mathsf{T}(\mathsf{N} \cap \mathsf{E}) = \mathsf{T}(\mathsf{N}\mathsf{T} \cap \mathsf{E}) = \mathsf{T}\mathsf{N}^\smile \mathsf{E}$ using that N is a vector.
2. $\mathsf{L}^\smile \mathsf{T} = \mathsf{E}^\smile \mathsf{N}\mathsf{T}\,\mathsf{T} = \mathsf{E}^\smile \mathsf{N}\mathsf{T} = \mathsf{E}^\smile \mathsf{N}$ using the preceding claim and that N is a vector.
3. $\mathsf{L}\mathsf{T} = \mathsf{T}\mathsf{N}^\smile \mathsf{E}\mathsf{T} = \mathsf{T}\mathsf{N}^\smile \mathsf{T} = \mathsf{T}$ using the first claim and Theorems 1.1 and 8.4.
4. L is up-closed using the first claim and Theorems 7.5 and 7.6.
5. $\mathsf{T}\mathsf{L} = \mathsf{T}\,\mathsf{T}(\mathsf{N} \cap \mathsf{E}) = \mathsf{T}(\mathsf{N} \cap \mathsf{E}) = \mathsf{L}$, whence $\mathsf{L}^\smile$ is a vector.
6. $\mathsf{E}\mathsf{T}\mathsf{L} = \mathsf{T}\mathsf{L} = \mathsf{L}$ using Theorem 1.1 and the preceding claim. Hence $\mathsf{T}\mathsf{L} \subseteq \mathsf{E}\backslash\mathsf{L}$ using (4).
7. $\overline{\mathsf{L}(\mathsf{E}\backslash\mathsf{N})} = \overline{\mathsf{T}\mathsf{N}^\smile \mathsf{E}(\mathsf{E}\backslash\mathsf{N})} = \overline{\mathsf{T}\mathsf{N}^\smile \mathsf{N}} = \overline{\mathsf{T}\,\mathsf{T}} = \overline{\mathsf{T}} = \mathsf{T}$ using the first claim and Theorems 1.5 and 8.3.
8. $\overline{(\overline{\mathsf{E}} \cap \mathsf{L})\mathsf{T}} = \overline{(\overline{\mathsf{E}} \cap \mathsf{T}\mathsf{N}^\smile \mathsf{E})\mathsf{T}} = \overline{\overline{\mathsf{E}}\mathsf{E}^\smile \mathsf{N}\mathsf{T}} = \overline{\overline{\mathsf{E}}\mathsf{E}^\smile \mathsf{N}} = \overline{\overline{\mathsf{E}}\mathsf{E}^\smile \mathsf{N}} = (\mathsf{E}/\mathsf{E})\mathsf{N} = \mathsf{I}\mathsf{N} = \mathsf{N}$ using the first claim, that N is a vector and Theorems 8.2 and 1.2.
9. $\mathsf{N} \cap \mathsf{E} = \mathsf{N} \cap \mathsf{T}(\mathsf{N} \cap \mathsf{E}) = \mathsf{N} \cap \mathsf{L}$ using Theorem 8.5.
10. $\overline{\mathsf{L}^d} = \mathsf{L}\mathsf{C} = \mathsf{T}\mathsf{N}^\smile \mathsf{E}\mathsf{C} = \mathsf{T}\mathsf{N}^\smile \overline{\mathsf{E}} = \mathsf{T}(\mathsf{N}\mathsf{T} \cap \overline{\mathsf{E}}) = \mathsf{T}(\mathsf{N} \cap \overline{\mathsf{E}})$ using the first claim, (24) and that N is a vector. Hence

$$\mathsf{L} \cup \overline{\mathsf{L}^d} = \mathsf{T}\mathsf{N}^\smile \mathsf{E} \cup \mathsf{T}\mathsf{N}^\smile \overline{\mathsf{E}} = \mathsf{T}\mathsf{N}^\smile (\mathsf{E} \cup \overline{\mathsf{E}}) = \mathsf{T}\mathsf{N}^\smile \mathsf{T} = \mathsf{T}$$

using the first claim and Theorem 8.4. Moreover

$$\mathsf{L} \cap \overline{\mathsf{L}^d} = \mathsf{T}\mathsf{L} \cap \mathsf{T}(\mathsf{N} \cap \overline{\mathsf{E}}) = \mathsf{T}(\mathsf{N} \cap \overline{\mathsf{E}} \cap \mathsf{T}\mathsf{L}) = \mathsf{T}(\overline{\mathsf{E}} \cap \mathsf{N} \cap \mathsf{L}) = \mathsf{T}(\overline{\mathsf{E}} \cap \mathsf{N} \cap \mathsf{E}) = \mathsf{T}\mathsf{O} = \mathsf{O}$$

using the fifth claim and the ninth claim. Thus $\mathsf{L}^d = \mathsf{L}$.
11. $\mathsf{L} ; R = \mathsf{L}(\mathsf{E}\backslash R) = \mathsf{T}\mathsf{N}^\smile \mathsf{E}(\mathsf{E}\backslash R) = \mathsf{T}\mathsf{N}^\smile R = \mathsf{T}(\mathsf{N}\mathsf{T} \cap R) = \mathsf{T}(\mathsf{N} \cap R)$ using the first claim, Theorem 1.5 and that N is a vector.
12. Using the preceding claim the backward implication holds by

$$\mathsf{L} = \mathsf{T}(\mathsf{N} \cap \mathsf{E}) \subseteq \mathsf{T}(\mathsf{N} \cap R) = \mathsf{L} ; R$$

The forward implication holds since the ninth claim and the eleventh claim and Theorem 8.5 imply

$$\mathsf{N} \cap \mathsf{E} = \mathsf{N} \cap \mathsf{L} \subseteq \mathsf{N} \cap (\mathsf{L} ; R) = \mathsf{N} \cap \mathsf{T}(\mathsf{N} \cap R) = \mathsf{N} \cap R$$

13. This holds by replacing $\subseteq$ with $=$ in the preceding two calculations.
14. This follows from the preceding claim and the ninth claim. □

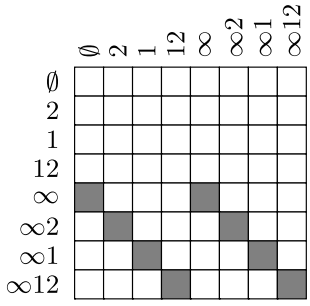
The computational interpretation of L is as follows. In each state, the best choice for the angel is the set $\{\infty\}$, whence the demon will choose ∞; thus ∞ is the unique outcome. For every multirelation R such that $\mathsf{L} \subseteq R$, the angel can force this outcome by choosing the set $\{\infty\}$. For every multirelation R such that $R \subseteq \mathsf{L}$, the demon can force this outcome since any set that can be chosen by the angel contains ∞.

4.3. Adding the special state

In Section 5.3 we will use the relation $\mathsf{K} : 2^A \leftrightarrow 2^A$ whose converse adds ∞ to a set. It is given by

$$\mathsf{K} = \big(\mathsf{E} \div (\mathsf{E} \cup \mathsf{N})\big)$$

A calculation shows $\mathsf{K}_{XY} \Leftrightarrow X = Y \cup \{\infty\}$. For $A = \{\infty, 1, 2\}$ its matrix is



The following result gives properties of K.

Theorem 10.

1. K *is bijective.*
2. $\mathsf{K} \subseteq \mathsf{L}^\smile\mathsf{T}$.
3. $\mathsf{K}^\smile \subseteq \mathsf{S}$ *and* $\mathsf{I} \subseteq \mathsf{SK}$ *and* K *is antisymmetric.*
4. $\mathsf{L}^\smile\mathsf{T} \cap \mathsf{I} \subseteq \mathsf{K}$ *and* $\mathsf{L}^\smile\mathsf{T} \cap \mathsf{I} \subseteq \mathsf{K}^\smile$.
5. $\mathsf{KK} = \mathsf{K}$ *and* $\mathsf{KK}^\smile \subseteq \mathsf{K} \subseteq \mathsf{K}^\smile\mathsf{K}$ *and* $\mathsf{KK}^\smile \subseteq \mathsf{K}^\smile \subseteq \mathsf{K}^\smile\mathsf{K}$.
6. $R\mathsf{K}^\smile = R \cap \mathsf{L}$ *if* R *is up-closed.*

Proof.

1. This follows immediately from (17).
2. This is implied by the following calculation, which uses Theorems 8.2 and 9.2:

$$\mathsf{K} = \big(\mathsf{E} \div (\mathsf{E} \cup \mathsf{N})\big) \subseteq \mathsf{E}^\smile / (\mathsf{E} \cup \mathsf{N})^\smile = \overline{\bar{\mathsf{E}}^\smile (\mathsf{E} \cup \mathsf{N})} = \overline{\bar{\mathsf{E}}^\smile \mathsf{E} \cup \bar{\mathsf{E}}^\smile \mathsf{N}} = \overline{\bar{\mathsf{E}}^\smile \mathsf{E}} \cap \overline{\bar{\mathsf{E}}^\smile \mathsf{N}} = \overline{\mathsf{E}^\smile \bar{\mathsf{E}}}^\smile \cap \bar{\bar{\mathsf{E}}}^\smile \mathsf{N}$$

$$= (\mathsf{E} \backslash \mathsf{E})^\smile \cap \mathsf{E}^\smile \mathsf{N} = \mathsf{S}^\smile \cap \mathsf{L}^\smile\mathsf{T}$$

3. The preceding calculation also implies $\mathsf{K}^\smile \subseteq \mathsf{S}$, whence $\mathsf{I} \subseteq \mathsf{SK}$ using the first claim and $\mathsf{K} \cap \mathsf{K}^\smile \subseteq \mathsf{S}^\smile \cap \mathsf{S} \subseteq \mathsf{I}$ using (19).
4. The above calculation also yields $\mathsf{L}^\smile\mathsf{T} \cap \mathsf{S}^\smile = \mathsf{E}^\smile / (\mathsf{E} \cup \mathsf{N})^\smile$. Moreover $\mathsf{I} = \mathsf{I}^\smile \subseteq \mathsf{S}^\smile$ using (19), and $\mathsf{E} \subseteq \mathsf{E} \cup \mathsf{N}$ implies $\mathsf{I} \subseteq \mathsf{E} \backslash (\mathsf{E} \cup \mathsf{N})$ using (4). Hence

$$\mathsf{L}^\smile\mathsf{T} \cap \mathsf{I} \subseteq \mathsf{L}^\smile\mathsf{T} \cap \mathsf{S}^\smile \cap \big(\mathsf{E} \backslash (\mathsf{E} \cup \mathsf{N})\big) = \big(\mathsf{E}^\smile / (\mathsf{E} \cup \mathsf{N})^\smile\big) \cap \big(\mathsf{E} \backslash (\mathsf{E} \cup \mathsf{N})\big) = \big(\mathsf{E} \div (\mathsf{E} \cup \mathsf{N})\big) = \mathsf{K}$$

   Thus $\mathsf{L}^\smile\mathsf{T} \cap \mathsf{I} = (\mathsf{L}^\smile\mathsf{T} \cap \mathsf{I})^\smile \subseteq \mathsf{K}^\smile$.
5. $(\mathsf{E} \cup \mathsf{N})\mathsf{K} = \mathsf{EK} \cup \mathsf{NK} = \mathsf{E}(\mathsf{E} \div (\mathsf{E} \cup \mathsf{N})) \cup \mathsf{NK} = \mathsf{E} \cup \mathsf{N} \cup \mathsf{NK} \subseteq \mathsf{E} \cup \mathsf{NT} = \mathsf{E} \cup \mathsf{N}$ using (18) and that N is a vector. Hence $\mathsf{K} \subseteq (\mathsf{E} \cup \mathsf{N}) \backslash (\mathsf{E} \cup \mathsf{N})$ using (4). But also

$$\mathsf{K} = \big(\mathsf{E} \div (\mathsf{E} \cup \mathsf{N})\big) \subseteq \mathsf{E}^\smile / (\mathsf{E} \cup \mathsf{N})^\smile \subseteq (\mathsf{E} \cup \mathsf{N})^\smile / (\mathsf{E} \cup \mathsf{N})^\smile$$

   using (2). Together $\mathsf{K} \subseteq ((\mathsf{E} \cup \mathsf{N}) \div (\mathsf{E} \cup \mathsf{N}))$. Hence

$$\mathsf{K}^\smile \subseteq \big((\mathsf{E} \cup \mathsf{N}) \div (\mathsf{E} \cup \mathsf{N})\big) = \big((\mathsf{E} \cup \mathsf{N}) \div \mathsf{E}\big)\big(\mathsf{E} \div (\mathsf{E} \cup \mathsf{N})\big) = \mathsf{K}^\smile\mathsf{K}$$

   using (12) and Theorem 1.3. This implies $\mathsf{K} \subseteq \mathsf{K}^\smile\mathsf{K}$ and $\mathsf{K}^\smile\mathsf{K}^\smile \subseteq \mathsf{K}^\smile$ using the first claim. Therefore $\mathsf{KK}^\smile \subseteq \mathsf{K}^\smile$ again using the first claim and also $\mathsf{KK} \subseteq \mathsf{K}$. Thus $\mathsf{KK}^\smile \subseteq \mathsf{K}$, which implies $\mathsf{K} \subseteq \mathsf{KK}$ using the first claim again.
6. $R\mathsf{K}^\smile \subseteq R\mathsf{S} = R$ using the third claim and that R is up-closed. Moreover $R\mathsf{K}^\smile \subseteq R\mathsf{TL} \subseteq \mathsf{TL} = \mathsf{L}$ using the second claim and Theorem 9.5. Also $R \cap \mathsf{L} = (R \cap \mathsf{TL})\mathsf{I} = R(\mathsf{L}^\smile\mathsf{T} \cap \mathsf{I}) \subseteq R\mathsf{K}^\smile$ using Theorem 9.5 and the fourth claim. □

5. Approximation

The solution of the recursive specification $R = f(R)$ is the least fixpoint of the function f in a suitable approximation order. Instantiating f with the identity function yields the endless loop, which is represented by the multirelation L. Hence L should be the least element of the approximation order. Because L is neither O nor T, the subset and the superset order cannot be used for approximation. In this section, we use RelView to derive a suitable approximation order.

5.1. Deriving an approximation order

The general idea is to look for a relation that satisfies a few properties expected of approximation, but to not impose unnecessary constraints. Let $\sqsubseteq$ denote the approximation relation on up-closed multirelations; reasonable properties are:

- $\sqsubseteq$ is a partial order,
- L is the $\sqsubseteq$-least element,
- $\cup$, $\cap$, ; and $\cdot^d$ are $\sqsubseteq$-isotone.

We implement a RelView program that constructs the $\subseteq$-least relation $\sqsubseteq$ that satisfies these properties. In the following we describe the relational constructions; the implementation is given in Appendix A.

Because $\sqsubseteq$ applies only to up-closed multirelations, we first need to enumerate these. The following result gives the set of up-closed multirelations as a vector using the parallel product of relations.

Theorem 11. *The vector* $\overline{(\mathsf{E}^\smile(\mathsf{I}\|\mathsf{S})\cap\bar{\mathsf{E}}^\smile)\mathsf{T}} : 2^{A\times 2^B} \leftrightarrow \{\star\}$ *represents the set of up-closed multirelations of type* $A \leftrightarrow 2^B$.

Proof. Let $p_1 : A \times 2^B \leftrightarrow A$ and $p_2 : A \times 2^B \leftrightarrow 2^B$ be projections. Then

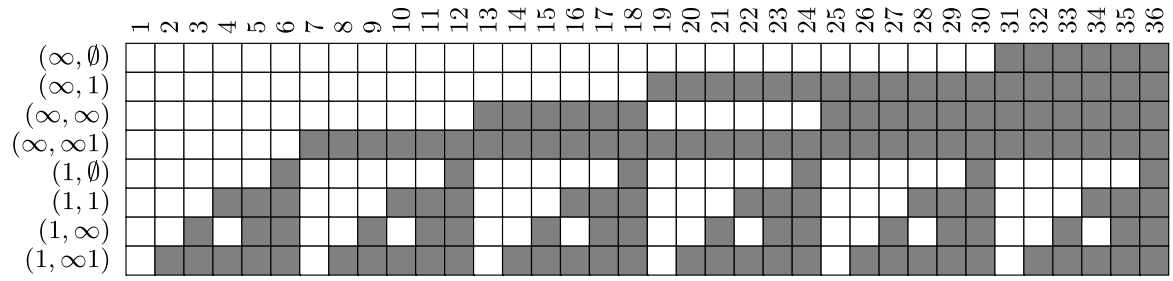
$$
\begin{array}{ll}
 & R : A \leftrightarrow 2^B \text{ is up-closed} \\
\Leftrightarrow & \forall x, Y, Z : R_{xY} \wedge Y \subseteq Z \Rightarrow R_{xZ} \\
\Leftrightarrow & \forall x, Y, Z : \mathsf{E}_{(x,Y)R} \wedge \mathsf{S}_{YZ} \Rightarrow \mathsf{E}_{(x,Z)R} \\
\Leftrightarrow & \forall x, Y, Z : \mathsf{E}^\smile{}_{R(x,Y)} \wedge \mathsf{S}_{YZ} \Rightarrow \mathsf{E}^\smile{}_{R(x,Z)} \\
\Leftrightarrow & \forall x, Y, Z : (\exists u : \mathsf{E}^\smile{}_{Ru} \wedge p_{1ux} \wedge p_{2uY}) \wedge \mathsf{S}_{YZ} \Rightarrow \mathsf{E}^\smile{}_{R(x,Z)} \\
\Leftrightarrow & \forall x, Z : \neg\exists Y : (\exists u : \mathsf{E}^\smile{}_{Ru} \wedge p_{1ux} \wedge p_{2uY}) \wedge \mathsf{S}_{YZ} \wedge \neg\mathsf{E}^\smile{}_{R(x,Z)} \\
\Leftrightarrow & \forall x, Z : \neg\exists u : \mathsf{E}^\smile{}_{Ru} \wedge p_{1ux} \wedge (\exists Y : p_{2uY} \wedge \mathsf{S}_{YZ}) \wedge \bar{\mathsf{E}}^\smile{}_{R(x,Z)} \\
\Leftrightarrow & \forall x, Z : \neg\exists u : \mathsf{E}^\smile{}_{Ru} \wedge p_{1ux} \wedge (p_2\mathsf{S})_{uZ} \wedge (\exists v : \bar{\mathsf{E}}^\smile{}_{Rv} \wedge p_{1vx} \wedge p_{2vZ}) \\
\Leftrightarrow & \neg\exists u, v, x, Z : \mathsf{E}^\smile{}_{Ru} \wedge p_{1ux} \wedge (p_2\mathsf{S})_{uZ} \wedge \bar{\mathsf{E}}^\smile{}_{Rv} \wedge p_{1vx} \wedge p_{2vZ} \\
\Leftrightarrow & \neg\exists u, v : \mathsf{E}^\smile{}_{Ru} \wedge \bar{\mathsf{E}}^\smile{}_{Rv} \wedge (\exists x : p_{1ux} \wedge p_1{}^\smile{}_{xv}) \wedge \big(\exists Z : (p_2\mathsf{S})_{uZ} \wedge p_2{}^\smile{}_{Zv}\big) \\
\Leftrightarrow & \neg\exists u, v : \mathsf{E}^\smile{}_{Ru} \wedge \bar{\mathsf{E}}^\smile{}_{Rv} \wedge (p_1 p_1{}^\smile)_{uv} \wedge (p_2\mathsf{S}p_2{}^\smile)_{uv} \\
\Leftrightarrow & \neg\exists v : \big(\exists u : \mathsf{E}^\smile{}_{Ru} \wedge (p_1\mathsf{I}p_1{}^\smile \cap p_2\mathsf{S}p_2{}^\smile)_{uv}\big) \wedge \bar{\mathsf{E}}^\smile{}_{Rv} \\
\Leftrightarrow & \neg\exists v : \big(\mathsf{E}^\smile(\mathsf{I}\|\mathsf{S})\big)_{Rv} \wedge \bar{\mathsf{E}}^\smile{}_{Rv} \\
\Leftrightarrow & \neg\exists v : \big(\mathsf{E}^\smile(\mathsf{I}\|\mathsf{S})\cap\bar{\mathsf{E}}^\smile\big)_{Rv} \\
\Leftrightarrow & \neg\overline{\big((\mathsf{E}^\smile(\mathsf{I}\|\mathsf{S})\cap\bar{\mathsf{E}}^\smile)\mathsf{T}\big)}_{R\star} \\
\Leftrightarrow & \overline{(\mathsf{E}^\smile(\mathsf{I}\|\mathsf{S})\cap\bar{\mathsf{E}}^\smile)\mathsf{T}}_{R\star}
\end{array}
$$

The ranges of quantified variables are $x \in A$ and $Y, Z \subseteq B$ and $u, v \in A \times 2^B$. □

A construction described in [2] transforms this vector to a relation that contains one column for each up-closed multirelation:

$$\mathsf{E}\,\mathrm{inj}\Big(\overline{\big((\mathsf{E}^\smile(\mathsf{I}\|\mathsf{S})\cap\bar{\mathsf{E}}^\smile)\mathsf{T}\big)}\Big)^\smile$$

For every vector $V \neq \mathsf{O}$, $\mathrm{inj}(V)$ is an injective mapping such that $\mathrm{inj}(V)^\smile\mathsf{T} = V$. Each column of the resulting relation is the row-wise representation of the entries of an up-closed multirelation. For example, the following relation is obtained for $A = B = \{\infty, 1\}$:



Columns 1, 15, 16, 31, 36 contain the multirelations O, L, E, N, T, respectively.

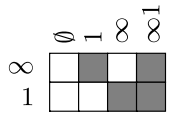
The range of this relation is the set of up-closed multirelations, and hence the domain of the approximation relation. We apply the following algorithm to compute the approximation relation based on the above requirements:

$\sqsubseteq := \{(\mathsf{L}, R) \mid R \text{ up-closed}\}^*$
while $\sqsubseteq$ changes do
  for each $(Q, R) \in \sqsubseteq$ do
    for each up-closed P do
      $\sqsubseteq := \sqsubseteq \cup \{(P \cup Q, P \cup R), (P \cap Q, P \cap R), (Q\ ;\ P, R\ ;\ P), (P\ ;\ Q, P\ ;\ R), (Q^d, R^d)\}$
  $\sqsubseteq := \sqsubseteq^*$

The initialisation makes L the $\sqsubseteq$-least element. The update in the for-loops ensures that $\cup$, $\cap$, ; and ^{d} are $\sqsubseteq$-isotone. Forming the reflexive-transitive closure * ensures that $\sqsubseteq$ is a preorder. The only requirement not guaranteed is that $\sqsubseteq$ is antisymmetric.

The RelView implementation of this algorithm uses library functions to convert between multirelations of type $A \leftrightarrow 2^B$ and vectors of type $A \times 2^B \leftrightarrow \{\star\}$. The vector representation is used to index a multirelation in the above enumeration with the help of a symmetric quotient. The index is used to extract entries from $\sqsubseteq$ and to add new entries to $\sqsubseteq$. The multirelation representation is used to calculate the multirelations by which $\sqsubseteq$ is updated.

Applying the algorithm for $A = B = \{\infty, 1\}$ gives $\sqsubseteq = \mathsf{T}$, which is not antisymmetric. It is therefore necessary to restrict the set of up-closed multirelations. Comparing with relational computation models such as those of [17,8] we find that some up-closed multirelations represent non-strict computations. For example, column 21 of the above enumeration contains the multirelation



In state ∞ the outcome is 1; in state 1 the outcome is ∞. Thus the computation delivers an outcome even if the preceding computation fails to terminate.

5.2. Strict multirelations

A multirelation R is *strict* if $\mathsf{L} = \mathsf{L}\ ;\ R$. Letting the endless loop be a left annihilator of sequential composition characterises strictness also in various relational computation models [12]. For multirelations, strictness is equivalent to each of $\mathsf{N} \cap \mathsf{E} = \mathsf{N} \cap R$ and $\mathsf{N} \cap \mathsf{L} = \mathsf{N} \cap R$ by Theorems 9.13 and 9.9.

The following result gives the set of strict multirelations as a vector using the tupling of relations.

Theorem 12. *The vector* $((\mathsf{E}^\smile/[\mathsf{I}, \mathsf{E}]) \cap (\bar{\mathsf{E}}^\smile/[\mathsf{I}, \bar{\mathsf{E}}]))\mathsf{NT} : 2^{A \times 2^A} \leftrightarrow \{\star\}$ *represents the set of strict multirelations of type* $A \leftrightarrow 2^A$.

Proof.

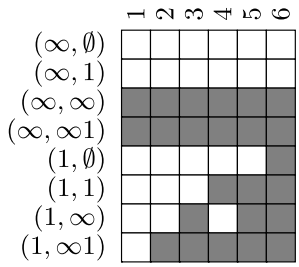
$$
\begin{aligned}
& R : A \leftrightarrow 2^A \quad \text{is strict} \\
\Leftrightarrow\ & \mathsf{N} \cap \mathsf{E} \subseteq R \wedge \mathsf{N} \cap R \subseteq \mathsf{E} \\
\Leftrightarrow\ & \big(\forall x, Y : (\mathsf{N} \cap \mathsf{E})_{xY} \Rightarrow R_{xY}\big) \wedge \big(\forall x, Y : (\mathsf{N} \cap R)_{xY} \Rightarrow \mathsf{E}_{xY}\big) \\
\Leftrightarrow\ & (\forall x, Y : \mathsf{N}_{xY} \wedge \mathsf{E}_{xY} \Rightarrow R_{xY}) \wedge (\forall x, Y : \mathsf{N}_{xY} \wedge R_{xY} \Rightarrow \mathsf{E}_{xY}) \\
\Leftrightarrow\ & \big(\forall x, Y : x = \infty \wedge \mathsf{E}_{xY} \Rightarrow (x, Y) \in R\big) \wedge \big(\forall x, Y : x = \infty \wedge (x, Y) \in R \Rightarrow \mathsf{E}_{xY}\big) \\
\Leftrightarrow\ & (\forall x, Y : x = \infty \wedge \mathsf{E}_{\infty Y} \Rightarrow \mathsf{E}_{(x,Y)R}) \wedge \big(\forall x, Y : x = \infty \wedge \neg \mathsf{E}_{\infty Y} \Rightarrow (x, Y) \notin R\big) \\
\Leftrightarrow\ & (\forall x, Y : \mathsf{I}_{\infty x} \wedge \mathsf{E}_{\infty Y} \Rightarrow \mathsf{E}_{(x,Y)R}) \wedge \big(\forall x, Y : \mathsf{I}_{\infty x} \wedge \bar{\mathsf{E}}_{\infty Y} \Rightarrow \bar{\mathsf{E}}_{(x,Y)R}\big) \\
\Leftrightarrow\ & \big(\forall x, Y : [\mathsf{I}, \mathsf{E}]_{\infty(x,Y)} \Rightarrow \mathsf{E}^\smile{}_{R(x,Y)}\big) \wedge \big(\forall x, Y : [\mathsf{I}, \bar{\mathsf{E}}]_{\infty(x,Y)} \Rightarrow \bar{\mathsf{E}}^\smile{}_{R(x,Y)}\big) \\
\Leftrightarrow\ & \big(\mathsf{E}^\smile/[\mathsf{I}, \mathsf{E}]\big)_{R\infty} \wedge \big(\bar{\mathsf{E}}^\smile/[\mathsf{I}, \bar{\mathsf{E}}]\big)_{R\infty} \\
\Leftrightarrow\ & \exists x : \big(\mathsf{E}^\smile/[\mathsf{I}, \mathsf{E}]\big)_{Rx} \wedge \big(\bar{\mathsf{E}}^\smile/[\mathsf{I}, \bar{\mathsf{E}}]\big)_{Rx} \wedge x = \infty \\
\Leftrightarrow\ & \exists x, Y : \big((\mathsf{E}^\smile/[\mathsf{I}, \mathsf{E}]) \cap (\bar{\mathsf{E}}^\smile/[\mathsf{I}, \bar{\mathsf{E}}])\big)_{Rx} \wedge \mathsf{N}_{xY} \wedge \mathsf{T}_{Y\star} \\
\Leftrightarrow\ & \big(((\mathsf{E}^\smile/[\mathsf{I}, \mathsf{E}]) \cap (\bar{\mathsf{E}}^\smile/[\mathsf{I}, \bar{\mathsf{E}}]))\mathsf{NT}\big)_{R\star}
\end{aligned}
$$

The ranges of quantified variables are $x \in A$ and $Y \subseteq A$. □

We apply the above construction again to enumerate all strict, up-closed multirelations as the columns of a relation:

$$\mathsf{E}\ \mathrm{inj}\big(\overline{(\mathsf{E}^\smile(\mathsf{I}\|\mathsf{S}) \cap \bar{\mathsf{E}}^\smile)\mathsf{T}} \cap \big((\mathsf{E}^\smile/[\mathsf{I}, \mathsf{E}]) \cap (\bar{\mathsf{E}}^\smile/[\mathsf{I}, \bar{\mathsf{E}}])\big)\mathsf{NT}\big)^\smile$$

For example, the following relation is obtained for $A = B = \{\infty, 1\}$:



Columns 1, 3, 4, 6 contain the multirelations $\mathsf{N}\cap\mathsf{L}$, L, E, $\overline{\mathsf{N}}\cup\mathsf{L}$, respectively. The matrix comprises columns 13–18 of the matrix in Section 5.1.

The following result shows that many multirelational operations preserve strictness.

Theorem 13.

1. L, E, $\mathsf{N}\cap\mathsf{L}$ *and* $\overline{\mathsf{N}}\cup\mathsf{L}$ *are strict.*
2. $Q\ ;\ R$ *is strict if* Q *and* R *are strict and* Q *is up-closed.*
3. R^d *is strict if* R *is strict.*
4. $\bigcup_{i\in I} R_i$ *and* $\bigcap_{i\in I} R_i$ *are strict if* R_i *is strict for each* $i\in I$ *and* $I\neq\emptyset$.

Proof.

1. These claims follow from Theorems 9.13 and 9.9 and $\mathsf{N}\cap(\mathsf{N}\cap\mathsf{L})=\mathsf{N}\cap\mathsf{L}$ and $\mathsf{N}\cap(\overline{\mathsf{N}}\cup\mathsf{L})=\mathsf{N}\cap\mathsf{L}$.
2. $\mathsf{L}\ ;\ (Q\ ;\ R)=(\mathsf{L}\ ;\ Q)\ ;\ R=\mathsf{L}\ ;\ R=\mathsf{L}$ using Theorem 7.2 as Q is up-closed and that Q and R are strict.
3. $\mathsf{L}\ ;\ R^d=\mathsf{L}^d\ ;\ R^d=(\mathsf{L}\ ;\ R)^d=\mathsf{L}^d=\mathsf{L}$ using Theorems 9.10, 9.4 and 7.3 and that R is strict.
4. $\mathsf{N}\cap\bigcup_{i\in I} R_i=\bigcup_{i\in I}(\mathsf{N}\cap R_i)=\bigcup_{i\in I}(\mathsf{N}\cap\mathsf{L})=\mathsf{N}\cap\mathsf{L}$ using that R_i is strict. The assumption $I\neq\emptyset$ is needed for the last equality. Moreover $\mathsf{N}\cap\bigcap_{i\in I} R_i=\bigcap_{i\in I}(\mathsf{N}\cap R_i)=\bigcap_{i\in I}(\mathsf{N}\cap\mathsf{L})=\mathsf{N}\cap\mathsf{L}$ using that $\cap$ distributes over non-empty intersections and that R_i is strict. □

A strict multirelation is obtained from a multirelation R by applying the function $s(R)=(\mathsf{N}\cap\mathsf{L})\cup(\overline{\mathsf{N}}\cap R)$. The following result shows properties of s.

Theorem 14.

1. *Both the image of* s *and the fixpoints of* s *are precisely the strict multirelations.*
2. s *is* $\subseteq$*-isotone and idempotent.*
3. $s(\mathsf{O})=s(\mathsf{N})=\mathsf{N}\cap\mathsf{L}$ *and* $s(\mathsf{T})=\overline{\mathsf{N}}\cup\mathsf{L}$ *and* $s(\mathsf{L})=\mathsf{L}$ *and* $s(\mathsf{E})=\mathsf{E}$.

Proof.

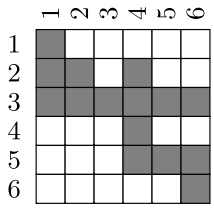
1. $\mathsf{N}\cap s(R)=\mathsf{N}\cap((\mathsf{N}\cap\mathsf{L})\cup(\overline{\mathsf{N}}\cap R))=(\mathsf{N}\cap\mathsf{N}\cap\mathsf{L})\cup(\mathsf{N}\cap\overline{\mathsf{N}}\cap R)=(\mathsf{N}\cap\mathsf{L})\cup\mathsf{O}=\mathsf{N}\cap\mathsf{L}$ shows that $s(R)$ is strict. A strict multirelation R is a fixpoint of s since

$$s(R)=(\mathsf{N}\cap\mathsf{L})\cup(\overline{\mathsf{N}}\cap R)=(\mathsf{N}\cap R)\cup(\overline{\mathsf{N}}\cap R)=(\mathsf{N}\cup\overline{\mathsf{N}})\cap R=\mathsf{T}\cap R=R$$

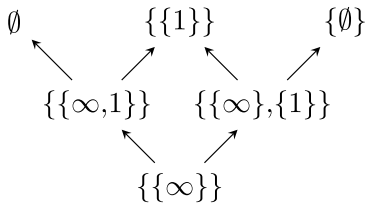
   A fixpoint of s is clearly in the image of s.
2. s is $\subseteq$-isotone since $\cap$ and $\cup$ are $\subseteq$-isotone. By the preceding claim, $s(s(R))=s(R)$ since $s(R)$ is strict.
3. These claims follow by simple calculations using the definition of s and Theorem 9.9. □

5.3. Deriving an approximation order (continued)

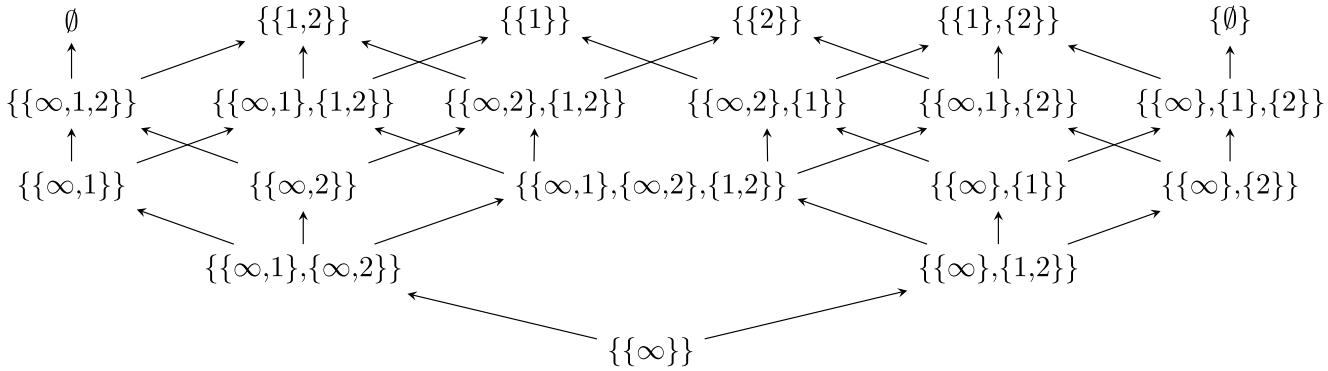
We apply the algorithm of Section 5.1 for $A=B=\{\infty,1\}$, but this time restricted to the strict, up-closed multirelations. The result is



Rows and columns are numbered according to the enumeration produced in Section 5.2. RelView easily confirms that this relation is antisymmetric, hence a partial order. Its Hasse-diagram computed by RelView is shown below. Every node is labelled with the image of state 1 in the corresponding multirelation, that is, with the outcome for input 1 to be interpreted as in Section 3.4. In each set, only the $\subseteq$-minimal subsets are given; their upward closure is not included:



Applying the algorithm of Section 5.1 for $A = B = \{\infty, 1, 2\}$ also yields a partial order. Its Hasse-diagram is:



Again the upward closure is omitted: for example, $\{\emptyset\}$ represents the set 2^A and $\{\{\infty, 1\}, \{2\}\}$ represents the set $\{\{\infty, 1\}, \{\infty, 2\}, \{\infty, 1, 2\}, \{2\}, \{1, 2\}\}$.

Denote the above order by $\sqsubseteq_2$. We observe the following about two sets related by this order $Xs \sqsubseteq_2 Ys$:

- If Xs contains a set X that does not contain ∞, then also Ys contains X.
- If Ys contains a set Y, then Xs contains Y or a set X such that $\infty \in X$ and Y contains all elements of X except perhaps ∞.

The underlying interpretation is that ∞ in a set in Xs may be replaced by any number of elements when going from Xs to Ys; a set in Xs that contains ∞ may be omitted in Ys and a set in Xs that does not contain ∞ must remain in Ys. Our observations hold for $A = \{\infty, 1\}$ and for $A = \{\infty, 1, 2\}$, but we expect that they hold for arbitrary A. This generalises the Egli–Milner order to sets of sets.

The above observations are formalised by

$$\begin{aligned} Xs \sqsubseteq_2 Ys \quad \Leftrightarrow \quad & (\forall X \in Xs : \infty \notin X \Rightarrow X \in Ys) \wedge \\ & \big(\forall Y \in Ys : \exists X \in Xs : X = Y \vee \big(\infty \in X \wedge X \subseteq Y \cup \{\infty\}\big)\big) \end{aligned}$$

The approximation order on strict, up-closed multirelations is $\sqsubseteq_2$ applied pointwise to each state:

$$Q \sqsubseteq R \quad \Leftrightarrow \quad \forall x \in A : Q(x) \sqsubseteq_2 R(x)$$

Here $Q(x) = \{Y \mid Q_{xY}\}$ is the image of x under Q and similarly $R(x)$ is the image of x under R. The following result expresses $\sqsubseteq$ in terms of relations.

Theorem 15. *$Q \sqsubseteq R \Leftrightarrow R \cap \mathsf{L} \subseteq Q \subseteq R \cup \mathsf{L}$ if Q and R are up-closed.*

Proof. $Q \sqsubseteq R$ is equivalent to

$$\begin{aligned} \forall x \in A : & \big(\forall X \in Q(x) : \infty \notin X \Rightarrow X \in R(x)\big) \wedge \\ & \big(\forall Y \in R(x) : \exists X \in Q(x) : X = Y \vee \big(\infty \in X \wedge X \subseteq Y \cup \{\infty\}\big)\big) \end{aligned}$$

The first term is equivalently transformed by

$$\begin{aligned}
& \forall x : \forall X \in Q\,(x) : \infty \notin X \Rightarrow X \in R(x) \\
\Leftrightarrow\ & \forall x, X : Q_{xX} \Rightarrow (\infty \notin X \Rightarrow R_{xX}) \\
\Leftrightarrow\ & \forall x, X : Q_{xX} \Rightarrow (\infty \in X \vee R_{xX}) \\
\Leftrightarrow\ & \forall x, X : Q_{xX} \Rightarrow (\mathsf{L}_{xX} \vee R_{xX}) \\
\Leftrightarrow\ & \forall x, X : Q_{xX} \Rightarrow (R \cup \mathsf{L})_{xX} \\
\Leftrightarrow\ & Q \subseteq R \cup \mathsf{L}
\end{aligned}$$

The ranges of quantified variables are $x \in A$ and $X \subseteq A$. The second term is equivalently transformed by

$$\begin{aligned}
& \forall x : \forall Y \in R(x) : \exists X \in Q\,(x) : X = Y \vee \big(\infty \in X \wedge X \subseteq Y \cup \{\infty\}\big) \\
\Leftrightarrow\ & \forall x, Y : R_{xY} \Rightarrow \exists X : Q_{xX} \wedge \big(X = Y \vee \big(\infty \in X \wedge X \subseteq Y \cup \{\infty\}\big)\big) \\
\Leftrightarrow\ & \forall x, Y : R_{xY} \Rightarrow \exists X : Q_{xX} \wedge \big(\mathsf{I}_{XY} \vee \big(\infty \in X \wedge \exists Z : X \subseteq Z \wedge Z = Y \cup \{\infty\}\big)\big) \\
\Leftrightarrow\ & \forall x, Y : R_{xY} \Rightarrow \exists X : Q_{xX} \wedge \big(\mathsf{I}_{XY} \vee \big((\exists y : \mathsf{L}_{yX}) \wedge \exists Z : \mathsf{S}_{XZ} \wedge \mathsf{K}_{ZY}\big)\big) \\
\Leftrightarrow\ & \forall x, Y : R_{xY} \Rightarrow \exists X : Q_{xX} \wedge \big(\mathsf{I}_{XY} \vee \big((\mathsf{TL})_{YX} \wedge (\mathsf{SK})_{XY}\big)\big) \\
\Leftrightarrow\ & \forall x, Y : R_{xY} \Rightarrow \exists X : Q_{xX} \wedge \big(\mathsf{I}_{XY} \vee \big((\mathsf{TL})^{\smile} \cap \mathsf{SK}\big)_{XY}\big) \\
\Leftrightarrow\ & \forall x, Y : R_{xY} \Rightarrow \exists X : Q_{xX} \wedge \big(\mathsf{I} \cup \big((\mathsf{TL})^{\smile} \cap \mathsf{SK}\big)\big)_{XY} \\
\Leftrightarrow\ & \forall x, Y : R_{xY} \Rightarrow \big(Q\,\big(\mathsf{I} \cup \big((\mathsf{TL})^{\smile} \cap \mathsf{SK}\big)\big)\big)_{xY} \\
\Leftrightarrow\ & R \subseteq Q\,\big(\mathsf{I} \cup \big((\mathsf{TL})^{\smile} \cap \mathsf{SK}\big)\big)
\end{aligned}$$

The ranges of quantified variables are $x, y \in A$ and $X, Y \subseteq A$. The result simplifies to

$$\begin{aligned}
& R \subseteq Q\,\big(\mathsf{I} \cup \big((\mathsf{TL})^{\smile} \cap \mathsf{SK}\big)\big) = Q\,\mathsf{I} \cup Q\,(\mathsf{L}^{\smile}\mathsf{T} \cap \mathsf{SK}) = Q \cup (Q \cap \mathsf{TL})\mathsf{SK} = Q \cup (Q \cap \mathsf{TL})\mathsf{K} \\
& = Q \cup Q\,(\mathsf{L}^{\smile}\mathsf{T} \cap \mathsf{K}) = Q \cup Q\,\mathsf{K} = Q\,\mathsf{I} \cup Q\,\mathsf{SK} = Q\,(\mathsf{I} \cup \mathsf{SK}) = Q\,\mathsf{SK} = Q\,\mathsf{K}
\end{aligned}$$

using that Q is up-closed and Theorems 9.4, 7.6, 7.7, 10.2 and 10.3. This is equivalent to $R\mathsf{K}^{\smile} \subseteq Q$ using Theorem 10.1, and thus to $R \cap \mathsf{L} \subseteq Q$ by Theorem 10.6 since R is up-closed. □

5.4. Median

Given the characterisation of Theorem 15, we find that $\sqsubseteq$ is the semilattice order induced by the median operation. By Theorems 7.7 and 13.4 the strict, up-closed multirelations are closed under $\cup$ and $\cap$ and thus form a distributive lattice. Hence we can look at the ternary median operation [9,6,5]:

$$(P, Q, R) = (P \cap Q) \cup (Q \cap R) \cup (R \cap P)$$

It is self-dual and a collection of its symmetries is given in [19]. Consider the instance

$$Q \sqcap R = (\mathsf{L}, Q, R) = (\mathsf{L} \cap Q) \cup (Q \cap R) \cup (R \cap \mathsf{L})$$

Our previous result [10, Proposition 20] specialises to multirelations as follows.

Theorem 16. *The strict, up-closed multirelations form a semilattice with meet $\sqcap$, semilattice order $\sqsubseteq$ and $\sqsubseteq$-least element L. The following hold*:

1. $Q \sqsubseteq R \Leftrightarrow Q = Q \sqcap R$.
2. *$\cap$, $\cup$ and $\sqcap$ distribute over each other.*
3. $Q \subseteq R \Leftrightarrow Q \cap \mathsf{L} \subseteq R \wedge Q \subseteq R \cup \mathsf{L}$.
4. $Q \sqcap R = (R \cap \mathsf{L}) \cup Q = R \cap (\mathsf{L} \cup Q)$ *if* $Q \subseteq R$.

Further results based on the median instantiate to multirelations. They include representations of $\sqsubseteq$-least (pre-)fixpoints in terms of $\subseteq$-least (pre-)fixpoints and $\subseteq$-greatest (post-)fixpoints. We present [10, Theorem 24] as an example.

Theorem 17. *Let f be a $\subseteq$- and $\sqsubseteq$-isotone function on strict, up-closed multirelations. Let μf be the $\subseteq$-least fixpoint of f. Let νf be the $\subseteq$-greatest fixpoint of f. Then the $\sqsubseteq$-least fixpoint ξf of f exists and satisfies $\xi f = \mu f \sqcap \nu f = (\nu f \cap \mathsf{L}) \cup \mu f$.*

We prove the following new result, which also holds in arbitrary pointed distributive lattices.

Theorem 18. *$\sqsubseteq$ is the $\subseteq$-least partial order with least element L and isotone operations $\cup$ and $\cap$.*

Proof. $\cup$ and $\cap$ are $\sqsubseteq$-isotone by the distributivity properties in Theorem 16. Let $\preccurlyeq$ be a partial order with least element L such that $\cup$ and $\cap$ are $\preccurlyeq$-isotone. Let $Q \sqsubseteq R$, whence $R \cap \mathsf{L} \subseteq Q \subseteq R \cup \mathsf{L}$. Hence $(R \cap \mathsf{L}) \cup Q = Q = Q \cap (R \cup \mathsf{L})$. First, $\mathsf{L} \preccurlyeq R$ implies $R \cup \mathsf{L} \preccurlyeq R \cup R = R$, and therefore $Q = Q \cap (R \cup \mathsf{L}) \preccurlyeq Q \cap R$. Second, $\mathsf{L} \preccurlyeq R$ implies $R \cap \mathsf{L} \preccurlyeq R \cap R = R$, hence $Q = Q \cup (R \cap \mathsf{L}) \preccurlyeq Q \cup R$, and therefore $Q \cap R \preccurlyeq (Q \cup R) \cap R = R$. Together, $Q \preccurlyeq Q \cap R \preccurlyeq R$. □

The following result shows that other operations on multirelations are $\sqsubseteq$-isotone, too.

Theorem 19. $^{.d}$ *and* ; *are* $\sqsubseteq$*-isotone on the strict, up-closed multirelations.*

Proof. Let $Q \sqsubseteq R$, whence $R \cap \mathsf{L} \subseteq Q \subseteq R \cup \mathsf{L}$. First,

$$R^d \cap \mathsf{L} = R^d \cap \mathsf{L}^d = (R \cup \mathsf{L})^d \subseteq Q^d \subseteq (R \cap \mathsf{L})^d = R^d \cup \mathsf{L}^d = R^d \cup \mathsf{L}$$

using Theorems 9.10, 5.7, 5.1 and 5.8. Hence $Q^d \sqsubseteq R^d$. Second,

$$(R \,;\, P) \cap \mathsf{L} = (R \,;\, P) \cap (\mathsf{L} \,;\, P) = (R \cap \mathsf{L}) \,;\, P \subseteq Q \,;\, P \subseteq (R \cup \mathsf{L}) \,;\, P = (R \,;\, P) \cup (\mathsf{L} \,;\, P) = (R \,;\, P) \cup \mathsf{L}$$

using that P is strict, that R is up-closed and Theorems 9.4, 7.8, 3.1 and 3.7. Hence $Q \,;\, P \sqsubseteq R \,;\, P$. Third,

$$\begin{aligned}(P \,;\, R) \cap \mathsf{L} &= P(\mathsf{E}\backslash R) \cap \mathsf{TL} = P\big((\mathsf{E}\backslash R) \cap \mathsf{TL}\big) \subseteq P\big((\mathsf{E}\backslash R) \cap (\mathsf{E}\backslash \mathsf{L})\big) = P\big(\mathsf{E}\backslash (R \cap \mathsf{L})\big) \\ &= P \,;\, (R \cap \mathsf{L}) \subseteq P \,;\, Q \subseteq P \,;\, (R \cup \mathsf{L}) = P\big(\mathsf{E}\backslash (R \cup \mathsf{L})\big) = P\overline{\mathsf{E}^\smile \overline{R \cup \mathsf{L}}} = P\overline{\mathsf{E}^\smile (\overline{R} \cap \overline{\mathsf{L}})} \\ &= P\overline{\mathsf{E}^\smile (\overline{R} \cap \mathsf{T}\overline{\mathsf{L}})} = P\overline{\mathsf{E}^\smile \overline{R}} \cap \mathsf{T}\overline{\mathsf{L}} = P(\overline{\mathsf{E}^\smile \overline{R}} \cup \mathsf{T}\overline{\mathsf{L}}) = P(\mathsf{E}\backslash R) \cup P\mathsf{T}\overline{\mathsf{L}} \subseteq (P \,;\, R) \cup \mathsf{L}\end{aligned}$$

using Theorems 9.5 and 9.6, (6), Theorems 3.1 and 9.5, and that $P\mathsf{T}\overline{\mathsf{L}} \subseteq \mathsf{L}$ by a Schröder equivalence. Hence $P \,;\, Q \sqsubseteq P \,;\, R$. □

6. Conclusion

This paper shows that relations are useful for investigating multirelations and extending the underlying state space to represent infinite computations. Our development essentially benefited from exploring the approximation order with RelView, which led to the compact algebraic characterisation. In future work we will apply the given method to derive approximation orders for further computation models.

Acknowledgements

I thank the anonymous referees for their helpful comments. I thank Gunther Schmidt for welcoming me at RelMiCS 2005 in a friendly and encouraging way even though I was uninitiated to Schröder equivalences and the Dedekind formula.

Appendix A. RelView implementation

```
sub(R)    = epsi(R) \ epsi(R).           { subset relation of size 2^dom(R) }
dual(R)   = -R * syq(epsi(R),-epsi(R)).  { dual of multirelation R }
comp(R,S) = R * (epsi(R) \ S).           { composition of multirelations R and S }

Upclosed(R)                              { enumerate all up-closed multirelations of size R }
  DECL Prod = PROD(R*R^,R^*R)
       P,E,V
  BEG
    P = p-1(Prod)
    E = epsi(P)
    V = -((E^ * par(I(R*R^),sub(R)) & -E^) * Ln1(P))
    RETURN E * inj(V)^                   { every column gives a multirelation as a vector }
  END

StrictUpclosed(N)                        { enumerate all strict, up-closed multirelations }
  DECL Prod = PROD(N*N^,N^*N)
       E,EP,U,S
  BEG
    E = epsi(N)
    EP = epsi(p-1(Prod))
    U = -dom(EP^ * par(I(N*N^),sub(N)) & -EP^)
    S = dom(((EP^ / [I(N*N^),E]) & (-EP^ / [I(N*N^),-E])) * N)
    RETURN EP * inj(U & S)^              { every column gives a multirelation as a vector }
  END
```

```
Loop(N) = L(N*N^) * (N & epsi(N)).       { multirelation representing the endless loop }

Strict(U,N)                              { select strict multirelations from up-closed ones }
  DECL LL
  BEG
    LL = Loop(N)
    RETURN U * inj((r2v(LL & N) \ U)^ & (U \ r2v(LL | -N)))^
  END

col_index(U,R) = syq(U,r2v(R) * L1n(U)). { point indexing multirelation R from columns of U }

isol(U,X,Y,Z) = col_index(U,comp(X,Z)) & col_index(U,comp(Y,Z))^.
isor(U,X,Y,Z) = col_index(U,comp(Z,X)) & col_index(U,comp(Z,Y))^.
isoj(U,X,Y,Z) = col_index(U,X | Z) & col_index(U,Y | Z)^.
isom(U,X,Y,Z) = col_index(U,X & Z) & col_index(U,Y & Z)^.
isod(U,X,Y)   = col_index(U,dual(X)) & col_index(U,dual(Y))^.

APX(U,N)                                 { least order satisfying constraints stated below }
  DECL LL,A,Aprev,Atodo,PA,X,Y,Utodo,PU,Z
  BEG
    LL = Loop(N)
    A = refl(trans(col_index(U,LL)))     { order has least element LL }
    Aprev = O(A)
    WHILE -eq(A,Aprev) DO                { order changed? }
      Aprev = A
      Atodo = A
      WHILE -empty(Atodo) DO             { process all pairs (X,Y) in current order }
        PA = atom(Atodo)
        Atodo = Atodo & -PA
        X = v2r(U * dom(PA),LL)
        Y = v2r(U * dom(PA^),LL)
        Utodo = Ln1(U^)
        WHILE -empty(Utodo) DO           { combine with all multirelations Z in U }
          PU = point(Utodo)
          Utodo = Utodo & -PU
          Z = v2r(U * PU,LL)             { multirelational operations are isotone }
          A = A | isol(U,X,Y,Z) | isor(U,X,Y,Z) | isoj(U,X,Y,Z) | isom(U,X,Y,Z) | isod(U,X,Y)
        OD
      OD
      A = refl(trans(A))
    OD
    RETURN A
  END

Hasse(A) = A & -I(A) & -((A & -I(A)) * (A & -I(A))).

APXrel(U,N)                              { order based on relational definition }
  DECL LL,A,Todo,P,Q,R                   { gives the same result as APX }
  BEG
    LL = Loop(N)
    A = O(U^ * U)
    Todo = L(A)
    WHILE -empty(Todo) DO                { process all pairs (Q,R) of multirelations in U }
      P = atom(Todo)
      Todo = Todo & -P
      Q = v2r(U * dom(P),LL)
      R = v2r(U * dom(P^),LL)            { evaluate relational definition for Q and R }
      IF incl(R & LL,Q) & incl(Q,R | LL) THEN
        A = A | P
      FI
    OD
    RETURN A
  END

{ The following functions are taken from RelView libraries. }
```

```
par(R,S)                                  { parallel composition of relations R and S }
  DECL ProdDom = PROD(R*R^,S*S^)
       ProdRan = PROD(R^*R,S^*S)
  BEG
    RETURN (p-1(ProdDom) * R * p-1(ProdRan)^) & (p-2(ProdDom) * S * p-2(ProdRan)^)
  END

r2v(R)                                    { transform relation R:A<->B to vector A*B<->1 }
  DECL Prod = PROD(R*R^,R^*R)
  BEG
    RETURN dom(p-1(Prod) * R & p-2(Prod))
  END

v2r(V,R)                                  { transform vector V:A*B<->1 to relation A<->B }
  DECL Prod = PROD(R*R^,R^*R)             { the result has the same type as R }
  BEG
    RETURN p-1(Prod)^ * (p-2(Prod) & V * L1n(R))
  END
```

References

[1] R.-J. Back, J. von Wright, Refinement Calculus, Springer, New York, 1998.
[2] R. Berghammer, Ordnungen, Verbände und Relationen mit Anwendungen, second edition, Springer, 2012.
[3] R. Berghammer, G. Schmidt, The RELVIEW-system, in: C. Choffrut, M. Jantzen (Eds.), STACS 91, in: Lecture Notes in Computer Science, vol. 480, Springer, 1991, pp. 535–536.
[4] R. Berghammer, G. Schmidt, H. Zierer, Symmetric quotients and domain constructions, Inf. Process. Lett. 33 (3) (1989) 163–168.
[5] G. Birkhoff, Lattice Theory, third edition, Colloquium Publications, vol. XXV, American Mathematical Society, 1967.
[6] G. Birkhoff, S.A. Kiss, A ternary operation in distributive lattices, Bull. Am. Math. Soc. 53 (8) (1947) 749–752.
[7] A. Cavalcanti, J. Woodcock, S. Dunne, Angelic nondeterminism in the unifying theories of programming, Form. Asp. Comput. 18 (3) (2006) 288–307.
[8] S. Dunne, Recasting Hoare and He's Unifying Theory of Programs in the context of general correctness, in: A. Butterfield, G. Strong, C. Pahl (Eds.), 5th Irish Workshop on Formal Methods, Electronic Workshops in Computing, The British Computer Society, 2001.
[9] A.A. Grau, Ternary Boolean algebra, Bull. Am. Math. Soc. 53 (6) (1947) 567–572.
[10] W. Guttmann, Fixpoints for general correctness, J. Log. Algebr. Program. 80 (6) (2011) 248–265.
[11] W. Guttmann, Algebras for iteration and infinite computations, Acta Inform. 49 (5) (2012) 343–359.
[12] W. Guttmann, Unifying lazy and strict computations, in: W. Kahl, T.G. Griffin (Eds.), Relational and Algebraic Methods in Computer Science, in: Lecture Notes in Computer Science, vol. 7560, Springer, 2012, pp. 17–32.
[13] W. Guttmann, Algebras for correctness of sequential computations, Sci. Comput. Program. (2013), http://dx.doi.org/10.1016/j.scico.2013.08.008.
[14] W. Guttmann, Extended designs algebraically, Sci. Comput. Program. 78 (11) (2013) 2064–2085.
[15] W.H. Hesselink, Programs, Recursion and Unbounded Choice, Cambridge University Press, 1992.
[16] W.H. Hesselink, Multirelations are predicate transformers. Available from http://www.cs.rug.nl/~wim/pub/whh318.pdf, 2004.
[17] C.A.R. Hoare, J. He, Unifying Theories of Programming, Prentice Hall Europe, 1998.
[18] D. Jacobs, D. Gries, General correctness: A unification of partial and total correctness, Acta Inform. 22 (1) (1985) 67–83.
[19] O. Klinke, On the 90-degree-lemma, Technical report, University of Birmingham, 2008. Available from http://epapers.bham.ac.uk/53/.
[20] C.E. Martin, S.A. Curtis, I. Rewitzky, Modelling angelic and demonic nondeterminism with multirelations, Sci. Comput. Program. 65 (2) (2007) 140–158.
[21] G. Nelson, A generalization of Dijkstra's calculus, ACM Trans. Program. Lang. Syst. 11 (4) (1989) 517–561.
[22] R. Parikh, Propositional logics of programs: new directions, in: M. Karpinski (Ed.), Foundations of Computation Theory, in: Lecture Notes in Computer Science, vol. 158, Springer, 1983, pp. 347–359.
[23] I. Rewitzky, Binary multirelations, in: H. de Swart, E. Orłowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments, in: Lecture Notes in Computer Science, vol. 2929, Springer, 2003, pp. 256–271.
[24] I. Rewitzky, C. Brink, Monotone predicate transformers as up-closed multirelations, in: R. Schmidt (Ed.), Relations and Kleene Algebra in Computer Science, in: Lecture Notes in Computer Science, vol. 4136, Springer, 2006, pp. 311–327.
[25] G. Schmidt, Partiality I: Embedding relation algebras, J. Log. Algebr. Program. 66 (2) (2006) 212–238.
[26] G. Schmidt, Relational Mathematics, Cambridge University Press, 2011.
[27] G. Schmidt, C. Hattensperger, M. Winter, Heterogeneous relation algebra, in: C. Brink, W. Kahl, G. Schmidt (Eds.), Relational Methods in Computer Science, Springer, Wien, 1997, pp. 39–53, Chapter 3.
[28] G. Schmidt, T. Ströhlein, Relationen und Graphen, Springer, 1989.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 212–224

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Hopscotch—reaching the target hop by hop

Peter Höfner [a,c,*], Annabelle McIver [b]

[a] NICTA, Australia
[b] Department of Computing, Macquarie University, Australia
[c] Computer Science and Engineering, University of New South Wales, Australia

ARTICLE INFO

Article history:
Available online 12 February 2014

Keywords:
Routing
Semiring
Kleene algebra
Path algebra

ABSTRACT

Concrete and abstract relation algebras have widespread applications in computer science. One of the most famous is graph theory. Classical relations, however, only reason about connectivity, not about the length of a path between nodes. Generalisations of relation algebra, such as the min-plus algebra, use numbers allowing the treatment of weighted graphs. In this way one can for example determine the length of shortest paths (e.g. Dijkstra's algorithm). In this paper we treat matrices that carry even more information, such as the "next hop" on a path towards a destination. In this way we can use algebra not only for determining the length of paths, but also the concrete path. We show how paths can be reconstructed from these matrices, hop by hop. We further sketch an application for message passing in wireless networks.

It is our pleasure to dedicate this paper to Gunther Schmidt at the occasion of his 75th birthday. Gunther has been working in the area of relation algebra for many years, publishing countless papers and two books. His first book particularly addresses the relationship between relations and graphs. Relations are an excellent tool for analysing unweighted graphs; subsequently weighted graphs were "algebraised" using matrices over reals or naturals, such as the min-plus algebra, a long-term interest of Gunther. In this paper we continue this line of research and show how to treat matrices that contain even more information, not only numbers. In this respect we illustrate, as Gunther did on so many occasions, that matrices are a powerful, concise and easy-to-understand tool in computer science. So, all the best Gunther, and many more years of health and success!

1. Introduction

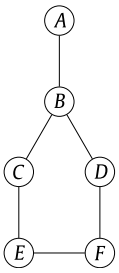
Concrete and abstract relation algebras have widespread applications in computer science. Using concrete relation algebra, it is easy and concise to characterise and calculate with concepts such as orderings, equivalence relations, reflexive closures, Church–Rosser theorems, etc. [1–5]. Concrete relation algebra can be represented either by sets of relations (pairs) or by matrices over the Boolean algebra; it is well known that both representations are isomorphic.

Abstract relation algebra generalises matrices or sets and uses an axiomatic approach. This abstraction allows inherently algebraic reasoning (often equational), which can be automated [6].

One of the most famous applications of relation algebra is graph theory (e.g. [4]). Characterising (unweighted) graphs by their adjacency matrices allows calculations on graphs to be performed algebraically. For example, the reflexive transitive closure can be characterised by the least fixpoint of $x = a \cdot x + 1$, and hence easily calculated by the Knaster–Tarski fixpoint

* Corresponding author.
E-mail addresses: peter.hoefner@nicta.com.au (P. Höfner), annabelle.mciver@mq.edu.au (A. McIver).

http://dx.doi.org/10.1016/j.jlap.2014.02.009
2352-2208/



(a) 6-nodes graph

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

(b) Corresponding Relation

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} \infty & 1 & \infty & \infty & \infty & \infty \\ 1 & \infty & 1 & 1 & \infty & \infty \\ \infty & 1 & \infty & \infty & 1 & \infty \\ \infty & 1 & \infty & \infty & \infty & 1 \\ \infty & \infty & 1 & \infty & \infty & 1 \\ \infty & \infty & \infty & 1 & 1 & \infty \end{pmatrix} \end{array}$$

(c) Corresponding Matrix (in min-plus algebra)

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 2 & 3 & 3 \\ 1 & 0 & 1 & 1 & 2 & 2 \\ 2 & 1 & 0 & 2 & 1 & 2 \\ 2 & 1 & 2 & 0 & 2 & 1 \\ 3 & 2 & 1 & 2 & 0 & 1 \\ 3 & 2 & 2 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

(d) Reflexive transitive closure (in min-plus algebra)

Fig. 1. Graph, relation and reflexive transitive closure (min-plus algebra).

theorem. However, the reflexive transitive closure of a relation only determines whether two nodes are connected via a path or disconnected. Relation algebra cannot be used to determine the length of a path between nodes, nor to handle weighted graphs.

To overcome this deficiency (abstract) relation algebra was generalised and the developed techniques were extended to semirings, Kleene algebras and similar structures (e.g. [7–9]). For weighted graphs the max-plus algebra [10], the min-plus algebra[1] (e.g. [11]), the min-max algebra and fuzzy relations [12] turned out to be useful. Using these semiring-like structures it is possible to formally derive algorithms, such as Dijkstra's shortest-path algorithm or the algorithm of Floyd and Warshall [13]. Other applications for this type of algebra include refinement [14], knowledge representation [15], social choice theory [16], preference modelling [17,18] and program analysis [19,20].

This paper aims to formalise concepts of (wireless) networks algebraically. Based on previous work [21], we show that path finding in a matrix-presentation of a network is not an easy task and that some new theory is needed. As a result, this paper presents new definitions of "hops" and "paths" in an algebraic setting and discusses fundamental properties. To guarantee applicability, we relate our formalism to routing protocols for wireless mesh networks. In this paper we focus on characterising the basic concepts, not on algebraic reasoning.

The paper is organised as follows. In Section 2 we explain the problem of describing paths algebraically in the context of routing protocols for wireless mesh networks. Section 3 recalls the basic algebraic constructions we need, and in Section 4.1 we set out an algebraic development for paths as sequences of nodes. Finally, in Section 5, we illustrate the ideas on a small example taken from routing.

2. A hidden path

Let us first present the problem at hand. For this purpose, we assume that the reader is familiar with relations and relation algebra.[2] The remaining necessary algebraic foundations are formally introduced in Section 3.

2.1. The "classical" encoding

It is well known that matrices over Boolean algebras can be used to describe (unweighted) graphs by their adjacency matrices. This (essentially) yields relation algebra. An example is given in Fig. 1(b), which encodes the graph presented in Fig. 1(a). It is also known that the reflexive transitive closure operation (Kleene star) determines the connectivity between nodes. In the example there is a path between all nodes, i.e., all nodes are connected with each other. Hence the reflexive transitive closure is the all-relation $\top$ ($\top_{ij} = 1$ for all i, j). However, this result neither indicates the distances between nodes nor the actual paths.

The min-plus semiring can be used to determine path lengths. This algebra uses $\mathbb{R}_{\geqslant 0} \cup \{\infty\}$ as carrier set, defines addition as min, multiplication as $+$, and Kleene star as $a^* = 0$. As usual, these operations can be lifted to matrices. Fig. 1(c) shows the characterisation of the graph of Fig. 1(a), using the min-plus algebra; Fig. 1(d) the reflexive, transitive closure—it

[1] This algebra is sometimes called tropical semiring.

[2] The definitions of abstract and concrete relation algebras are given in Appendix A.

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,\infty) & (C,1) & (D,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) & (\varepsilon,\infty) & (E,1) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (F,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (C,1) & (\varepsilon,\infty) & (\varepsilon,\infty) & (F,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (D,1) & (E,1) & (\varepsilon,\infty) \end{pmatrix} \end{array}$$

(a) Matrix using next hops and hop count

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (B,1) & (B,2) & (B,2) & (B,3) & (B,3) \\ (A,1) & (\varepsilon,0) & (C,1) & (D,1) & (C,2) & (D,2) \\ (B,2) & (B,1) & (\varepsilon,0) & (B,2) & (E,1) & (E,2) \\ (B,2) & (B,1) & (B,2) & (\varepsilon,0) & (F,2) & (F,1) \\ (C,3) & (C,2) & (C,1) & (F,2) & (\varepsilon,0) & (F,1) \\ (D,3) & (D,2) & (E,2) & (D,1) & (E,1) & (\varepsilon,0) \end{pmatrix} \end{array}$$

(b) Reflexive transitive closure

Fig. 2. Matrix representation of Fig. 1(a) and Kleene star, using next hops.

shows the distances between nodes, but still no paths. An entry labelled with ∞ indicates that there is not path between nodes.

2.2. Wireless mesh networks and routing

To encode paths in a matrix-model, one can calculate with language-like models (see e.g. [13]). In this paper we use an algebraic model inspired by *routing tables* as they are used by nodes in a (wireless) network to forward data packets; in particular routing tables used in wireless mesh networks. *Wireless mesh networks* (WMNs) are self-organising ad-hoc networks that support broadband communication without relying on a wired backbone. Instead of having a central component (router), each node in the network acts as an independent router. Route finding and route maintenance are critical for the performance of these networks. A widely used routing algorithm for WMNs is the Ad hoc On-Demand Distance Vector (AODV) routing protocol [22].

For the purpose of this paper, we use AODV as motivating example; the presented theory, however, applies to other routing protocols as well. AODV is a reactive protocol, i.e., route discovery processes are only initiated when needed. As for most routing protocols, the basic idea of AODV is inspired by (a variant of) Dijkstra's shortest path algorithm. This shortest path algorithm keeps a record of the paths from a source, extending those paths in each iteration of the algorithm. AODV is based on a distributed version of this idea. Instead of having global knowledge, each node maintains a routing table, which is updated whenever messages are received.

2.3. Next hops and routing tables

Since Dijkstra's shortest path algorithm can be modelled by an algebraic approach [23,13], it is no surprise that similar techniques can be used for modelling routing tables [21]: entries in the table contain information for each possible destination, including the next hop (not the entire path) together with the total number of hops required to reach the destination. The *next hop* specifies a neighbouring node where the packet has to be sent to reach the destination; the *hop count* specifies the length of the entire path. As before the length of a non-existent path has length ∞; in case no next hop exists—either no path exists or it is the trivial path with hop count 0—we use the symbol ε.

As for relations and the min-plus algebra, any graph can be encoded as a matrix using the introduced pairs (Fig. 2(a)). Fig. 2(b) shows a matrix indicating *all* shortest paths of the graph of Fig. 1(a) (this matrix can be determined by the Kleene star of the matrix of Fig. 2(a)). In our model [21], we use pairs (m, x) as matrix entries, where m indicates the next hop and x the hop count. Formally, we assume an ordered set $(\Sigma, \preceq)$ of node names, and calculate using $(\Sigma \times \mathbb{N}\backslash\{0\}) \cup \{(\varepsilon, 0), (\varepsilon, \infty)\}$, where $\varepsilon \notin \Sigma$. The two special entries $(\varepsilon, 0)$ and (ε, ∞)[3] hence no next hop must be given.

A (global) *snapshot* is a $n \times n$ matrix of such pairs. Each line of a snapshot describes a *routing table* of the corresponding node. In the presented example (Figs. 1 and 2), the routing table of node F is represented by the matrix' last row. F's entry for destination A (first column) has its next hop registered as D with a hop count 3, whereas the intermediate nodes D and B have the next hops B and A, and hop counts 2 and 1, respectively.

In [21] it is shown that the set of all snapshots form a Kleene algebra, which allows algebraic modifications.

2.4. The ad hoc on-demand distance vector protocol

Each routing table (row of a snapshot) represents the local knowledge of a node. Local knowledge is often neither complete nor optimal, i.e., there might be routes in the network, which have not yet been discovered; or the routing protocol might have failed to establish the shortest route—these discovered routes are indicated by the precise matrix entries.

Routing tables (and snapshots) are updated as a result of message passing. In case of AODV, a route discovery process of node A searching for a route to node D (the destination) starts with A broadcasting a route request (RREQ) message, which is received by all nodes within A's transmission range. If an intermediate node (not the destination) receives a RREQ message and does not have a valid entry for the destination in its routing table, the request is forwarded by re-broadcasting

[3] We only use pairs to be consistent with other elements of $\Sigma \times \mathbb{N}\backslash\{0\}$. state that there is a trivial path of length 0 and no path at all (a path of length ∞).

the RREQ message. During this forwarding process, the intermediate node updates its routing table and adds a "reverse route" entry with destination A into its routing table, indicating via which next hop A can be reached, and the distance in number of hops.

As soon as the RREQ is received by the destination itself or by a node that knows a valid route to the destination, a route reply (RREP) is generated. In contrast to RREQ messages, a RREP message is unicast, i.e., it is only sent to a single node, not to all nodes within transmission range. The RREP message travels from its generator (either D or an intermediate node knowing a route to D) back along the established route towards A, the originator of the RREQ message. All intermediate nodes on the selected route will process the RREP message and, in most cases, forward it towards A. By passing a RREP message towards A, a node adds a "forward route" entry to its routing table.

The route discovery process is completed when the RREP reaches node A; an end-to-end route from A to D has been established, and data packets can start to flow. Full details can be found in RFC 3561 [22], the de facto standard of AODV.

2.5. Message passing

We observe that messages are either *broadcast* or *unicast*: broadcast messages are intended for any node in transmission range, whereas unicast messages are addressed to a single node. Thus unicast messages are only received by the addressee, and any other node which picks up the message simply ignores it. Message passing can be modelled by algebraic matrix operations [21]. At the algebraic level message sending is expressed by $a + b + b \cdot c$, for some algebraic elements a, b, c (details are given later).

It has been shown that broadcasting a message can "easily" modelled in algebra, as long as the current topology b is given. Unicasting, however, is a much harder task since knowledge about the next hops needs to be distilled from the snapshot. To explain this desire we revisit the example an consider again the matrix of Fig. 2(b).

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (B,1) & (B,2) & (B,2) & (B,3) & (B,3) \\ (A,1) & (\varepsilon,0) & (C,1) & (D,1) & (C,2) & (D,2) \\ (B,2) & (B,1) & (\varepsilon,0) & (B,2) & (E,1) & (E,2) \\ (B,2) & (B,1) & (B,2) & (\varepsilon,0) & (F,2) & (F,1) \\ (C,3) & (C,2) & (C,1) & (F,2) & (\varepsilon,0) & (F,1) \\ (D,3) & (D,2) & (E,2) & (D,1) & (E,1) & (\varepsilon,0) \end{pmatrix} \end{array}$$

The first entry in the last row only indicates the existence of a path from F to A of length 3, with given next hop D. A message needs to be sent to D first (via the link $(D,1)$). Node D, after it has received the packet from F, inspects its own routing table (4th row of the matrix) and determines that its own path to A has B as next hop. Hence the packet is forwarded via the link $(B,1)$. This forwarding continues until the destination A is reached. This "hopping from node to node" is one of the fundamental ideas of routing in (wireless) networks. It allows flexible changes in the path information without informing every node.

Assuming that one can distill the marked entries from the matrix, then this matrix can be used as topology-matrix b in the above equation, which then models unicast algebraically. The main contribution of this paper is to determine an algebraic presentation of paths, which paves the way for algebraic modelling and algebraic reasoning for networks in general and routing in particular. As an example it should be possible to give algebraic characterisations of properties for routing protocols, that should be guaranteed:

- *Packet delivery*: when a node has information about a destination D, then the intermediate nodes do so as well;
- *Loop freedom*: at no time should there be a non-trivial loop in the snapshot.

These properties might be obvious for algorithms such as Dijkstra's shortest path. However, we have shown that in AODV snapshots only reflect local knowledge about the networks and hence theses properties have to be guaranteed. Both properties can be characterised over paths; hence it is necessary to distill paths out of a given matrix. Of course one can just use domain-theoretic reasoning, but a purely algebraic approach brings several advantages, such equational reasoning and proof automation with off-the-shelf automated theorem provers [6].

3. Kleene algebras, products and special elements

In this section we briefly recapitulate the algebraic theory needed. It is well known that semirings and Kleene algebras model sequential composition, (non-deterministic) choice and, in case of Kleene algebra, finite iteration in a first-order equational calculus.

Formally, an *idempotent semiring* (*i-semiring*) is a quintuple $(S, +, 0, \cdot, 1)$ such that $(S, +, 0)$ is an idempotent commutative monoid, $(S, \cdot, 1)$ is a monoid, multiplication distributes over addition and 0 is an annihilator w.r.t. multiplication $\cdot$. The *natural order* $\leqslant$ on S is given by $a \leqslant b \Leftrightarrow a + b = b$. An i-semiring induces an *upper semilattice* in which $a + b$ is the supremum of a and b, and 0 is the least element.

A *Kleene algebra* is an idempotent semiring S extended by an operation $^*\colon S \to S$ for iterating an element an arbitrary but finite number of times. Such an operation has to satisfy the *star unfold* and the *star induction* axioms

$$1 + a \cdot a^* = a^*, \qquad b + a \cdot c \leqslant c \;\Rightarrow\; a^* \cdot b \leqslant c,$$
$$1 + a^* \cdot a = a^*, \qquad b + c \cdot a \leqslant c \;\Rightarrow\; b \cdot a^* \leqslant c.$$

The (direct) product of two i-semirings (Kleene algebras) forms again an i-semiring (a Kleene algebra) if all operations are defined component-wise, i.e., $(a_1, a_2) + (b_1, b_2) = (a_1 + a_2, b_1 + b_2)$, $(a_1, a_2) \cdot (b_1, b_2) = (a_1 \cdot a_2, b_1 \cdot b_2)$, and $(a_1, a_2)^* = (a_1^*, a_2^*)$. In this case, the natural order is also component-wise. If the natural order should coincide with a lexicographical order, the product becomes more intricate.

Theorem 3.1. *(See [21].) Assume two totally ordered sets $(N, \preceq)$ and $(H, \sqsubseteq)$ with an isotone binary operation $\circ$ on N and a binary operation $\bullet$ on H that is strictly isotone, i.e., $a \sqsubset b \;\Rightarrow\; (c \bullet a \sqsubset c \bullet b) \wedge (a \bullet c \sqsubset b \bullet c)$.*

On the set $\mathbb{M} = (N \times H) \cup \{\bot, \top\}$ addition and multiplication is defined as

$$(m, x) + (n, y) = \begin{cases} (m, x) & \text{if } (x \sqsubset y) \vee (x = y \wedge m \preceq n), \\ (n, y) & \text{otherwise}, \end{cases}$$
$$(m, x) \cdot (n, y) = (m \circ n, x \bullet y),$$

for $(m, x), (n, y) \in N \times H$. For the special elements $\bot$ and $\top$ choice is defined by $\bot + r = r + \bot = r$ and $\top + r = r + \top = \top$, and multiplication by $\bot \cdot r = r \cdot \bot = \bot$ and $\top \cdot r = r \cdot \top = r$, for all $r \in \mathbb{M}$.

(a) *The structure $S = (\mathbb{M}, +, \top, \cdot, \bot)$ is an i-semiring, if $a \sqsubset a \bullet b$ for all $a, b \in H$; the natural order coincides with the lexicographical order $(n, y) \leqslant (m, x) \Leftrightarrow (x \sqsubset y) \vee (x = y \wedge m \preceq n)$; $\bot$ is the least and $\top$ the greatest element.*

(b) *Under the conditions of Part* (a)*, setting $r^* = \top$ for all $r \in \mathbb{M}$ turns S into a Kleene algebra.*

If we assume an ordered set $(\Sigma, \preceq)$ of node identifiers and the hop count to be an element of $\mathbb{N}\backslash\{0\}$, this theorem shows that there is an underlying Kleene algebra behind the entries of snapshots (see Section 2). In this algebra $(\varepsilon, 0)$ relates to $\top$ and (ε, ∞) to $\bot$. Note that addition chooses the better route. Therefore the natural order reads as "worse than". In particular, $(n, y) \leqslant (m, x)$ means that the entry (n, y) is worse than (m, x). Most often this implies that (m, x) has a smaller hop count than (n, y). In the remainder we refer to this algebra as *next-hop semiring*.

If elements of i-semirings/Kleene algebras characterise single elements or transitions, whole transition systems are usually encoded by matrices.

Theorem 3.2. *(See e.g. [24].) Standard matrix addition and multiplication turns the family $M(n, K)$ of $n \times n$ matrices over a Kleene algebra K into a Kleene algebra; the zero matrix is neutral w.r.t. $+$, and the identity matrix I w.r.t. multiplication. In case K has a greatest element $\top$, the matrix containing $\top$ at all positions is the greatest element in the matrix model.*

The matrix algebra over the next-hop semiring is called *routing algebra*.

For our definition of paths, we also need the concept of irreducibility. Assume an upper semilattice $(S, \leqslant)$ with the least element 0 and $+$ as join operation. An element $a \in S$ is *join-irreducible* if

$$\forall b, c: \quad a = b + c \;\Rightarrow\; (a = b \vee a = c).$$

The next-hop semiring is selective, i.e., $a + b \in \{a, b\}$, hence every element is join-irreducible. Since addition on matrices is defined point-wise, the property lifts nicely to matrices: a matrix is join-irreducible if all but one entries are equal to 0 and if the single entry is join-irreducible w.r.t. the underlying algebra. In the routing algebra this means that there is at most one entry different to (ε, ∞).

Assume an i-semiring with greatest element. In any matrix model, if an element a is join-irreducible, the equation $\top \cdot a \cdot \top$ fills the entire matrix with the non-trivial entry occurring in a. An example, using routing algebra, reads as follows

$$\top \cdot \begin{pmatrix} (A,2) & (\varepsilon,0) & (\varepsilon,0) \\ (\varepsilon,0) & (\varepsilon,0) & (\varepsilon,0) \\ (\varepsilon,0) & (\varepsilon,0) & (\varepsilon,0) \end{pmatrix} \cdot \top = \begin{pmatrix} (A,2) & (A,2) & (A,2) \\ (A,2) & (A,2) & (A,2) \\ (A,2) & (A,2) & (A,2) \end{pmatrix}.$$

In case multiplication is defined, an element a is *multiplicative-irreducible* if

$$\forall b, c: \quad a = b \cdot c \;\Rightarrow\; (a = b \vee a = c).$$

In the next-hop semiring a is multiplicative-irreducible iff $a = (*, 1)$ (for some $* \in \Sigma$), $a = (\varepsilon, 0)$ or $a = (\varepsilon, \infty)$, meaning that a route cannot be split up into two "proper subroutes". Since multiplication is not point-wise on matrices, the property does not lift nicely. In the next section we will discuss alternatives to this definition that can be successfully lifted to matrices.

$$\begin{pmatrix} (\varepsilon, \mathbf{0}) & (\varepsilon, \infty) & (\varepsilon, \infty) \\ (\varepsilon, \infty) & (\varepsilon, \infty) & (\varepsilon, \infty) \\ (\varepsilon, \infty) & (\varepsilon, \infty) & (\varepsilon, \infty) \end{pmatrix}$$

Fig. 3. A pre-node.

4. Building paths using bricks

4.1. Nodes and tests

A path should always end up in a node. The simplest path is a node with a connection to itself. In the routing algebra a node has exactly one entry different to (ε, ∞). Moreover, this value needs to be the multiplicative unit $(\varepsilon, 0)$ and on the diagonal. An example is given in Fig. 3.

Definition 4.1. Assume an i-semiring S with greatest element $\top$. An element $a \in S$ is a *pre-node* if it is a subidentity ($a \leqslant 1$), join-irreducible, and satisfies $\top \cdot a \cdot \top = \top$.

In models based on matrices, the subidentity-property forces the non-trivial value to sit on the diagonal.

In the setting of relation algebra, Schmidt and Ströhlein defined the concept of points [25]. They were inspired by the "general theory of graphs". The intuition behind both definitions is the same—modelling single nodes in a graph. A point in a concrete relation algebra is modelled as a matrix that has exactly one row filled with 1's. Although pre-nodes and points are not equivalent, they are closely related; see Appendix A for more details.

Pre-nodes and points are also related to the concept of tests [26]. A *test semiring* [26] is a pair $(S, \mathsf{test}(S))$, where S is an i-semiring and $\mathsf{test}(S) \subseteq [0, 1]$ is a Boolean subalgebra of the set $[0, 1]$ of S such that $0, 1 \in \mathsf{test}(S)$ and join and meet in $\mathsf{test}(S)$ coincide with addition $(+)$ and multiplication $(\cdot)$. By $\neg$ we denote complementation in $\mathsf{test}(S)$. Since the test algebra forms a Boolean algebra, the following shunting rule holds for $p, q, r \in \mathsf{test}(S)$

$$p \cdot q \leqslant r \;\Leftrightarrow\; p \leqslant \neg q + r.$$

Idempotent semirings admit at least the test algebra $\{0, 1\}$ and can have different test algebras. A semiring with test algebra $\{0, 1\}$ is called *discrete*.

In the matrix model, the maximal test algebra contains all diagonal matrices with tests of the underlying algebra on its diagonal. Hence Theorem 3.2 can be extended to hold for Kleene algebras with tests.

Lemma 4.2. *Assume a matrix-semiring $M(n, S)$ of $n \times n$ matrices over an i-semiring S. The following properties hold*:

(1) *The identity matrix I equals the sum of all pre-nodes.*
(2) *For two pre-nodes a, b with $a \neq b$, $a \cdot b = 0$.*
(3) *Every node is a (join-irreducible) element of the maximal test algebra, but*
(4) *not every join-irreducible test element is a pre-node.*

Proof. Parts (1) and (2) are straightforward, using arguments on the structure of pre-nodes. Part (3) follows by setting $\neg a = \sum_{b \in \{c \mid c \text{ is pre-node},\ c \neq a\}} b$ for a node a. Basically, the idea is to create a diagonal matrix, which entries are 0, where the pre-nodes a had a non-zero entry, and vice versa. Note, that the complement of a is in general not a pre-node. For Part (4) we give a counter example. Assume $S = \mathsf{test}(S) = \{0, p, \neg p, 1\}$; all operations are given by the Boolean algebra of tests. We now consider 2×2-matrices. The matrix $\left(\begin{smallmatrix} p & 0 \\ 0 & 0 \end{smallmatrix}\right)$ is a test in the matrix-semiring, but not a pre-node, since

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} p & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} p & p \\ p & p \end{pmatrix} \neq \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}. \qquad \square$$

This lemma shows that the rule $\top \cdot a \cdot \top = \top$ is crucial for our definition of pre-nodes. However, it also shows that pre-nodes often behave similar to tests.

Proposition 4.3. *Assume a matrix-algebra over an i-semiring (a Kleene algebra) that has a discrete test algebra that is a maximal test algebra. Then all tests are sums of pre-nodes; in case of 0, it is the empty sum.*

Proof. Tests in a matrix-algebra that has a discrete test algebra that is a maximal test algebra are diagonal matrices. The entries on the diagonal are either 1 or 0. Since matrices with exactly one 1 on the diagonal are pre-nodes, the claim is trivial. $\square$

In the general setting of semirings Lemma 4.2 does not hold; counter examples can easily generated by automated tools, such as Mace4 [27]. For example, the following generated example presents a counter example of Lemma 4.2(2).

$+$	0	a	1	$\top$
0	0	a	1	$\top$
a	a	a	1	$\top$
1	1	1	1	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$

$\cdot$	0	a	1	$\top$
0	0	0	0	0
a	0	a	a	$\top$
1	0	a	1	$\top$
$\top$	0	$\top$	$\top$	$\top$

All elements are join-irreducible, a and 1 are the only pre-nodes; but obviously $a \cdot 1 = a$. This counter example also illustrates that 1 cannot be a pre-node, except it is the only one.

It has been shown in several situations that tests are a useful and powerful (e.g. [9,28]). Since we want to make use of both concepts pre-nodes and tests, we combine them.

Definition 4.4. A *node* of an i-semiring is an element which is both a pre-node and a test.

For an i-semiring S, the set of all nodes is denoted by $\mathcal{N}_S$.

Premultiplying an arbitrary matrix in the routing algebra by a pre-node, a node, or a test selects certain rows; hence the routing table information about particular nodes can be selected. Postmultiplying selects columns, i.e., the matrix is "restricted" to information about a particular node or a set of nodes.

It has been shown that i-semirings and Kleene algebras can be equipped with forward and backward modal operators [29], which can be defined via tests. These operators provide a concise and convenient algebraic framework for calculi such as the wp calculus and predicate transformer semantics, but they have also been used in the context of routing [21]. In this paper we will use this concept to model paths. The (modal) forward box-operator $|\_] : S \to (\mathsf{test}(S) \to \mathsf{test}(S))$ is defined by

$$p \leqslant |a]q \;\Leftrightarrow\; p \cdot a \cdot \neg q \leqslant 0, \quad \text{and} \quad |a \cdot b]p = |a]\big(|b]p\big).$$

The forward diamond-operator $|\_\rangle$ is the de Morgan dual of this operation, i.e., $|a\rangle p = \neg|a]\neg p$. This operator will be used to determine whether there is a route from p to q in a (checking $p \leqslant |a\rangle q$). The backward operators are defined symmetrically by $p \leqslant [a|q \Leftrightarrow \neg q \cdot a \cdot p \leqslant 0$, $[a \cdot b|p = [a|([b|p)$ and $\langle a|p = \neg[a|\neg p$. Others have derived many properties for modal Kleene algebra (e.g., [29,30]); here, we only list properties used in the remainder. For an i-semiring S with $a \in S$, $p, q \in \mathsf{test}(S)$, we have

$$p \cdot a = 0 \;\Leftrightarrow\; \langle a|p = 0, \qquad |a\rangle p \leqslant q \;\Leftrightarrow\; p \leqslant [a|q, \quad \text{and} \quad |p \cdot a\rangle q = p \cdot |a\rangle q.^{4}$$

Note that diamond- and box-operators bind more tightly than addition and multiplication. Test semirings equipped with $|\_]$ and $|\_\rangle$ are called *modal*.

It is straightforward to see that both the next-hop semiring and the routing algebra are modal. Modal operators can be used to check whether a route from p to q is known at a snapshot a, checking $p \leqslant |a\rangle q$. The dual inequality $p \leqslant \langle a|q$ checks whether p is a known destination of q.

4.2. Bricks

Based on the definition of nodes we can now turn to paths. A path should list all intermediate nodes as 1-hop neighbours and then finally reach the destination, where it "loops" forever. Remember Fig. 2(b): the path from F to D is given by 1-hop neighbours—a packet travelling along that path needs to be sent via the 1-hop connection to D, then to B and finally to A. That means that only entries of the form $(*, 1)$, $(\varepsilon, 0)$ and (ε, ∞) should be allowed.

If we are able to characterise matrices that have at most one entry of the form $(*, 1)$ or $(\varepsilon, 0)$ (all other entries being (ε, ∞)), then paths can be built from these elements. A matrix with one entry of the form $(*, 1)$ would plead for something like multiplicative-irreducibility on matrix level. However, as mentioned before the definition of multiplicative-irreducibility does not lift to matrices. A counter example for this property, using elements of the routing algebra, is the following

$$\begin{pmatrix} (\varepsilon,\infty) & (B,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) \end{pmatrix} = \begin{pmatrix} (B,1) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (A,3) \end{pmatrix} \cdot \begin{pmatrix} (\varepsilon,\infty) & (\varepsilon,0) \\ (\varepsilon,\infty) & (\varepsilon,\infty) \end{pmatrix}.$$

Intuitively, the matrix on the left-hand side should be irreducible since the path $(B, 1)$ cannot be split into two non-trivial paths. However it can be split into two matrices, both different from the original one. The example also shows that the splitting matrices can contain arbitrary entries; in particular entries with a hop count larger than 1. To exclude matrices with multiple non-trivial entries, we can make use of join-irreducibility; to allow different positioning of an element in a matrix, we will fill up the matrix using the "$\top \cdot x \cdot \top$-construction" and compare the results.

[4] Symmetric rules for the other type of box- and diamond operator hold, but are not listed explicitly.

Definition 4.5. An element a of an i-semiring S is called a *brick* if

(1) a is join-irreducible, and
(2) $\forall b, c\colon a = b \cdot c \;\Rightarrow\; (\top \cdot a \cdot \top = \top \cdot b \cdot \top \vee \top \cdot a \cdot \top = \top \cdot c \cdot \top)$.

The element $\left(\begin{smallmatrix}(\varepsilon,\infty) & (B,1)\\ (\varepsilon,\infty) & (\varepsilon,\infty)\end{smallmatrix}\right)$ is a brick. For an i-semiring, we denote the set of all bricks by $\mathcal{B}_S$, i.e., $\mathcal{B}_S = \{a \in S \mid a \text{ is a brick}\}$.

Lemma 4.6. *In the routing algebra, an element a is a brick if and only if it contains at most one entry of the form $(*, 1)$ or $(\varepsilon, 0)$; all other entries are (ε, ∞).*

Proof. Join-irreducibility does not allow multiple non-(ε, ∞) entries. Assume a matrix M that has an entry with hop count larger than 1, say $(B, 3)$, at position i, j, and that satisfies (2). In the next-hop semiring such a single entry can be split into two non-trivial entries: for example, $(B, 3) = (B, 1) \cdot (A, 2)$. It is easy to construct matrices N and O such that $M = N \cdot O$. In detail, all entries of N and O are equal to (ε, ∞), except N_{ij} and O_{jk}, which are $(B, 1)$ and $(A, 2)$ respectively (for some k). Neither $\top \cdot N \cdot \top = \top \cdot M \cdot \top$ nor $\top \cdot N \cdot \top = \top \cdot O \cdot \top$ hold, hence (2) is not satisfied, a contradiction. □

Corollary 4.7. *In any algebra that satisfies the Tarski-rule $a \neq 0 \;\Rightarrow\; \top \cdot a \cdot \top = \top$ the second item of Definition 4.5 is a tautology. Therefore, in such algebras, a brick is equivalent to a join-irreducible element.*

However, most i-semirings do not satisfy the Tarski-rule, e.g., the next-hop semiring and the routing algebra. Hence the concepts of join-irreducibility and bricks are different.

4.3. Paths

Finally, we can define paths as a sequence of bricks (between nodes) leading to a single node, the destination. Before we give a formal definition, we look at some examples of paths and non-paths. The first example, depicted in Fig. 4, shows a simple path, leading from C via B to A. We notice that a matrix, modelling a path, should only contain entries of the form $(*, 1)$, (ε, ∞) and $(\varepsilon, 0)$ (as mentioned before). We further notice that every row has exactly one entry; and every column has exactly one entry as well, except the column containing $(\varepsilon, 0)$, which has two entries. All paths we want to characterise should be infinite. That means a path either loops at a single node at its end or forms a loop (2nd example of Fig. 4); the third example shows an example, which is not a path. All nodes on a path have at most one successor and at most one predecessor. The only exception is the final node: since at this node the path can loop forever, it has up to two predecessors.

We now present the main definition of this paper.

Definition 4.8. A *path* of an i-semiring S is an element $a \in S$ satisfying the following three properties.

(1) $\exists B \subseteq \mathcal{B}_S\colon\; a = \sum_{b \in B} b$;
(2) $\forall p, q \in \mathcal{N}_S\colon\; q \cdot a \cdot p \neq 0 \;\Rightarrow\; p \cdot a \neq 0$;
(3) $\forall p \in \mathcal{N}_S\colon\; \neg p \cdot |a\rangle p \in \mathcal{N}_S \wedge \langle a| p \in \mathcal{N}_S$.

In the routing algebra (1) means that the matrix consists of $(\varepsilon, 0)$, (ε, ∞) and $(*, 1)$-entries only; (2) guarantees a successor of a path (every node must have (at least one)): if node p can be reached from q via a then the path a must have a continuation from p. (3) restricts paths to have at most one successor and at most one predecessor; with the one exception mentioned before. In the routing algebra this definition allows degenerate paths, meaning that there are snapshots that are paths and that have $(\varepsilon, 0)$ somewhere outside the diagonal. However, in routing tables this never occurs, since all path have at least length 1.

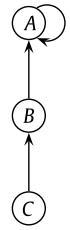
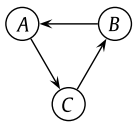
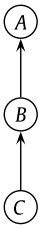
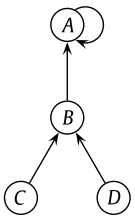
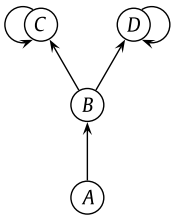
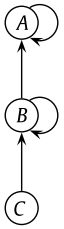
It is not convenient to work with this definition due to the inequations. However, Part (2) can be read as $p \cdot a \leqslant 0 \;\Rightarrow\; q \cdot a \cdot p \leqslant 0$. Hence, by using the formulas of page 218, we get the following result.

Proposition 4.9. *Definition 4.8(2) is equivalent to $\langle a| p \leqslant 0 \;\Rightarrow\; p \cdot \langle a| q \leqslant 0,\; \forall p, q \in \mathcal{N}_S$.*

In the routing algebra, a snapshot contains a path, if both matrices coincide at all positions where the path is not (ε, ∞).

Definition 4.10. Assume an i-semiring S. A path $b \in \mathcal{B}_S$ is *contained* in an element a, written as $b \sqsubseteq a$, if

$$\forall p, q \in \mathcal{N}_S\colon \quad p \cdot b \cdot q = p \cdot a \cdot q \vee p \cdot b \cdot q = 0.$$

Graph[a]	Matrix representation	Description
	$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) \end{pmatrix} \end{array}$	One of the most simple paths; it starts at C, takes B as intermediate node and reaches node A. Paths should only contain entries of the form $(*,1)$, (ε,∞) and $(\varepsilon,0)$.
	$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} (\varepsilon,\infty) & (\varepsilon,\infty) & (C,1) \\ (A,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) \end{pmatrix} \end{array}$	Paths are allowed to have non-trivial loops.
	$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) \end{pmatrix} \end{array}$	We assume that every node has at least one successor node. Since A does not have a successor, this is not a path (by our understanding).
	$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \end{pmatrix} \end{array}$	No node can have more than one predecessor (except the destination); hence no path.
	$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (C,1) & (D,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) \end{pmatrix} \end{array}$	Every node has exactly one successor; hence no path.
	$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,0) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,\infty) \end{pmatrix} \end{array}$	Paths are not allowed to have loops at intermediate nodes. (This coincides with the multiple successor and multiple predecessor rules.)

[a] The graphical representation does not show the hop count. In this figure we assume self-loops to have hop count 0 and all other connections to have hop count 1.

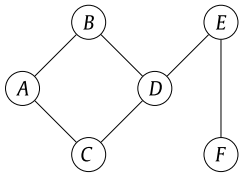
Fig. 4. Two paths and four non-paths.

Proposition 4.11. *Let S be an i-semiring.*

(1) *The relation $\sqsubseteq$ is reflexive.*
(2) *The relation $\sqsubseteq$ is transitive, i.e., for all $a \in S$ and $b, c \in \mathcal{B}_S$*

$$b \sqsubseteq c \wedge c \sqsubseteq a \;\Rightarrow\; b \sqsubseteq a.$$

The proof is straightforward, using the definition.



$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (B,1) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\mathbf{B},\mathbf{2}) & (B,1) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) \end{pmatrix} \end{array}$$

Fig. 5. Graph and snapshot.

5. A brief application in routing

After setting up the definitions, we now briefly turn to an application in routing. Here, we sketch modelling aspects, i.e., we show how certain routing properties can be characterised by algebra. Using the presented characterisation to reason about concrete protocols is part of future work.

5.1. Properties of routing protocols

We first model the packet delivery property: if a node has information about a destination, then there is a path in the snapshot leading to the destination.

In the routing algebra, a node A has information about destination D (with respect to a given snapshot) if the corresponding entry in the snapshot is not equal to (ε,∞). In Fig. 2(b), node F has information about A, since the corresponding entry is $(D,3)$. This can be checked by restricting the snapshot to the corresponding nodes.

Algebraically, a node p has information about node d w.r.t. a if $p\cdot a\cdot a\neq 0$. By this we can define packet delivery.

Definition 5.1. Let K be a Kleene algebra and $a\in K$. A node p satisfies the *packet delivery* property for node q if a contains a path that is consistent with the corresponding position:

$$p\cdot a\cdot q\neq 0 \;\Rightarrow\; \exists b\in\mathcal{B}_K\colon\quad b\sqsubseteq a\wedge p\cdot b^*\cdot q=p\cdot a\cdot q.$$

Example 5.2. Assume the snapshot depicted in Fig. 5. This snapshot, which we denote M, does not contain many paths; it is a typical example of a network running AODV where not many messages have been sent. In fact, only A broadcast a message in the past to its neighbours B and C, and B forwarded this message.

Let us look at D's routing table. It is easy to see that D satisfies the packet delivery property for A. However, if the entry would be $(C,2)$—instead of $(B,2)$—the property would not hold anymore, since M contains no path from D via C to A.

In the same vein as packet delivery, we can characterise loop freedom. Informally, loop freedom states that at no time there should be a non-trivial loop in a snapshot. We can use the transitive closure to check for non-trivial paths. As usual, the *transitive closure* of an element a of a Kleene algebra is defined as $a^+=a\cdot a^*$.

Definition 5.3. An element a of a Kleene algebra K contains a loop if

$$\exists b\in\mathcal{B}_K,\ p\in\mathcal{N}_K\colon\quad b\sqsubseteq a\wedge p\cdot b^+\cdot p\neq 0\wedge p\cdot b^+\cdot p\neq 1.$$

An element is *loop free* if it does not contain a loop.

5.2. Broadcast, unicast and forward

As mentioned earlier message sending can be expressed by

$$a+b\cdot(1+c),$$

where a,b,c are elements of an i-semiring S. By distributivity, this expression consists of three parts: a, b and $b\cdot c$. Informally, a describes the system before the message is sent (hence it is not part of the message itself); it is a snapshot which then, after the message has been delivered, is updated by the two other summands. The node b represents the current topology[5] and establishes a 1-hop connection between nodes. The third term $(b\cdot c)$ transmits knowledge c via the topology; during sending the knowledge is adapted to the topology. We refer to [21] for details.

From this algebraic perspective broadcast and unicast is the same. But, the topology must be adapted in a way that it only offers the links used.

[5] By a topology we understand a matrix which characterises a particular graph, i.e., it only contains entries of the form $(*,1)$, (ε,∞) and $(\varepsilon,0)$.

Example 5.4. Let us look at the topology that corresponds to the graph given in Fig. 5:

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (B,1) & (C,1) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,0) & (\varepsilon,\infty) & (D,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,\infty) & (\varepsilon,0) & (D,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (C,1) & (\varepsilon,0) & (E,1) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (D,1) & (\varepsilon,0) & (F,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (E,1) & (\varepsilon,0) \end{pmatrix} \end{array}$$

If node D wants to broadcast a message then the topology should be restricted to the 4th row; this is easily achieved by premultiplying the topology by node D.

In AODV, route replies are unicast back to the originator of the route discovery process. We have to restrict the topology or more precisely the snapshot in such a way that it contains only the single path back to the originator. If we consider the snapshot given in Fig. 5, and D needs to unicast a packet to A, we have to pick a path starting with the entry $(D,1)$. This can be done by the path selected for the packet delivery property. If this path is restricted to the sender, the packet is sent to the next hop.

Propagating and forwarding the message through the whole network can be expressed by using Kleene star.

$$a + b \cdot \lfloor b^* \rangle p + b^* \cdot p.$$

This equation, which was derived in [21], is now true for both, broadcast and unicast, depending on the topology chosen. Remember that for unicast a path is chosen.

6. Conclusion and outlook

In this paper we have treated matrices carrying information useful for routing algorithms. For modelling routing tables we have used pairs as entries of the matrices. These pairs consist of the length of a path and the next hop on the path. Using examples of wireless mesh networks we have argued that a path is a concept needed to reason about routing. We have then shown how to treat paths algebraically and have established a relationship to routing algorithms. Having the concept of paths solved one problem that occurred when modelling a unicast-mechanism, and which was mentioned in [21].

The approach we follow in [21] and in this paper is based on ideas of Sobrinho and Griffin [31–33] and of Carré [34] Sobrinho and Griffin were the first to bring algebraic reasoning into the realm of hop-by-hop routing. Sobrinho [31] states that he follows an algorithmic approach while others use a matrix approach [34]. One of our aim is to combine these two approaches. A recent contribution, also based on algebraic principles, is NetKAT [35]. It aims to provide a solid mathematical foundation for new network programming language based on Kleene algebras with tests [26]. This approach is orthogonal to ours: whereas our goal is to be able to derive and verify routing algorithms, the aim of NetKAT to use the algebraic foundation to ensure that languages built on top of it satisfy some universal properties.

So far we concentrated on modelling crucial properties of routing protocols algebraically; formal reasoning using the given definitions is part of future work. Moreover we hope to find even simpler characterisations for bricks, paths and the properties of packet delivery and loop freedom: this would be extremely useful since inequations, as they appear in the definitions, are inconvenient for algebraic reasoning.

So far the routing algebra is based on pairs storing the next hop and the hop count. Routing protocols used in industry save additional information inside the routing tables. An example are sequence numbers which indicate the freshness of routes. However, when including this into the routing algebra, we cannot use Theorem 3.1 any longer and the resulting matrix algebra does not satisfy distributivity nor associativity laws. A solution to this is to use module-like structures such as Kleene modules [36]; a detailed analysis, however, on the relationship between our treatment of paths, the modelling of AODV (as presented in [21]) and Kleene modules is part of future work.

> '*Copilowish* [11] *"enjoyed the full benefits of the matrix approach" and regretted that this "elegant machinery is apparently too little known." We think these feelings can be shared today.*' [6]
>
> Gunther Schmidt [25]

Acknowledgement

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

[6] The reference [11] in the quote is the same as [37] in our references.

Appendix A. Relation algebra

Description of binary relations and their basic operations can be found in most introductory courses on discrete mathematics. There are many textbooks were more information can be found (e.g. [4,38]).

A binary (homogeneous) relation R over a set M is a subset of $M \times M$—a set of ordered pairs. Since relations are sets, unions $R \cup S$, intersections $R \cap S$ and complements $\overline{R}$ of relations can be easily defined. In this case the set of all binary relations forms a Boolean algebra. For two relations R and S, the *relational product* $R; S$ is the set of all pairs (x, y) such that there exists a $z \in M$ with $(x, z) \in R$ and $(z, y) \in S$. The *converse* R^T of a relation R is the set of all pairs (y, x) with $(x, y) \in R$. The *identity relation* $\mathbb{I}_M$ on M is the set containing all pairs (x, x) with $x \in M$. The structure $(\mathcal{P}(M \times M), \cup, ;, \overline{\ }, ^\mathsf{T}, \mathbb{I}_M)$ is called *concrete relation algebra* of all binary relations over M. A representation as 0-1-matrix, as done in this paper, is straightforward.

To define relation algebras more abstractly, binary relations are replaced by arbitrary elements of some carrier set M and a set of equational axioms is given. A (*abstract*) *relation algebra* is a structure $(M, +, ;, \overline{\ }, ^\mathsf{T}, \mathbb{I})$ satisfying the axioms

$$(R + S) + T = R + (S + T), \qquad R + S = S + R, \qquad R = \overline{\overline{R} + \overline{S}} + \overline{\overline{R} + S},$$
$$(R; S); T = R; (S; T), \qquad (R + S); T = R; T + S; T, \qquad R; \mathbb{I} = R,$$
$$R^{\mathsf{T}\mathsf{T}} = R, \qquad (R + S)^\mathsf{T} = R^\mathsf{T} + S^\mathsf{T}, \qquad R^\mathsf{T}; \overline{R; S} + \overline{S} = \overline{S}.$$

Since relation algebras are Boolean algebras, they form partially ordered sets with respect to $R \leqslant S \Leftrightarrow R + S = S$; the meet of Boolean algebra is defined as $R \cap S = \overline{\overline{R} + \overline{S}}$, the greatest elements $\top = R + \overline{R}$ and least element $\mathbf{0} = R \cap \overline{R}$.[7]

In this appendix, we freely use well-known facts about relation algebra, such as isotonicity of all operators (except complement, which is antitone). Moreover, if needed we freely switch between matrix representation, sets and algebraic reasoning (whatever is more convenient).

Schmidt and Ströhlein define the concept of points in [25]. A *point* is a relation R satisfying the following three properties.

$$R = R; \top, \qquad R \neq \mathbf{0}, \qquad R; R^\mathsf{T} \subseteq \mathbb{I}.$$

We want to relate the concepts of pre-nodes and points in relation algebra.

Theorem A.1. *In a concrete relation algebra over a non-empty-set M, the concepts of point and node are closely related.*

(1) *R is a point $\Rightarrow$ $R \cap \mathbb{I}_M$ is a node.*
(2) *R is a node $\Rightarrow$ $R; \top_M$ is a point.*[8]

Proof. Assume a concrete relation algebra over a (non-empty) set M.

(1) $R \cap \mathbb{I}_M \subseteq \mathbb{I}_M$ is trivial; since the Tarski rule holds for concrete relation algebra, it remains to show join-irreducibility. In the concrete relation algebra, a point is a matrix with one row of 1's (see e.g. [4]). The intersection with $\mathbb{I}_M$ yields a matrix with exactly one entry equals to 1, which is join-irreducible.
(2) $R; \top_M = R; \top_M; \top_M$ is obvious. By $\top_M; R; \top_M = \top_M$, we have that $R \neq \mathbf{0}$, and hence $R; \top_M \neq \mathbf{0}$. The proof of $(R; \top_M); (R; \top_M)^\mathsf{T} \subseteq \mathbb{I}_M$ on matrix-level (similar as before): by join-irreducibility R has only one entry that is not equal $\mathbf{0}$. Hence $R; \top_M$ is a matrix with one row of 1's, for this, by the definition of matrix multiplication, the claim follows. □

For abstract relation algebra the claims do not hold; automated counter example generators such as Mace4 [27] can generate easily examples.

References

[1] B. Carré, Graphs and Networks, Oxford Applied Mathematics and Computing Science Series, Clarendon Press, 1979.
[2] G. Schmidt, T. Ströhlein, Relationen und Graphen, Mathematik für Informatiker, Springer, 1989.
[3] E. Mayr, G. Schmidt, G. Tinhofer (Eds.), Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, vol. 903, Springer, 1995.
[4] G. Schmidt, T. Ströhlein, Relations and Graphs: Discrete Mathematics for Computer Scientists, Springer, 1993.
[5] G. Schmidt, Relational Mathematics, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2011.
[6] P. Höfner, G. Struth, On automating the calculus of relations, in: A. Armando, P. Baumgartner, G. Dowek (Eds.), Automated Reasoning, in: Lecture Notes in Artificial Intelligence, vol. 5195, Springer, 2008, pp. 50–66.

[7] Schmidt often includes the Tarski rule $a \neq \mathbf{0} \Rightarrow \top; a; \top = \top$ as an axiom, whereas Maddux does not. Here, we use the more general case and skip this rule.
[8] The greatest element in a concrete relation algebra is denoted by $\top_M$.

[7] J.H. Conway, Regular Algebra and Finite Machines, Chapman & Hall, 1971.
[8] D. Kozen, On Kleene algebras and closed semirings, in: MFCS '90: Proceedings on Mathematical Foundations of Computer Science 1990, in: Lecture Notes in Computer Science, Springer, 1990, pp. 26–47.
[9] D. Kozen, On Hoare logic and Kleene algebra with tests, ACM Trans. Comput. Log. 1 (2000) 60–76.
[10] B. Heidergott, G.J. Olsder, J.W. van der Woude, Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus algebra and its Applications, Princeton Series in Applied Mathematics, Princeton University Press, 2006.
[11] W. Kuich, Semirings and formal power series: Their relevance to formal languages and automata, in: Handbook of Formal Languages, vol. 1, Springer, 1997, pp. 609–677.
[12] Y. Kawahara, On the cardinality of relations, in: R. Schmidt (Ed.), Relations and Kleene Algebra in Computer Science, in: Lecture Notes in Computer Science, vol. 4136, Springer, 2006, pp. 251–265.
[13] P. Höfner, B. Möller, Dijkstra, Floyd and Warshall meet Kleene, Form. Asp. Comput. 24 (2012) 459–476.
[14] J. von Wright, Towards a refinement algebra, Sci. Comput. Program. 51 (2004) 23–45.
[15] B. Möller, Modal knowledge and game semirings, Comput. J. 56 (2013) 53–69.
[16] G. Schmidt, Relational concepts in social choice, in: W. Kahl, T. Griffin (Eds.), Relational and Algebraic Methods in Computer Science, in: Lecture Notes in Computer Science, vol. 7560, Springer, 2012, pp. 278–293.
[17] G. Schmidt, R. Berghammer, Relational measures and integration in preference modeling, J. Log. Algebr. Program. 76 (2008) 112–129.
[18] B. Möller, P. Roocks, M. Endres, An algebraic calculus of database preferences, in: J. Gibbons, P. Nogueira (Eds.), Mathematics of Program Construction, in: Lecture Notes in Computer Science, vol. 7342, Springer, 2012, pp. 241–262.
[19] J. Desharnais, B. Möller, G. Struth, Algebraic notions of termination, Log. Methods Comput. Sci. 7 (2011).
[20] W. Guttmann, Unifying lazy and strict computations, in: W. Kahl, T. Griffin (Eds.), Relational and Algebraic Methods in Computer Science, in: Lecture Notes in Computer Science, vol. 7560, Springer, 2012, pp. 17–32.
[21] P. Höfner, A. McIver, Towards an algebra of routing tables, in: H. de Swart (Ed.), Relational and Algebraic Methods in Computer Science, in: Lecture Notes in Computer Science, vol. 6663, Springer, 2011, pp. 212–229.
[22] C. Perkins, E. Belding-Royer, S. Das, Ad hoc on-demand distance vector (AODV) routing, RFC 3561 (Experimental), http://www.ietf.org/rfc/rfc3561.txt, 2003.
[23] R. Backhouse, B. Carré, Regular algebra applied to path-finding problems, J. Inst. Math. Appl. (1975).
[24] D. Kozen, A completeness theorem for Kleene algebras and the algebra of regular events, Inf. Comput. 110 (1994) 366–390.
[25] G. Schmidt, T. Ströhlein, Relation algebras: Concept of points and representability, Discrete Math. 54 (1985) 83–92.
[26] D. Kozen, Kleene algebra with tests, ACM Trans. Program. Lang. Syst. 19 (1997) 427–443.
[27] W.W. McCune, Prover9 and Mace4, http://www.cs.unm.edu/~mccune/prover9 (accessed December 2, 2013).
[28] C. Bolduc, J. Desharnais, Static analysis of programs using omega algebra with tests, in: W. MacCaull, M. Winter, I. Düntsch (Eds.), Relational Methods in Computer Science, in: Lecture Notes in Computer Science, vol. 3929, Springer, 2006, pp. 60–72.
[29] J. Desharnais, B. Möller, G. Struth, Modal Kleene algebra and applications—A survey, J. Relat. Methods Comput. Sci. 1 (2004) 93–131.
[30] J. Desharnais, B. Möller, G. Struth, Kleene algebra with domain, ACM Trans. Comput. Log. 7 (2006) 798–833.
[31] J. Sobrinho, Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet, IEEE/ACM Trans. Netw. 10 (2002) 541–550.
[32] J. Sobrinho, Network routing with path vector protocols: Theory and applications, in: Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '03, ACM Press, 2003, pp. 49–60.
[33] T. Griffin, J. Sobrinho, Metarouting, Comput. Commun. Rev. 35 (2005) 1–12.
[34] B. Carré, Graphs and Networks, Oxford Applied Mathematics & Computing Science Series, Oxford University Press, 1980.
[35] C. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, D. Walker, NetKAT: Semantic foundations for networks, in: Symposium on Principles of Programming Languages (POPL 14), ACM, 2014, pp. 113–126.
[36] H. Leiß, Kleene modules and linear languages, J. Log. Algebr. Program. 66 (2006) 185–194.
[37] I. Copilowish, Matrix development of the calculus of relations, J. Symb. Log. 13 (1948) 193–203.
[38] R. Maddux, Relation Algebras, Studies in Logic and the Foundations of Mathematics, vol. 150, Elsevier, 2006.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 225–234

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Towards "mouldable code" via nested code graph transformation

Wolfram Kahl [1]

McMaster University, Hamilton, Ontario, L8S 4K1, Canada

ARTICLE INFO

Article history:
Available online 12 February 2014

ABSTRACT

Program transformation is currently de facto restricted to abstract syntax tree rewriting. However, many program transformation patterns, in particular in the realm of high-performance code generation, can more naturally be understood and expressed as *graph* transformations. We describe the conceptual organisation of a system based on application of algebraic graph transformation rules to data-flow and control-flow graphs, and outline the work, both theoretical and of implementation nature, that still needs to be done to realise this long-term project.

1. Introduction

An important part of the supporting infrastructure for optimising code generators, in particular in compilers, consists of analyses of graphs, especially control-flow graphs and data-flow graphs. In addition, many of the optimisations enabled by these analyses are then usefully understood as graph transformations.

Interestingly, the program transformation literature almost exclusively concentrates on transformation of (higher-order) abstract syntax *trees*. It is of course customary in a compiler context to parse the given programs into abstract syntax trees, and then extract from these the necessary information to construct the graphs to be used for data-flow and control-flow analyses using attribute grammars, while still considering the abstract syntax trees as the internal representation of the program.

However, many of the transformations that are used for code optimisation, in particular in the back-ends of compilers, act on patterns that can usefully be thought of as *graph* patterns, and the resulting transformations are frequently explained as *graph transformations* applied to the control-flow graph and the data-flow graph, which are usually considered as only *derived* from the programs being transformed.

In this paper, we instead adopt the approach of *representing* (parts of) programs as graphs, in particular as nested graphs with control-flow graphs and data-flow graphs at different levels. Mathematically, we represent all these graphs as directed hypergraphs, and use consistent drawing conventions. In directed hypergraphs, hyperedges (drawn as rectangles) can have multiple input nodes (connected by "input tentacles" drawn as arrows from the nodes to the edges) and multiple output nodes (connected by "output tentacles" drawn as arrows from the edges to the nodes). The graph itself may also have an input/output interface consisting of two lists of nodes; input nodes are indicated by numbered triangles above, and output nodes by numbered triangles below. Fig. 1 shows a data-flow graph (which is therefore acyclic) representing the expression $2 + (7 + (3 + 5 \cdot x) \cdot x) \cdot x$ with x standing for the single input; this evaluates a polynomial at x using Horner's rule. Fig. 2 is

E-mail address: kahl@cas.mcmaster.ca.

[1] This research is supported by the National Science and Engineering Research Council of Canada, NSERC.

http://dx.doi.org/10.1016/j.jlap.2014.02.010
2352-2208/

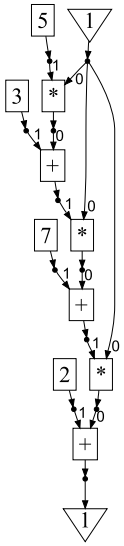


Fig. 1. Data-flow graph.

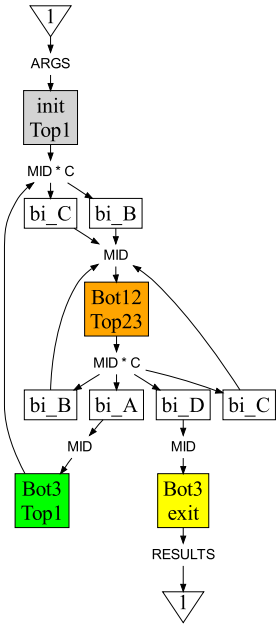


Fig. 2. Control-flow graph.

a control flow graph for a particular way to re-arrange the nested loops of matrix multiplication [1]. Control-flow graphs, like finite automata, have a single start state (drawn as an input), and all hyperedges are actually single-input single-output edges.

1.1. Data-flow-graph transformations

For example, common subexpression elimination can be understood as forming the quotient of a data-flow graph by a congruence relation; "dead-variable elimination" can be understood as transitioning to the sub-graph of a data-flow graph that is reachable from the declared output nodes.

Many cases of "strength reduction", i.e., of replacing special cases of more expensive operations with less expensive operations, are graph rewriting rules that correspond directly to term rewriting rules, such as $2 * x \to \mathsf{shiftLeft}(x, 2)$. However, non-linear rules such as $x + x \to \mathsf{shiftLeft}(x, 2)$ are more typically thought of as local graph rewriting rules where the two arguments of "+" have to be the same node, than as non-linear term or graph rewriting rules where the two sub-terms matched to x are compared for equality or isomorphism (or, more precisely, bisimilarity).

A more intricate situation arises in SIMD-isation: Assume that SIMD arithmetic for vectors of four elements are available, but a given program contains only three independent additions that can be grouped into a vector operation at a certain point. Since that vector operation will always perform four additions, that forth addition, if considered as an individual operation, is technically dead code since its result will not be used. This particular SIMD-isation transformation can therefore be considered as dead-variable introduction, followed by factoring multiple occurrences of isomorphic subgraphs into a single copy using SIMD operations instead of individual operations, again as data-flow graph transformation.

1.2. Control-flow-graph transformations

Many other common optimisation techniques can usefully be understood as control-flow graph transformations, for example:

- Dead-code elimination can be understood as transitioning to the sub-graph of a control-flow graph that is reachable from the declared entry nodes.
- Loop unrolling can be used to reduce the run-time impact of loop overhead; it will typically be a control-flow transformation that also affects the loop control logic, which may therefore also affect some data manipulations.

One of the purposes of many control-flow graph transformations, including loop unrolling, is to enable further data-flow optimisations. For this purpose, it makes sense to consider the edges of the control-flow graph to be labelled with data-flow graphs corresponding to "straight-line code", also called "basic blocks".[2]

[2] Integrating both data-flow and control-flow aspects in a single non-nested graph structure is feasible only for very restricted purposes. In general, one particularly obvious problem is that data-flow "sharing" and control-flow "branching" are incompatible interpretations of multiple outgoing edges.

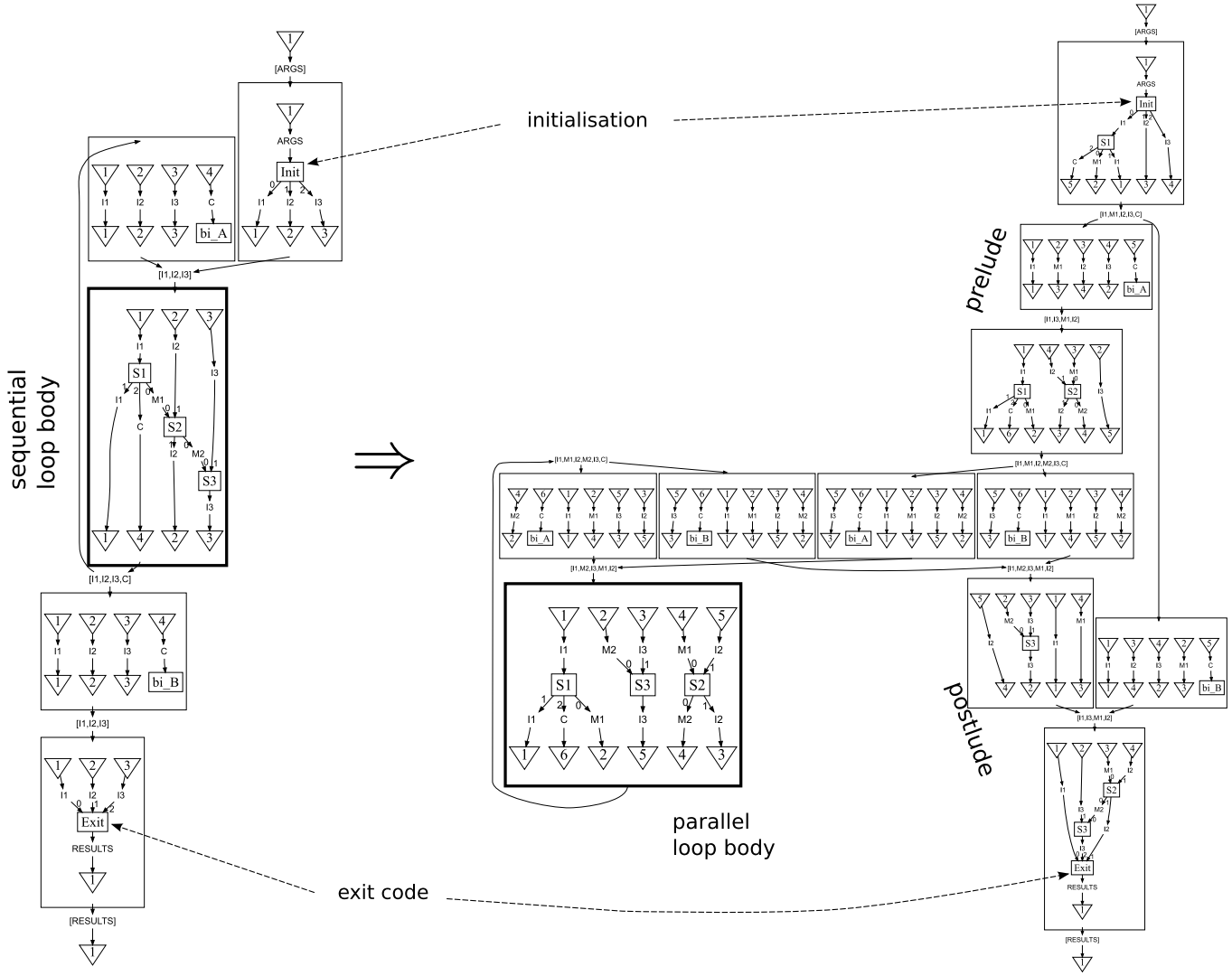


Fig. 3. Software pipelining as nested code graph transformation.

1.3. Combined data- and control-flow-graph transformations

On such nested graphs, more complex transformations can be formulated. One example is software pipelining [2], which can be considered as sequential decomposition of the original loop body followed by parallel reconstitution of the result loop body with adjustments to the control flow and copying of components into a new prelude and postlude, as sketched in Fig. 3.

1.4. Moving to graph representations for program transformation

In this paper, we make the case that formulating program transformations in terms of *graph* transformation is useful not only for explaining these transformations, but also for specifying and implementing them, and for proving them correct. One important goal of this approach is to make the transformational approach to high-performance program development more accessible to domain experts in scientific and high-performance computing, who may not be familiar with the concepts of the current programming-language-centred approaches to program transformation, but who are more likely able to map the control- and data-flow structures of their programs, *understood as graphs and graph patterns*, to the resource models that govern performance of implementations. An interface through which such domain experts can directly leverage their understanding of the resource aspects of their programs and the available computation infrastructure will enable successful special-purpose optimisations far more easily than current approaches concentrating on abstract syntax (tree) transformation strategies.

With this programme, I am striving to bring to fruition Gunther Schmidt's vision of "mouldable code", achieved via graph-based program transformation, that I had the privilege to witness him develop during the last two decades of the 20th century.

1.5. Overview

The next section highlights the history of program transformation, while Section 3 summarises Gunther Schmidt contributions to this field, putting his vision of "mouldable code" into the context of recent "grand challenges". On graph structures it is harder to prove semantics-preservation of transformation rule applications than on (abstract syntax) tree structures; we describe the issues involved in Section 4. In Section 5 we outline the principles guiding our formalisation of nestable code graphs in a way that is usable both also as an implementation, and we present a general approach to monoidal categories of "interfaced objects" in Section 6.

2. Program transformation

The birth of program transformation as a field of research is frequently identified with the early work of Burstall and Darlington, in particular [3], which most importantly introduced the "unfold-fold" style of calculating improved versions of functional programs. Such functional programs, represented typically using different mixtures of term rewriting rules and λ-expressions, remain the starting point for a large part of the literature on program transformation.

One long-running effort in this area was F.L. Bauer's "CIP" project [4–7] trying to realise "Computer-aided Intuition-guided Programming" via program transformation in a wide-spectrum language using conventional string-based syntax. (Within that project, Steinbrüggen [8] came close to inventing second-order rewriting — which was independently defined properly as "Combinatory Reduction Systems" by Klop [9].)

A very similar idea underlies compilation by program transformation, as for example demonstrated by Kelsey and Hudak [10], where the transformations are however applied fully automatically.

In their surveys [11,12] of program transformation, Pettorossi and Proietti concentrate on functional and logic programming, emphasise the need for compositional semantics, and the value of program transformation for "programming in the small". They also point out overlap with program synthesis, and suggest further development of the "rules + strategies" approach of Burstall and Darlington. Paige [13] makes many of the same points, and also argues that program transformation shines in the specialised subfields of functional program derivation, partial evaluation, and finite differencing; although he defines program transformation independent of program representation, he does not appear to seriously consider any alternative to the conventional approach based on abstract syntax trees and λ-expressions.

For the special branch of functional program derivation, the "Algebra of Programming" of Bird and de Moor [14] represents at the same time a polished gemstone of this particular approach, and also a move away from the constraints of programming languages, by embedding their calculational derivations into the context of relation-algebraic categories.

In what appears to be the most recent survey on a program transformation topic [15], Visser discusses the use of directed acyclic graphs (DAGs) as efficient implementation of abstract syntax trees, and mentions a number of approaches that add backlinks to the syntax tree to represent loops in control-flow graphs, but, essentially due to the tree-centred view of the resulting graph structure, has to discard considerations to use this graph structure as the object of transformation. In his more recent work, Visser adopts the motto of "Language definition by transformation" [16].

Looking at the recent research activity in the field of program transformations, the following words of Paige seem to still apply today:

> "In order to create confidence in the products of transformational systems we need to prove correctness of specifications and transformations. Currently, this is too labor intensive to be practical. Progress in computational logic, in languages and systems that support the development and reuse of 'proofware' is a major challenge, that, I believe, must be overcome for a viable program development technology to emerge from a transformational approach." [13]

3. Towards "mouldable code"

While many of his close colleagues were involved in the CIP project mentioned above, Gunther Schmidt formed the impression[3] that program transformation should not be acting on strings or terms, but instead on graph representations, where he was specifically thinking of DAG representations of λ-expressions enriched with variable binding edges. The first incarnation of the "Higher Object Programming System" (HOPS) striving to implement these ideas was presented in [17]. A second implementation followed [18,19], which was also used for first experiments of offering a more explicit data-flow view besides the traditional λ-expression-DAG view [20], and for strategy-driven λ-expression-DAG-based program transformation [21]. A third system added actual term graph layout with sharing, interactive literate programs with code in the form of term graph constructor declarations and term graph transformation rules, and a new approach to integrate the type information into the graph structure [22,23].

Also in the 1990s, Gunther Schmidt started some work on transferring the ideas of attribute grammars to term graphs, which finally resulted in the publication of [24].

[3] Personal communication.

In the landmark paper [25], Hoare uses 17 criteria to define "grand challenge", and proceeds to argue that a "verifying compiler" is an appropriate such grand challenge for current computer science research.

Since then, the first *verified* compiler has been described by Leroy [26], which is however on a different track: as Hoare writes [25]: "A verifying compiler uses mathematical and logical reasoning to check the correctness of the programs that it compiles." Much progress has been made also in this direction, particularly in line with Hoare's emphasis on being able to deal with "a broad selection of legacy code", see for example the KeY project [27] and Frama-C [28]. Use of legacy languages of course brings with it a number of problems, and Hoare concludes: "The long-term solution to these problems is to discard legacy [...] This could well be the topic of different grand challenges. One of these would involve design of a new programming language and compiler, especially designed to support verification."

Krone et al. [29] follow up on Hoare's paper, and make the argument that "thorough, verification-driven language re-design" is actually a necessary subgoal for eventually being able to meet the verifying compiler grand challenge.

I would now like to argue that beyond a language with inherent support for verification, and a compiler that can prove its compiled programs correct, we also need support for correctness-preserving program transformation. This will enable us to direct adaptation of correct code towards different non-functional requirements, in particular to adapt novel combinations of correct components to the particular hardware resource constraints of, for example, some future embedded-system setting.

In the 1990s, Gunther Schmidt frequently summarised his ultimate vision for the HOPS project as realising "mouldable code" (German: *knetbare Programme*), according to my recollection emphasising the following points:

- Programs conceptually structured as graphs,
- Program development is supported by a graph-based GUI,
- Programs are written in a programming language that facilitates correctness proofs,
- Program development is supported by a powerful transformation system that allows power-users to "turn the programs inside out"
  - for the purpose of fusion and other efficiency-improving adaptations,
  - and also for systematically and without impacting correctness adding what would later become known as "aspects" [30].

This "mouldable code" project, understood as including at least combined data-flow and control-flow graph transformations with full mathematical correctness proofs for the transformation mechanisms and for individual transformations, has turned out to be a rather long-term project, with the prototype systems developed so far essentially only exploring particular corners of the resulting design space. Checking the five criteria of Gray [31] for good long-range research goals (which Hoare includes in his grand challenge criteria), I would argue that this "mouldable code" project is clearly "challenging", "useful", and "incremental". It might be disputable whether it satisfies the fourth criterion, namely, whether is "understandable" to the degree that "It is also great to be able to tell your friends and family what you actually do" [31], but with certified programs and proofs recently gaining more widespread attention, the general understandability of such a project should also increase considerably. For the fifth criterion "testable", I would argue that a "simple test" will likely be forthcoming once an actual implementation of a "mouldable code" system is put to use on challenging projects where only a transformational approach can deliver the performance improvement factors that will be necessary for certain computing applications to even become feasible.

4. Semantics of hypergraph code representations

A key ingredient to any trustworthy transformation system is a precise mathematical semantics that comes with useful semantics preservation properties for the transformation mechanism. It is typical to ask for "compositional semantics" for this purpose (see e.g. [12]), which is especially natural where compositional syntax is used, i.e., conventional tree-shaped expression syntax.

Where programs are represented as different kind of graphs, the syntactical structure is not necessarily perceived as compositional in the strict sense anymore, so the more general formulation given above becomes appropriate.

Let us take as a starting point simple acyclic data-flow graphs (many-rooted term graphs), typically representing (tuples of) expressions in a functional programming language, but without variable-binding λ-abstractions.

Gadducci [32,33] showed that these form free so-called "gs-monoidal" categories, which provides us with a concept of compositionality for term graphs considered as syntax, and also provides "semantics for free". (A similar formalisation, taking also cycles into account, was given by Hasegawa [34].)

Transformation of such term graphs naturally strives to use the algebraic approach to graph transformation [35], where graph homomorphisms identify the matchings of the rule sides into the application graphs, and (normally) a rewriting step is constructed as a double-pushout (DPO) diagram (Figs. 4–6).

With term graphs in general, the unmodified DPO approach is not usable, since in the term graph rewriting step, the gluing graph G and the host graph H need to be taken from a "larger" category than the rule sides and application graphs, and for this larger category, the "semantics for free" mentioned above is not available. For the more complicated case of term graphs with λ-binders and cycles, [36] proposed a way of adapting the DPO approach to a "fibred approach" to rewriting [37]. The resulting rewriting concept was proven to preserve the variable-binding invariants, but no attempt was made to

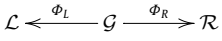


Fig. 4. DPO rule.

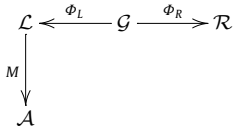


Fig. 5. ... with matching to application graph.

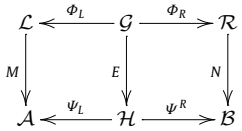


Fig. 6. DPO transformation step.

prove semantics preservation — it appears that even today, and even for the simpler case of term graphs without variable binding, there is still no semantics preservation proof of any adaptation of the DPO to term graph transformation.

For control-flow graphs, the natural theory is that of Kleene algebras (or Kleene categories). Although the category of control-flow graphs is not directly isomorphic a free Kleene algebra, it is isomorphic after quotienting with bisimulation equivalence. In this way, we still have Kleene-algebra "semantics for free", but resolution of DPO graph rewriting issues, in particular violations of the gluing condition, become interesting questions that apparently also have not yet been addressed.

For both data- and control-flow, a natural and easily-understood semantics therefore arises easily from their respective mathematical structures. However, semantics preservation of transformation rules, even when formulated as DPO rule in the categoric graph transformation setting, still poses significant hurdles:

- The proofs of general semantics preservation theorems will likely have to carefully align rule side matching homomorphisms with semantics-compatible decompositions of the application graphs, which will lead to technically rather involved proofs.
- The implementation of these transformation mechanisms should be proven correct, too.

While even combined data- and control-flow graph transformation, including software pipelining, are useful for transforming code intended to execute on single cores, we are also interested in multi-core parallelism [38,39]. The associated concurrent control flow can be modelled using a variant of Petri nets, where transitions are labelled with single-core control flow graphs (which again contain straight-line data-flow graphs as their edge labels). For using this approach to enable modular construction of distributed systems, a promising formalisation are the "open" Petri nets of [40], which have a compositional process semantics. Baldan et al. [41] show semantics-preserving DPO "reconfigurations" for these open Petri nets.

5. Formalising typed directed hypergraphs

The approach to high-performance code generation using graph-based program transformation has already led to first successes in the generation of special function vector libraries [42,43], using a Haskell implementation of the data-flow code graphs described in [44]. We also started work on more complex combined control- and data-flow optimisation [1]. However, due to the increased complexity and size of the graphs we are dealing with, and due to the fact that the Haskell type system, which lacks full dependent types, cannot easily help in ensuring complex nested code graph invariants, we are currently working on formalisations and implementations of code graph transformation infrastructure in the dependently-typed programming language Agda [45], using the RATH-Agda libraries [46,47] for categorical interfaces. We now summarise the basic design decisions guiding this effort.

All three levels of our nested code graphs, as described in the previous section, consist of typed directed hypergraphs with interfaces:

- Data-flow graphs, or term graphs, are most usefully represented using the "jungle" approach of [48,49], where nodes are labelled with types, and directed hyperedges with operations, which are connected (via "source tentacles") to the sequence of nodes representing their arguments, and (via "target tentacles") to the sequence of nodes representing their results. In most contexts, these operations are restricted to producing only one result; the code graphs described in [44] propose to allow multiple results, motivated in particular by the availability of multi-result instructions on some machines, as for example the `div` instructions on MIPS, which return the quotient and the remainder in two result registers. Normally, no node may occur twice in the concatenation of all edge result sequences, and each non-input node has to occur.
  The interface to a data-flow graph consists of a sequence of input nodes, which are not results of any edge, and a sequence of output nodes.
- Our control-flow graphs have data-flow graphs as edge labels, and therefore can naturally be equipped with node labels representing the type of the associated states, as lists of the node types of the input respectively output node sequences of the labels of the incident edges. (For the semantics, this means that we use the typed Kleene algebra of [50], called "Kleene categories" in [51].)
  All edges have exactly one source node and one target node; this implies that the two possible behaviours of branch instructions have to be modelled as two separate edges, as is visible in Fig. 3. All non-start nodes have to have an outgoing edge.
  The interface to a control-flow graph consists of a start state, and an exit state, where the exit state must not be the source node of any control flow edges.

- Our concurrent control-flow graphs are open Petri nets following [40], but with transitions labelled with control-flow graphs.
  Transitions can have multiple source and target nodes, called "places" in the Petri net literature, which usually uses multisets of source and target nodes. Since in our approach, we naturally pass also data when we pass control, we use the type of the passed data to label the places, and organise the source respectively target places of a transition into a sequence, so that the concatenation of their types produces the output respectively input types of the control-flow graphs labelling the incident edges.
  The interface to our concurrent control-flow graphs follows the open Petri net approach, and consists of a set of input places, and a set of output places.

A common data structure that is to be used at all three levels needs to have the following parameters:

- A type of node labels (types)
- Two container types for source and target nodes of edges (singleton or list)
- A type of edge labels, indexed by source and target types
- A recipe for relating edge label type indices with source and target node labels
- Two container types for input and output nodes of graphs.

For being able to let the type system of the implementation language check the type system of the application, that is, that the recipes for relating edge labels with node labels are being followed, a dependently-typed language is indispensable.

Initial experiments with using Agda for term graph formalisations [52] led to the insight that using Sets or Setoids for representing node and edge sets is not practical, since, for example, a pushout algorithm that works for arbitrary Setoids can only produce a Setoid of equivalence classes as result, which is not useful as a data structure for further computations. In order to achieve a reusable implementation that will itself maximally profit from any "mouldable code" inspired transformation infrastructure, we made the design decision to abstract from the implementation categories for sets and functions, respectively for sets and relations. Node and edge sets are therefore presented as objects of a parameter category $\mathcal{C}$.

A natural and typical instance for this parameter has natural numbers as objects, and as morphisms from m to n it has m-element arrays of natural numbers less than n, that is, a programming-oriented category that is equivalent to the category of all finite sets. First steps in the considerable effort required to implement basic reusable categoric operations for $\mathcal{C}$, including pushouts, using also a concrete relation category are described in [53].

For programming on top of this basic layer of abstraction, we will not only have $\mathcal{C}$-objects representing the sets of nodes and edges, but we also need objects representing the set node labels, and the set of node sequences for edge incidence and graph input and output. However, in particular the set of node sequences is not finite. Since also functions starting from this set, including the identity function, are not finite, most datatype representations of functions (and relations) cannot implement a full category interface. Since we typically do not need to represent functions *from* the set of all node sequences, but only functions from finite sets *to* it, we move to semigroupoids as "categories without identities" [47,54]. For our abstract development, we add a semigroupoid $\mathcal{S}$ extension of $\mathcal{C}$ as additional parameter, together with assumptions that $\mathcal{S}$ contains sequence types, etc.

For formulating the edge label consistency conditions at the level of abstraction over $\mathcal{C}$ and $\mathcal{S}$, we propose to use "dependent objects", an abstraction of dependent setoids[4] to the semigroupoid setting related to "type categories" as presented by [58].

Our current formalisation of dependent objects is sufficient to abstract the Setoid-based term graph formalisations of [52], but in order to use code graphs formalised in this way as edge labels for the code graphs of the next level, we need to transfer more of the type category machinery of [58] to the semigroupoid setting, which still is ongoing work.

6. Interfaced directed hypergraphs

Using the DPO approach to graph transformation for rewriting of code graphs suffers, at any level, from the necessity of special treatment for interfaces. One common aspect in all three levels of our code graphs is that the interface to a graph always consists of an input and an output part, and the two are of the same shape. To our knowledge, the formalisation of this set-up in this section is new; it is been completely formalised in Agda.

Definition 6.1. An *interface setting* consists of a category $\mathcal{G}$ of "interface-free" objects, a category $\mathfrak{J}$ of "interfaces", and a functor F from $\mathfrak{J}$ to $\mathcal{G}$.

[4] The concept of dependent setoids apparently is due to Bishop [55], see also Palmgren [56]. Sacerdoti Coen and Tassi write:

"Dependent setoids are types that are defined in a context of variables that range over setoids. A dependent setoid $T(x)$ is a dependent type T over x equipped with a function that maps elements of $T(x)$ to elements of $T(y)$ for all y equivalent to x. Dependent setoids largely complicate the technicalities involved in the use of setoids, since two terms inhabiting respectively $T(x)$ and $T(y)$ for equivalent but different x and y, have different types and thus they cannot even be compared without converting one of them from $T(x)$ to $T(y)$ beforehand. Hence there is a long tradition in type theory of avoiding dependent setoids as much as possible." [57]

For an object i of $\mathfrak{J}$ and an object G of $\mathcal{G}$ we define $\mathsf{i} \mapsto \mathsf{G}$ to be the homset, in $\mathcal{G}$, from $\mathsf{F}(\mathsf{i})$ to G. □

Definition 6.2. Given an interface setting and two objects i and j of $\mathfrak{J}$, we define the category $\mathcal{D}\,\mathsf{i}\,\mathsf{j}$ of "interfaced objects from i to j" as follows:

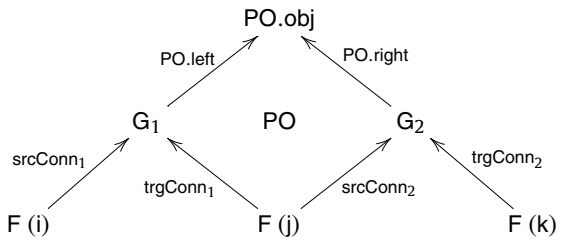
- An object consists of an object G of $\mathcal{G}$, and two morphisms $\mathsf{srcConn} : \mathsf{i} \mapsto \mathsf{G}$ and $\mathsf{trgConn} : \mathsf{j} \mapsto \mathsf{G}$
- A morphism from G_1 to G_2 is a morphism Φ from G_1 to G_2 in $\mathcal{G}$ such that $\mathsf{srcConn}_1 \fatsemi \Phi = \mathsf{srcConn}_2$ and $\mathsf{trgConn}_1 \fatsemi \Phi = \mathsf{trgConn}_2$ in $\mathcal{G}$.
- Composition and identities are carried over from $\mathcal{G}$. □

Applications of the DPO approach to code graphs, including to term graphs, and to open Petri nets in [41], have all horizontal morphisms in the DPO diagram of Fig. 6 taken from $\mathcal{D}$, that is, interface preserving, but the vertical morphisms taken directly from $\mathcal{G}$ and ignoring the interfaces. This heterogeneous set-up imposes additional application conditions that guarantee that the bottom morphisms and graphs all satisfy the additional conditions of $\mathcal{D}$. (This heterogeneous setup can be formalised using the more involved category-theoretic concept of Grothendieck opfibrations, where co-cartesian arrows replace pushouts [37,59].)

On the other hand, the interfaces give rise to external composition principles:

Definition 6.3. Assuming a choice of binary coproducts in $\mathfrak{J}$ which are preserved by F, and a choice of pushouts and a choice of coproducts in $\mathcal{G}$, we define:

- *Sequential composition* of an object D_1 from $\mathcal{D}\,\mathsf{i}\,\mathsf{j}$ and an object D_2 from $\mathcal{D}\,\mathsf{j}\,\mathsf{k}$ produces an object $\mathsf{D}_1 \fatsemi \mathsf{D}_2$ from $\mathcal{D}\,\mathsf{i}\,\mathsf{k}$, constructed via the pushout of $\mathsf{trgConn}_1$ and $\mathsf{srcConn}_2$:



- *Parallel composition* of an object D_1 from $\mathcal{D}\,\mathsf{i}_1\,\mathsf{j}_1$ and an object D_2 from $\mathcal{D}\,\mathsf{i}_2\,\mathsf{j}_2$ produces an object $\mathsf{D}_1 \otimes \mathsf{D}_2$ from $\mathcal{D}\,(\mathsf{i}_1 + \mathsf{i}_2)\,(\mathsf{j}_1 + \mathsf{j}_2)$, constructed componentwise via coproducts. □

Sequentional and parallel composition of Petri nets are not considered in [41]; the composition of open Petri nets there is actually more fine-grained: It composes an object D_1 from $\mathcal{D}\,\mathsf{i}_1\,\mathsf{j}_1$ and an object D_2 from $\mathcal{D}\,\mathsf{i}_2\,\mathsf{j}_2$ via the pushout of an arbitrary span of monos in $\mathcal{G}$, producing an object of $\mathcal{D}\,(\mathsf{i}_1' \cup \mathsf{i}_2')\,(\mathsf{j}_1' \cup \mathsf{j}_2')$ where interfaces may be reduced by being mapped to inner places in the pushout object, and merged by the pushout operation.

Where we do have sequentional and parallel composition, we obtain:

Theorem 6.4. *Let a choice of finite coproducts in $\mathfrak{J}$ which are preserved by F be given, and a choice of pushouts and a choice of coproducts in $\mathcal{G}$.*

Then $\mathcal{D}$ is a symmetric monoidal category defined as follows:

- *Objects are interfaces, the objects of $\mathfrak{J}$.*
- *Morphisms from i to j are isomorphism classes of objects of $\mathcal{D}\,\mathsf{i}\,\mathsf{j}$.*
- *Sequential and parallel composition are as defined in Definition 6.3.*
- *The natural transformations of identity, associativity, and exchange all have morphisms that are "wire isos" in $\mathcal{D}\,\mathsf{i}\,\mathsf{j}$, where* srcConn *and* trgConn *both are isomorphisms* (*taken from the monoidal category structure of coproducts in $\mathfrak{J}$*). □

Adding non-iso wires for duplication and termination, $\mathcal{D}$ can even be extended to a gs-monoidal category, but for jungles we need to add the additional constraint that srcConn is mono; this property is satisfied by all necessary wire objects and also preserved by sequentional and parallel composition.

Without this constraint, we can add co-wires for join to be used either for control flow join, or, as described in [44], for dataflow alternatives.

7. Conclusion and outlook

The monoidal categories of code graphs obtained in the previous section can be used for programmed code graph generation. Another important use is for high-level justification of semantics preservation. At the data-flow level, this will

need to extend the decomposition of jungles as gs-monoidal expressions given in [33] to respect matching morphisms, so that DPO transformation steps can essentially be mapped to term rewriting steps of gs-monoidal expressions. Working out the details for this, and transferring this approach to the control-flow levels is still ongoing work.

In summary, although significant progress has been made on many fronts, it is also obvious that many of the involved theories and tools have been becoming available only fairly recently, and in particular in the field of semantics-preserving graph transformation, significant theoretical work still needs to be done.

Even recently formulated "grand challenges" obviously aim lower than Gunther Schmidt's vision of mouldable code through graph transformation. This clearly shows that he was decades ahead of his time when he started to develop this vision; I am grateful to have become a part of this endeavour.

References

[1] C.K. Anand, W. Kahl, MultiLoop: Efficient software pipelining for modern hardware, in: B. Spencer, M.-A. Storey, D. Stewart (Eds.), Proc. CASCON 2007, IBM Centre for Advanced Studies, Toronto, 2007, pp. 260–263.
[2] V.H. Allan, R.B. Jones, R.M. Lee, S.J. Allan, Software pipelining, ACM Comput. Surv. 27 (1995) 367–432.
[3] R. Burstall, J. Darlington, A transformation system for developing recursive programs, J. ACM 24 (1977) 44–67.
[4] F. Bauer, R. Berghammer, M. Broy, W. Dosch, F. Geiselbrechtinger, R. Gnatz, E. Hangel, W. Hesse, B. Krieg-Brückner, A. Laut, T. Matzner, B. Möller, F. Nickl, H. Partsch, P. Pepper, K. Samelson, M. Wirsing, H. Wössner, The Munich Project CIP. Vol. I: The Wide Spectrum Language CIP-L, LNCS, vol. 183, Springer-Verlag, Berlin/Heidelberg/New York, 1985.
[5] F. Bauer, H. Ehler, A. Horsch, B. Möller, H. Partsch, O. Paukner, P. Pepper, The Munich Project CIP. Vol. II: The Transformation System CIP-S, LNCS, vol. 292, Springer-Verlag, Berlin, 1987.
[6] F. Bauer, B. Möller, H. Partsch, P. Pepper, Formal program construction by transformations — computer-aided, intuition-guided programming, IEEE Trans. Softw. Eng. 15 (1989) 165–180.
[7] M. Broy, M. Wirsing (Eds.), Methods of Programming, Selected Papers of the CIP-project, LNCS, vol. 544, Springer, 1991.
[8] R. Steinbrüggen, The use of nested scheme parameters in the system CIP, in: R. Wilhelm (Ed.), GI — 10. Jahrestagung, Saarbrücken, in: Informatik Fachberichte, vol. 33, Springer-Verlag, 1980, p. 106, extended abstract.
[9] J.W. Klop, Combinatory reduction systems, Mathematical Centre Tracts 127, PhD thesis, Centre for Mathematics and Computer Science, Amsterdam, 1980.
[10] R. Kelsey, P. Hudak, Realistic compilation by program transformation — detailed summary, in: 16th POPL, ACM Press, 1989, pp. 281–292.
[11] A. Pettorossi, M. Proietti, Rules and strategies for transforming functional and logic programs, ACM Comput. Surv. 28 (1996) 360–414.
[12] A. Pettorossi, M. Proietti, Future directions in program transformation, ACM SIGPLAN Not. 32 (1997) 99–102. Position Statement at the Workshop on Strategic Directions in Computing Research, MIT, Cambridge, MA, USA, June 14–15, 1996, http://dx.doi.org/10.1145/251595.251610. Also published in: ACM Comput. Surv. 28 (4es) (December 1996), Article 171, http://dx.doi.org/10.1145/242224.242445.
[13] R. Paige, Future directions in program transformations, SIGPLAN Not. 32 (1997) 94–98.
[14] R.S. Bird, O. de Moor, Algebra of Programming, International Series in Computer Science, vol. 100, Prentice Hall, 1997.
[15] E. Visser, A survey of strategies in rule-based program transformation systems, in: Reduction Strategies in Rewriting and Programming, J. Symb. Comput. 40 (2005) 831–873 (special issue).
[16] L.C.L. Kats, E. Visser, The Spoofax language workbench. Rules for declarative specification of languages and IDEs, in: M. Rinard (Ed.), Proceedings of the 25th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2010, October 17–21, 2010, Reno, NV, USA, 2010, pp. 444–463.
[17] H. Zierer, G. Schmidt, R. Berghammer, An interactive graphical manipulation system for higher objects based on relational algebra, in: G. Tinhofer, G. Schmidt (Eds.), Graph-Theoretic Concepts in Computer Science, WG '86, in: LNCS, vol. 246, Springer, 1986, pp. 68–81.
[18] A. Bayer, W. Kahl, The **H**igher-**O**bject **P**rogramming **S**ystem "HOPS", in: B. Buth, R. Berghammer (Eds.), Systems for Computer-Aided Specification, Development and Verification, Bericht Nr. 9416, Universität Kiel, 1994, pp. 154–171.
[19] A. Bayer, B. Grobauer, W. Kahl, P. Kempf, F. Schmalhofer, G. Schmidt, M. Winter, The Higher Object Programming System HOPS, Technical Report, Inst. für Informatik der Univ. der Bundeswehr München, 1996. Internal Report. 206 pp.
[20] A. Bayer, F. Derichsweiler, HOPS as a link between functional and data-flow oriented programming, in: R. Berghammer, F. Simon (Eds.), Programming Languages and Fundamentals of Programming, Bericht Nr. 9717, Universität Kiel, Institut für Informatik und Praktische Mathematik, 1997, pp. 209–217, Bericht Nr. 9717.
[21] F. Derichsweiler, Strategy Driven Program Transformation within the Higher Object Programming System HOPS, in: A. Poetzsch-Heffter, J. Meyer (Eds.), Programmiersprachen und Grundlagen der Programmierung, Informatik Berichte 263 — 1/2000, FU Hagen, 1999, pp. 165–172.
[22] W. Kahl, Internally typed second-order term graphs, in: J. Hromkovič, O. Sýkora (Eds.), Graph Theoretic Concepts in Computer Science, WG '98, in: LNCS, vol. 1517, Springer, 1998, pp. 149–163.
[23] W. Kahl, The term graph programming system HOPS, in: R. Berghammer, Y. Lakhnech (Eds.), Tool Support for System Specification, Development and Verification, in: Advances in Computing Science, Springer-Verlag, Wien, ISBN 3-211-83282-3, 1999, pp. 136–149.
[24] W. Kahl, F. Derichsweiler, Declarative term graph attribution for program generation, J. Univers. Comput. Sci. 7 (2001) 54–70.
[25] T. Hoare, The verifying compiler: A grand challenge for computing research, J. ACM 50 (2003) 63–69.
[26] X. Leroy, Formal verification of a realistic compiler, Commun. ACM 52 (2009) 107–115.
[27] B. Beckert, R. Hähnle, P.H. Schmitt (Eds.), Verification of Object-Oriented Software: The KeY Approach, LNCS, vol. 4334, Springer-Verlag, 2007.
[28] P. Cuoq, F. Kirchner, N. Kosmatov, V. Prevosto, J. Signoles, B. Yakobowski, Frama-c: a software analysis perspective, in: Proceedings of the 10th International Conference on Software Engineering and Formal Methods, SEFM'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 233–247.
[29] J. Krone, W.F. Ogden, M. Sitaraman, B.W. Weide, Refocusing the verifying compiler grand challenge, Tech. Report RSRG-08-01, School of Computing, Clemson University, Clemson, SC, 2008.
[30] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Christina Lopes, Jean-Marc Loingtier, John Irwin, Aspect-oriented programming, in: M. Aksit, S. Matsuoka (Eds.), ECOOP'97 — Object-Oriented Programming, in: LNCS, vol. 1241, Springer, 1997, pp. 220–242.
[31] J. Gray, What next?: A dozen information-technology research goals, J. ACM 50 (2003) 41–57.
[32] F. Gadducci, On the algebraic approach to concurrent term rewriting, PhD thesis, Università degli studi di Pisa, Dipartimento di Informatica, 1996.
[33] A. Corradini, F. Gadducci, An algebraic presentation of term graphs, via gs-monoidal categories, Appl. Categ. Struct. 7 (1999) 299–331.
[34] M. Hasegawa, Models of sharing graphs, a categorical semantics of let and letrec, PhD thesis, University of Edinburgh, 1997.
[35] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer, Fundamentals of Algebraic Graph Transformation, Springer, 2006.
[36] W. Kahl, Algebraische Termgraphersetzung mit gebundenen Variablen, Reihe Informatik, Herbert Utz Verlag Wissenschaft, München, ISBN 3-931327-60-4, 1996; also Doctoral Diss. at Univ. der Bundeswehr München, Fakultät für Informatik.

[37] W. Kahl, A fibred approach to rewriting — how the duality between adding and deleting cooperates with the difference between matching and rewriting, Technical Report 9702, Fakultät für Informatik, Universität der Bundeswehr München, 1997.
[38] C.K. Anand, W. Kahl, Synthesizing and verifying multicore parallelism in categories of nested code graphs, in: M. Alexander, W. Gardner (Eds.), Process Algebra for Parallel and Distributed Processing, in: CRC Computational Science Series, vol. 2, Chapman & Hall, 2009, pp. 3–45.
[39] M. Dobrogost, C.K. Anand, W. Kahl, Verified multicore parallelism using atomic verifiable operations, in: M.Y. Qadri, S.J. Sangwine (Eds.), Multicore Technology: Architecture, Reconfiguration, and Modeling, CRC Press, 2013, pp. 107–151.
[40] P. Baldan, A. Corradini, H. Ehrig, R. Heckel, Compositional semantics for open Petri nets based on deterministic processes, Math. Struct. Comput. Sci. 15 (2005) 1–35.
[41] P. Baldan, A. Corradini, H. Ehrig, R. Heckel, B. König, Bisimilarity and behaviour-preserving reconfigurations of open Petri nets, in: T. Mossakowski, U. Montanari, M. Haveraaen (Eds.), Algebra and Coalgebra in Computer Science, in: LNCS, vol. 4624, Springer, Berlin, Heidelberg, 2007, pp. 126–142.
[42] C.K. Anand, W. Kahl, An optimized Cell BE special function library generated by Coconut, IEEE Trans. Comput. 58 (2009) 1126–1138.
[43] IBM, Cell BE SDK 3.1, This is the latest version of IBM's Cell SDK and includes 32 single-precision special functions in in-lineable generated C header files, and as a library of long-vector functions scheduled by Coconut, and 32 double-precision special functions in in-lineable generated C header files, and as a library of long-vector functions all generated by Coconut., 2008.
[44] W. Kahl, C.K. Anand, J. Carette, Control-flow semantics for assembly-level data-flow graphs, in: W. McCaull, M. Winter, I. Düntsch (Eds.), 8th Intl. Seminar on Relational Methods in Computer Science, RelMiCS 8, Feb. 2005, in: LNCS, vol. 3929, Springer, 2006, pp. 147–160.
[45] U. Norell, Towards a practical programming language based on dependent type theory, PhD thesis, Department of Computer Science and Engineering, Chalmers University of Technology, 2007.
[46] W. Kahl, Dependently-typed formalisation of relation-algebraic abstractions, in: H. de Swart (Ed.), Relational and Algebraic Methods in Computer Science, RAMiCS 2011, in: LNCS, vol. 6663, Springer, 2011, pp. 230–247.
[47] W. Kahl, Relation-algebraic theories in agda — RATH-Agda-2.0.0, Mechanically checked Agda theories available for download, with 467 pages literate document output, http://RelMiCS.McMaster.ca/RATH-Agda/, 2014.
[48] B. Hoffmann, D. Plump, Implementing term rewriting by jungle evaluation, Inform. Théor. Appl. (Theor. Inform. Appl.) 25 (1991) 445–472.
[49] A. Corradini, F. Rossi, Hyperedge replacement jungle rewriting for term-rewriting systems and logic programming, Theor. Comput. Sci. 109 (1–2) (1993) 7–48.
[50] D. Kozen, Typed Kleene algebra, Technical Report 98-1669, Computer Science Department, Cornell University, 1998.
[51] W. Kahl, Refactoring heterogeneous relation algebras around ordered categories and converse, J. Relat. Methods Comput. Sci. 1 (2004) 277–313.
[52] W. Kahl, Dependently-typed formalisation of typed term graphs, in: R. Echahed (Ed.), Proc. of 6th International Workshop on Computing with Terms and Graphs, TERMGRAPH 2011, in: EPTCS, vol. 48, 2011, pp. 38–53.
[53] W. Kahl, Towards certifiable implementation of graph transformation via relation categories, in: W. Kahl, T.G. Griffin (Eds.), Relational and Algebraic Methods in Computer Science, RAMiCS 2012, in: LNCS, vol. 7560, Springer, 2012, pp. 82–97.
[54] W. Kahl, Relational semigroupoids: Abstract relation-algebraic interfaces for finite relations between infinite types, J. Log. Algebr. Program. 76 (2008) 60–89.
[55] E. Bishop, Foundations of Constructive Analysis, McGraw-Hill, 1967.
[56] E. Palmgren, Constructivist and structuralist foundations: Bishop's and Lawvere's theories of sets, Ann. Pure Appl. Log. 163 (2012) 1384–1399.
[57] C. Sacerdoti Coen, E. Tassi, Formalizing overlap algebras in Matita, Math. Struct. Comput. Sci. 21 (2011) 1–31.
[58] A.M. Pitts, Categorical logic, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), Handbook of Logic in Computer Science, vol. 5, Oxford University Press, 2001, pp. 39–128.
[59] R. Banach, A fibration semantics for extended term graph rewriting, in: M. Sleep, M. Plasmeijer, M. van Eekelen (Eds.), Term Graph Rewriting: Theory and Practice, Wiley, 1993, pp. 91–100.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 235–248

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Arrow's Theorem for incomplete relations

Roger D. Maddux

Department of Mathematics, Iowa State University, United States

ARTICLE INFO

Article history:
Available online 5 March 2014

ABSTRACT

Two theorems are presented that extend Arrow's Theorem to relations that are not necessarily complete. Let U be a set with three or more elements, let $\mathcal{W}$ be the set of weak orderings of U, let $\mathcal{L}$ be the set of linear orderings of U, and let f be an n-ary function mapping $\mathcal{W}^n$ to $\mathcal{W}$. By Arrow's Impossibility Theorem, if f satisfies Arrow's Condition P ("Pareto") and Condition 3 ("independence of irrelevant alternatives") then f violates Arrow's condition of non-dictatorship, which implies that the restriction of f to linear orderings is a projection function, i.e., there is some $k \in \{1, \ldots, n\}$ such that $f(R_1, \ldots, R_n) = R_k$ for all linear orderings $R_1, \ldots, R_n \in \mathcal{L}$. Thus Arrow's Theorem provides a characterization of projection functions on linear orderings as those that satisfy two conditions that clearly hold for projection functions, but which are sufficient (by Arrow's Theorem) to imply a function is a projection function. We prove a similar characterization in Theorem 3 and use it to derive Arrow's characterization for projection functions on linear orderings. Arrow's Theorem does not characterize projection functions on weak orderings, but Theorem 3 does characterize projection functions on a strictly larger class than the weak orderings, namely the transitive co-transitive relations. Along the way we prove, as a consequence of Theorem 2, that if a transitive-valued multivariate relational function f satisfies relational versions of Arrow's Conditions P and 3, and maps equivalence relations on U to transitive relations on U, then for some $D_0 \subseteq \{1, \ldots, n\}$, $f(R_1, \ldots, R_n)$ is just the intersection of all R_i for $i \in D_0$.

1. Introduction

In the spring of 1940, when Alfred Tarski was a visiting professor at the City College of New York, he gave a seminar attended by Kenneth Arrow, who said,

> "It was a great course, Calculus of Relations. His organization was beautiful—I could tell that immediately—and he was thorough. In fact what I learned from him played a role in my own later work—not so much the particular theorems but the language of relations was immediately applicable to economics. I could express my problems in those terms."
> [1, p. 134]

At Tarski's request, Arrow proofread Tarski's book, Introduction to Logic [2].

E-mail address: maddux@iastate.edu.

http://dx.doi.org/10.1016/j.jlap.2014.02.012
2352-2208/

"I also owe many thanks to Mr. K.J. Arrow for his help in reading proofs."

[2, p. xvii]

The part of Tarski's book on the Calculus of Relations was used by Arrow to formulate his problems and prove the Impossibility Theorem [3]. Amartya Sen and Kotaro Suzumura wrote:

"Kenneth Arrow founded the modern form of social choice theory in a path-breaking contribution at the middle of the twentieth century. We—the editors of the Handbook of Social Choice and Welfare other than Arrow—want to begin this final volume by noting the continuing need to read Arrow's decisive contribution in his epoch-making book, Social Choice and Individual Values, which started off the contemporary round of research on social choice theory."

[4, p. 3]

This paper examines Arrow's book [3] for its contribution to the calculus of relations. In particular, Arrow's Theorem provides a characterization of projection functions on linear orderings. Arrow's Theorem is about relations that are both transitive and complete (weak orderings). These properties are appropriate for applications in social welfare theory, but completeness is not essential for many steps in Arrow's proof. Avoiding completeness leads to Theorems 2 and 3. Theorem 3 extends Arrow's characterization of projection functions to a much wider class of relations, while Theorem 2 provides a characterization of functions whose output is the intersection of some particular set of input relations. The proofs use the notion of a decisive set, but there is a special new trick in the proof that supersets of decisive sets are decisive.

The paper is structured as follows. Section 2 starts is a review of the calculus of relations. Arrow's Theorem and characterization of projection functions on linear orderings are presented in Section 3. Arrow's conditions and the variants used here are compared in Sections 4 and 5. The essential properties required of the input sets are isolated in Section 6 in the definitions of "diverse" and "very diverse" sets of relations. All the key lemmas concerning decisive set are gathered in Section 7. The Intersection Theorem 2 and Projection Theorem 3 are presented in Section 8 and Section 9. Finally, the structure of transitive and co-transitive relations is studied in Section 10.

2. Calculus of relations, basic definitions

Let the *universe* U be a set with at least three elements (Schröder's convention).

Definition 1. For any $x, y \in U$, the *ordered pair* $\langle x, y\rangle$ is defined by

$$\langle x, y\rangle = \big\{\{x\}, \{x, y\}\big\}.$$

This particular definition is due to Kuratowski [5], but any definition could be used if it has the following key property.

$$(\forall x, y, u, v)\Big(\big(\langle x, y\rangle = \langle u, v\rangle\big) \Leftrightarrow \big((x = u) \wedge (y = v)\big)\Big).$$

Definition 2. Define the *set of ordered pairs* of elements of U by

$$U^2 = \big\{\langle x, y\rangle\colon\ x, y \in U\big\}.$$

A *relation* is a subset of U^2. $\mathcal{P}(U^2)$ (the set of all subsets of U^2) is the *set of relations* on U. Define the *diversity relation* on U by

$$0' = \big\{\langle x, y\rangle\colon\ x, y \in U,\ x \neq y\big\}$$

and the *identity relation* on U by

$$1' = \big\{\langle x, x\rangle\colon\ x \in U\big\}.$$

We write "xRy" for "$\langle x, y\rangle \in R$" and use other logical notation, such as $\neg$ ("not"), $\wedge$ ("and"), $\vee$ ("or"), $\Rightarrow$ ("implies"), $(\forall x, y \in U)$ ("for all x and y in U"), $\exists$ ("there exists"), etc. Since relations are sets of ordered pairs, they are subject to various operations on sets.

Definition 3. For relations $R, S \subseteq U^2$, their *union* $R \cup S$, *intersection* $R \cap S$, *complement* $\overline{R}$ (with respect to U^2), *converse* R^{-1}, and *relative product* $R;S$ are defined by

$$R \cap S := \big\{\langle x, y\rangle\colon\ xRy \wedge xSy\big\},$$
$$R \cup S := \big\{\langle x, y\rangle\colon\ xRy \vee xSy\big\},$$
$$\overline{R} := \big\{\langle x, y\rangle\colon\ x, y \in U \wedge \neg(xRy)\big\},$$

$$R^{-1} := \{\langle x, y\rangle\colon\ yRx\},$$
$$R;S := \{\langle x, y\rangle\colon\ (\exists z)(xRz \wedge zSy)\}.$$

By Schröder's convention (that U has at least three elements), $0';0' = U^2$. If U has exactly two elements, then $0';0' = 1' \neq U^2$. If U has exactly one element, then $0';0' = 0' = \emptyset \neq 1' = U^2$.

Definition 4. Assume $R \subseteq U^2$. R is

- *reflexive* if $(\forall x \in U)(xRx)$,
- *irreflexive* if $(\forall x \in U)\neg(xRx)$,
- *connected* if $(\forall x, y \in U)(x \neq y \Rightarrow xRy \vee yRx)$,
- *complete* if $(\forall x, y \in U)(xRy \vee yRx)$,
- *symmetric* if $(\forall x, y \in U)(xRy \Rightarrow yRx)$,
- *anti-symmetric* if $(\forall x, y \in U)(xRy \Rightarrow (x = y \vee y\overline{R}x))$,
- *transitive* if $(\forall x, y, z \in U)(xRy \wedge yRz \Rightarrow xRz)$,
- *co-transitive* if $\overline{R}$ is transitive,
- *quasi-transitive* if $R \cap \overline{R^{-1}}$ is transitive,
- an *equivalence relation* if R is transitive, symmetric, and reflexive,
- a *partial ordering* if R is transitive and reflexive,
- a *weak ordering* if R is transitive and complete, and
- a *linear ordering* if R is transitive, complete, and anti-symmetric.

$\mathcal{T}$ is the set of transitive relations on U. $\mathcal{V}$ is the set of relations on U that are both transitive and co-transitive. $\mathcal{W}$ is the set of weak orderings of U. $\mathcal{L}$ is the set of linear orderings of U.

We will make frequent use of the fact that all the following statements are equivalent: R is transitive, R^{-1} is transitive, $R;R \subseteq R$, $R^{-1};R^{-1} \subseteq R^{-1}$, $R^{-1};\overline{R} \subseteq \overline{R}$, $\overline{R};R^{-1} \subseteq \overline{R}$, $\overline{R^{-1}};R \subseteq \overline{R^{-1}}$, and $R;\overline{R^{-1}} \subseteq \overline{R^{-1}}$.

3. Arrow's Theorem

In Arrow's application of the calculus of relations to social welfare, U is a set of alternative social states, $\{1, \ldots, n\}$ is society, and each member i of society chooses a weak ordering R_i of the alternative social states.

> "Definition 4: By a *social welfare function* will be meant a process or rule which, for each set of individual [weak] orderings $R_1, \ldots, R_n$ for alternative social states (one [weak] ordering for each individual), states a corresponding social [weak] ordering of alternative social states, R."
>
> [3, p. 23]

In our notation, a social welfare function is a function f that takes arbitrary inputs $R_1, \ldots, R_n$ from $\mathcal{W}$, the set of weak orderings of U, and produces a corresponding output $R = f(R_1, \ldots, R_n)$ in $\mathcal{W}$. Arrow states a problem and proves it has no solution.

> "The question raised is that of constructing a social ordering of all conceivable alternative social states from any given set of individual orderings of those social states, the method of construction being in accordance with the value judgments of citizens' sovereignty and rationality as expressed in Conditions 1–5."
>
> [3, p. 31]

> "In the following proof we assume a given social welfare function satisfying Conditions 1–5 and show that the assumption leads to a contradiction."
>
> [3, p. 51]

In the 1963 edition [3], Arrow was able to

> "show that some of the conditions can be replaced by the Pareto principle.... Since the Pareto principle is universally accepted, the new set of conditions will be easier to compare with other formulations of the problem of social choice."
>
> [3, p. 96]

The Pareto principle is Condition P below. Arrow replaced Conditions 1 and 2 by the stronger Condition 1′, "[t]o meet an important objection raised by Blau," who found "an error in the original statement of the theorem" [6].

"Condition 1′: *All logically possible orderings are admissible*."

[3, p. 97]

Condition 1′ says the domain of f is $\mathcal{W}^n$, the n-fold Cartesian product of $\mathcal{W}$. The 1963 version of the theorem reads,

"Theorem 2: *Conditions* 1′, 3, P, *and* 5 *are inconsistent*."

[3, p. 97]

To state Conditions 3, P, and 5 we need to first retrace how Arrow arrives at the use of weak orderings.

"It is assumed further that … the chooser considers in turn all possible pairs of alternatives, say x and y, and for each such pair he makes one and only one of three decisions: x is preferred to y, x is indifferent to y, or y is preferred to x. The decisions made for different pairs are assumed to be consistent with each other, so, for example, if x is preferred to y and y to z then x is preferred to z; similarly, if x is indifferent to y and y to z, then x is indifferent to z."

[3, p. 12^{9-15}]

Let P be the relation 'is preferred to', and I the relation 'is indifferent to'. In the first sentence above Arrow assumes that P, I, and P^{-1} are pairwise disjoint relations that apply to all pairs of alternatives, that is,

$$\emptyset = P \cap I = P \cap P^{-1} = I \cap P^{-1}, \tag{1}$$

$$U^2 = P \cup I \cup P^{-1}, \tag{2}$$

and in the second sentence he assumes that P and I are transitive, i.e.,

$$P;P \subseteq P, \tag{3}$$

$$I;I \subseteq I. \tag{4}$$

"Instead of working with two relations, it will be slightly more convenient to use a single relation 'preferred or indifferent.'"

[3, p. 12]

In current notation, this amounts to making the following definition.

$$R = P \cup I. \tag{5}$$

Arrow restates some of his earlier assumptions as two axioms concerning R.
Arrow's Axiom I. R is complete. ("*For all x and y, either xRy or yRx*." [3, p. 13])
Arrow's Axiom II. R is transitive. ("*For all x, y, and z, xRy and yRz imply xRz*." [3, p. 13])

"A relation satisfying both Axioms I and II is termed a weak ordering or sometimes simply an ordering."

[3, p. 13]

The set of relations on U satisfying Axioms I and II is $\mathcal{W}$. Every ordering $R \in \mathcal{W}$ is transitive, but

"…we ordinarily feel that not only the relation R but also the relations of (strict) preference and of indifference are transitive. We shall show that, by defining preference and indifference suitably in terms of R all the usually desired properties of preference patterns obtain."

[3, p. 14^{6-7}]

Arrow's Definition 1. $P = \overline{R^{-1}}$. ("*xPy is defined to mean not yRx*." [3, p. 14])
Arrow's Definition 2. $I = R \cap R^{-1}$. ("*xIy means xRy and yRx*." [3, p. 14])

Arrow could have started with P and I satisfying (1)–(4), defined R by (5), and proved Axioms I and II and Definitions 1 and 2, but he chose instead to start with a complete transitive relation R, define P and I by Definitions 1 and 2, and prove (1)–(4). The inputs R_i and the output $R = f(R_1, \ldots, R_n)$ of the social welfare function f should be weak orderings, because

"…it will be assumed that individuals are rational, by which is meant that the ordering relations R_i satisfy Axioms I and II. The problem will be to construct an ordering relation R for society as a whole that will also reflect rational choice-making so R may also be assumed to satisfy Axioms I and II."

[3, p. 19^{15-19}]

Every social welfare function f determines a choice function C according to Arrow's Definition 3: If $S \subseteq U$, then

$$C(S) = \{x\colon\ x \in S \wedge (\forall y \in S)(xf(R_1, \ldots, R_n)y)\}.$$

("$C(S)$ *is the set of all alternatives* x *in* S *such that, for every* y *in* S, xRy." [3, p. 15])

> "Condition 3: *Let* $R_1, \ldots, R_n$ *and* $R'_1, \ldots, R'_n$ *be two sets of individual orderings and let* C *and* C' *be the corresponding social choice functions. If, for all individuals* i *and all* x *and* y *in a given environment* S, xR_iy *if and only if* xR'_iy, *then* $C(S)$ *and* $C'(S)$ *are the same*."
>
> [3, p. 27]

In Conditions P and 5, the strict preference relation of individual i is $P_i = \overline{R_i^{-1}}$.

> "Condition P: *If* xP_iy *for all* i, *then* xPy."
>
> [3, p. 96]

> "Definition 6: *A social welfare function is said to be* dictatorial *if there exists an individual* i *such that, for all* x *and* y, xP_iy *implies* xPy *regardless of the orderings* $R_1, \ldots, R_n$ *of all individuals other than* i, *where* P *is the social preference relation corresponding to* $R_1, \ldots, R_n$."
>
> [3, p. 30]

> "Condition 5: *The social welfare function is not to be dictatorial* (non-dictatorship)."
>
> [3, p. 30]

Arrow's Theorem 2 can be formally restated (entirely in terms of R_i) as follows.

Theorem 1 *(Arrow's Theorem). Assume* $\mathcal{W}$ *is the set of weak orderings of a set* U *with at least three elements,* $2 \leqslant n < \omega$, *and* $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$. *If* f *maps* $\mathcal{W}^n$ *into* $\mathcal{W}$ (*Condition* 1′), f *satisfies Arrow's Condition* P,

$$(\forall R_1, \ldots, R_n \in \mathcal{W})\ (\forall x, y \in U)\quad \Bigg(\bigwedge_{1 \leqslant i \leqslant n} \big(x\overline{R_i^{-1}}y\big) \Rightarrow x\overline{\big(f(R_1, \ldots, R_n)\big)^{-1}}y\Bigg),$$

and f *satisfies Arrow's Condition* 3,

$$(\forall R_1, \ldots, R_n,\ R'_1, \ldots, R'_n \in \mathcal{W})\ (\forall S \subseteq U)$$
$$\Bigg((\forall x, y \in S)\Bigg(\bigwedge_{1 \leqslant i \leqslant n} \big(xR_iy \Leftrightarrow xR'_iy\big)\Bigg) \Rightarrow C(S) = C'(S)\Bigg),$$

then f *violates Condition* 5: *there is some* $k \in \{1, \ldots, n\}$ (*called a "dictator"*) *such that*

$$(\forall R_1, \ldots, R_n \in \mathcal{W})\ (\forall x, y \in U)\quad \big(x\overline{R_k^{-1}}y \Rightarrow x\overline{\big(f(R_1, \ldots, R_n)\big)^{-1}}y\big). \tag{6}$$

Arrow's Theorem does not characterize functions on weak orderings, imposing only the restriction in (6) that, for some $k \in \{1, \ldots, n\}$, $f(R_1, \ldots, R_n) \subseteq R_k$ for all weak orderings $R_1, \ldots, R_n \in \mathcal{W}$. However, if R_k is a linear ordering then equality does hold because $f(R_1, \ldots, R_n)$ must be complete and no proper subset of a linear ordering is a complete relation. Hence the following conclusion can be added to Arrow's Theorem.

$$(\forall R_1, \ldots, R_n \in \mathcal{L})\quad \big(f(R_1, \ldots, R_n) = R_k\big). \tag{7}$$

In Section 9 we prove (7) from the hypotheses of Theorem 1 by applying the Projection Theorem 3, which uses a relational version of Arrow's Condition 3 to characterize projection functions on the class of transitive co-transitive relations, thus extending Arrow's characterization from the linear orderings to a class strictly larger than the class of weak orderings.

4. Condition 3 and independence

It has long been common to formulate variations on Arrow's Condition 3 in purely relational terms without reference to the choice function C. This may well be due to the fact that, since $x \in C(S)$ implies $xf(R_1, \ldots, R_n)x$, every multivariate relational operator $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$ that has only irreflexive outputs satisfies Condition 3 simply because $C(S) = \emptyset$ for all $S \subseteq U$.

Definition 5. Assume $2 \leqslant n < \omega$, $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$, and $\mathcal{R} \subseteq \mathcal{P}(U^2)$. f satisfies **IIA** on $\mathcal{R}$ if

$$(\forall R_1, \ldots, R_n,\ R'_1, \ldots, R'_n \in \mathcal{R})\ (\forall x, y \in U) \tag{8}$$
$$\left(x \neq y \wedge \bigwedge_{1 \leqslant i \leqslant n} \left(xR_i y \Leftrightarrow xR'_i y \right) \Rightarrow \left(xf(R_1, \ldots, R_n) y \Leftrightarrow xf\left(R'_1, \ldots, R'_n\right) y \right) \right).$$

The next lemma shows the relation between **IIA** and Condition 3. It will be needed in the proof of (7) in Section 9.

Lemma 1.

1. ***IIA*** *on* $\mathcal{W}$ *implies Condition* 3.
2. *Condition* 3 *implies* ***IIA*** *on* $\mathcal{L}$.
3. *Condition* 3 *does not imply* ***IIA*** *on* $\mathcal{W}$.

Proof. It is easy to prove that **IIA** on $\mathcal{W}$ implies Condition 3. For the second part, suppose $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$, f maps weak orderings to weak orderings ($\mathcal{W}^n$ into $\mathcal{W}$), and Condition 3 holds. We will prove that f satisfies **IIA** on $\mathcal{L}$ using only the restriction of Condition 3 to 2-element sets $S \subseteq U$. First, since $\mathcal{L} \subseteq \mathcal{W}$, we get a consequence of Condition 3 by replacing $\mathcal{W}$ with $\mathcal{L}$. Then we let $S = \{x, y\}$ with $x \neq y$ in Condition 3, we replace the concluding equation $C(\{x, y\}) = C'(\{x, y\})$ with the equivalent formula

$$\left((xRx \wedge xRy) \Leftrightarrow \left(xR'x \wedge xR'y \right) \right) \wedge \left((yRx \wedge yRy) \Leftrightarrow \left(yR'x \wedge yR'y \right) \right)$$

where $R = f(R_1, \ldots, R_n)$ and $R' = f(R'_1, \ldots, R'_n)$, and obtain

$$(\forall R_1, \ldots, R_n,\ R'_1, \ldots, R'_n \in \mathcal{L})\ (\forall x, y \in U)$$
$$\left(x \neq y \wedge \bigwedge_{1 \leqslant i \leqslant n} \underline{\left((xR_i x \Leftrightarrow xR'_i x) \right.} \wedge \left(xR_i y \Leftrightarrow xR'_i y \right) \wedge \left(yR_i x \Leftrightarrow yR'_i x \right) \underline{\left. \wedge \left(yR_i y \Leftrightarrow yR'_i y \right) \right)} \right.$$
$$\left. \Rightarrow \left((\underline{xRx \wedge} xRy) \Leftrightarrow \left(\underline{xR'x \wedge} xR'y \right) \right) \wedge \left((yRx \underline{\wedge yRy}) \Leftrightarrow \left(yR'x \underline{\wedge yR'y} \right) \right) \right)$$

Linear orderings are reflexive, and the relations R and R' are also reflexive by the assumptions on f, so the underlined portions can be deleted (because the formulas in them are always true), leaving

$$(\forall R_1, \ldots, R_n,\ R'_1, \ldots, R'_n \in \mathcal{L})\ (\forall x, y \in U)$$
$$\left(x \neq y \wedge \bigwedge_{1 \leqslant i \leqslant n} \underline{\left(xR_i y \Leftrightarrow xR'_i y \right)} \wedge \underline{\left(yR_i x \Leftrightarrow yR'_i x \right)} \right) \Rightarrow \left(xRy \Leftrightarrow xR'y \right) \wedge \left(yRx \Leftrightarrow yR'x \right) \Big)$$

The underlined formulas are equivalent because the input relations are linear orderings. To see this, use $\breve{L} \cap 0' = L^{-1} \cap 0'$ for all $L \in \mathcal{L}$ in the third step:

$$\begin{array}{lllll} xR_i y \Leftrightarrow xR'_i y & \text{iff} & x\overline{R_i} y \Leftrightarrow x\overline{R'_i} y & & \text{by logic} \\ & \text{iff} & xR_i^{-1} y \Leftrightarrow xR_i'^{-1} y & & R_i, R'_i \in \mathcal{L},\ x \neq y \\ & \text{iff} & yR_i x \Leftrightarrow yR'_i x & & \end{array}$$

We may delete the second underlined formula (because it is redundant), delete the final conjunct (making the condition no stronger), and get a consequence of Condition 3 which is, after eliminating abbreviations, exactly **IIA** on $\mathcal{L}$. Finally, for an example to show that Condition 3 does not imply **IIA** on $\mathcal{W}$, let

$$f(R_1, R_2, \ldots, R_n) = R_1^{-1}$$

for all $R_1, R_2, \ldots, R_n \in \mathcal{P}(U^2)$. Since f preserves transitivity and completeness, it maps weak orderings to weak orderings. We will show Condition 3 holds for f. Assume $R_1, \ldots, R_n,\ R'_1, \ldots, R'_n \in \mathcal{W}$, $S \subseteq U$, and, for every $i \in \{1, \ldots, n\}$,

$$R_i \cap S^2 = R'_i \cap S^2. \tag{9}$$

We wish to show that

$$f(R_1, \ldots, R_n) \cap S^2 = f\left(R'_1, \ldots, R'_n\right) \cap S^2, \tag{10}$$

because (10) implies $C(S) = C'(S)$. By the definition of f, (10) is equivalent to

$$R_1^{-1} \cap S^2 = R_1'^{-1} \cap S^2. \tag{11}$$

We get (11) from (9) by setting $i = 1$ and applying $^{-1}$ to both sides. Thus Condition 3 holds. To see that f fails to satisfy **IIA** on $\mathcal{W}$, choose distinct $x, y \in U$ and choose weak orderings $R_1, \ldots, R_n, R_1', \ldots, R_n' \in \mathcal{W}$, so that xR_1y, $xR_1'y$, yR_1x, $y\overline{R_1'}x$, and $R_i = R_i'$ for $2 \leqslant i \leqslant n$. (Note that $R_1 \notin \mathcal{L}$.) Then

$$R_i \cap \{\langle x, y\rangle\} = R_i' \cap \{\langle x, y\rangle\} \tag{12}$$

for all $i \in \{1, \ldots, n\}$, so the hypothesis of the implication in **IIA** holds, but its conclusion fails, i.e.,

$$f(R_1, \ldots, R_n) \cap \{\langle x, y\rangle\} \neq f(R_1', \ldots, R_n') \cap \{\langle x, y\rangle\},$$

because $xR_1^{-1}y$ but not $xR_1'^{-1}y$. □

5. Condition P and unanimity

Definition 6. Assume $\mathcal{R} \subseteq \mathcal{P}(U^2)$, $2 \leqslant n < \omega$, and $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$. f satisfies **Unan** on $\mathcal{R}$ if

$$(\forall R_1, \ldots, R_n \in \mathcal{R})\ (\forall x, y \in U)$$
$$\left(\left(x \neq y \wedge \bigwedge_{1 \leqslant i \leqslant n} (xR_iy)\right) \Rightarrow xf(R_1, \ldots, R_n)y\right). \tag{13}$$

Note that **Unan** is equivalent to

$$(\forall R_1, \ldots, R_n \in \mathcal{R}) \quad \left(\bigcap_{1 \leqslant i \leqslant n} R_i \cap 0' \subseteq f(R_1, \ldots, R_n)\right),$$

while Condition P (applied to $\mathcal{R}$ instead of $\mathcal{W}$) is equivalent to

$$(\forall R_1, \ldots, R_n \in \mathcal{R}) \quad \left(f(R_1, \ldots, R_n) \cap 0' \subseteq \bigcup_{1 \leqslant i \leqslant n} R_i\right).$$

6. Diverse sets of relations

Definition 7. A set $\mathcal{R} \subseteq \mathcal{P}(U^2)$ of relations on U is *diverse* if (14)–(17) hold, and *very diverse* if (14)–(19) hold.

$$(\forall x, y, z \in U) \quad \left(|\{x, y, z\}| = 3 \Rightarrow (\exists R \in \mathcal{R})(x\overline{R}y \wedge x\overline{R}z \wedge zRy)\right), \tag{14}$$
$$(\forall x, y, z \in U) \quad \left(|\{x, y, z\}| = 3 \Rightarrow (\exists R \in \mathcal{R})(xRy \wedge xRz \wedge zRy)\right), \tag{15}$$
$$(\forall x, y, z \in U) \quad \left(|\{x, y, z\}| = 3 \Rightarrow (\exists R \in \mathcal{R})(x\overline{R}y \wedge x\overline{R}z \wedge zRy)\right), \tag{16}$$
$$(\forall x, y, z \in U) \quad \left(|\{x, y, z\}| = 3 \Rightarrow (\exists R \in \mathcal{R})(x\overline{R}y \wedge xRz \wedge z\overline{R}y)\right), \tag{17}$$
$$(\forall x, y, z \in U) \quad \left(|\{x, y, z\}| = 3 \Rightarrow (\exists R \in \mathcal{R})(xRy \wedge x\overline{R}z \wedge zRy)\right), \tag{18}$$
$$(\forall x, y, z \in U) \quad \left(|\{x, y, z\}| = 3 \Rightarrow (\exists R \in \mathcal{R})(xRy \wedge xRz \wedge z\overline{R}y)\right). \tag{19}$$

Conditions (14)–(19) are part of a set of eight conditions obtained by filling in the blanks of the pattern $x\_\_y \wedge x\_\_z \wedge z\_\_y$ with either "R" or "$\overline{R}$". Six of these conditions are required for a set $\mathcal{R}$ of relations to be very diverse. The remaining two conditions are

$$(\forall x, y, z \in U) \quad \left(|\{x, y, z\}| = 3 \Rightarrow (\exists R \in \mathcal{R})(xRy \wedge x\overline{R}z \wedge z\overline{R}y)\right),$$
$$(\forall x, y, z \in U) \quad \left(|\{x, y, z\}| = 3 \Rightarrow (\exists R \in \mathcal{R})(x\overline{R}y \wedge xRz \wedge zRy)\right).$$

These conditions say for every 3-element subset of U there is a relation in $\mathcal{R}$ that fails to be co-transitive on that subset, and a relation in $\mathcal{R}$ that fails to be transitive on that subset. These are the only two conditions (out of eight) that do not hold for the set $\mathcal{V}$ of transitive co-transitive relations on U, so $\mathcal{V}$ is very diverse. $\mathcal{V}$ is examined more closely in the last section.

The set of equivalence relations on U is diverse. In fact, if $\mathcal{R}$ is the set consisting of U^2, 1', and the equivalence relations on U with exactly two equivalence classes, then $\mathcal{R}$ is diverse. However, the set of equivalence relations on U is not very diverse, because if R is an equivalence relation then $xRy \wedge zRy$ implies xRz, contrary to what is required in (18), and $xRy \wedge xRz$ implies zRy, contrary to (19). Thus both (18) and (19) fail for any set of equivalence relations. Similarly, $\{0'\} \cup \{\{\langle x, y\rangle\}\colon\ x, y \in U, x \neq y\}$ is diverse but not very diverse.

For any $x, y, z \in U$, let $xyz = \{\langle x, z\rangle, \langle x, y\rangle, \langle y, z\rangle\} \in U^2$, and let $\mathcal{R} = \{xyz\colon\ x, y, z \in U, |\{x, z, y\}| = 3\}$. Then $\mathcal{R}$ contains only transitive relations and is very diverse. To prove this, consider any three elements $x, y, z \in U$. For (14) let $R = yzx$, for (15) let $R = xzy$, for (16) let $R = zyx$, for (17) let $R = yxz$, for (18) let $R = zxy$, and for (19) let $R = xyz$. $\mathcal{R}$ is a very diverse set of transitive relations, but $\mathcal{R}$ contains relations that are not co-transitive, so $\mathcal{R} \nsubseteq \mathcal{V}$. Let

$$\mathcal{R}' = \Big\{xyz \cup \big(\{x, y, z\} \times \big(U \cap \overline{\{x, y, z\}}\big)\big)\colon\ x, y, z \in U,\ \big|\{x, z, y\}\big| = 3\Big\}.$$

Then $\mathcal{R}'$ is very diverse and $\mathcal{R} \subseteq \mathcal{V}$. We can add 1' to every set in $\mathcal{R}'$ and get a very diverse set of transitive, co-transitive, reflexive relations. By making them a bit bigger we get a very diverse set of weak orderings. The set of linear orderings with x, y, z "at the bottom" (or "at the top") is very diverse.

7. Decisive sets

Throughout this section we make the assumptions of Theorem 2: U is a set with at least three elements, $2 \leqslant n < \omega$, $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$, $\mathcal{R}$ is a diverse set of relations on U, f maps $\mathcal{R}^n$ into $\mathcal{T}$, and f satisfies **Unan** and **IIA** on $\mathcal{R}$. Arrow's (revised) notion of decisive set [3, p. 98], originally stated for preference relations, is formulated here for relations in general.

Definition 8. Assume $D \subseteq \{1, \ldots, n\}$ and $x, y \in U$. D is *decisive for* $\langle x, y\rangle$ if

$$(\forall R_1, \ldots, R_n \in \mathcal{R}) \quad \big(D = \{i\colon\ xR_i y\} \Rightarrow xf(R_1, \ldots, R_n)y\big). \tag{20}$$

Let $[D]$ be the set of pairs $\langle x, y\rangle$ of distinct elements of U such that D is decisive for $\langle x, y\rangle$. D is *decisive* if $[D] \cap 0' \neq \emptyset$.

By **Unan**, $[\{1, \ldots, n\}] = 0'$, so $\{1, \ldots, n\}$ is decisive. We'll show now that $[D] \cap 0'$ is either $\emptyset$ or $0'$, and the decisive sets form a filter in the Boolean algebra of all subsets of $\{1, \ldots, n\}$.

Lemma 2. *For distinct $x, y, z \in U$, if $x[D]z$ and $z[E]y$ then $x[D \cap E]y$.*

Proof. Assume $|\{x, y, z\}| = 3$, $x[D]z$, $z[E]y$, and $R_1, \ldots, R_n \in \mathcal{R}$. To show that $D \cap E$ is decisive for $\langle x, z\rangle$, we assume $D \cap E = \{i\colon\ xR_i y\}$ and prove that $xf(R_1, \ldots, R_n)y$. By (14), (15), (16), and (17), there are relations $R_{(14)}, R_{(15)}, R_{(16)}, R_{(17)} \in \mathcal{R}$, respectively, with the properties listed below, from which we choose $R'_1, \ldots, R'_n \in \{R_{(14)}, \ldots, R_{(17)}\}$ as follows:

$x\overline{R_{(14)}}y$	$x\overline{R_{(14)}}z$	$z\overline{R_{(14)}}y$	$R'_i = R_{(14)}$	if $i \in \overline{D \cap E}$
$xR_{(15)}y$	$xR_{(15)}z$	$zR_{(15)}y$	$R'_i = R_{(15)}$	if $i \in D \cap E$
$x\overline{R_{(16)}}y$	$x\overline{R_{(16)}}z$	$zR_{(16)}y$	$R'_i = R_{(16)}$	if $i \in \overline{D} \cap E$
$x\overline{R_{(17)}}y$	$xR_{(17)}z$	$z\overline{R_{(17)}}y$	$R'_i = R_{(17)}$	if $i \in D \cap \overline{E}$

Let $R = f(R_1, \ldots, R_n)$ and $R' = f(R'_1, \ldots, R'_n)$. Then we have $xR'z$ by $x[D]z$ (see the second and fourth columns) and $zR'y$ by $z[E]y$ (see the third and fourth columns), so $xR'y$ since outputs of f are transitive. We have arranged that $D \cap E = \{i\colon\ xR'_i y\}$ (see the first and fourth columns), but we assumed $D \cap E = \{i : xR_i y\}$, so $\bigwedge_{1 \leqslant i \leqslant n}(xR_i y \Leftrightarrow xR'_i y)$. It follows, by **IIA** and $xR'y$, that xRy, as desired. □

Lemma 3. *If $x, y, z \in U$ are distinct, $D \subseteq \{1, \ldots, n\}$, and $x[D]y$, then $u[D]v$ for all distinct $u, v \in \{x, y, z\}$.*

Proof.

1.	$x[D]y$	hypothesis
2.	$y\big[\{1, \ldots, n\}\big]x$	**Unan**
3.	$z\big[\{1, \ldots, n\}\big]x$	**Unan**
4.	$z[D]y$	1, 3, Lemma 2
5.	$z[D]x$	4, 2, Lemma 2
6.	$y\big[\{1, \ldots, n\}\big]z$	**Unan**
7.	$x[D]z$	1, 6, Lemma 2

8. $y[D]z$ 2, 7, Lemma 2
9. $y[D]x$ 5, 8, Lemma 2

In steps 1, 4, 5, 7–9, we have assumed or proved $u[D]v$ in all six cases that arise by choosing distinct $u, v \in \{x, y, z\}$. □

Lemma 4. *If D is decisive then $[D] = 0'$.*

Proof. Let $D \subseteq \{1, \ldots, n\}$ be a decisive subset. Since D is decisive, there must be distinct $x, y \in U$ such that $x[D]y$. Assume $u, v \in U$ and $u \neq v$. We wish to show $u[D]v$. Then $|\{x, y, u, v\}| > 1$ since $x \neq y$. If $|\{x, y, u, v\}|$ is 2 or 3, we may choose $z \in U$ distinct from x, y such that $\{x, y, u, v\} \subseteq \{x, y, z\}$. Such an element z is either u, v, or exists by the assumption that U has at least three elements. Then $|\{x, y, z\}| = 3$, $x[D]y$, $u, v \in \{x, y, z\}$, and $u \neq v$, hence $u[D]v$ by Lemma 3. We therefore may assume $|\{x, y, u, v\}| = 4$. From Lemma 3, applied to the three-element sets $\{x, y, u\}$ and $\{x, y, v\}$, we have $u[D]x$ and $x[D]v$, hence $u[D]v$ by Lemma 2. □

Lemma 5. *Intersections of decisive sets are decisive.*

Proof. Assume $D, E \subseteq \{1, \ldots, n\}$ are both decisive. Let $x, y, z \in U$ be three distinct elements. Then $x[D]y$ and $y[E]z$ since $[D] = [E] = 0'$ by Lemma 4, so $x[D \cap E]z$ by Lemma 2, hence $D \cap E$ is decisive. □

The proof of the next lemma has a new trick, a division into two cases that later converge. This allows us to avoid the last two diversity conditions (the two needed for a set of relations to be "very" diverse), and extend the theorem to diverse sets, in particular the set of equivalence relations.

Lemma 6. *Supersets of decisive sets are decisive.*

Proof. Assume $D \subseteq E \subseteq \{1, \ldots, n\}$ and D is decisive. Then there are $y, z \in U$ such that $y[D]z$ and $y \neq z$. Since U has at least three elements, we may choose $x \in U$ distinct from y, z. We will prove $x[E]y$, which implies that E is decisive. Assume $R_1, \ldots, R_n \in \mathcal{R}$ and $E = \{i\colon\ xR_i y\}$. We wish to show $xf(R_1, \ldots, R_n)y$. By (14), (15), (16) with x and z interchanged, and (17), there are relations $R_{(14)}, R_{(15)}, R_{(16)}, R_{(17)} \in \mathcal{R}$, respectively, with these properties:

1. $x\overline{R_{(14)}}y$ $x\overline{R_{(14)}}z$ $z\overline{R_{(14)}}y$ by (14)
2. $xR_{(15)}y$ $xR_{(15)}z$ $zR_{(15)}y$ by (15)
3. $z\overline{R_{(16)}}y$ $z\overline{R_{(16)}}x$ $xR_{(16)}y$ by (16) with z, x switched
4. $x\overline{R_{(17)}}y$ $xR_{(17)}z$ $z\overline{R_{(17)}}y$ by (17)

For each $i \in \{1, \ldots, n\}$ let

5. $$R'_i = \begin{cases} R_{(15)} & \text{if } i \in D \\ R_{(16)} & \text{if } i \in \overline{D} \cap E \\ R_{(17)} & \text{if } i \in \overline{E} \text{ and } xR_{(16)}z \\ R_{(14)} & \text{if } i \in \overline{E} \text{ and } x\overline{R_{(16)}}z \end{cases}$$
6. $xR'_i y$ $xR'_i z$ $zR'_i y$ for every $i \in D$ by 5, 2

If $xR_{(16)}z$ then

7. $xR'_i y$ $xR'_i z$ $z\overline{R'_i}y$ for every $i \in \overline{D} \cap E$ by 5, 3, $xR_{(16)}z$
8. $x\overline{R'_i}y$ $xR'_i z$ $z\overline{R'_i}y$ for every $i \in \overline{E}$ by 5, 4, $xR_{(16)}z$

but if $x\overline{R_{(16)}}z$ then

7′. $xR'_i y$ $x\overline{R'_i}z$ $z\overline{R'_i}y$ for every $i \in \overline{D} \cap E$ by 5, 3, $x\overline{R_{(16)}}z$
8′. $x\overline{R'_i}y$ $x\overline{R'_i}z$ $z\overline{R'_i}y$ for every $i \in \overline{E}$ by 5, 1, $x\overline{R_{(16)}}z$

In either case, we have

9. $xf(R'_1, \ldots, R'_n)z$ by 6, 7, 8, $x[\{1, \ldots, n\}]z$ or 6, 7′, 8′, $x[D]z$

10. $zf\left(R'_1,\ldots,R'_n\right)y$ by 6, 7, 8, $x[D]z$ or 6, 7′, 8′, $z[D]y$
11. $xf\left(R'_1,\ldots,R'_n\right)y$ by 9, 10, f maps into $\mathcal{T}$
12. $E=\left\{i\colon xR'_iy\right\}$ by 6, 7, 8, or 6, 7′, 8′
13. $E=\{i\colon xR_iy\}$ by assumption
14. $\bigwedge\limits_{1\leqslant i\leqslant n}\left(xR_iy\Leftrightarrow xR'_iy\right)$ by 12, 13
15. $xf(R_1,\ldots,R_n)y$ by 14, **IIA** □

The next lemma says the complement of a decisive set is "decisive for the complementary relation".

Lemma 7. *Suppose that $D\subseteq\{1,\ldots,n\}$, $x,y\in U$, and not $x[D]y$. Then*

$$(\forall R_1,\ldots,R_n\in\mathcal{R})\quad\left(D=\{i\colon xR_iy\}\Rightarrow x\overline{f(R_1,\ldots,R_n)}y\right)\tag{21}$$

Proof. Assume $R_1,\ldots,R_n\in\mathcal{R}$ and $D=\{i\colon xR_iy\}$. We have assumed that (20) is false, so there are $R'_1,\ldots,R'_n\in\mathcal{R}$ such that

$$\left(\forall i\in\{1,\ldots,n\}\right)\quad\left(i\in D\Leftrightarrow xR'_iy\right)$$

and $x\overline{f(R'_1,\ldots,R'_n)}y$. Now it follows from

$$\left(\forall i\in\{1,\ldots,n\}\right)\quad(i\in D\Leftrightarrow xR_iy)$$

and

$$\left(\forall i\in\{1,\ldots,n\}\right)\quad\left(i\in D\Leftrightarrow xR'_iy\right)$$

that

$$\left(\forall i\in\{1,\ldots,n\}\right)\quad\left(xR_iy\Leftrightarrow xR'_iy\right),$$

so from **IIA** and $x\overline{f(R'_1,\ldots,R'_n)}y$ we get $x\overline{f(R_1,\ldots,R_n)}y$, as desired. □

8. The Intersection Theorem

Theorem 2 *(Intersection Theorem). Assume U is a set with at least three elements, $2\leqslant n<\omega$, $f:\mathcal{P}(U^2)^n\to\mathcal{P}(U^2)$, $\mathcal{R}$ is a diverse set of relations on U, f maps $\mathcal{R}^n$ into the set $\mathcal{T}$ of transitive relations on U, and f satisfies **Unan** and **IIA** on $\mathcal{R}$. Then there is some $D_0\subseteq\{1,\ldots,n\}$ such that*

$$(\forall R_1,\ldots,R_n\in\mathcal{R})\quad\left(f(R_1,\ldots,R_n)\cap 0'=\bigcap_{i\in D_0}R_i\cap 0'\right).\tag{22}$$

Proof. Let D_0 be the intersection of all the decisive subsets of $\{1,\ldots,n\}$. There are only finitely many subsets of $\{1,\ldots,n\}$, hence D_0 is decisive by (an inductive proof based on) Lemma 5. We wish to prove (22). Consider any $R_1,\ldots,R_n\in\mathcal{R}$ and any $x,y\in U$. Suppose $x\neq y$ and xR_iy for all $i\in D_0$, i.e., $\langle x,y\rangle\in\bigcap_{i\in D_0}R_i\cap 0'$ (the right hand side of (22)). Let $E=\{i\colon xR_iy\}$. Then $D_0\subseteq E\subseteq\{1,\ldots,n\}$ by definitions, so E is decisive by Lemma 6, consequently $x[E]y$ by Lemma 4, so $xf(R_1,\ldots,R_n)y$ by definitions. This shows $f(R_1,\ldots,R_n)\supseteq\bigcap_{i\in D_0}R_i\cap 0'$. To show

$$f(R_1,\ldots,R_n)\cap 0'\subseteq\bigcap_{i\in D_0}R_i\tag{23}$$

(which implies the other direction of (22)), assume $xf(R_1,\ldots,R_n)y$ for distinct $x,y\in U$. Again, we let $E=\{i\colon xR_iy\}$ and prove $x[E]y$. Toward this end assume $R'_1,\ldots,R'_n\in\mathcal{R}$ and assume

$$E=\left\{i\colon xR'_iy\right\}.\tag{24}$$

We must now show $xf(R'_1,\ldots,R'_n)y$. From the definition of E and (24) we get $\bigwedge_{1\leqslant i\leqslant n}(xR_iy\Leftrightarrow xR'_iy)$, hence $xf(R_1,\ldots,R_n)y\Leftrightarrow xf(R'_1,\ldots,R'_n)y$ by **IIA**, so $xf(R'_1,\ldots,R'_n)y$ because we assumed $xf(R_1,\ldots,R_n)y$. We have just proved $x[E]y$ and have

assumed $x \neq y$, so E is decisive, hence $D_0 \subseteq E$ since D_0 is the intersection of all decisive subsets of $\{1, \ldots, n\}$, including E. Therefore, for every $i \in D_0$, we have $i \in E$, hence xR_iy, that is, $\langle x, y\rangle \in \bigcap_{i\in D_0} R_i$. This completes the proof of (23) and with it, (22) and all of Theorem 2. □

We cannot conclude that D_0 is not empty. For an example to show this, let $\mathcal{R} = \mathcal{P}(U^2)$ and for all $R_1, \ldots, R_n \in \mathcal{R}$, let

$$f(R_1, \ldots, R_n) = U^2.$$

Then $\mathcal{R}$ is very diverse and the output relation U^2 is transitive and co-transitive, so f maps $\mathcal{R}$ into $\mathcal{V}$, **Unan** and **IIA** hold, and $D_0 = \emptyset$.

We cannot conclude that D_0 is a proper subset of $\{1, \ldots, n\}$. For an example to show this, let $\mathcal{R}$ be the set of equivalence relations on U. Then $\mathcal{R}$ is diverse (but not very diverse). Let f be defined for all relations $R_1, \ldots, R_n \in \mathcal{P}(U^2)$ by

$$f(R_1, \ldots, R_n) = \bigcap_{1\leqslant i\leqslant n} R_i. \tag{25}$$

It is easy to see that **Unan** and **IIA** hold on $\mathcal{R}$. Furthermore, f preserves transitivity, reflexivity, and symmetry, so, applied to equivalence relations in $\mathcal{R}$, f produces an equivalence relation, which is transitive. Therefore f maps $\mathcal{R}^n$ into $\mathcal{T}$. The function f satisfies all the hypotheses, and hence the conclusion, of Theorem 2, but only with $D_0 = \{1, \ldots, n\}$.

A related example, due to Sen [7, Theorem V], is obtained by applying the operator of Lemma 9 in Section 10 to the previous example.

$$f(R_1, \ldots, R_n) = \bigcap_{1\leqslant i\leqslant n} R_i \cup \overline{\left(\bigcap_{1\leqslant i\leqslant n} R_i\right)^{-1}}. \tag{26}$$

The functions defined on weak orderings by (25) and (26) satisfy **IIA** on $\mathcal{W}$, Condition 3, Condition P, and Condition 5. Since the inputs are transitive, the output of (25) is always transitive but may not be complete, even if the inputs are linear orderings. On the other hand, the output of (26) is complete (and quasi-transitive), but may not be transitive even if the inputs are linear orderings. These examples show that Arrow's Theorem fails if the outputs are required to be only transitive, or only quasi-transitive and complete.

9. The Projection Theorem

The next theorem shows that if the output relations are co-transitive as well as transitive, and the set $\mathcal{R}$ of inputs is very diverse (not just diverse), then D_0 is either empty or contains a single element.

Theorem 3 *(Projection Theorem). Assume U is a set with at least three elements, $2 \leqslant n < \omega$, $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$, $\mathcal{R}$ is a very diverse set of relations on U, f maps $\mathcal{R}^n$ into the set $\mathcal{V}$ of transitive co-transitive relations on U, and f satisfies **Unan** and **IIA** on $\mathcal{R}$. Then either*

$$(\forall R_1, \ldots, R_n \in \mathcal{R}) \quad \big(f(R_1, \ldots, R_n) \cap 0' = 0'\big), \tag{27}$$

or else there is some $k \in \{1, \ldots, n\}$ such that

$$(\forall R_1, \ldots, R_n \in \mathcal{R}) \quad \big(f(R_1, \ldots, R_n) \cap 0' = R_k \cap 0'\big). \tag{28}$$

Proof. This is a continuation of the proof of Theorem 2. So far we have proved (22) with $\mathcal{R} = \mathcal{V}$. We now make use of the additional diversity hypotheses (18) and (19), and the hypothesis that all output relations are co-transitive, to show that D_0 has at most one element.

Suppose $2 \leqslant |D_0|$. Then we may partition D_0 into two nonempty disjoint parts, say D' and D'', where $D' \cap D'' = \emptyset$ and $D_0 = D' \cup D''$. Let $\overline{D_0} = \{i\colon\ i \in \{1, \ldots, n\},\ i \notin D_0\}$. Thus we have a partition of $\{1, \ldots, n\}$ into three pairwise disjoint sets, D', D'', $\overline{D_0}$. Although $\overline{D_0}$ may be empty, D' and D'' are nonempty proper subsets of D_0. Since D_0 is the intersection of all decisive subsets of $\{1, \ldots, n\}$, no proper nonempty subset of D_0 can be decisive for any ordered pair in 0', because of Lemma 4. Choose distinct $x, y, z \in U$ using the hypothesis that $|U| \geqslant 3$. Then D' is not decisive for $\langle z, y\rangle$, D'' is not decisive for $\langle x, z\rangle$, and D_0 is decisive for $\langle x, z\rangle$ by Lemma 4. By (18) and (19) and (14) there are relations $R_{(18)}, R_{(19)}, R_{(14)} \in \mathcal{R}$, respectively, with these properties:

1. $xR_{(18)}y$ $\quad x\overline{R_{(18)}}z$ $\quad zR_{(18)}y$ $\quad$ by (18), $y \neq z$
2. $xR_{(19)}y$ $\quad xR_{(19)}z$ $\quad z\overline{R_{(19)}}y$ $\quad$ by (19), $x \neq z$
3. $x\overline{R_{(14)}}y$ $\quad x\overline{R_{(14)}}z$ $\quad z\overline{R_{(14)}}y$ $\quad$ by (14)

Choose $R_1, \ldots, R_n \in \mathcal{R}$ by setting, for each $i \in \{1, \ldots, n\}$,

4. $R_i = \begin{cases} R_{(18)} & \text{if } i \in D' \\ R_{(19)} & \text{if } i \in D'' \\ R_{(14)} & \text{if } i \in \overline{D_0} \end{cases}$

5. $xR_i y \quad x\overline{R_i}z \quad zR_i y$ for all $i \in D'$ by 1, 4

6. $xR_i y \quad xR_i z \quad z\overline{R_i}y$ for all $i \in D''$ by 2, 4

7. $x\overline{R_i}y \quad x\overline{R_i}z \quad z\overline{R_i}y$ for all $i \in \overline{D_0}$ by 3, 4

8. $x\overline{f(R_1, \ldots, R_n)}z$ by 5, 6, 7, not $x\big[D''\big]z$, Lemma 7

9. $z\overline{f(R_1, \ldots, R_n)}y$ by 5, 6, 7, not $z\big[D'\big]y$, Lemma 7

10. $xf(R_1, \ldots, R_n)y$ by 5, 6, 7, $x[D_0]y$

It follows that $\overline{f(R_1, \ldots, R_n)}$ is not transitive, hence $f(R_1, \ldots, R_n)$ is not co-transitive and not in $\mathcal{V}$, contrary to the hypothesis that f is an operator taking values in $\mathcal{V}$. The assumption $2 \leqslant |D_0|$ has yielded a contradiction, so $|D_0| \leqslant 1$. If $|D_0| = 0$ then (27), and if $|D_0| = 1$ then $D_0 = \{k\}$ for some $k \in \{1, \ldots, n\}$ such that (28). □

Proof of (7). Here we prove (7) under the hypotheses of Theorem 1. Assume U is a set with at least three elements, $2 \leqslant n < \omega$, $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$, f maps $\mathcal{W}^n$ into $\mathcal{W}$ (Condition 1′), and f satisfies Condition P and Condition 3.

To apply Theorem 3 with $\mathcal{R} = \mathcal{L}$, we must check that the hypotheses hold. We have assumed U is a set with at least three elements, $2 \leqslant n < \omega$, and $f : \mathcal{P}(U^2)^n \to \mathcal{P}(U^2)$. $\mathcal{R}$ (the set of linear orderings of U) is very diverse. By hypothesis, f maps $\mathcal{W}^n$ into $\mathcal{W}$, but $\mathcal{R}^n = \mathcal{L}^n \subset \mathcal{W}^n$ and $\mathcal{W} \subset \mathcal{V}$, so f maps $\mathcal{R}^n$ into $\mathcal{V}$. We proved in Lemma 1 that Condition 3 implies **IIA** on $\mathcal{L} = \mathcal{R}$. It remains to prove **Unan** from Condition P. Assume $x, y \in U$, $R_1, \ldots, R_n \in \mathcal{R} = \mathcal{L}$, $x \neq y$, and $xR_i y$ for all $i \in \{1, \ldots, n\}$. Note that for each linear ordering $R_i \in \mathcal{L}$ we have

$$R_i \cap 0' = \overline{R_i^{-1}} \cap 0'. \tag{29}$$

From $x \neq y$ and $xR_i y$ for all $i \in \{1, \ldots, n\}$ we therefore conclude that $x\overline{R_i^{-1}}y$ for all $i \in \{1, \ldots, n\}$, hence $x\overline{(f(R_1, \ldots, R_n))^{-1}}y$ by Condition P, which implies $xf(R_1, \ldots, R_n)y$ by the completeness of $f(R_1, \ldots, R_n) \in \mathcal{W}$.

The hypotheses of Theorem 3 have been met with $\mathcal{R} = \mathcal{L}$, so we have either (27) or (28). However, Condition P rules out (27), because together they imply

$$0' \subseteq \bigcup_{1 \leqslant i \leqslant n} R_i$$

for all $R_1, \ldots, R_n \in \mathcal{R} = \mathcal{L}$, but this is false since we can find, for any pair $\langle x, y \rangle \in 0'$, linear orderings $R_i \in \mathcal{R} = \mathcal{L}$ such that $x\overline{R_i}y$ for all $i \in \{1, \ldots, n\}$. Then $\langle x, y \rangle$ is in the set on the left but not in the set on the right. Therefore, (28) holds with $\mathcal{R} = \mathcal{L}$. However, R_k is reflexive (it's a linear ordering) and $f(R_1, \ldots, R_n)$ is reflexive (it's a weak ordering because f maps $\mathcal{L}^n \subseteq \mathcal{W}^n$ into $\mathcal{W}$), so (28) tells us that the restriction of f to $\mathcal{L}$ is actually a projection function (onto coordinate k), that is, (7) holds. □

10. Weak orderings vs. transitive co-transitive relations

As we show here, the transitive co-transitive relations (the subject of the Projection Theorem) do not differ much from the weak orderings (the subject of Arrow's Theorem). First the basic inclusions.

Lemma 8. $\mathcal{L} \subset \mathcal{W} \subset \mathcal{V} \subset \mathcal{T}$ (*all inclusions are proper*).

Proof. The first inclusion is immediate from definitions, and is proper because U^2 is an example of a weak (but not linear) ordering. The third inclusion is also immediate from definitions, and is proper because 1' is transitive but not co-transitive. If R is a weak ordering, then $\overline{R} \subseteq R^{-1}$ by the completeness of R and $\overline{R}; R^{-1} \subseteq \overline{R}$ by the transitivity of R, so $\overline{R}$ is transitive because $\overline{R}; \overline{R} \subseteq \overline{R}; R^{-1} \subseteq \overline{R}$. This shows $\mathcal{W} \subseteq \mathcal{V}$. The inclusion is proper because $\emptyset$ is transitive and co-transitive, hence $\emptyset \in \mathcal{V}$, while $\emptyset$ is neither reflexive, connected, nor complete, hence $\emptyset \notin \mathcal{W}$. □

$\mathcal{W}$ is closed under conversion (e.g., the converse of $\leqslant$ is $\geqslant$) but not complementation (the complement $>$ of $\leqslant$ is irreflexive). $\mathcal{V}$ is closed under both conversion and complementation, so $\mathcal{V}$ contains the strict preference relations of all the weak orderings. As the following lemma shows, the unary relational operator used in (26), which reproduces the original weak ordering from its preference relation, associates a weak ordering with every transitive co-transitive relation.

Lemma 9. *Assume $R \in \mathcal{V}$ and $W = R \cup \overline{R^{-1}}$. Then W is a weak ordering whose strict preference relation is $R \cap \overline{R^{-1}}$, and whose indifference relation is*

$$E = \left(R \cap R^{-1}\right) \cup \left(\overline{R} \cap \overline{R^{-1}}\right).$$

Furthermore, E is an equivalence relation whose equivalence classes are linearly ordered by the relation $<$, which is defined for equivalence classes X and Y by

$$X < Y \quad \Leftrightarrow \quad X \times Y \subseteq R \cap \overline{R^{-1}}.$$

Proof. Assume R is transitive and co-transitive. Then R and $\overline{R}$ are both transitive, but converses of transitive relations are transitive, so R^{-1} and $\overline{R^{-1}}$ are also transitive. Let

$$E_0 = R \cap R^{-1} \quad \text{and} \quad E_1 = \overline{R} \cap \overline{R^{-1}}, \quad \text{so } E = E_0 \cup E_1 \text{ and } W = R \cup E_1 \tag{30}$$

Intersections of transitive relations are transitive, so E_0 and E_1 are transitive. For *any* relation R, $R \cap R^{-1}$ is symmetric, so E_0 and E_1 are symmetric as well as transitive. The two relations E_0 and E_1 are obviously disjoint. To see that their union E contains the identity relation observe that

$$1' \cap R = \left(1' \cap R\right)^{-1} = 1'^{-1} \cap R^{-1} = 1' \cap R^{-1},$$

so $1' \cap R = 1' \cap R \cap R^{-1} \subseteq E_0$, and, similarly, $1' \cap \overline{R} = 1' \cap \overline{R} \cap \overline{R^{-1}} \subseteq E_1$, hence

$$1' = 1' \cap U^2 = 1' \cap (R \cup \overline{R}) = \left(1' \cap R\right) \cup \left(1' \cap \overline{R}\right) \subseteq E_0 \cup E_1 = E.$$

Thus E is reflexive. E is symmetric because it is the union of two symmetric relations. R is transitive, therefore

$$R;R \subseteq R, \qquad R^{-1};\overline{R} \subseteq \overline{R}, \qquad \overline{R};R^{-1} \subseteq \overline{R} \tag{31}$$

and R is co-transitive, therefore

$$\overline{R};\overline{R} \subseteq \overline{R}, \qquad R;\overline{R^{-1}} \subseteq R, \qquad \overline{R^{-1}};R \subseteq R. \tag{32}$$

We have

$$E_0;E_1 = E_1;E_0 = \emptyset \tag{33}$$

because

$$\begin{aligned} E_0;E_1 &\subseteq \left(R^{-1};\overline{R}\right) \cap \left(R;\overline{R^{-1}}\right) && \text{by (30)} \\ &\subseteq \overline{R} \cap R = \emptyset && \text{by (31), (32)} \\ E_1;E_0 &= E_1^{-1};E_0^{-1} = (E_0;E_1)^{-1} = \emptyset^{-1} = \emptyset \end{aligned}$$

E is transitive because

$$\begin{aligned} E;E &= (E_0 \cup E_1);(E_0 \cup E_1) && \text{by (30)} \\ &= (E_0;E_0) \cup (E_1;E_0) \cup (E_0;E_1) \cup (E_1;E_1) \\ &\subseteq (E_0;E_0) \cup \emptyset \cup \emptyset \cup (E_1;E_1) && \text{by (33)} \\ &\subseteq E_0 \cup E_1 = E && E_0, E_1 \text{ are transitive, (30)} \end{aligned}$$

Thus E is an equivalence relation on U. W is transitive because

$$\begin{aligned} W;W &= (R \cup E_1);(R \cup E_1) && \text{by (30)} \\ &= (R;R) \cup (R;E_1) \cup (E_1;R) \cup (E_1;E_1) \\ &\subseteq R \cup \left(R;\overline{R^{-1}}\right) \cup \left(\overline{R^{-1}};R\right) \cup E_1 && \text{(31), (30), } E_1 \text{ is transitive} \\ &\subseteq R \cup R \cup R \cup E_1 && \text{by (32)} \\ &= W && \text{by (30)} \end{aligned}$$

and W is complete because

$$\begin{aligned} W \cup W^{-1} &= (R \cup E_1) \cup (R \cup E_1)^{-1} && \text{by (30)} \\ &= R \cup E_1 \cup R^{-1} \cup E_1 && E_1 \text{ is symmetric} \\ &= R \cup R^{-1} \cup \left(\overline{R} \cap \overline{R^{-1}}\right) && \text{by (30)} \\ &= R \cup R^{-1} \cup \overline{R \cup R^{-1}} = U^2. \end{aligned}$$

Therefore W is a weak ordering. The strict preference relation of W is, by Arrow's Definition 1, $\overline{W^{-1}} = \overline{(R \cup \overline{R^{-1}})^{-1}} = \overline{R^{-1} \cup \overline{R}} = \overline{R^{-1}} \cap R = R \cap \overline{R^{-1}}$, and the indifference relation of W is, by Arrow's Definition 2,

$$\begin{aligned} W \cap W^{-1} &= \left(R \cup \overline{R^{-1}}\right) \cap \left(R \cup \overline{R^{-1}}\right)^{-1} \\ &= \left(R \cup \overline{R^{-1}}\right) \cap \left(R^{-1} \cup \overline{R}\right) \\ &= \left(R \cap R^{-1}\right) \cup \left(\overline{R} \cap \overline{R^{-1}}\right) \\ &= E \end{aligned}$$

For the transitivity of $<$, assume $X < Y < Z$ for some equivalence classes X, Y, Z of E. Then

$$\begin{aligned} X \times Z &= (X \times Y);(Y \times Z) && Y \neq \emptyset \\ &\subseteq \left(R \cap \overline{R^{-1}}\right);\left(R \cap \overline{R^{-1}}\right) && X < Y < Z \\ &\subseteq (R;R) \cap \left(\overline{R^{-1}};\overline{R^{-1}}\right) \\ &\subseteq R \cap \overline{R^{-1}} && R, \overline{R^{-1}} \text{ are transitive} \end{aligned}$$

Next we let $P = R \cap \overline{R^{-1}}$ and prove

$$E;P;E \subseteq P. \tag{34}$$

By (30) and (31), we have

$$E_0;P \subseteq R;P \subseteq R;R \cap R;\overline{R^{-1}} \subseteq P,$$

and furthermore, by (30) and (32),

$$E_1;P \subseteq \overline{R^{-1}};P \subseteq \overline{R^{-1}};R \cap \overline{R^{-1}};\overline{R^{-1}} \subseteq P,$$

so, by the last two equations,

$$E;P = (E_0;P) \cup (E_1;P) \subseteq P.$$

By a similar calculation, $P;E \subseteq P$, so (34) holds. To prove $<$ is complete, assume X and Y are (non-empty) equivalence classes of E. If there is a pair $\langle x, y\rangle \in X \times Y$ such that $\langle x, y\rangle \in P$ then, by (34),

$$X \times Y = E;\{\langle x, y\rangle\};E \subseteq E;P;E \subseteq P,$$

hence $X < Y$. If not, then $(X \times Y) \cap P = \emptyset$. Taking converses, we get

$$\emptyset = \emptyset^{-1} = (X \times Y)^{-1} \cap P^{-1} = (Y \times X) \cap R^{-1} \cap \overline{R},$$

so $Y \times X \subseteq \overline{R^{-1} \cap \overline{R}} = P$, hence $Y < X$. □

References

[1] A.B. Feferman, S. Feferman, Alfred Tarski: Life and Logic, Cambridge University Press, 2004, Vi+426 pp.
[2] A. Tarski, Introduction to Logic, 2nd edition, Oxford University Press, New York, 1941 (revised 1946).
[3] K.J. Arrow, Social Choice and Individual Values, 1st edition, John Wiley & Sons, New York, 1951 (2nd edition, 1963).
[4] K.J. Arrow, A. Sen, K. Suzumura, Kenneth Arrow on social choice theory, in: Handbook of Social Choice and Welfare, vol. II, Elsevier BV, 2011, Ch. Thirteen.
[5] K. Kuratowski, Sur la notion de l'ordre dans la théorie des ensembles, Fundam. Math. 2 (1921) 161–171.
[6] J.H. Blau, The existence of social welfare functions, Econometrica 25 (1957) 302–313.
[7] A. Sen, Quasi-transitivity, rational choice, and collective decisions, Rev. Econ. Stud. 36 (1969) 381–393.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 249–262

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

A relation-algebraic approach to the "Hoare logic" of functional dependencies

José N. Oliveira [a]

[a] *High Assurance Software Lab, INESC TEC and Univ. Minho, 4710-070 Braga, Portugal*

ARTICLE INFO

Article history:
Available online 18 February 2014

Keywords:
Relational mathematics
Program calculi
Data dependency theory

ABSTRACT

Abstract algebra has the power to *unify* seemingly disparate theories once they are encoded into the same abstract formalism. This paper shows how a relation-algebraic rendering of both database dependency theory and Hoare programming logic purports one such unification, in spite of the latter being an algorithmic theory and the former a data theory. The approach equips relational data with *functional types* and an associated type system which is useful for database operation type checking and optimization.
The prospect of a generic, unified approach to both programming and data theories on top of libraries already available in automated deduction systems is envisaged.

> "*Hardly anybody confronted with practical problems knows how to apply relational calculi* [*for which*] *there is almost no broadly available computer support* [...] *We feel, however, that the situation is about to change dramatically as relational mathematics develops and computer power exceeds previous expectations.*"
>
> [Gunther Schmidt [1]]

1. Introduction

In a paper addressing the influence of Alfred Tarski (1901–1983) in computer science, Solomon Feferman [2] quotes the following statement by his colleague John Etchemendy: "*You see those big shiny Oracle towers on Highway 101? They would never have been built without Tarski's work on the recursive definitions of satisfaction and truth*".

The 'big shiny Oracle towers' are nothing but the headquarters of Oracle Corporation, the giant database software provider sited in the San Francisco Peninsula. Still Feferman [2]: "*Does Larry Ellison know who Tarski is or anything about his work?* [...] *I learned subsequently from Jan Van den Bussche that* [...] *he marks the reading of Codd's seminal paper as the starting point leading to the Oracle Corporation.*"

Bussche [3] had in fact devoted attention to relating Codd and Tarski's work: "*We conclude that Tarski produced two alternatives for Codd's relational algebra: cylindric set algebra, and relational algebra with pairing* [...] *For example, we can represent the ternary relation* $\{(a, b, c), (d, e, f)\}$ *as* $\{(a, (b, c)), (d, (e, f))\}$". Still Bussche [3]:

> "*Using such representations, we leave it as an exercise to the reader to simulate Codd's relational algebra in* RA^+ [relational algebra with pairing]".

E-mail address: jno@di.uminho.pt.

http://dx.doi.org/10.1016/j.jlap.2014.02.013
2352-2208/

To the best of the author's knowledge, nobody has thus far addressed this *exercise* in a thorough and generic way. Instead, standard relational database theory [4,5] includes a well-known relation algebra but this is worked out in set theory and quantified logic, far from the objectives of Tarski's life-long pursuit in developing methods for elimination of quantifiers from logic expressions. This effort ultimately lead to his *formalization of set theory without variables* [6].

The topic has acquired recent interest with the advent of work on implementing extensions of Tarski's algebra in automated deduction systems such as *Isabelle* [7] or *Prover9* and the associated counterexample generator *Mace4* [8]. This offers a potential for automation which has not been acknowledged by the database community. In this context, it is worth mentioning an early concern of the founding fathers of the standard theory [9]:

"[*A*] *general theory that ties together dependencies, relations and operations on relations is still lacking*".

More than 30 years later, this concern is still justified, as database programming standards remain insensitive to techniques such as formal verification and extended static checking [10] which are regarded more and more essential to ensuring quality in complex software systems.

Contribution The remainder of this paper will show how an algebraic treatment of standard data dependency theory along the *exercise* proposed by Bussche equips relational data with *functional types* and an associated type system which can be used to *type check* database operations.

Interestingly, such a *typed* approach to database programming will be shown to relate to other programming logics such as e.g. *Hoare logic* [11] or that of *strongest invariant functions* [12] which has been used in the analysis of while statements, for instance.

On the whole, the approach has a *unifying theories of programming* [13] flavour, even though the exercise will not be carried out in "canonical" UTP.

Outline Section 2 introduces functional dependencies (FD) and shows how to convert the standard definition into the Tarskian, quantifier-free style. The parallel between the *functions as types* approach which emerges from such a conversion and a similar treatment of Hoare logic starts in Section 3. Section 4 shows that, in essence, *injectivity* is what matters in FDs and gives a correspondingly simpler definition of FD which is used in Section 5 to re-factor the standard theory into a *type system of FDs*. Section 6 shows how to use this type system to type check database operations and Section 7 shows how to calculate query optimizations from FDs. The last sections conclude and give an account of related and future work.

Some technical details are omitted from the current paper for conciseness. All can be found in a technical report available on-line [14].

2. Introducing functional dependencies

In standard relational data processing, real life objects or entities are recorded by assigning values to their observable properties or *attributes*. A database *table* is a collection of such attribute assignments, one per object, such that all values of a particular attribute (say i) are of the same type (say A_i). For n such attributes, a *relational database file* T can be regarded as a set of n-tuples, that is, $T \subseteq A_1 \times \cdots \times A_n$. A *relational database* is just a collection of several such relations, or tables.

Attribute names normally replace natural numbers in the identification of attributes. The enumeration of all attribute names in a database table, for instance $S = \{\text{PILOT}, \text{FLIGHT}, \text{DATE}, \text{DEPARTS}\}$ concerning the airline scheduling system given as example in [4], is a finite set called the table's *scheme*. This scheme captures the *syntax* of the data. What about *semantics*? Even non-experts in airline scheduling will accept "business rules" such as, for instance: *a single pilot is assigned to a given flight, on a given date*. This restriction is an example of a so-called *functional dependency* (FD) among attributes, which can be stated more formally by writing "FLIGHT DATE $\rightarrow$ PILOT" to mean that *attribute* PILOT *is functionally dependent on* FLIGHT *and* DATE, or that FLIGHT, DATE *functionally determine* PILOT.

Data dependencies help in capturing the *meaning* of relational data. Data dependency theory involves not only functional dependencies (FD) but also multi-valued dependencies (MVD). Both are central to the standard theory, where they are addressed in an axiomatic way. Maier [4] provides the following definition for FD-satisfiability:

Definition 1. Given subsets $x, y \subseteq S$ of the relation scheme S of an n-ary relation T, this relation is said to satisfy functional dependency $x \rightarrow y$ iff all pairs of tuples $t, t' \in T$ which "agree" on x also "agree" on y, that is,

$$\forall t, t' : t, t' \in T \quad \Rightarrow \quad \big(t[x] = t'[x] \quad \Rightarrow \quad t[y] = t'[y]\big) \tag{1}$$

(The notation $t[x]$ in (1) means "the values exhibited by tuple t for the attributes in x".) □

How does one express formula (1) in Tarski's relation algebra style, without the two-dimensional universal quantification and logical implications inside? For so doing we need to settle some notation. To begin with, $t[x]$ is better written as $x(t)$, where x is identified with the *projection function* associated to attribute set x. Regarding x and y in (1) as such functions we write:

$$\forall t, t' : t, t' \in T \quad \Rightarrow \quad \big(x(t) = x\big(t'\big) \quad \Rightarrow \quad y(t) = y\big(t'\big)\big) \tag{2}$$

Next, we observe that, given a function $f : A \to B$, the binary relation $R \subseteq A \times A$ which checks whether two values of A have the same image under f[1] — that is, $a' R a \equiv f(a') = f(a)$ — can be written alternatively as $a'(f^\circ \cdot f)a$. Here, f° denotes the *converse* of f — that is, $a(f^\circ)b$ holds iff $b = f(a)$ — and the dot $(\cdot)$ denotes the extension of function composition to binary relations[2]:

$$b(R \cdot S)c \quad \equiv \quad \exists a : b\ R\ a\ \wedge\ a\ S\ c \tag{3}$$

Using converse and composition the rightmost implication of (2) can be rewritten into $t(x^\circ \cdot x)t' \Rightarrow t(y^\circ \cdot y)t'$, for all $t, t' \in T$. Implications such as this can be expressed as relation inclusions, following the definition

$$R \subseteq S \quad \equiv \quad \forall b, a : b\ R\ a \quad \Rightarrow \quad b\ S\ a \tag{4}$$

However, just stating the inclusion $x^\circ \cdot x \subseteq y^\circ \cdot y$ would be a gross error, for the double scope of the quantification ($t \in T\ \wedge\ t' \in T$) would not be taken into account. To handle this, we first unnest the two implications of (2),

$$\forall t, t' : \big(t \in T\ \wedge\ t' \in T\ \wedge\ t\big(x^\circ \cdot x\big)t'\big) \quad \Rightarrow \quad t(y^\circ \cdot y)t'$$

and treat the antecedent $t \in T\ \wedge\ t' \in T\ \wedge\ t(x^\circ \cdot x)t'$ independently, by replacing the set of tuples T by the binary relation $[\![T]\!]$ defined as follows[3]:

$$b[\![T]\!]a \quad \equiv \quad b = a\ \wedge\ a \in T \tag{5}$$

Note that $t \in T$ can be expressed in terms of $[\![T]\!]$ by $\exists u : t[\![T]\!]u$ and similarly for $t' \in T$. Then:

$$\begin{array}{ll} & \big(t \in T\ \wedge\ t' \in T\ \wedge\ t\big(x^\circ \cdot x\big)t'\big) \\ \equiv & \quad \{\text{expansion of } t \in T \text{ and } t' \in T\} \\ & \exists u, u' : t[\![T]\!]u\ \wedge\ t'[\![T]\!]u'\ \wedge\ t(x^\circ \cdot x)t' \\ \equiv & \quad \{\wedge \text{ is commutative; } u = t \text{ and } u' = t'\text{; converse}\} \\ & \exists u, u' : t[\![T]\!]u\ \wedge\ u\big(x^\circ \cdot x\big)u'\ \wedge\ u'[\![T]\!]^\circ t' \\ \equiv & \quad \{\text{composition (3) twice}\} \\ & t\big([\![T]\!] \cdot x^\circ \cdot x \cdot [\![T]\!]^\circ\big)t' \qquad \square \end{array}$$

Finally, by putting this together with $t(y^\circ \cdot y)t'$ we obtain

$$[\![T]\!] \cdot x^\circ \cdot x \cdot [\![T]\!]^\circ \subseteq y^\circ \cdot y \tag{6}$$

as a quantifier-free relation algebra expression meaning the same as (1).

Generalization To reassure the reader worried about the doubtful practicality of derivations such as the above, we would like to say that we don't need to do it over and over again: inequality (6), our Tarskian alternative to the original textbook definition (1), is all we need for calculating with functional dependencies. Moreover, we can start this by actually expanding the scope of the definition from sets of tuples $[\![T]\!]$ and attribute functions $(x,\ y)$ to arbitrary binary relations R and suitably typed functions f and g:

$$R \cdot f^\circ \cdot f \cdot R^\circ \subseteq g^\circ \cdot g \tag{7}$$

In this wider setting, R can be regarded not only as a piece of data but also as the specification of a non-deterministic computation, or even the transition relation of a finite-state automaton; and f (resp. g) as a function which observes the input (resp. output) of R. Put back into quantified logic, such a wider notion of a functional dependency will expand as follows:

$$\forall a', a : f\big(a'\big) = f(a) \quad \Rightarrow \quad \big(\forall b', b : b'\ R\ a'\ \wedge\ b\ R\ a \quad \Rightarrow \quad g\big(b'\big) = g(b)\big) \tag{8}$$

[1] This is known as the *nucleus* [12] or *kernel* [15] of a function f.

[2] Thus composition of both functions are relations should be read backwards. This is consistent with $b\ f\ a$ (function f regarded as a special case of relation) meaning $b = f(a)$ and not $a = f(b)$.

[3] This is a standard way of encoding a set T as a *partial identity* [1], thus called since $[\![T]\!] \subseteq id$. The set of all such relations forms a Boolean algebra which reproduces the usual algebra of sets. Moreover, partial identities are symmetric ($[\![T]\!]^\circ = [\![T]\!]$) and such that $[\![S]\!] \cdot [\![T]\!] = [\![S]\!] \cap [\![T]\!]$. Also known as *coreflexives* [16] or as *monotypes* [17], partial identities are special cases of *tests* in Kleene algebras with tests [18].

In words: *inputs a, a' to R which are indistinguishable by f can only lead to outputs indistinguishable by g*. Notationally, we will convey this interpretation by writing $R : f \to g$ or $f \xrightarrow{R} g$. We can still say that R satisfies FD $f \to g$, in particular wherever R is a piece of data. As can be easily checked, $f(a') = f(a)$ is an equivalence relation which, in the wider setting, can be regarded as the *semantics* of the datatype which R takes inputs from (think of $f : A \to B$ as a *semantic* function mapping a syntactic domain A into a semantic domain B), and similarly for g concerning the output type.

Summing up, the functions f and g in (7) can be regarded as *types* for R. Some type assertions of this kind will be very easy to check, for instance $id : f \to f$, just by replacing $R, f, g := id, f, f$ in (7) and simplifying. But type inference will be easier to calculate on top of the even simpler (re)statement of (7) which is given next.

3. Functions as types

Before proceeding let us record two properties of the relational operators *converse* and *composition*[4]:

$$(R \cdot S)^\circ = S^\circ \cdot R^\circ \tag{9}$$

$$\left(R^\circ\right)^\circ = R \tag{10}$$

Moreover, it will be convenient to have a name for the relation $R^\circ \cdot R$ which, for R a function f, is the equivalence relation "indistinguishable by f" seen above. We define

$$\ker R \triangleq R^\circ \cdot R \tag{11}$$

and read $\ker R$ as "the *kernel* of R". Clearly, $a'(\ker R)a$ means $\exists b : b\ R\ a' \wedge b\ R\ a$ and therefore $\ker R$ measures the *injectivity* of R: the larger it is the larger the set of inputs which R is unable to distinguish (= the less *injective* R is).

We capture this by introducing a preorder on relations which compares their *injectivity*:

$$R \leqslant S \triangleq \ker S \subseteq \ker R \tag{12}$$

As an example, take two list functions: *elems* computing the set of all elements of a list and *bagify* keeping the bag of such elements. The former loses more information (order and multiplicity) than the latter, which only forgets about order. Thus $elems \leqslant bagify$. A function f (relation in general) will be *injective* iff $\ker f \subseteq id$ ($id \leqslant f$), which easily converts to the usual definition: $f(a') = f(a) \Rightarrow a' = a$.

Summing up: for functions or any totally defined relations R and S,[5] $R \leqslant S$ means that *R is less or as injective as S*; for possibly partial R and S, it will mean that *R less injective or more defined than S*. Therefore, for *total* relations R the preorder is universally bounded, $! \leqslant R \leqslant id$, where the infimum is captured by constant function ! which maps every argument to a given (predefined) value, the choice of which is irrelevant.[6] The kernel of ! is therefore the largest possible, denoted by $\top$ (for "top"). The other bound is trivial to check, since $\ker id = id$, this arising from the well-known fact that id is the unit of composition. In general, $id \leqslant R$ means that R is injective.

Equipped with this ordering, we may spruce up our relational characterization of the $f \xrightarrow{R} g$ type assertion, or functional dependency (FD):

$$\begin{array}{ll} & f \xrightarrow{R} g \\ \equiv & \{\text{definition (7)}\} \\ & R \cdot f^\circ \cdot f \cdot R^\circ \subseteq g^\circ \cdot g \\ \equiv & \{\text{converses (9), (10); kernel (11)}\} \\ & \ker(f \cdot R^\circ) \subseteq \ker g \\ \equiv & \{(12)\text{: } g \text{ is "}\textit{less or as injective}\text{ as } f \text{ w.r.t. } R\text{"}\} \\ & g \leqslant f \cdot R^\circ \qquad \square \end{array}$$

We thus reach a rather compact formula for expressing functional dependencies, whose layout invites us to actually swap the direction of the arrow notation (but, of course, this is optional and just a matter of taste):

[4] It may help to recall the same properties from elementary linear algebra, once converse is interpreted as *matrix transposition* and composition as *matrix–matrix multiplication* [1].

[5] A relation R is totally defined (or *entire*) iff $id \subseteq \ker R$.

[6] Thus $! \cdot f = !$, for all f. Also note that $R \leqslant S$ is a preorder, not a partial order, meaning that two relations indistinguishable with respect to their degree of injectivity can be different.

Definition 2. Given an arbitrary binary relation $R \subseteq A \times B$ and functions $f : B \to D$ and $g : A \to C$, the "type assertion" $g \xleftarrow{R} f$ meaning that R *satisfies FD* $f \to g$ is captured by the equivalence:

$$g \xleftarrow{R} f \equiv g \leqslant f \cdot R^\circ \qquad \square \tag{13}$$

There are two main advantages in definition (13), besides saving ink. The most important is that it profits from the relational calculus of injectivity which will be addressed in the following section. The other is that it makes it easy to bridge with other programming logics, as is seen next.

Parallel with Hoare logic As is widely known, Hoare logic is based on triples of the form $\{p\}R\{q\}$, with the standard interpretation: "*if the assertion p is true before initiation of a program R, then the assertion q will be true on its completion*" [11].

Let program R be identified with the relation which captures its state transition semantics and predicates p (and q) be identified with relation $s'[\![p]\!]s \equiv s' = s \ \wedge \ p(s)$ (similarly for q) in which the reader identifies the earlier trick of converting sets to partial identities (Section 2). Note how $[\![p]\!]$ can be regarded as the semantics of a statement which checks $p(s)$ and does not change state, failing otherwise. In relation algebra the Hoare triple is captured by

$$\{p\}R\{q\} \quad \equiv \quad \mathsf{rng}\left(R \cdot [\![p]\!]\right) \subseteq [\![q]\!] \tag{14}$$

meaning that the outputs of R (given by the range operator rng) for inputs pre-conditioned by p fall inside q[7]; that is, q is *weaker* than the strongest (*liberal*) post-condition $slp(R, p)$, something we can express by writing

$$\{p\}R\{q\} \quad \equiv \quad q \leqslant p \cdot R^\circ \tag{15}$$

under a suitable preorder $\leqslant$ expressing that q is less constrained than $p \cdot R^\circ$[8]:

$$R \leqslant S \quad \equiv \quad \mathsf{dom}\, S \subseteq \mathsf{dom}\, R \tag{16}$$

In spite of the different semantic context, there is a striking formal similarity between formulas (15) and (13) suggesting that Hoare logic and the logic we want to build for FDs share the same mathematics once expressed in relation algebra. Such similarities will become apparent in the sequel, particularly whereupon we write $p \xrightarrow{R} q$ (or the equivalent $q \xleftarrow{R} p$) for $\{p\}R\{q\}$ to put the two notations closer to each other. In this way, rules such as e.g. that of composition, $\{p\}R\{q\} \ \wedge \ \{q\}S\{r\} \Rightarrow \{p\}R; S\{r\}$ become reminiscent of labeled transition systems[9]:

$$p \xrightarrow{R} q \ \wedge \ q \xrightarrow{S} r \quad \Rightarrow \quad p \xrightarrow{R;S} r \tag{17}$$

We will check the FD equivalent to composition rule (17) shortly.

4. A calculus of injectivity ($\leqslant$)

One of the advantages of relation algebra is its easy "tuning" to special needs, which we will illustrate below concerning the algebra of injectivity. We give just an example, taken from [14]; the reader is referred to this report for technical details.

We start by considering two rules of relation algebra which prove very useful in program calculation:

$$f \cdot R \subseteq S \quad \equiv \quad R \subseteq f^\circ \cdot S \tag{18}$$

$$R \cdot f^\circ \subseteq S \quad \equiv \quad R \subseteq S \cdot f \tag{19}$$

In these equivalences,[10] which are widely known as *shunting rules* [23,1], f is required to be a (total) function. In essence, they let one trade a function f from one side to the other of a $\subseteq$-equation just by taking converses. (This is akin to "changing sign" in trading terms in inequations of elementary algebra.)

It would be useful to have similar rules for the injectivity preorder, which we have chosen as support for our definition of a FD (13). Such rules turn out to be quite easy to infer, as is the case of the following Galois connection for trading a function f with respect to injectivity

$$R \cdot f \leqslant S \quad \equiv \quad R \leqslant S \cdot f^\circ \tag{20}$$

[7] See e.g. [19]. Term $\mathsf{rng}(R \cdot [\![p]\!])$ instantiates the semiring *diamond* combinator of [20]. Wehrman et al. [21] give an even simpler semantics for Hoare triples: $P\{R\}Q \equiv R \cdot P \subseteq Q$, that is, P is at most the *weakest pre-specification* (residual relation) $R \setminus Q$, where $b(R \setminus Q)a$ means $\forall c : c \ R \ b \Rightarrow c \ Q \ a$ [22].

[8] Details: by definition, $\mathsf{dom}\, R = \ker R \cap id$ and $\mathsf{rng}\, R = \mathsf{dom}\,(R^\circ)$ (converse duality). Starting from (14), triple $\{p\}R\{q\}$ asserts $\mathsf{rng}(R \cdot [\![p]\!]) \subseteq [\![q]\!]$, itself the same as $\mathsf{dom}([\![p]\!] \cdot R^\circ) \subseteq \mathsf{dom}\,[\![q]\!]$ (15) by converse duality and the fact that the domain of a partial identity is itself. Parentheses $[\![\_]\!]$ are dropped for improved readability.

[9] Forward composition $R; S$ means the same as $S \cdot R$.

[10] Technically, these equivalences should be regarded as (families of) Galois connections.

calculated as follows:

$$
\begin{array}{ll}
& R \cdot f \leqslant S \\
\equiv & \quad \{\text{definition (12); converses (9), (10); kernel (11)}\} \\
& \ker S \subseteq f^\circ \cdot (\ker R) \cdot f \\
\equiv & \quad \{\text{shunting rules (18), (19)}\} \\
& f \cdot \ker S \cdot f^\circ \subseteq \ker R \\
\equiv & \quad \{\text{converses, kernel and definition (12) again}\} \\
& R \leqslant S \cdot f^\circ \qquad \square
\end{array}
$$

Below we put shunting rule (20) at work in the derivation of a *trading*-rule which will enable handling composite antecedent and consequent functions in FDs:

$$g \xleftarrow{h \cdot R \cdot k^\circ} f \;\equiv\; g \cdot h \prec\!\xleftarrow{R} f \cdot k \tag{21}$$

Thanks to (20), the calculation of (21) is immediate:

$$
\begin{array}{ll}
& g \xleftarrow{h \cdot R \cdot k^\circ} f \\
\equiv & \quad \{\text{definition (13); converses}\} \\
& g \leqslant f \cdot k \cdot R^\circ \cdot h^\circ \\
\equiv & \quad \{\text{shunting rule (20)}\} \\
& g \cdot h \leqslant (f \cdot k) \cdot R^\circ \\
\equiv & \quad \{\text{definition (13)}\} \\
& g \cdot h \prec\!\xleftarrow{R} f \cdot k \qquad \square
\end{array}
$$

Another result about relational injectivity which will help in the sequel is

$$X \leqslant R \cup S \quad \equiv \quad X \leqslant R \;\wedge\; X \leqslant S \;\wedge\; R^\circ \cdot S \subseteq \ker X \tag{22}$$

where $R \cup S$ is the union of relations R and S. For $X := id$, (22) tells that $R \cup S$ is injective *iff* both R and S are injective and *don't "equivocate" each other*: wherever bSa and bRc hold, $c = a$. The calculation of (22) follows:

$$
\begin{array}{ll}
& X \leqslant R \cup S \\
\equiv & \quad \{\text{definitions of } \leqslant \text{ (12) and kernel (11)}\} \\
& (R \cup S)^\circ \cdot (R \cup S) \subseteq \ker X \\
\equiv & \quad \{\text{distribution of converse and composition over union}\} \\
& (R^\circ \cdot R) \cup (R^\circ \cdot S) \cup (S^\circ \cdot R) \cup (S^\circ \cdot S) \subseteq \ker X \\
\equiv & \quad \{\text{kernel (11)}\} \\
& \ker R \cup (R^\circ \cdot S) \cup (S^\circ \cdot R) \cup \ker S \subseteq \ker X \\
\equiv & \quad \{\text{universal property: } R \cup S \subseteq X \equiv R \subseteq X \;\wedge\; S \subseteq X;\ (12)\} \\
& X \leqslant R \;\wedge\; R^\circ \cdot S \subseteq \ker X \;\wedge\; S^\circ \cdot R \subseteq \ker X \;\wedge\; X \leqslant S \\
\equiv & \quad \{\text{the intermediate conjuncts are the same (taking converses)}\} \\
& X \leqslant R \;\wedge\; R^\circ \cdot S \subseteq \ker X \;\wedge\; X \leqslant S \qquad \square
\end{array}
$$

Hoare logic counterparts Galois connection (20) holds with no further change once $\leqslant$ is replaced by the preorder adopted for Hoare triples (16), the reasoning being the same. Fact (22) is even simpler for such a preorder, as the third conjunct

disappears.[11] Finally, the Hoare logic counterpart of (21) is

$$q \xleftarrow{h \cdot R \cdot k^\circ} p \quad \equiv \quad wp(h,q) \xleftarrow{R} wp(k,p) \tag{23}$$

where $wp(f,p) = \mathsf{dom}\,([\![p]\!] \cdot f)$ denotes the weakest-precondition for *function* f to ensure p on the output.[12] The first steps of the proof of (23) are the same as those of (21), leading to $q \cdot h \leqslant p \cdot k \cdot R^\circ$ (abbreviating $[\![p]\!]$, $[\![q]\!]$ to p, q). But the calculation requires further reasoning in this case because predicates p, q are (relationally) partial identities (tests) and therefore are not at the same level as functions. Since domain is idempotent, dom can be added to any of R or S in $R \leqslant S$ and thus dom is a self-adjoint concerning (16):

$$\mathsf{dom}\,R \leqslant S \quad \equiv \quad R \leqslant \mathsf{dom}\,S \tag{24}$$

Then:

$$\begin{array}{ll} & q \cdot h \leqslant \mathsf{dom}\,(p \cdot k \cdot R^\circ) \\ \equiv & \{\text{domain: } \mathsf{dom}\,(R \cdot S) = \mathsf{dom}\,(\mathsf{dom}\,R \cdot S)\} \\ & q \cdot h \leqslant \mathsf{dom}\,(\mathsf{dom}\,(p \cdot k) \cdot R^\circ) \\ \equiv & \{(24)\} \\ & \mathsf{dom}\,(q \cdot h) \leqslant \mathsf{dom}\,(p \cdot k) \cdot R^\circ \\ \equiv & \{\text{weakest precondition for functions: } wp(f,p) = \mathsf{dom}\,(p \cdot f)\} \\ & wp(h,q) \leqslant wp(k,p) \cdot R^\circ \\ \equiv & \{\text{Hoare triple (15)}\} \\ & wp(h,q) \xleftarrow{R} wp(k,p) \qquad \square \end{array}$$

5. Building a type system of FDs

The machinery set up in the previous sections is enough for developing a type system whereby *dependencies, relations and operations on relations are tied together*, as envisaged by Beeri et al. [9].

Composition rule FDs on relations which matching antecedent and consequent functions (as types) compose:

$$y \xleftarrow{S \cdot R} x \quad \Leftarrow \quad y \xleftarrow{S} z \quad \wedge \quad z \xleftarrow{R} x \tag{25}$$

Proof.

$$\begin{array}{ll} & h \xleftarrow{S} g \quad \wedge \quad g \xleftarrow{R} f \\ \equiv & \{(13) \text{ twice}\} \\ & h \leqslant g \cdot S^\circ \quad \wedge \quad g \leqslant f \cdot R^\circ \\ \Rightarrow & \{\leqslant\text{-monotonicity of } (\cdot S^\circ);\ \text{converse (9)}\} \\ & h \leqslant g \cdot S^\circ \quad \wedge \quad g \cdot S^\circ \leqslant f \cdot (S \cdot R)^\circ \\ \Rightarrow & \{\leqslant\text{-transitivity}\} \\ & h \leqslant f \cdot (S \cdot R)^\circ \\ \equiv & \{(13) \text{ again}\} \\ & h \xleftarrow{S \cdot R} f \qquad \square \end{array}$$

For R and S the same database table (tuple set), this rule subsumes Armstrong axiom F5 (Transitivity) in the standard FD theory [4]. For R and S regarded as describing computations, rule (25) is the FD counterpart of the rule of composition in Hoare logic, recall (17).[13]

[11] This happens because dom distributes through union, while ker does not. Both versions of (20) are instances of a generic result concerning *Galois connection lifting*, see appendix D.6 of [14].

[12] Terms such as $h \cdot R \cdot k^\circ$ denote programs which begin by reversing a function, proceeding as R and then updating the state by another function; for instance, program `{x := x-1; R; x := 2*x}` on a single-variable state `x` is denoted by relational term $(2*) \cdot R \cdot (1+)^\circ$, for some subprogram R.

[13] The proof is the same, as both (12) and (16) are preorders (thus transitive) compatible with relational composition. Recall that $R; S = S \cdot R$.

Consequence (weakening/strengthening) rule

$$k \xleftarrow{R} h \quad \Leftarrow \quad k \leqslant g \ \wedge \ g \xleftarrow{R} f \ \wedge \ f \leqslant h \tag{26}$$

Proof. See [14], where this rule is shown to subsume and generalize standard Armstrong axioms F2 (*Augmentation*) and F4 (*Projectivity*). In the parallel with Hoare logic, it corresponds to the two *rules of consequence* [11] which, put together and writing triples as arrows, becomes

$$q' \xleftarrow{R} p' \quad \Leftarrow \quad q' \Leftarrow q \ \wedge \ q \xleftarrow{R} p \ \wedge \ p \Leftarrow p'$$

for a program R and assertions p, q, p', q'. (Note that implications $q' \Leftarrow q$ etc. correspond to $[\![q]\!] \subseteq [\![q']\!]$ and therefore to $q' \leqslant q$ once brackets $[\![\,]\!]$ are dropped for simplicity.)

Reflexivity We have seen already that

$$f \xleftarrow{id} f \tag{27}$$

holds trivially. This rule, which corresponds to the "*skip*" rule of Hoare logic, $p \xleftarrow{skip} p$ is easily shown to hold for any set T,

$$f \xleftarrow{[\![T]\!]} f \tag{28}$$

as FDs are downward closed (that is, preserved by sub-relations, by monotonicity). Rule (28) is known as Armstrong axiom F1 (*Reflexivity*).

Note in passing that (25) and (27) together define a *category* whose objects are functions (types) and whose morphisms (arrows) are FDs.

6. Type checking database operations

Merging (union) Let us proceed to an example of database operation *type checking*: we want to know what it means for the merging of two database files to satisfy a particular functional dependency $f \longrightarrow g$. That is, we want to find a *sufficient* condition for the union $R \cup S$ of two relations R and S to be of type $f \longrightarrow g$. The relational algebra of injectivity does most of the work:

$$\begin{array}{ll} & g \xleftarrow{R \cup S} f \\ \equiv & \quad \{\text{definition (13); converse distributes by union}\} \\ & g \leqslant f \cdot (R^\circ \cup S^\circ) \\ \equiv & \quad \{\text{relational composition distributes through union}\} \\ & g \leqslant f \cdot R^\circ \cup f \cdot S^\circ \\ \equiv & \quad \{\text{algebra of injectivity (22); definition (13) again, twice}\} \\ & g \xleftarrow{R} f \ \wedge \ g \xleftarrow{S} f \ \wedge \ R \cdot \ker f \cdot S^\circ \subseteq \ker g \\ \equiv & \quad \{\text{introduce "mutual dependency" shorthand}\} \\ & g \xleftarrow{R} f \ \wedge \ g \xleftarrow{S} f \ \wedge \ g \xleftarrow{R,S} f \qquad \square \end{array}$$

The "mutual dependency" shorthand $g \xleftarrow{R,S} f$ introduced in the last step for $R \cdot \ker f \cdot S^\circ \subseteq \ker g$ can be read as a generalization of the standard definition of FD to *two* relations instead of *one* — just generalize the second R in (8) to some S. For R and S two sets of tuples, it means that grabbing one tuple from one set and another tuple from the other set, if they cannot be distinguished by f then they will remain indistinguishable by g.

It should be stressed that the bottom line of the calculation expresses not only a *sufficient* but also a *necessary* condition for $g \xleftarrow{R \cup S} f$ to hold, as all steps are equivalences. Summing up, rule

$$g \xleftarrow{R \cup S} f \quad \equiv \quad g \xleftarrow{R} f \ \wedge \ g \xleftarrow{S} f \ \wedge \ g \xleftarrow{R,S} f \tag{29}$$

holds.[14]

[14] The counterpart of (29) in Hoare logic is $\{p\}R\{q\} \ \wedge \ \{p\}S\{q\} \equiv \{p\}(R \cup S)\{q\}$ written directly in the original triple notation, where $R \cup S$ denotes the non-deterministic choice between R and S.

Type checking other database operations will follow the same scheme. Below we handle in detail one particular such operation, *relational join* [4]. This is justified not only for its relevance in data processing but also because it brings about other standard FD rules not yet addressed.

Joining (pairing) Recall from Section 1 how [3] explains the relevance of Tarski's work on *pairing* in relation algebra by illustrating how a ternary (in general, n-ary) relation $\{(a,b,c),(d,e,f)\}$ gets represented by a binary one, $\{(a,(b,c)),(d,(e,f))\}$.

Pairing is not only useful for ensuring that sets of arbitrarily long (but finite) tuples are representable by binary relations but also for defining the *join* operator ($\vartriangle$) on such sets. This operator turns out to be particularly handy to formalize in case the two sets of tuples are already represented as relations R and S:

$$(a,b)(R \vartriangle S)c \quad \equiv \quad a\ R\ c \ \wedge\ b\ S\ c \tag{30}$$

Interestingly, relational join behaves as a *least upper bound* with respect to the injectivity preorder:

$$R \vartriangle S \leqslant T \quad \equiv \quad R \leqslant T \ \wedge\ S \leqslant T \tag{31}$$

This arises from fact[15]

$$\ker(R \vartriangle S) \quad = \quad \ker R \cap \ker S \tag{32}$$

as follows:

$$\begin{array}{ll} & R \vartriangle S \leqslant T \\ \equiv & \quad \{(12) \text{ and } (32)\} \\ & \ker T \subseteq (\ker R) \cap (\ker S) \\ \equiv & \quad \{\text{universal property: } X \subseteq R \cap S \equiv X \subseteq R \ \wedge\ X \subseteq S\} \\ & \ker T \subseteq \ker R \ \wedge\ \ker T \subseteq \ker S \\ \equiv & \quad \{(12) \text{ twice}\} \\ & R \leqslant T \ \wedge\ S \leqslant T \qquad \square \end{array}$$

This combinator, termed *split* in [23], *fork* in [24] and *strict fork* in [1], turns out to be more general than its use in data processing suggests. In particular, when R and S are functions f and g, $f \vartriangle g$ is the obvious function which pairs the outputs of f and g: $(f \vartriangle g)x = (f(x), g(x))$. Think for instance of the projection function f_x (resp. f_y) which, in the context of Definition 1 yields $t[x]$ (resp. $t[y]$) when applied to a tuple t. Then $(f_x \vartriangle f_y)t = (t[x], t[y]) = t[xy]$, where xy denotes the union of attributes x and y [4]. So, attribute union corresponds to joining the corresponding projection functions. This gives us a quite uniform framework for handling both relational join and compound attributes. To make notation closer to what is common in data dependency theory we will abbreviate $f_x \vartriangle f_y$ to $f_x f_y$ and this even further to xy, identifying (as we did before) each attribute (say x) with the corresponding projection function (say f_x).

Keeping abbreviation fg of $f \vartriangle g$ (for functions), from (31) it is easy to derive facts $! \leqslant f \leqslant id$, $f \leqslant fg$ and $g \leqslant fg$. This is consistent with the use of juxtaposition to denote "sets of attributes". Likewise, $\leqslant$ can be regarded as expressing "attribute inclusion" in this context: the more attributes one observes the more injective the projection function corresponding to such attributes is.[16]

A first illustration of this unified framework is given below: the (generic) calculation of the so-called Armstrong axioms F3 (*Additivity*) and F4 (*Projectivity*).[17] This is done in one go, for arbitrary (suitably typed) R, f, g, h[18]:

$$gh \xleftarrow{R} f \quad \equiv \quad g \xleftarrow{R} f \ \wedge\ h \xleftarrow{R} f \tag{33}$$

Calculation:

$$\begin{array}{ll} & gh \xleftarrow{R} f \\ \equiv & \quad \{(13); \text{ expansion of shorthand } gh\} \\ & g \vartriangle h \leqslant f \cdot R^\circ \end{array}$$

[15] Fact (32) follows immediately from $(R \vartriangle S)^\circ \cdot (X \vartriangle Y) = R^\circ \cdot X \cap S^\circ \cdot Y$ [23].

[16] Note how ! mimics the empty set and id mimics the whole set of attributes, enabling one to "see the whole thing" and thus discriminating as much as possible.

[17] See [4].

[18] In the Hoare logic counterpart of this rule, gh stands for the product $g \times h$ of predicates g and h defined by $(g \times h)(b,a) = g(b) \ \wedge\ h(a)$. The rule, derived in [15], ensures that the category of FDs has products.

$$
\begin{array}{ll}
\equiv & \{\text{universal property of } \triangle \text{ (31)}\} \\
& g \leqslant f \cdot R^\circ \ \wedge \ h \leqslant f \cdot R^\circ \\
\equiv & \{\text{(13) twice}\} \\
& g \xleftarrow{R} f \ \wedge \ h \xleftarrow{R} f \qquad \square
\end{array}
$$

The typing rule for the join $R \triangle S$ of two relations R and S is calculated in the same way. The reasoning involves properties of the projection functions $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$:

$$
\begin{array}{ll}
& g \xleftarrow{R} f \ \wedge \ h \xleftarrow{S} f \\
\Rightarrow & \{\pi_1 \cdot (R \triangle S) \subseteq R \text{ and } \pi_2 \cdot (R \triangle S) \subseteq S;\ \text{FDs are downward closed}\} \\
& g \xleftarrow{\pi_1 \cdot (R \triangle S)} f \ \wedge \ h \xleftarrow{\pi_2 \cdot (R \triangle S)} f \\
\equiv & \{\text{trading (21) twice}\} \\
& g \cdot \pi_1 \xleftarrow{R \triangle S} f \ \wedge \ h \cdot \pi_2 \xleftarrow{R \triangle S} f \\
\equiv & \{\text{F3} + \text{F4 (33)}\} \\
& (g \cdot \pi_1) \triangle (h \cdot \pi_2) \xleftarrow{R \triangle S} f \\
\equiv & \{\text{product of functions: } f \times g = (f \cdot \pi_1) \triangle (g \cdot \pi_2)\} \\
& g \times h \xleftarrow{R \triangle S} f \qquad \square
\end{array}
$$

7. Beyond the type system: database operation optimization

As explained above, FD theory (resp. Hoare logic) can be regarded as a type system whose rules help in reasoning about data models (resp. programs) without going into the semantic intricacies of data business rules (resp. program meanings).

When compared to the quantified expression of Definition 1, quantifier-free equivalent (13) looks simpler and is therefore expected to be easier to use in practice. This section gives two illustrations of this, one concerned with query optimization and the other with optimizing lexicographic sorting of database files.

FDs for query optimization This example, taken from [5], is also addressed by [25]: one wants to optimize the conjunctive query

$$\left\{(d, a') \mid (t, d, a) \in \mathit{Movies},\ (t', d', a') \in \mathit{Movies},\ t = t'\right\} \tag{34}$$

over a database file *Movies*(*Title*, *Director*, *Actor*) into a query accessing this file only once, knowing that FD *Title* → *Director* holds.

Using abbreviations M, t, d and a for (respectively) *Movies*, *Title*, *Director* and *Actor*, we want to solve for X the equation

$$d \cdot M \cdot (\ker t) \cdot M \cdot a^\circ = X \tag{35}$$

— whose left hand side is the relational equivalent of (34)[19] — aiming at a solution X containing only one instance of M. The equation is solved by taking FD $d \xleftarrow{M} t$ itself as starting point and trying to re-write it into something one recognizes as an instance of (35):

$$
\begin{array}{ll}
& d \xleftarrow{M} t \\
\equiv & \{(13)\} \\
& d \leqslant t \cdot M^\circ \\
\equiv & \{\text{expanding (11), (12); } M^\circ = M \text{ since } M \text{ is a partial identity}\} \\
& M \cdot t^\circ \cdot t \cdot M \subseteq d^\circ \cdot d \\
\equiv & \{\text{composition } (\cdot M) \text{ with a partial identity [14]}\} \\
& M \cdot t^\circ \cdot t \cdot M \subseteq d^\circ \cdot d \cdot M
\end{array}
$$

[19] As the interested reader may check by introducing the variables back. Note how $\ker t$ expresses $t = t'$ and projection functions d (for *Director*) and a (for *Actor*) work over tuple (t, d, a) and tuple (t', d', a'), respectively. The use of the same letters for data variables and the corresponding projection functions should help in comparing the two versions of the query.

$$\Rightarrow \quad \{\text{shunting (18), (19); monotonicity of } (\cdot a^\circ)\text{; kernel (11)}\}$$
$$d \cdot M \cdot (\ker t) \cdot M \cdot a^\circ \subseteq d \cdot M \cdot a^\circ \qquad \square$$

We thus find $d \cdot M \cdot a^\circ$ as a candidate solution for X. To obtain $X = d \cdot M \cdot a^\circ$ it remains to check the converse inclusion:

$$d \cdot M \cdot a^\circ \subseteq d \cdot M \cdot (\ker t) \cdot M \cdot a^\circ$$
$$\Leftarrow \quad \{id \subseteq \ker t \text{ because kernels of functions are equivalence relations}\}$$
$$d \cdot M \cdot a^\circ \subseteq d \cdot M \cdot M \cdot a^\circ$$
$$\equiv \quad \{M \cdot M = M \cap M = M \text{ because } M \text{ is a partial identity}\}$$
$$d \cdot M \cdot a^\circ \subseteq d \cdot M \cdot a^\circ \qquad \square$$

Altogether, FD $d \xleftarrow{M} t$ grants the solution $X = d \cdot M \cdot a^\circ$ to Eq. (35) — that is

$$X = \{(d, a') \mid (t, d, a') \in Movies\}$$

— which optimizes the given query by only visiting the movies file once.[20]

Optimizing lexicographic sorting This example calculates an improvement in lexicographic sorting of database files subject to FDs. Let $\leqslant$ and $\sqsubseteq$ be two preorders of the same type. By the expression $\leqslant \triangleright \sqsubseteq$ we mean the lexicographic order

$$a(\leqslant \triangleright \sqsubseteq)a' \quad \equiv \quad a \leqslant a' \ \wedge \ (a \geqslant a' \Rightarrow a \sqsubseteq a')$$

which gives priority to $\leqslant$, that is,

$$\leqslant \triangleright \sqsubseteq \quad = \quad \leqslant \cap (\leqslant^\circ \Rightarrow \sqsubseteq) \tag{36}$$

where relational implication is the upper adjoint of intersection:

$$R \cap S \subseteq X \quad \equiv \quad R \subseteq (S \Rightarrow X) \tag{37}$$

Now suppose that T is a database file whose schema includes attribute x (resp. y) whose domain is ordered by partial order $\leqslant_x$ (resp. $\leqslant_y$). Thus the tuples of T can be ordered not only by the preorders

$$t \leqslant_a^T t' \quad \equiv \quad t, t' \in T \ \wedge \ a(t) \leqslant_a a(t)' \tag{38}$$

for $a \in \{x, y\}$, but also by lexicographic combinations thereof, e.g. $\leqslant_x^T \triangleright \leqslant_y^T$. However, such lexicographic preorders can be simplified in presence of FDs. Below we calculate a sufficient condition for such a lexicographic preorder to reduce to one of its components, for instance:

$$\leqslant_x^T \triangleright \leqslant_y^T \quad = \quad \leqslant_x^T \quad \Leftarrow \quad y \xleftarrow{T} x \tag{39}$$

The relation-algebraic calculation of rule (39) goes in the same style as before[21]:

$$\leqslant_x^T \triangleright \leqslant_y^T \quad = \quad \leqslant_x^T$$
$$\equiv \quad \{(36);\ X \cap Y = X \text{ equivalent to } X \subseteq Y\}$$
$$\leqslant_x^T \subseteq ((\leqslant_x^T)^\circ \Rightarrow \leqslant_y^T)$$
$$\equiv \quad \{\text{Galois connection (37)}\}$$
$$\leqslant_x^T \cap (\leqslant_x^T)^\circ \subseteq \leqslant_y^T$$
$$\equiv \quad \{\text{variable-free versions of (38) for } a \in \{x, y\}\text{; converses}\}$$
$$T \cdot x^\circ \cdot \leqslant_x \cdot x \cdot T \cap T \cdot x^\circ \cdot \leqslant_x^\circ \cdot x \cdot T \subseteq T \cdot y^\circ \cdot \leqslant_y \cdot y \cdot T$$
$$\equiv \quad \{\text{distributions over intersection, as } x \cdot T \text{ is univalent; converses}\}$$
$$T \cdot x^\circ \cdot (\leqslant_x \cap \leqslant_x^\circ) \cdot x \cdot T \subseteq T \cdot y^\circ \cdot \leqslant_y \cdot y \cdot T$$

[20] By the way: symmetry between a and d in calculation step $d \cdot M \cdot t^\circ \cdot t \cdot M \cdot a^\circ \subseteq d \cdot M \cdot a^\circ$ above immediately tells that FD $a \xleftarrow{M} t$ would also enable the proposed optimization.

[21] The fourth step in the reasoning relies on prop. 5.3 of [1]: for univalent Q, distribution law $(R \cap S) \cdot Q = (R \cdot Q) \cap (S \cdot Q)$ holds — and therefore (taking converses) so does $Q^\circ \cdot (R \cap S) = (Q^\circ \cdot R) \cap (Q^\circ \cdot S)$.

$$
\begin{array}{ll}
\equiv & \{\leqslant_x \text{ is antisymmetric: } \leqslant_x \cap \leqslant_x^\circ = id\} \\
& T \cdot x^\circ \cdot x \cdot T \subseteq y^\circ \cdot \leqslant_y \cdot y \\
\Leftarrow & \{\leqslant_y \text{ is reflexive}\} \\
& T \cdot x^\circ \cdot x \cdot T \subseteq y^\circ \cdot y \\
\equiv & \{(6); (13)\} \\
& y \xleftarrow{T} x \qquad \square
\end{array}
$$

8. Conclusions

> "*The great merit of algebra is as a powerful tool for exploring family relationships over a wide range of different theories.* (...) *It is only their algebraic properties that emphasize the family likenesses* (...) *Algebraic proofs by term rewriting are the most promising way in which computers can assist in the process of reliable design.*"
>
> [Hoare and Jifeng [13]]

There is a growing interest in algebraic reasoning in computer science able to eventually promote calculational techniques in software engineering, hopefully unifying seemingly disparate theories once they are encoded into the same abstractions. Relation algebra [1] is particularly apt in this respect.

The current paper shows how a relation-algebraic rendering of both data dependency theory and Hoare logic purports one such unification, in spite of the latter being an algorithmic theory and the former a data theory, thanks to both algorithms and data structures being expressed in the *unified language of binary relations*.

In short (and informally), both logics rely on triples: *something* (a data set; a program) lies between an *antecedent* and a *consequent* observation (a data attribute; a state assertion); there is an ordering (injectivity; definition) on observations; triples express that antecedent observations are "enough" for consequents to hold "modally through" what is in between.

Triples are nicely captured by *arrows*, whose end-points can be regarded as *types*. On the data side, our approach equips relational data with *functional types* and an associated type system which can be used to *type check* database operations and optimize queries by calculation once they are written as Tarskian, quantifier-free formulas.

As formal verification is becoming more and more widespread to ensure quality in complex software systems, we believe our approach may contribute to *unified* formal verification tools blending in the same framework extended static checking and database programming.

Back to the opening story, surely Tarski's work on satisfaction and truth is relevant to computer science. But Etchemendy's answer could have been better tuned to the particular context of database technology suggested by the Oracle towers landscape:

> [...] "*They would never have been built without Tarski's work on the calculus of binary relations.*"

9. Related and future work

Functional dependencies have been characterized relationally by checking the determinism of the relations obtained by projecting tuple sets by antecedent and consequent attributes [26,27]. This alternative definition is equivalent to the one followed in the current paper.[22] As a generalization, Jaoua et al. [27] also study so-called *difunctional* dependencies.

Dependencies in relational databases have also been expressed using so-called *indiscernibility relations* [28]. Freyd and Scedrov [16] develop a τ-category theory of relations based on *monic n-tuples*. Concepts such as *table*, *column*, *short column* etc. fit into the spirit of (pointfree) data dependency and database theory and should be carefully studied in the context of the current paper.

Wisnesky [25] addresses the semantic optimization of monad comprehensions in functional programming by generalizing results from relational database theory. As this theory relies on the *powerset* monad, whose comprehensions correspond to database queries, by handling similar optimizations in relation algebra (as we did in Section 7) we have followed the well-known shift towards the Kleisli adjoint category.

As is well-known, this shift can be generalized to any other monad. Wisnesky [25] includes queries on probabilistic databases, this time relying on the (finite support) *distribution* monad. As shown by Oliveira [29], the "Kleisli shift" w.r.t. this monad leads to typed *linear algebra*. Wong and Butz [30] introduce Bayesian embedded multivalued dependencies as necessary and sufficient conditions for lossless decomposition of probabilistic relations. Lossless decomposition and multivalued dependencies have been handled by Oliveira [14] in the same way as FDs in the current paper. The prospect of calculating with data dependencies in probabilistic systems *directly* in matrix algebra is an interesting prospect for future

[22] See section *Generic relational projections* in [14].

work, in line with the consistent use of matrix notation by Schmidt [1] in relation algebra. Whether this carries over to probabilistic Hoare logic [31] remains to be seen.

Other ways of relating data dependency theory with algorithmic reasoning can be devised. For instance, Mili et al. [12] reason about while-loops $w = (\textbf{while}\ t\ \textbf{do}\ b)$ in terms of so-called *strongest invariant functions*, where invariant functions f, ordered by injectivity, are such that $f \cdot [\![t]\!] = f \cdot b \cdot [\![t]\!]$ holds. A simple argument in relation algebra shows this equivalent to $f \cdot b \cdot [\![t]\!] \subseteq f$, thus entailing FD $f \xleftarrow{b \cdot [\![t]\!]} f$. How much of our FD relation-algebraic approach could be applied in this setting is open to research. This includes finding a meaningful counterpart of the *rule of iteration* [11] at data level, a topic not addressed in the current paper.

Another law not considered in the correspondence between Hoare logic and functional dependencies is the *axiom of assignment*, $\{p[e/x]\}x := e\{p\}$ where $p[e/x]$ denotes the predicate which is obtained from p by replacing all occurrences of x by e. This axiom is interesting because it relies on the state structure of imperative programs: variables which hold data. Assignment $x := e$ means *selective updating*: program variable x is updated to e. A possible data-level counterpart to such selective updating is the SQL `update` command, which is of the form `UPDATE` R `SET` f `WHERE` x, meaning: *update all tuples in R which satisfy selection criterion x by tuple-transformation f, leaving the rest unchanged*. While the semantics of this operation is easy to encode in relation algebra, the parallel is somewhat artificial and needs further analysis. In general, future work should identify which generic properties of the $\leqslant$ relation on types are common to both frameworks and derive a more general kernel theory which both are instances of.

Last but not least, another prospect for future work concerns automated reasoning. RelView [32] is a well-known system that calculates with relations "beyond toy size". Many applications of relation algebra have been handled successfully in this tool. Algebraic structures such as idempotent semirings and Kleene algebras (which relation algebra is an instance of) have also been shown to be amenable to automation by e.g. Höfner and Struth [8] and Struth [7]. Möller et al. [33] encode a database preference theory into idempotent semiring algebra and show how to use Prover9 to discharge proofs. Model checking of extended static checks in tools such as e.g. the Alloy Analyser also blends well with algebraic-relational models [34].

The implementation of a generic, unified approach to both data and program theories on top of libraries already available in such automated deduction systems is a prospect for long term research.

Acknowledgements

This work is funded by ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the *FCT – Fundação para a Ciência e a Tecnologia* (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-028923.

The author would like to thank the anonymous referees for insightful comments which significantly improved the quality of the original submission. Feedback and exchange of ideas with Ryan Wisnesky about pointfree query reasoning are also gratefully acknowledged.

References

[1] G. Schmidt, Relational Mathematics, in: Encyclopedia of Mathematics and Its Applications, vol. 132, Cambridge University Press, 2010.
[2] S. Feferman, Tarski's influence on computer science, Log. Methods Comput. Sci. 2 (2006), 1–1–13.
[3] J. Bussche, Applications of Alfred Tarski's ideas in database theory, in: CSL'01: Proceedings of the 15th International Workshop on Computer Science Logic, Springer-Verlag, London, UK, 2001, pp. 20–37.
[4] D. Maier, The Theory of Relational Databases, Computer Science Press, 1983.
[5] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison–Wesley, 1995.
[6] A. Tarski, S. Givant, A Formalization of Set Theory without Variables, AMS Colloquium Publications, vol. 41, American Mathematical Society, Providence, Rhode Island, 1987.
[7] G. Struth, Isabelle repository for relational and algebraic methods, URL: http://staffwww.dcs.shef.ac.uk/people/G.Struth/isa, 2011.
[8] P. Höfner, G. Struth, Automated reasoning in Kleene algebra, in: Proceedings, CADE-21, Springer-Verlag, 2007, pp. 279–294.
[9] C. Beeri, R. Fagin, J. Howard, A complete axiomatization for functional and multivalued dependencies in database relations, in: D. Smith (Ed.), Proc. 1977 ACM SIGMOD, Toronto, ACM, NY, USA, 1977, pp. 47–61.
[10] C. Flanagan, K. Leino, M. Lillibridge, G. Nelson, J. Saxe, R. Stata, Extended static checking for Java, in: PLDI, 2002, pp. 234–245.
[11] C. Hoare, An axiomatic basis for computer programming, Commun. ACM 12 (10) (1969) 576–580, 583.
[12] A. Mili, J. Desharnais, J. Gagné, Strongest invariant functions: Their use in the systematic analysis of while statements, Acta Inform. 22 (1985) 47–66.
[13] C. Hoare, H. Jifeng, Unifying Theories of Programming, Series in Computer Science, Prentice-Hall International, 1998.
[14] J. Oliveira, Pointfree foundations for (generic) lossless decomposition, Technical Report TR-HASLab:3:2011, HASLab/INESC TEC & U. Minho, 2011, URL: https://repositorium.sdum.uminho.pt/handle/1822/24648.
[15] J. Oliveira, Extended Static Checking by Calculation using the Pointfree Transform, LNCS, vol. 5520, Springer-Verlag, 2009, pp. 195–251.
[16] P. Freyd, A. Scedrov, Categories, Allegories, Mathematical Library, vol. 39, North-Holland, 1990.
[17] H. Doornbos, R. Backhouse, J. van der Woude, A calculational approach to mathematical induction, Theor. Comput. Sci. 179 (1997) 103–135.
[18] D. Kozen, Kleene algebra with tests, ACM Trans. Program. Lang. Syst. 19 (1997) 427–443.
[19] D. Kozen, On Hoare logic and Kleene algebra with tests, ACM Trans. Comput. Log. 1 (2000) 60–76.
[20] J. Desharnais, B. Möller, G. Struth, Kleene algebra with domain, ACM Trans. Comput. Log. 7 (2006) 798–833.
[21] I. Wehrman, C. Hoare, P.W. O'Hearn, Graphical models of separation logic, Inf. Process. Lett. 109 (2009) 1001–1004.
[22] C. Hoare, J. He, The weakest prespecification, Inf. Process. Lett. 24 (1987) 127–132.
[23] R. Bird, O. de Moor, Algebra of Programming, Series in Computer Science, Prentice-Hall International, 1997.
[24] M. Frias, G. Baum, A. Haeberer, Fork algebras in algebra, logic and computer science, Fundam. Inform. (1997) 1–25.

[25] R. Wisnesky, Minimizing Monad comprehensions, Technical Report TR-02-11, Harvard University, Cambridge, Massachusetts, 2011.
[26] G. Schmidt, T. Ströhlein, Relations and Graphs: Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1993.
[27] A. Jaoua, N. Belkhiter, H. Ounalli, T. Moukam, Databases, in: C. Brink, W. Kahl, G. Schmidt (Eds.), Relational Methods in Computer Science, Springer-Verlag New York, Inc., New York, NY, USA, 1997, pp. 197–210.
[28] H. Okuma, W. MacCaull, Y. Kawahara, Informational representability for contexts in Dedekind categories, Technical Report trcs208, Kyushu University, 2003, URL: http://www.i.kyushu-u.ac.jp/doitr/trcs208.ps.gz.
[29] J. Oliveira, Towards a linear algebra of programming, Form. Asp. Comput. 24 (2012) 433–458.
[30] S. Wong, C. Butz, The implication of probabilistic conditional independence and embedded multivalued dependency, in: 8th Conf. on Inf. Processing and Management of Uncertainty in K.-B. Systems, IPMU00, 2000, pp. 876–881.
[31] G. Barthe, B. Grégoire, S. Béguelin, Probabilistic relational Hoare logics for computer-aided security proofs, in: MPC'12, 2012, pp. 1–6.
[32] R. Berghammer, Computing and visualizing Banks sets of dominance relations using relation algebra and RelView, J. Log. Algebr. Program. 82 (2013) 123–136.
[33] B. Möller, P. Roocks, M. Endres, An algebraic calculus of database preferences, in: MPC 2012, in: LNCS, vol. 7342, Springer, Berlin, Heidelberg, 2012, pp. 241–262.
[34] J. Oliveira, M. Ferreira, Alloy meets the algebra of programming: A case study, IEEE Trans. Softw. Eng. 39 (2013) 305–326.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 263–283

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Relations into algebras of probabilistic distributions

Norihiro Tsumagari [a,*], Hitoshi Furusawa [b], Yasuo Kawahara [c,1]

[a] Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-8502, Japan
[b] Department of Mathematics and Computer Science, Kagoshima University, Kagoshima 890-0065, Japan
[c] Kyushu University, Fukuoka 819-0395, Japan

ARTICLE INFO

Article history:
Available online 12 February 2014

Keywords:
Algebras of probabilistic distributions
Convex relations
Associativity and distributivity of convex composition
Relational calculus

ABSTRACT

The paper proposes two types of convex relations into algebras of probabilistic distributions as a relational algebraic foundation of semantic domains of probabilistic systems [4,7,8]. Following previous results by Tsumagari [16], we particularly focus on the associative law for the convex compositions defined via bounded combinations of probabilistic distributions, and prove that the convex compositions are associative for convex relations.

1. Introduction

The concept of rings is basic in mathematics as a framework of numbers. Recently from a view point on algebraic study [1] of semantic domains for distributed algorithms, the importance of variants of rings, such as Kleene algebras [5] and idempotent semirings, has been increased. It is well known that the set of all binary relations on a set forms a typical example of complete idempotent semirings.

When constructing a concrete model of semirings with preferable properties, we have to first focus our attention on the associativity of possible composition. For relations $\alpha : X \rightharpoonup Y$ and $\beta : Y \rightharpoonup Z$ the ordinary composite $\alpha\beta : X \rightharpoonup Y$ is defined as

$$(x, z) \in \alpha\beta \quad \leftrightarrow \quad \exists y \in Y.\ (x, y) \in \alpha \wedge (y, z) \in \beta.$$

Of course, the ordinary composition of relations is associative. A multirelation is a relation of a form $\alpha : X \rightharpoonup \wp(Y)$, where $\wp(X)$ denotes the power set of X. Depending on applications, two definitions of composition of multirelations are known. One of them is called the *reachability composition* studied by Peleg [13] and Goldblatt [3] for concurrent dynamic logic. The reachability composition $\alpha \cdot \beta$ of multirelations $\alpha : X \rightharpoonup \wp(Y)$ and $\beta : Y \rightharpoonup \wp(Z)$ is defined by

$$(x, T) \in \alpha \cdot \beta \quad \leftrightarrow \quad \exists U \in \wp(Y).\ \left[(x, U) \in \alpha \wedge \exists \{T_y\}_{y \in U} \subseteq \wp(Z).\ \forall y \in U.\ (y, T_y) \in \beta \wedge T = \bigcup_{y \in U} T_y\right].$$

Another composition of multirelations is given by Parikh and Rewitzky. Their composition $\alpha; \beta : X \rightharpoonup \wp(Z)$ of multirelations $\alpha : X \rightharpoonup \wp(Y)$ and $\beta : Y \rightharpoonup \wp(Z)$ is defined by

$$(x, T) \in \alpha; \beta \quad \leftrightarrow \quad \exists U \in \wp(Y).\ \left[(x, U) \in \alpha \wedge \forall y \in U.\ (y, T) \in \beta\right].$$

* Corresponding author.
E-mail addresses: ntsuma@kurims.kyoto-u.ac.jp (N. Tsumagari), furusawa@sci.kagoshima-u.ac.jp (H. Furusawa), kawahara@i.kyushu-u.ac.jp (Y. Kawahara).
[1] Professor Emeritus.

http://dx.doi.org/10.1016/j.jlap.2014.02.014
2352-2208/

It is readily seen that the definition of $\alpha;\beta$ is making use of the membership relation and residual composition. For the associativity of this composition we need a condition called *up-closed*. Up-closed multirelations provide a model of Parikh's game logic [11,12]. Rewitzky [6,14] studied them as a semantic domain of predicate transformer semantics of nondeterministic programming language. Further Nishizawa, Tsumagari and Furusawa [10] demonstrated that the set of all up-closed multirelations forms a complete idempotent left semiring (complete IL-semiring) introduced by Möller [9].

On the other hand, McIver et al. [4,7,8] introduced a semantic domain of probabilistic programs and probabilistic Kleene algebra, and indicated that probabilistic Kleene algebras are useful to simplify a model of probabilistic distributed systems. Based on their works, Tsumagari [16] initially introduced two probabilistic (non-numerical) models of complete IL-semirings with the set of maps from a set into the unit interval $[0,1]$, and studied *probabilistic multirelations* and the point-wise convexity of them. The point-wise convexity plays an important rôle for both models to satisfy the associativity of composition.

The aim of the paper is to expand Tsumagari's work [16] and to give a relational foundation for relations into algebras of probabilistic distributions. Following his work we will reformulate probabilistic multirelations as certain convex relations, together with stepwise refinement. Then we will clarify how the convexity works in the associativity of composition of convex relations.

In Section 2 we review the basic properties of algebras consisting of maps from a set into the unit interval $[0,1]$ together with scalar products, multiplications and bounded sums. Section 3 studies convex combinations of probabilistic distributions. In Section 4 we introduce convex composition of relations into algebras of probabilistic distributions by using convex combinations. In Section 5 we show the associative law of convex composition. Section 6 introduces two types of convex relations, and study the distributive laws of convex composition over joins. Section 7 summarizes this work.

Notation. In the paper we will denote by I a singleton set. A (binary) relation α from a set X to a set Y, written $\alpha : X \rightharpoonup Y$, is a subset $\alpha \subseteq X \times Y$. The empty relation $\emptyset_{XY} : X \rightharpoonup Y$ and the universal relation $\nabla_{XY} : X \rightharpoonup Y$ are defined by $\emptyset_{XY} = \emptyset$ and $\nabla_{XY} = X \times Y$, respectively. The converse of a relation $\alpha : X \rightharpoonup Y$ is denoted by $\alpha^\sharp$. The identity relation $\{(x,x) \mid x \in X\}$ over X is denoted by id_X. The ordinary composition of relations (which include functions) will be denoted by juxtaposition. For example, the composite of a relation $\alpha : X \rightharpoonup Y$ followed by $\beta : Y \rightharpoonup Z$ is denoted by $\alpha\beta$, and of course the composition of functions $f : X \to Y$ and $g : Y \to Z$ by fg. Also the traditional notation $f(x)$ will be written by xf as a composite of functions $x : I \to X$ and $f : X \to Y$. However, the evaluation of a map $p : X \to [0,1]$ at $x \in X$ will be expressed by $p_{[x]} \in [0,1]$. Note that the symbols of multiplication for reals and the ordinary composition of relations are omitted. Some proofs refer the point axiom (PA) and the Dedekind formula (DF_*), i.e.

$$\text{(PA)} \quad \bigsqcup_{x \in X} x = \nabla_{IX},$$

$$(\mathrm{DF}_*) \quad \alpha\beta \sqcap \gamma \sqsubseteq \big(\alpha \sqcap \gamma\beta^\sharp\big)\big(\beta \sqcap \alpha^\sharp\gamma\big),$$

where $x \in X$ is identified as a function $x : I \to X$. Note that (PA) is equivalent to $\mathrm{id}_X = \bigsqcup_{x \in X} x^\sharp x$ and that so is (DF_*) to $\alpha\beta \sqcap \gamma \sqsubseteq \alpha(\beta \sqcap \alpha^\sharp\gamma)$. See [15] for more details on basic properties of relations.

2. Maps to the unit interval

We consider maps from a set X to the unit interval $[0,1]$. Such a map $p : X \to [0,1]$ is often called a fuzzy set. The support $\lfloor p \rfloor$ of a map p is the subset of X defined by $\lfloor p \rfloor = \{x \in X \mid p_{[x]} > 0\}$. The set of all maps from X to $[0,1]$ will be denoted by $\mathcal{Q}(X)$. As we will discuss later, maps in $\mathcal{Q}(X)$ will be restricted as to be probabilistic (sub-)distributions. The point-wise order $\leqslant$ on $\mathcal{Q}(X)$ is a binary relation such that

$$p \leqslant q \quad \leftrightarrow \quad \forall x \in X.\ p_{[x]} \leqslant q_{[x]}$$

for $p, q \in \mathcal{Q}(X)$. For a real $k \in [0,1]$ a map $k_X : X \to [0,1]$ such that $k_{X[x]} = k$ for all $x \in X$ is called the *constant map over* X with value k. For $a \in X$ define a map $\dot{a} : X \to [0,1]$ by

$$\dot{a}_{[x]} = \begin{cases} 1 & \text{if } x = a, \\ 0 & \text{otherwise.} \end{cases}$$

The constant maps 0_X and 1_X over X are the least and the greatest elements of $\mathcal{Q}(X)$, respectively.

We introduce the following operators to discuss algebras of probabilistic distributions. For a real $k \in [0,1]$ and maps $p, q \in \mathcal{Q}(X)$ we define maps $k \cdot p$, $p * q$, $p \oplus q \in \mathcal{Q}(X)$ by

$$(k \cdot p)_{[x]} = kp_{[x]}, \qquad (p * q)_{[x]} = p_{[x]}q_{[x]}$$

and

$$(p \oplus q)_{[x]} = \min\{p_{[x]} + q_{[x]}, 1\}$$

for all $x \in X$, respectively. The set $\mathcal{Q}(X)$ forms an algebra called a prering.

Proposition 1. *Let $p, q \in \mathcal{Q}(X)$ and $k, k' \in [0,1]$. Then the following hold*:

(a) $(p * q) * r = p * (q * r)$, $p * q = q * p$,
(b) $(p \oplus q) \oplus r = p \oplus (q \oplus r)$, $p \oplus q = q \oplus p$,
(c) $k \cdot p = k_X * p$, $0 \cdot p = 0_X$, $1 \cdot p = p$, $(kk') \cdot p = k \cdot (k' \cdot p)$,
(d) $p \oplus 0_X = p$, $p \oplus 1_X = 1_X$,
(e) *If* $q_{[x]} + r_{[x]} \leqslant 1$ *for all* $x \in X$, *then* $p * (q \oplus r) = (p * q) \oplus (p * r)$,
(f) *If* $k + k' \leqslant 1$, *then* $(k + k') \cdot p = (k \cdot p) \oplus (k' \cdot p)$,
(g) $p \leqslant p' \wedge q \leqslant q' \to p \oplus q \leqslant p' \oplus q' \wedge p * q \leqslant p' * q'$.

Proof is omitted. □

In general the distributive laws $p * (q \oplus r) = (p * q) \oplus (p * r)$ and $k \cdot (q \oplus r) = (k \cdot q) \oplus (k \cdot r)$ do not always hold.
The following proposition shows the basic properties about the support of maps in $\mathcal{Q}(X)$.

Proposition 2. *Let $p, q, r \in \mathcal{Q}(X)$ and $k \in [0,1]$. Then the following hold:*

(a) $\lfloor 0_X \rfloor = \emptyset$, *and* $\lfloor k_X \rfloor = X$ *if* $k > 0$,
(b) $\lfloor \dot{a} \rfloor = \{a\}$,
(c) $\lfloor p * q \rfloor = \lfloor p \rfloor \cap \lfloor q \rfloor$,
(d) $\lfloor p \oplus q \rfloor = \lfloor p \rfloor \cup \lfloor q \rfloor$.

Proof is omitted. □

Proposition 3. *Let $p, q \in \mathcal{Q}(X)$. Then*

$$q \leqslant p \quad \leftrightarrow \quad \exists t \in \mathcal{Q}(X).\ q = p * t.$$

Proof.
($\leftarrow$) $q_{[x]} = p_{[x]} t_{[x]} \leqslant p_{[x]}$ since $t_{[x]}, p_{[x]} \in [0,1]$.
($\rightarrow$) Assume $q \leqslant p$. Define a map $t : X \to [0,1]$ by

$$\forall x \in X.\ t_{[x]} = \begin{cases} \frac{q_{[x]}}{p_{[x]}} & \text{if } p_{[x]} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Then it is trivial that $q = p * t$. □

The *sum* of a map $p \in \mathcal{Q}(X)$ is the least upper bound of the set $\{\sum_{x \in F} p_{[x]} \mid F : \text{finite subset of } X\}$, which is denoted by $\|p\|$. It is well known that the sum $\|p\|$ of p exists iff the above set is bounded. Also $\lfloor p \rfloor$ is a countable subset of X if $\|p\|$ exists. (For each positive integer n define a subset $\lfloor p \rfloor_n$ of X by $\lfloor p \rfloor_n = \{x \in X \mid 1/n < p_{[x]}\}$. Then each $\lfloor p \rfloor_n$ has finite (at most $(n-1) \cdot |n|$) members and so $\lfloor p \rfloor = \bigcup_{n>0} \lfloor p \rfloor_n$ is countable.)
We define three types of maps in $\mathcal{Q}(X)$ which are used in this paper.

Definition 1. Three subsets of $\mathcal{Q}(X)$ are defined as follows:

(a) $p \in \mathcal{Q}_0(X) \leftrightarrow p \in \mathcal{Q}(X) \wedge \|p\| \leqslant 1$,
(b) $p \in \mathcal{Q}_1(X) \leftrightarrow p \in \mathcal{Q}(X) \wedge \|p\| = 1$,
(c) $p \in \mathcal{Q}_*(X) \leftrightarrow p \in \mathcal{Q}_0(X) \wedge \lfloor p \rfloor : \text{finite subset of } X$.

Note that $0_X \in \mathcal{Q}_*(X)$, $\dot{x} \in \mathcal{Q}_*(X) \cap \mathcal{Q}_1(X)$ and $p * q \in \mathcal{Q}_*(X)$ for all $x \in X$, $p, q \in \mathcal{Q}_*(X)$. Elements of $\mathcal{Q}_*(X)$ are probabilistic sub-distributions, and those of $\mathcal{Q}_*(X) \cap \mathcal{Q}_1(X)$ are probabilistic distributions. Essentially, McIver et al. [4,7,8] have studied either the case of $\mathcal{Q}_*(X)$ or the case of finite set X in order to develop models of probabilistic systems.
The restriction of the point-wise order $\leqslant : \mathcal{Q}(X) \rightharpoonup \mathcal{Q}(X)$ onto $\mathcal{Q}_\tau(X)$ is denoted by $\xi_X^\tau : \mathcal{Q}_\tau(X) \rightharpoonup \mathcal{Q}_\tau(X)$, that is,

$$\forall p, q \in \mathcal{Q}_\tau(X).\ (p, q) \in \xi_X^\tau \quad \leftrightarrow \quad p \leqslant q,$$

where the subscript/superscript τ is one of 0, 1, and $*$. Remark that the restricted order ξ_X^τ on $\mathcal{Q}_1(X)$ is discrete, that is, for $\xi_X^\tau = \mathrm{id}_{\mathcal{Q}_1(X)}$. Thus the order on $\mathcal{Q}_1(X)$ will not be used in the rest of the paper.
For $\tau \in \{0, *\}$ every map $t \in \mathcal{Q}(X)$ yields a map $t_* : \mathcal{Q}_\tau(X) \to \mathcal{Q}_\tau(X)$ by

$$\forall p \in \mathcal{Q}_\tau(X).\ p t_* = p * t.$$

Corollary 1. $(\xi_X^\tau)^\sharp = \bigsqcup_{t\in\mathcal{Q}(X)} t_*$ *for* $\tau \in \{0, *\}$.

Proof.

$$\begin{aligned}
(p,q) \in \left(\xi_X^\tau\right)^\sharp &\leftrightarrow q \leqslant p \\
&\leftrightarrow \exists t \in \mathcal{Q}(X).\, q = p * t = pt_* \quad \{\text{Proposition 3}\} \\
&\leftrightarrow \exists t \in \mathcal{Q}(X).\, (p,q) \in t_* \\
&\leftrightarrow (p,q) \in \bigsqcup_{t\in\mathcal{Q}(X)} t_*. \qquad \square
\end{aligned}$$

A map $e_X : X \to \mathcal{Q}_\tau(X)$ is defined by $xe_X = \dot{x}$ for each $x \in X$, where $\tau \in \{0, 1, *\}$. Also, for $\tau \in \{0, *\}$ we define a relation $\varepsilon_X^\tau : X \multimap \mathcal{Q}_\tau(X)$ by $\varepsilon_X^\tau = e_X(\xi_X^\tau)^\sharp$. As discussed in detail later, e_X and ε_X^τ are the units of convex composition over certain convex relations, respectively.

3. Convex combinations

Extending finite bounded sums

$$\bigoplus_{j=1}^{n} q_j = q_1 \oplus q_2 \oplus \cdots \oplus q_n$$

of maps $q_1, q_2, \ldots, q_n \in \mathcal{Q}(X)$, we will define the bounded sum of an arbitrary set of maps in $\mathcal{Q}(X)$. For a set $\{q_j \mid j \in J\}$ of maps in $\mathcal{Q}(Y)$ define a map $\bigoplus_{j\in J} q_j$ in $\mathcal{Q}(Y)$ by

$$\forall y \in Y.\, \left(\bigoplus_{j\in J} q_j\right)_{[y]} = \begin{cases} \sum_{j\in J} (q_j)_{[y]} & \text{if } \sum_{j\in J} (q_j)_{[y]} \leqslant 1, \\ 1 & \text{otherwise.} \end{cases}$$

Of course, we mean $(\bigoplus_{j\in J} q_j)_{[y]} = 1$ even if the sum $\sum_{j\in J} (q_j)_{[y]}$ diverges.

The support of bounded sums of a set of maps in $\mathcal{Q}(X)$ is given by the union of supports of their maps contained in the set.

Proposition 4. *For all subsets* $\{q_j \mid j \in J\} \subseteq \mathcal{Q}(X)$ *the following holds*:

$$\left\lfloor \bigoplus_{j\in J} q_j \right\rfloor = \bigcup_{j\in J} \lfloor q_j \rfloor.$$

Proof.

$$\begin{aligned}
y \notin \left\lfloor \bigoplus_{j\in J} q_j \right\rfloor &\leftrightarrow \left(\bigoplus_{j\in J} q_j\right)_{[y]} = 0 \\
&\leftrightarrow \sum_{j\in J} (q_j)_{[y]} = 0 \\
&\leftrightarrow \forall j \in J.\, (q_j)_{[y]} = 0 \\
&\leftrightarrow \forall j \in J.\, y \notin \lfloor q_j \rfloor \\
&\leftrightarrow \neg\left[\exists j \in J.\, y \in \lfloor q_j \rfloor\right] \\
&\leftrightarrow y \notin \bigcup_{j\in J} \lfloor q_j \rfloor. \qquad \square
\end{aligned}$$

Let $f : X \to \mathcal{Q}(Y)$ be a map. Define a map $f_\diamond : \mathcal{Q}(X) \to \mathcal{Q}(Y)$ by

$$pf_\diamond = \bigoplus_{x\in X} p_{[x]} \cdot (xf)$$

where $p \in \mathcal{Q}(X)$. The map $pf_\diamond$ is called a *convex combination* of p and f. We need this notion to raise the composition of convex relations.

Example 1. Set $X = \{x, y\}$ and define maps f, f', g, $h : X \to \mathcal{Q}(X)$ by $xf = yf = \dot{x}$, $xf' = yf' = \dot{y}$, $xg = \dot{x}$, $yg = 0_X$, $xh = \dot{y}$ and $yh = 0_X$. We have $f_\diamond$, $f'_\diamond$, $g_\diamond$ and $h_\diamond$ such that

$$pf_\diamond = p_{[x]} \cdot \dot{x} \oplus p_{[y]} \cdot \dot{x}, \qquad pf'_\diamond = p_{[x]} \cdot \dot{y} \oplus p_{[y]} \cdot \dot{y}, \qquad pg_\diamond = p_{[x]} \cdot \dot{x}, \qquad ph_\diamond = p_{[x]} \cdot \dot{y},$$

for all $p \in \mathcal{Q}(X)$. Especially for $p' \in \mathcal{Q}_1(X)$, $p' f_\diamond = \dot{x}$ and $p' f'_\diamond = \dot{y}$ hold.

The basic properties of convex combinations are listed below.

Proposition 5. *Let $k \in [0, 1]$, $p \in \mathcal{Q}_0(X)$ and $f : X \to \mathcal{Q}(Y)$. Then the followings hold*:

(a) $(pf_\diamond)_{[y]} = \sum_{x \in X} p_{[x]} (xf)_{[y]}$,
(b) $\|pf_\diamond\| = \sum_{x \in X} p_{[x]} \|xf\|$,
(c) $\lfloor pf_\diamond \rfloor = \bigcup_{x \in \lfloor p \rfloor} \lfloor xf \rfloor$,
(d) $0_X f_\diamond = 0_Y$,
(e) $\dot{x} f_\diamond = xf$,
(f) $p(e_X)_\diamond = p$,
(g) $p(\nabla_{XI} k_Y)_\diamond = (k\|p\|)_Y$,
(h) $k \cdot (pf_\diamond) = (k \cdot p) f_\diamond$.

Proof.
(a) $(pf_\diamond)_{[y]} = \sum_{x \in X} p_{[x]} (xf)_{[y]}$:

$$\begin{aligned} \sum_{x \in X} p_{[x]} (xf)_{[y]} &\leqslant \sum_{x \in X} p_{[x]} && \{xf \in \mathcal{Q}(Y)\} \\ &\leqslant 1 && \{p \in \mathcal{Q}_0(X)\}, \end{aligned}$$

$$\begin{aligned} (pf_\diamond)_{[y]} &= \min\left\{ \sum_{x \in X} p_{[x]} (xf)_{[y]}, 1 \right\} \\ &= \sum_{x \in X} p_{[x]} (xf)_{[y]}. \end{aligned}$$

(b) $\|pf_\diamond\| = \sum_{x \in X} p_{[x]} \|xf\|$:

$$\begin{aligned} \|pf_\diamond\| &= \sum_{y \in Y} (pf_\diamond)_{[y]} \\ &= \sum_{y \in Y} \sum_{x \in X} p_{[x]} (xf)_{[y]} && \{(a)\} \\ &= \sum_{x \in X} p_{[x]} \sum_{y \in Y} (xf)_{[y]} \\ &= \sum_{x \in X} p_{[x]} \|xf\|. \end{aligned}$$

(c) $\lfloor pf_\diamond \rfloor = \bigcup_{x \in \lfloor p \rfloor} \lfloor xf \rfloor$:

$$\begin{aligned} \lfloor pf_\diamond \rfloor &= \bigcup_{x \in X} \lfloor p_{[x]} \cdot (xf) \rfloor \\ &= \bigcup_{x \in \lfloor p \rfloor} \lfloor xf \rfloor. \end{aligned}$$

(d) $0_X f_\diamond = 0_Y$:

$$\begin{aligned} 0_X f_\diamond &= \bigoplus_{x \in X} (0_X)_{[x]} \cdot (xf) \\ &= \bigoplus_{x \in X} 0 \cdot (xf) \\ &= 0_Y && \{0 \cdot q = 0_Y \text{ if } q \in \mathcal{Q}(Y)\}. \end{aligned}$$

(e) $\dot{x}f_\diamond = xf$:

$$\begin{aligned}\dot{x}f_\diamond &= \bigoplus_{x'\in X} \dot{x}_{[x']} \cdot \left(x' f\right)\\ &= xf.\end{aligned}$$

(f) $p(e_X)_\diamond = p$:

$$\begin{aligned}\left(p(e_X)_\diamond\right)_{[x]} &= \sum_{x'\in X} p_{[x']}\left(x' e_X\right)_{[x]} \quad \{(a)\}\\ &= \sum_{x'\in X} p_{[x']}\dot{x}'_{[x]}\\ &= p_{[x]}.\end{aligned}$$

(g) $p(\nabla_{XI}k_Y)_\diamond = (k\|p\|)_Y$:

$$\begin{aligned}p(\nabla_{XI}k_Y)_\diamond &= \bigoplus_{x\in X} p_{[x]} \cdot (x\nabla_{XI}k_Y)\\ &= \bigoplus_{x\in X} p_{[x]} \cdot k_Y \qquad \{x\nabla_{XI} = \mathrm{id}_I\}\\ &= \left(\sum_{x\in X} p_{[x]}\right) \cdot k_Y \qquad \left\{p \in \mathcal{Q}_0(X),\ (a)\right\}\\ &= \|p\| \cdot k_Y\\ &= \left(\|p\|k\right)_Y \qquad \left\{k'\cdot k_Y = \left(k'k\right)_Y\right\}.\end{aligned}$$

(h) $k \cdot (pf_\diamond) = (k \cdot p) f_\diamond$:

$$\begin{aligned}\left(k \cdot (pf_\diamond)\right)_{[y]} &= k(pf_\diamond)_{[y]}\\ &= k\sum_{x\in X} p_{[x]}(xf)_{[y]} \qquad \{(a)\}\\ &= \sum_{x\in X}(k \cdot p)_{[x]}(xf)_{[y]}\\ &= (k \cdot p) f_\diamond. \qquad \square\end{aligned}$$

The convex combination also satisfies the following properties.

Proposition 6. *For $\tau \in \{0, 1, *\}$ the following hold*:

(a) *If $p \in \mathcal{Q}_\tau(X)$ and $f : X \to \mathcal{Q}_\tau(Y)$, then $pf_\diamond \in \mathcal{Q}_\tau(Y)$.*
(b) *If $p \in \mathcal{Q}_\tau(X)$ and $f : X \to \mathcal{Q}_\tau(Y)$, then there exist $p' \in \mathcal{Q}_\tau(\mathbb{N})$ and $f' : \mathbb{N} \to \mathcal{Q}_\tau(Y)$ such that $pf_\diamond = p' f'_\diamond$.*

Proof.
(a_0) $[p \in \mathcal{Q}_0(X) \wedge f : X \to \mathcal{Q}_0(X)] \to pf_\diamond \in \mathcal{Q}_0(Y)$:

$$\begin{aligned}\|pf_\diamond\| &= \sum_{x\in X} p_{[x]}\|xf\| \quad \{\text{Proposition 5(b)}\}\\ &\leqslant \sum_{x\in X} p_{[x]} \qquad \{xf \in \mathcal{Q}_0(Y)\}\\ &= \|p\|\\ &\leqslant 1 \qquad \{p \in \mathcal{Q}_0(X)\}.\end{aligned}$$

(a_1) $[p \in \mathcal{Q}_1(X) \wedge f : X \to \mathcal{Q}_1(X)] \to pf_\diamond \in \mathcal{Q}_1(Y)$:

$$\begin{aligned}\|pf_\diamond\| &= \sum_{x\in\lfloor p\rfloor} p_{[x]}\|xf\| && \{\text{Proposition 5(b)}\}\\ &= \sum_{x\in\lfloor p\rfloor} p_{[x]} && \{xf\in\mathcal{Q}_1(Y)\}\\ &= 1 && \{p\in\mathcal{Q}_1(Y)\}.\end{aligned}$$

(a_*) $[p\in\mathcal{Q}_*(X)\wedge f: X\to\mathcal{Q}_*(X)]\to pf_\diamond\in\mathcal{Q}_*(Y)$ is immediate from (a_0) and Proposition 5(c).
(b_0) $\forall p\in\mathcal{Q}_0(X)\ \forall f: X\to\mathcal{Q}_0(Y)\ \exists p'\in\mathcal{Q}_0(\mathbb{N})\ \exists f': \mathbb{N}\to\mathcal{Q}_0(Y).\ pf_\diamond = p'f'_\diamond$:

As already stated the support $\lfloor p\rfloor$ is countable if $\|p\|$ exists and so there is an injection $i:\lfloor p\rfloor\to\mathbb{N}$. Define a map $p'\in\mathcal{Q}(\mathbb{N})$ and a map $f':\mathbb{N}\to\mathcal{Q}(Y)$ by

$$p'_{[n]} = \begin{cases} p_{[x]} & \text{if } \exists x\in\lfloor p\rfloor.\ n = xi,\\ 0 & \text{otherwise}\end{cases}$$

and

$$nf' = \begin{cases} xf & \text{if } \exists x\in\lfloor p\rfloor.\ n = xi,\\ 0_Y & \text{otherwise,}\end{cases}$$

respectively. Remark that $n\in\lfloor p'\rfloor$ if and only if there exists $x\in\lfloor p\rfloor$ such that $n = xi$. Hence

$$\begin{aligned}pf_\diamond &= \bigoplus_{x\in X} p_{[x]}\cdot(xf)\\ &= \bigoplus_{n\in\mathbb{N}} p'_{[n]}\cdot(nf')\\ &= p'f'_\diamond.\end{aligned}$$

(b_*) In the case of $\tau = *$:

Let $p\in\mathcal{Q}_*(X)$ and $f: X\to\mathcal{Q}_*(Y)$, and take the same p' and f' defined in (b_0). Then it is clear that $p'\in\mathcal{Q}_*(\mathbb{N})$ and $f': \mathbb{N}\to\mathcal{Q}_*(Y)$.

(b_1) In the case of $\tau = 1$:

Let $p\in\mathcal{Q}_1(X)$ and $f: X\to\mathcal{Q}_1(Y)$, and take the same p' defined in (b_0) and define $f': X\to\mathcal{Q}_1(Y)$ by

$$nf' = \begin{cases} xf & \text{if } \exists x\in\lfloor p\rfloor.\ n = xi,\\ y_0 & \text{otherwise,}\end{cases}$$

where y_0 is an arbitrary point of Y. Then it is clear that $p'\in\mathcal{Q}_1(\mathbb{N})$ and $f': \mathbb{N}\to\mathcal{Q}_1(Y)$. □

4. Convex composition

In the rest of the paper the subscript τ is one of 0, 1 and $*$, unless otherwise stated. For a map $f: X\to\mathcal{Q}_\tau(Y)$ the convex combination induces a map $f_\diamond:\mathcal{Q}_\tau(X)\to\mathcal{Q}_\tau(Y)$ by Proposition 6(a). We now list some basic properties of the induced maps.

Proposition 7. *Let $f: X\to\mathcal{Q}_\tau(Y)$, $g: Y\to\mathcal{Q}_\tau(Z)$ and $h: X\to\mathcal{Q}_\tau(X)$ be maps. Then the following hold*:

(a) $f_\diamond g_\diamond = (fg_\diamond)_\diamond$,
(b) $e_X f_\diamond = f$,
(c) $(e_X)_\diamond = \mathrm{id}_{\mathcal{Q}_\tau(X)}$,
(d) $h\sqsubseteq f(\xi_X^\tau)^\sharp \to h_\diamond\sqsubseteq f_\diamond(\xi_X^\tau)^\sharp$ *for* $\tau\neq 1$,
(e) $(\xi_X^\tau)^\sharp f_\diamond\sqsubseteq f_\diamond(\xi_Y^\tau)^\sharp$ *for* $\tau\neq 1$,
(f) $(\nabla_{XI}0_Y)_\diamond = \nabla_{\mathcal{Q}_\tau(X)I}0_Y$ *for* $\tau\neq 1$.
(g) $(\xi_X^\tau)^\sharp h_\diamond(\xi_X^\tau)^\sharp = h_\diamond(\xi_X^\tau)^\sharp$ *for* $\tau\neq 1$.

Proof.
(a) $f_\diamond g_\diamond = (fg_\diamond)_\diamond$:

$$\begin{aligned}p(fg_\diamond)_\diamond &= \bigoplus_x p_{[x]}\cdot(xfg_\diamond)\\ &= \bigoplus_x \big(p_{[x]}\cdot(xf)\big)g_\diamond && \{\text{Proposition 5(h)}\}\end{aligned}$$

$$
\begin{aligned}
&= \bigoplus_x \bigoplus_y \big(p_{[x]}(xf)_{[y]}\big) \cdot (yg) \\
&= \bigoplus_y \bigoplus_x \big(p_{[x]}(xf)_{[y]}\big) \cdot (yg) \\
&= \bigoplus_y \Big(\sum_x p_{[x]}(xf)_{[y]}\Big) \cdot (yg) \quad \{\text{Proposition 5(a)}\} \\
&= \bigoplus_y (pf_\diamond)_{[y]} \cdot (yg) \\
&= (pf_\diamond) g_\diamond \\
&= p(f_\diamond g_\diamond).
\end{aligned}
$$

(b) $e_X f_\diamond = f$:

$$
\begin{aligned}
xe_X f_\diamond &= \dot{x} f_\diamond \quad \{xe_X = \dot{x}\} \\
&= xf \quad \{\text{Proposition 5(e)}\}.
\end{aligned}
$$

(c) $(e_X)_\diamond = \mathrm{id}_{\mathcal{Q}_\tau(X)}$ is direct from Proposition 5(f).
(d) $h \sqsubseteq f(\xi_X^\tau)^\sharp \to h_\diamond \sqsubseteq f_\diamond (\xi_X^\tau)^\sharp$:
For $p \in \mathcal{Q}_\tau(X)$ and $x \in X$ we have

$$
\begin{aligned}
(ph_\diamond)_{[x]} &= \sum_{x'} p_{[x']} \big(x'h\big)_{[x]} \\
&\leqslant \sum_{x'} p_{[x']} \big(x'f\big)_{[x]} \quad \big\{x'h \leqslant x'f \leftarrow h \sqsubseteq f\big(\xi_X^\tau\big)^\sharp\big\} \\
&= (pf_\diamond)_{[x]}
\end{aligned}
$$

which proves $ph_\diamond \leqslant pf_\diamond$ and hence $ph_\diamond \sqsubseteq pf_\diamond (\xi_X^\tau)^\sharp$.
(e) $(\xi_X^\tau)^\sharp f_\diamond \sqsubseteq f_\diamond (\xi_Y^\tau)^\sharp$:

$$
\begin{aligned}
&\to \quad p \leqslant p' \to pf_\diamond \leqslant p' f_\diamond \\
&\leftrightarrow \quad \xi_X^\tau \sqsubseteq f_\diamond \xi_Y^\tau f_\diamond^\sharp \\
&\leftrightarrow \quad \big(\xi_X^\tau\big)^\sharp \sqsubseteq f_\diamond \big(\xi_Y^\tau\big)^\sharp f_\diamond^\sharp \\
&\leftrightarrow \quad \big(\xi_X^\tau\big)^\sharp f_\diamond \sqsubseteq f_\diamond \big(\xi_Y^\tau\big)^\sharp \qquad \{f_\diamond : \text{tfn}\}.
\end{aligned}
$$

(f) $(\nabla_{XI} 0_Y)_\diamond = \nabla_{\mathcal{Q}_\tau(X)\, I} 0_Y$:

$$
\begin{aligned}
(\nabla_{XI} 0_Y)_\diamond &= \bigsqcup_{p \in \mathcal{Q}_\tau(X)} p^\sharp p (\nabla_{XI} 0_Y)_\diamond \quad \{(\text{PA})\} \\
&= \bigsqcup_{p \in \mathcal{Q}_\tau(X)} p^\sharp \big(\|p\| 0\big)_Y \quad \{\text{Proposition 5(g)}\} \\
&= \bigsqcup_{p \in \mathcal{Q}_\tau(X)} p^\sharp 0_Y \quad \{\|p\| 0 = 0\} \\
&= \nabla_{\mathcal{Q}_\tau(X)\, I} 0_Y \quad \{(\text{PA})\}.
\end{aligned}
$$

(g) $(\xi_X^\tau)^\sharp h_\diamond (\xi_X^\tau)^\sharp = h_\diamond (\xi_X^\tau)^\sharp$:

$$
\begin{aligned}
\big(\xi_X^\tau\big)^\sharp h_\diamond \big(\xi_X^\tau\big)^\sharp &\sqsubseteq h_\diamond \big(\xi_X^\tau\big)^\sharp \big(\xi_X^\tau\big)^\sharp \quad \{(\text{e})\} \\
&= h_\diamond \big(\xi_X^\tau\big)^\sharp \quad \big\{\big(\xi_X^\tau\big)^\sharp \big(\xi_X^\tau\big)^\sharp = \big(\xi_X^\tau\big)^\sharp\big\}.
\end{aligned}
$$

$$
\begin{aligned}
h_\diamond \big(\xi_X^\tau\big)^\sharp &= \mathrm{id}_{\mathcal{Q}_\tau(X)} h_\diamond \big(\xi_X^\tau\big)^\sharp \\
&\sqsubseteq \big(\xi_X^\tau\big)^\sharp h_\diamond \big(\xi_X^\tau\big)^\sharp. \qquad \square
\end{aligned}
$$

For a relation $\alpha : X \rightharpoonup \mathcal{Q}_\tau(Y)$ define a relation $\alpha_\diamond : \mathcal{Q}_\tau(X) \rightharpoonup \mathcal{Q}_\tau(Y)$ by

$$\alpha_\diamond = \bigsqcup_{f \sqsubseteq \alpha} f_\diamond,$$

where $f \sqsubseteq \alpha$ means that f is a map $f : X \to \mathcal{Q}_\tau(Y)$ such that $f \sqsubseteq \alpha$. This notion allows convex composition to be treated in ordinary relational calculus.

Remark. By the relational axiom of choice (AC) there exists a map $f \sqsubseteq \alpha$ iff α is total. Such a map f is often called a choice function of α. Also $\alpha_\diamond$ is total if α is total, and $\alpha_\diamond = \emptyset_{\mathcal{Q}_\tau(X)\mathcal{Q}_\tau(Y)}$ otherwise.

Example 2. Consider relations $\gamma = g \sqcup h$ and $\gamma' = h \sqcup e_X$ where $g, h : X \to \mathcal{Q}_*(X)$ appeared in Example 1. Since there is no maps included in $\gamma = g \sqcup h$ other than g and h, the identity $\gamma_\diamond = g_\diamond \sqcup h_\diamond$ holds. For a relation γ', the identity $\gamma'_\diamond = h_\diamond \sqcup e_{X\diamond}$ does not hold. Because γ' consists of four maps, that is $h \sqcup e_X = f' \sqcup g \sqcup h \sqcup e_X$ where $f', g : X \to \mathcal{Q}_*(X)$ are maps defined by $xf' = yf' = \dot{y}$, $xg = \dot{x}$, and $yg = 0_X$. Therefore $\gamma'_\diamond = f'_\diamond \sqcup g_\diamond \sqcup h_\diamond \sqcup e_{X\diamond}$ holds.

Proposition 8. *If $\alpha : X \rightharpoonup Y$ is a total relation, then $\alpha = \bigsqcup_{f \sqsubseteq \alpha} f$.*

Proof. Assume α is total. By the axiom of choice (AC) there is a function $f_0 : X \to Y$ such that $f_0 \sqsubseteq \alpha$. For each $(x_0, y_0) \in \alpha$ define a map $f : X \to Y$ by

$$\forall x \in X.\ xf = \begin{cases} y_0 & \text{if } x = x_0, \\ xf_0 & \text{otherwise.} \end{cases}$$

Then it is clear that $(x, q) \in f$ and so

$$\begin{aligned} f &= \bigsqcup_{x \in X} x^\sharp xf && \{(\mathrm{PA})\} \\ &= x_0^\sharp y_0 \sqcup \left(\bigsqcup_{x \neq x_0} x^\sharp x f_0 \right) \\ &\sqsubseteq \alpha \sqcup f_0 && \{x^\sharp x \sqsubseteq \mathrm{id}_X\} \\ &= \alpha && \{f_0 \sqsubseteq \alpha\}. \end{aligned}$$

Hence

$$\begin{aligned} (x_0, y_0) \in \alpha \quad &\to \quad \exists f.\ (x_0, y_0) \in f \wedge (f \sqsubseteq \alpha) \\ &\to \quad (x_0, y_0) \in \bigsqcup_{f \sqsubseteq \alpha} f, \end{aligned}$$

which shows $\alpha = \bigsqcup_{f \sqsubseteq \alpha} f$. □

A relation $\alpha : X \rightharpoonup \mathcal{Q}_\tau(Y)$ is called down-closed if it satisfies $\alpha(\xi_Y^\tau)^\sharp = \alpha$. The next proposition indicates that a relation α is total and down-closed iff it is 0-included [16], namely, $0_Y \in x\alpha$ for each $x \in X$.

Proposition 9. *Let $\tau \neq 1$. If $\alpha : X \rightharpoonup \mathcal{Q}_\tau(Y)$ is a total relation such that $\alpha(\xi_Y^\tau)^\sharp = \alpha$, then $\nabla_{XI} 0_Y \sqsubseteq \alpha$ (0-included).*

Proof. Assume α is total and $\alpha(\xi_Y^\tau)^\sharp = \alpha$. By the axiom of choice (AC) there is a function $f_0 : X \to \mathcal{Q}_*(Y)$ such that $f_0 \sqsubseteq \alpha$.

$$\begin{aligned} \nabla_{XI} 0_Y &= \bigsqcup_{x \in X} x^\sharp 0_Y && \{(\mathrm{PA})\} \\ &\sqsubseteq \bigsqcup_{x \in X} x^\sharp x f_0 (\xi_Y^\tau)^\sharp && \{\forall q \in \mathcal{Q}_\tau(Y).\ 0_Y \sqsubseteq q(\xi_Y^\tau)^\sharp\} \\ &= f_0(\xi_Y^\tau)^\sharp && \{(\mathrm{PA})\} \\ &\sqsubseteq \alpha(\xi_Y^\tau)^\sharp && \{f_0 \sqsubseteq \alpha\} \\ &= \alpha && \{\alpha(\xi_Y^\tau)^\sharp = \alpha\}. \end{aligned}$$ □

The diamond operator defined via convex combinations satisfies the following additional rules.

Proposition 10. *Let $\tau \neq 1$. For a map $f : X \to \mathcal{Q}_\tau(Y)$, a relation $\alpha : X \multimap \mathcal{Q}_\tau(Y)$ and $t \in \mathcal{Q}(Y)$ the following hold*:

(a) $f_\diamond t_* = (ft_*)_\diamond$,
(b) $\alpha_\diamond t_* \sqsubseteq (\alpha t_*)_\diamond$,
(c) $\alpha_\diamond (\xi_Y^\tau)^\sharp \sqsubseteq (\alpha(\xi_Y^\tau)^\sharp)_\diamond$,
(d) $(f(\xi_X^\tau)^\sharp)_\diamond = f_\diamond(\xi_X^\tau)^\sharp$.

Proof.
(a) $f_\diamond t_* = (ft_*)_\diamond$:
For $p \in \mathcal{Q}_\tau(X)$ we have

$$\begin{aligned}
p(f_\diamond t_*) &= (pf_\diamond)t_* \\
&= (pf_\diamond)\cdot t \\
&= \left(\bigoplus_{x\in X} p_{[x]}\cdot(xf)\right)\cdot t \\
&= \bigoplus_{x\in X}\left(p_{[x]}\cdot(xf)\right)\cdot t \\
&= \bigoplus_{x\in X} p_{[x]}\cdot\left((xf)\cdot t\right) \\
&= \bigoplus_{x\in X} p_{[x]}\cdot(xft_*) \\
&= p(ft_*)_\diamond.
\end{aligned}$$

(b) $\alpha_\diamond t_* \sqsubseteq (\alpha t_*)_\diamond$:

$$\begin{aligned}
\alpha_\diamond t_* &= \left(\bigsqcup_{f\sqsubseteq\alpha} f_\diamond\right)t_* \\
&= \bigsqcup_{f\sqsubseteq\alpha} f_\diamond t_* \\
&= \bigsqcup_{f\sqsubseteq\alpha} (ft_*)_\diamond \quad \{(a)\} \\
&\sqsubseteq \bigsqcup_{f'\sqsubseteq\alpha t_*} f'_\diamond \quad \{ft_* \sqsubseteq \alpha t_*\} \\
&= (\alpha t_*)_\diamond.
\end{aligned}$$

(c) $\alpha_\diamond(\xi_Y^\tau)^\sharp \sqsubseteq (\alpha(\xi_Y^\tau)^\sharp)_\diamond$:

$$\begin{aligned}
\alpha_\diamond\left(\xi_Y^\tau\right)^\sharp &= \alpha_\diamond\left(\bigsqcup_{t\in\mathcal{Q}(Y)} t_*\right) \quad \left\{\left(\xi_Y^\tau\right)^\sharp = \bigsqcup_{t\in\mathcal{Q}(Y)} t_*\right\} \\
&= \bigsqcup_{t\in\mathcal{Q}(Y)} \alpha_\diamond t_* \\
&\sqsubseteq \bigsqcup_{t\in\mathcal{Q}(Y)} (\alpha t_*)_\diamond \quad \{(b)\} \\
&\sqsubseteq \left(\alpha\left(\xi_Y^\tau\right)^\sharp\right)_\diamond \quad \left\{t_* \sqsubseteq \left(\xi_Y^\tau\right)^\sharp\right\}.
\end{aligned}$$

(d) $(f(\xi_X^\tau)^\sharp)_\diamond = f_\diamond(\xi_X^\tau)^\sharp$:

$$\begin{aligned}
\left(f\left(\xi_X^\tau\right)^\sharp\right)_\diamond &= \bigsqcup_{h\sqsubseteq f(\xi_X^\tau)^\sharp} h_\diamond \\
&\sqsubseteq f_\diamond\left(\xi_X^\tau\right)^\sharp \quad \{\text{Proposition 7(d)}\}.
\end{aligned}$$

The opposite direction $f_\diamond(\xi_X^\tau)^\sharp \sqsubseteq (f(\xi_X^\tau)^\sharp)_\diamond$ follows from (c). □

Now we will define a composition [4,7] for relations into algebras of probabilistic distributions.

Definition 2. Let $\alpha : X \rightharpoonup \mathcal{Q}_\tau(Y)$ and $\beta : Y \rightharpoonup \mathcal{Q}_\tau(Z)$ be relations. The convex composite $\alpha \circ \beta : X \rightharpoonup \mathcal{Q}_\tau(Z)$ of α followed by β is defined by

$$\alpha \circ \beta = \alpha\beta_\diamond.$$

Remark. In some aspects, convex composition seems to be concrete examples of Kleisli composition of the powerset monads studied by Eklund and Gähler [2]. However, in our case the composition chooses a map from latter argument in nondeterministic way, whereas Kleisli composition chooses in deterministic way.

We show the basic properties on convex composition of relations.

Proposition 11. *Let $\alpha, \alpha' : X \rightharpoonup \mathcal{Q}_\tau(Y)$, $\beta, \beta' : Y \rightharpoonup \mathcal{Q}_\tau(Z)$ and $\gamma : Z \rightharpoonup \mathcal{Q}_\tau(W)$ be relations. Then*

(a) $\beta \sqsubseteq \beta' \to \beta_\diamond \sqsubseteq \beta'_\diamond$,
(b) $\alpha \sqsubseteq \alpha' \wedge \beta \sqsubseteq \beta' \to \alpha \circ \beta \sqsubseteq \alpha' \circ \beta'$,
(c) $\alpha_\diamond\beta_\diamond \sqsubseteq (\alpha \circ \beta)_\diamond$,
(d) $(\alpha \circ \beta) \circ \gamma \sqsubseteq \alpha \circ (\beta \circ \gamma)$,
(e) $\alpha : \mathit{total} \to e_X\alpha_\diamond = \alpha$,
(f) $\alpha : \mathit{total} \to \alpha \circ \nabla_{YI}0_Z = \nabla_{XI}0_Z$ *for* $\tau \neq 1$,
(g) $\alpha : \mathit{total} \to 0_X \circ \alpha = 0_Y$ *for* $\tau \neq 1$,
(h) $(\alpha \circ \beta)(\xi_Z^\tau)^\sharp \sqsubseteq \alpha \circ \beta(\xi_Z^\tau)^\sharp$ *for* $\tau \neq 1$.

Proof.
(a) $\beta \sqsubseteq \beta' \to \beta_\diamond \sqsubseteq \beta'_\diamond$:
Assume $\beta \sqsubseteq \beta'$. Then

$$\begin{aligned}\beta_\diamond &= \bigsqcup_{g \sqsubseteq \beta} g_\diamond \\ &\sqsubseteq \bigsqcup_{g \sqsubseteq \beta'} g_\diamond \quad \{\beta \sqsubseteq \beta'\} \\ &= \beta'_\diamond.\end{aligned}$$

(b) $\alpha \sqsubseteq \alpha' \wedge \beta \sqsubseteq \beta' \to \alpha \circ \beta \sqsubseteq \alpha' \circ \beta'$:
Assume $\alpha \sqsubseteq \alpha'$ and $\beta \sqsubseteq \beta'$. Then

$$\begin{aligned}\alpha \circ \beta &= \alpha\beta_\diamond \\ &\sqsubseteq \alpha'\beta'_\diamond \quad \{\alpha \sqsubseteq \alpha', \beta \sqsubseteq \beta', (a)\} \\ &= \alpha' \circ \beta'.\end{aligned}$$

(c) $\alpha_\diamond\beta_\diamond \sqsubseteq (\alpha \circ \beta)_\diamond$:
Note that for maps $f : X \to \mathcal{Q}_\tau(Y)$ and $g : Y \to \mathcal{Q}_\tau(Z)$ such that $f \sqsubseteq \alpha$ and $g \sqsubseteq \beta$, we have

$$\begin{aligned}f_\diamond g_\diamond &= (fg_\diamond)_\diamond \quad \{\text{Proposition 7(a)}\} \\ &\sqsubseteq (\alpha\beta_\diamond)_\diamond \quad \{(a), (b)\} \\ &= (\alpha \circ \beta)_\diamond.\end{aligned}$$

Hence

$$\begin{aligned}\alpha_\diamond\beta_\diamond &= \Big(\bigsqcup_{f \sqsubseteq \alpha} f_\diamond\Big)\Big(\bigsqcup_{g \sqsubseteq \beta} g_\diamond\Big) \\ &= \bigsqcup_{f \sqsubseteq \alpha} \bigsqcup_{g \sqsubseteq \beta} f_\diamond g_\diamond \\ &\sqsubseteq (\alpha \circ \beta)_\diamond \quad \{f_\diamond g_\diamond \sqsubseteq (\alpha \circ \beta)_\diamond\}.\end{aligned}$$

(d) $(\alpha \circ \beta) \circ \gamma \sqsubseteq \alpha \circ (\beta \circ \gamma)$:

$$\begin{aligned}(\alpha \circ \beta) \circ \gamma &= (\alpha\beta_\diamond)\gamma_\diamond \\ &= \alpha(\beta_\diamond\gamma_\diamond) \\ &\sqsubseteq \alpha(\beta \circ \gamma)_\diamond \qquad \{(c)\} \\ &= \alpha \circ (\beta \circ \gamma).\end{aligned}$$

(e) $\alpha : \text{total} \to e_X\alpha_\diamond = \alpha$:

$$\begin{aligned}e_X\alpha_\diamond &= e_X\Big(\bigsqcup_{f \sqsubseteq \alpha} f_\diamond\Big) \\ &= \bigsqcup_{f \sqsubseteq \alpha} e_X f_\diamond \\ &= \bigsqcup_{f \sqsubseteq \alpha} f \qquad \{\text{Proposition 7(b)}\} \\ &= \alpha \qquad \{\alpha : \text{total}\}.\end{aligned}$$

(f) $\alpha : \text{total} \to \alpha \circ \nabla_{YI}0_Z = \nabla_{XI}0_Z$:

$$\begin{aligned}\alpha \circ \nabla_{YI}0_Z &= \alpha(\nabla_{YI}0_Z)_\diamond \\ &= \alpha\nabla_{\mathcal{Q}_\tau(Y)I}0_Z \qquad \{\text{Proposition 7(f)}\} \\ &= \nabla_{XI}0_Z \qquad \{\alpha : \text{total}\}.\end{aligned}$$

(g) $\alpha : \text{total} \to 0_X \circ \alpha = 0_Y$:

$$\begin{aligned}0_X \circ \alpha &= 0_X\Big(\bigsqcup_{f \sqsubseteq \alpha} f_\diamond\Big) \\ &= \bigsqcup_{f \sqsubseteq \alpha} 0_X f_\diamond \\ &= 0_Y \qquad \{\text{Proposition 5(d)}\}.\end{aligned}$$

(h) $(\alpha \circ \beta)(\xi_Z^\tau)^\sharp \sqsubseteq \alpha \circ \beta(\xi_Z^\tau)^\sharp$:

$$\begin{aligned}(\alpha \circ \beta)\big(\xi_Z^\tau\big)^\sharp &= (\alpha\beta_\diamond)\big(\xi_Z^\tau\big)^\sharp \\ &= \alpha\big(\beta_\diamond\big(\xi_Z^\tau\big)^\sharp\big) \\ &\sqsubseteq \alpha\big(\beta\big(\xi_Z^\tau\big)^\sharp\big)_\diamond \qquad \{\text{Proposition 10(c)}\} \\ &= \alpha \circ \beta\big(\xi_Z^\tau\big)^\sharp. \qquad \square\end{aligned}$$

By Proposition 7(c) and Proposition 11(e), if α is total then e_X is neutral for convex composition, that is, $\alpha \circ e_X = e_X \circ \alpha = \alpha$.

The following proposition shows that ε_X^τ is identity element for convex composition in the case of $\tau \neq 1$ and $\alpha(\xi_Y^\tau)^\sharp \sqsubseteq \alpha$ (down-closed).

Proposition 12. *Let $\alpha : X \multimap \mathcal{Q}_\tau(Y)$ be a total relation for $\tau \neq 1$. Then the following holds*:

(a) $\alpha \sqsubseteq \varepsilon_X^\tau \circ \alpha \sqsubseteq \alpha(\xi_Y^\tau)^\sharp$,
(b) $\alpha \sqsubseteq \alpha \circ \varepsilon_Y^\tau \sqsubseteq \alpha(\xi_Y^\tau)^\sharp$.

Proof.
(a) $\alpha \sqsubseteq \varepsilon_X^\tau \circ \alpha \sqsubseteq \alpha(\xi_Y^\tau)^\sharp$:

$$\begin{aligned}\alpha &= e_X\alpha_\diamond \qquad \{\alpha : \text{total}, \text{ Proposition 11(e)}\} \\ &\sqsubseteq e_X\big(\xi_X^\tau\big)^\sharp\alpha_\diamond \qquad \{e_X \sqsubseteq e_X\big(\xi_X^\tau\big)^\sharp = \varepsilon_X^\tau\} \\ &= e_X\big(\xi_X^\tau\big)^\sharp\Big(\bigsqcup_{f \sqsubseteq \alpha} f_\diamond\Big)\end{aligned}$$

$$\sqsubseteq e_X\Big(\bigsqcup_{f\sqsubseteq\alpha} f_\diamond\Big)\big(\xi^\tau_X\big)^\sharp \quad \{\text{Proposition 7(e)}\}$$
$$= e_X\alpha_\diamond\big(\xi^\tau_X\big)^\sharp$$
$$= \alpha\big(\xi^\tau_Y\big)^\sharp.$$

(b) $\alpha \sqsubseteq \alpha\circ\varepsilon^\tau_Y \sqsubseteq \alpha(\xi^\tau_Y)^\sharp$:

$$\alpha = \alpha(e_Y)_\diamond \quad \{\text{Proposition 7(c)}\}$$
$$\sqsubseteq \alpha\big(\varepsilon^\tau_Y\big)_\diamond \quad \{e_Y \sqsubseteq \varepsilon^\tau_Y\}$$
$$\sqsubseteq \alpha\big(\xi^\tau_Y\big)^\sharp \quad \{(\varepsilon^\tau_Y)_\diamond \sqsubseteq (\xi^\tau_Y)^\sharp\}. \qquad \square$$

5. Associative law

In this section we will introduce the convex relations and study the associative law of convex composition on their relations. For a relation $\gamma : Z \rightharpoonup \mathcal{Q}_\tau(W)$ define a relation $\gamma^\bullet : Z \rightharpoonup \mathcal{Q}_\tau(W)$ by

$$\forall z \in Z.\ z\gamma^\bullet = \nabla_{I\mathcal{Q}_\tau(\mathbb{N})}\circ\nabla_{\mathbb{N}I} z\gamma.$$

Note that $\rho^\bullet = \nabla_{I\mathcal{Q}_\tau(\mathbb{N})}\circ\nabla_{\mathbb{N}I}\rho$ for a relation $\rho : I \rightharpoonup \mathcal{Q}_\tau(W)$.

Remark. The definition of $\gamma^\bullet$ explicitly contains an element (or a variable) beyond preferable relational expressions.

The notion of $\gamma^\bullet$ derives a property called *convex* for relations to satisfy the associativity of convex composition. A relation $\gamma : Z \rightharpoonup \mathcal{Q}_\tau(W)$ is called convex if it satisfies $\gamma^\bullet = \gamma$.

Example 3. Consider on the same X as in previous examples. Define a relation $\alpha : X \rightharpoonup \mathcal{Q}_1(X)$ by $x\alpha = y\alpha = (\frac{1}{2})_X$. Then $\alpha^\bullet : X \rightharpoonup \mathcal{Q}_1(X)$ satisfies $\alpha^\bullet = \alpha$. However when we regard α as $\alpha : X \rightharpoonup \mathcal{Q}_*(X)$, $\alpha^\bullet : X \rightharpoonup \mathcal{Q}_*(X)$ satisfies $x\alpha^\bullet = y\alpha^\bullet = (\frac{1}{2})_X(\xi^*_X)^\sharp$, that is $\alpha^\bullet \neq \alpha$. For a relation $\gamma : X \rightharpoonup \mathcal{Q}_*(X)$ which appeared in Example 2, we obtain that $\gamma^\bullet \neq \gamma$ since $x\gamma^\bullet = \mathcal{Q}_*(X)$ though $x\gamma = \dot{x}\sqcup\dot{y}$.

Proposition 13. *Let $\gamma : Z \rightharpoonup \mathcal{Q}_\tau(W)$ be a relation.*

(a) *If γ is total, then $\gamma \sqsubseteq \gamma^\bullet$,*
(b) *$\nabla_{I\mathcal{Q}_\tau(Y)}\circ\nabla_{YI} z\gamma \sqsubseteq z\gamma^\bullet$ for all sets Y,*
(c) *$\gamma^{\bullet\bullet} \sqsubseteq \gamma^\bullet$,*
(d) *$\gamma^\bullet(\xi^\tau_W)^\sharp \sqsubseteq (\gamma(\xi^\tau_W)^\sharp)^\bullet$ $(\tau \neq 1)$.*

Proof. Set $\rho = z\gamma$. Then $\rho^\bullet = z\gamma^\bullet$, $\rho^{\bullet\bullet} = z\gamma^{\bullet\bullet}$ and $(\rho(\xi^\tau_W)^\sharp)^\bullet = z(\gamma(\xi^\tau_W)^\sharp)^\bullet$. Thus it suffices to show the following statements for ρ.
(a) $\gamma : \text{total} \to \rho \sqsubseteq \rho^\bullet$:

$$\rho = \nabla_{I\,\mathbb{N}}\nabla_{\mathbb{N}\,I}\rho \qquad \{\mathbb{N} \neq \emptyset\}$$
$$= \nabla_{I\,\mathbb{N}} e_{\mathbb{N}}(\nabla_{\mathbb{N}\,I}\rho)_\diamond \quad \{\text{Proposition 11(e)},\ \rho : \text{total}\}$$
$$\sqsubseteq \nabla_{I\mathcal{Q}_\tau(\mathbb{N})}(\nabla_{\mathbb{N}I}\rho)_\diamond.$$

(b) $\nabla_{I\mathcal{Q}_\tau(Y)}\circ\nabla_{YI}\rho \sqsubseteq \nabla_{I\mathcal{Q}_\tau(\mathbb{N})}\circ\nabla_{\mathbb{N}I}\rho$:

$$\nabla_{I\mathcal{Q}_\tau(Y)}(\nabla_{YI}\rho)_\diamond = \Big(\bigsqcup_{p\in\mathcal{Q}_\tau(Y)} p\Big)\Big(\bigsqcup_{f\sqsubseteq\nabla_{YI}\rho} f_\diamond\Big)$$
$$= \bigsqcup_{p\in\mathcal{Q}_\tau(Y)}\ \bigsqcup_{f\sqsubseteq\nabla_{YI}\rho} pf_\diamond$$
$$\sqsubseteq \bigsqcup_{p'\in\mathcal{Q}_\tau(\mathbb{N})}\ \bigsqcup_{f'\sqsubseteq\nabla_{\mathbb{N}I}\rho} (p'f'_\diamond) \qquad \{\text{Proposition 6(b)}\}$$
$$= \nabla_{I\mathcal{Q}_\tau(\mathbb{N})}(\nabla_{\mathbb{N}I}\rho)_\diamond.$$

(c) Recall that

$$\begin{aligned} q' \sqsubseteq \rho^{\bullet\bullet} \quad &\to \quad \exists p' \in \mathcal{Q}_\tau(\mathbb{N})\, \exists f' : \mathbb{N} \to \mathcal{Q}_\tau(X) \\ &\qquad q' = p' f'_\diamond \wedge \forall n \in \mathbb{N}.\, nf' \sqsubseteq \rho^\bullet \\ nf' \sqsubseteq \rho^\bullet \quad &\to \quad \exists p_n \in \mathcal{Q}_\tau(\mathbb{N})\, \exists f_n : \mathbb{N} \to \mathcal{Q}_\tau(X) \\ &\qquad nf' = p_n(f_n)_\diamond \wedge \forall m \in \mathbb{N}.\, mf_n \sqsubseteq \rho, \end{aligned}$$

and so

$$\begin{aligned} q' &= p' f'_\diamond \\ &= \bigoplus_{n\in\mathbb{N}} p'_{[n]}(nf') \\ &= \bigoplus_{n\in\mathbb{N}} p'_{[n]}(p_n(f_n)_\diamond) \\ &= \bigoplus_{n\in\mathbb{N}} p'(n)\Big(\bigoplus_{m\in\mathbb{N}} p_{n[m]}(mf_n)\Big) \\ &= \bigoplus_{n\in\mathbb{N}}\bigoplus_{m\in\mathbb{N}} p'_{[n]} p_{n[m]}(mf_n). \end{aligned}$$

Define $\hat{p} \in \mathcal{Q}_\tau(\mathbb{N}\times\mathbb{N})$ and $\hat{f} : \mathbb{N}\times\mathbb{N} \to \mathcal{Q}_\tau(X)$ by $(n,m)\hat{p} = p'_{[n]} p_{n[m]}$ and $(n,m)\hat{f} = mf_n$, respectively. Then

$$\begin{aligned} q' &= \hat{p}\hat{f}_\diamond \\ &\sqsubseteq \nabla_{I\mathcal{Q}_\tau(\mathbb{N}\times\mathbb{N})} \circ (\nabla_{\mathbb{N}\times\mathbb{N} I}\rho) \\ &\sqsubseteq \nabla_{I\mathcal{Q}_\tau(\mathbb{N})} \circ (\nabla_{\mathbb{N} I}\rho) \qquad \{(\mathrm{b})\} \\ &= \rho^\bullet. \end{aligned}$$

This proves $\rho^{\bullet\bullet} \sqsubseteq \rho^\bullet$.
(d) $\gamma^\bullet(\xi^\tau_W)^\sharp \sqsubseteq (\gamma(\xi^\tau_W)^\sharp)^\bullet$: $(\tau \neq 1)$

$$\begin{aligned} \rho^\bullet\big(\xi^\tau_W\big)^\sharp &= \nabla_{I\mathcal{Q}_*(\mathbb{N})}(\nabla_{\mathbb{N} I}\rho)_\diamond\big(\xi^\tau_W\big)^\sharp \\ &\sqsubseteq \nabla_{I\mathcal{Q}_*(\mathbb{N})}\big(\nabla_{\mathbb{N} I}\rho\big(\xi^\tau_W\big)^\sharp\big)_\diamond \quad \{\text{Proposition 10(c)}\} \\ &= \big(\rho\big(\xi^\tau_W\big)^\sharp\big)^\bullet. \qquad \square \end{aligned}$$

Now we define two kinds of convex relations, named $\mathcal{Q}_*$-convex relation and $\mathcal{Q}_1$-convex relation.

Definition 3. A relation $\alpha : X \rightharpoonup \mathcal{Q}_*(Y)$ is called *$\mathcal{Q}_*$-convex* if $\mathrm{id}_X \sqsubseteq \alpha\alpha^\sharp$ (total), $\alpha(\xi^*_Y)^\sharp = \alpha$ (down-closed) and $\alpha^\bullet = \alpha$ (convex). A relation $\alpha : X \rightharpoonup \mathcal{Q}_1(Y)$ is called *$\mathcal{Q}_1$-convex* if $\mathrm{id}_X \sqsubseteq \alpha\alpha^\sharp$ (total) and $\alpha^\bullet = \alpha$ (convex).

By Proposition 9, a $\mathcal{Q}_*$-convex relation α is 0-included, that is α satisfies $\nabla_{XI} 0_Y \sqsubseteq \alpha$.
We need the following lemma to derive the associative law of convex composition.

Lemma 1. *Let $f : Y \to \mathcal{Q}_\tau(W)$ be a map, and $\beta : Y \rightharpoonup \mathcal{Q}_\tau(Z)$ and $\gamma : Z \rightharpoonup \mathcal{Q}_\tau(W)$ relations. If $f \sqsubseteq \beta\gamma_\diamond$, then $f_\diamond \sqsubseteq \beta_\diamond(\gamma^\bullet)_\diamond$.*

Proof. Let $f \sqsubseteq \beta\gamma_\diamond$ and $p \in \mathcal{Q}_\tau(Y)$. Then
(1) $\exists g \sqsubseteq \beta.\ f \sqsubseteq g\gamma_\diamond$:
As $f \sqsubseteq \beta\gamma_\diamond$ it holds that

$$\begin{aligned} \mathrm{id}_Y &= ff^\sharp \sqcap \mathrm{id}_Y && \{f : \mathrm{tfn}\} \\ &\sqsubseteq \beta\gamma_\diamond f^\sharp \sqcap \mathrm{id}_Y && \{f \sqsubseteq \beta\gamma_\diamond\} \\ &\sqsubseteq \big(\beta \sqcap f(\gamma_\diamond)^\sharp\big)\big(\beta^\sharp \sqcap \gamma_\diamond f^\sharp\big) && \{(\mathrm{DF}_*)\}. \end{aligned}$$

Hence $\beta \sqcap f(\gamma_\diamond)^\sharp$ is total and by the axiom of choice (AC) there exists a tfn $g : Y \to \mathcal{Q}_\tau(Z)$ such that $g \sqsubseteq \beta \sqcap f(\gamma_\diamond)^\sharp$, which is equivalent to $g \sqsubseteq \beta$ and $f \sqsubseteq g\gamma_\diamond$.
(2) $\forall y \in Y\, \exists h_y \sqsubseteq \gamma.\ yf = yg(h_y)_\diamond$: Note that

$$\begin{aligned} yf &\sqsubseteq yg\gamma_\diamond && \{f \sqsubseteq g\gamma_\diamond\} \\ &= \bigsqcup_{h \sqsubseteq \gamma} ygh_\diamond && \left\{\gamma_\diamond = \bigsqcup_{h\sqsubseteq\gamma} h_\diamond\right\}. \end{aligned}$$

Thus there exists $h_y \sqsubseteq \gamma$ such that $yf = yg(h_y)_\diamond$.
(3) Define a map $r_z \in \mathcal{Q}(Y)$ by

$$r_{z[y]} = \begin{cases} \frac{p_{[y]}(yg)_{[z]}}{(pg_\diamond)_{[z]}} & \text{if } (pg_\diamond)_{[z]} > 0, \\ p_{[y]} & \text{otherwise.} \end{cases}$$

(4) $(pg_\diamond)_{[z]} r_{z[y]} = p_{[y]}(yg)_{[z]}$ and $r_z \in \mathcal{Q}_\tau(Y)$, i.e., $r_z \sqsubseteq \nabla_{I\mathcal{Q}_\tau(Y)}$:
If $\tau = 1$ then $(pg_\diamond)_{[z]} = 0$ implies $(yg)_{[z]} = 0$ for each $y \in Y$. Even if $\tau \neq 1$, $(pg_\diamond)_{[z]} = 0$ implies $p = 0_Y$ or $(yg)_{[z]} = 0$ for each $y \in Y$. In each case it is clear that $(pg_\diamond)_{[z]} r_{z[y]} = p_{[y]}(yg)_{[z]}$.
If $(pg_\diamond)_{[z]} = 0$ then $r_z = p \in \mathcal{Q}_\tau(Y)$. If $(pg_\diamond)_{[z]} \neq 0$ then

$$\begin{aligned} \|r_z\| &= \sum_y \frac{p_{[y]}(yg)_{[z]}}{(pg_\diamond)_{[z]}} \\ &= \frac{\sum_y p_{[y]}(yg)_{[z]}}{(pg_\diamond)_{[z]}} \\ &= \frac{(pg_\diamond)_{[z]}}{(pg_\diamond)_{[z]}} \\ &= 1, \end{aligned}$$

and $\lfloor r_z \rfloor \subseteq \lfloor p \rfloor$, since $p_{[y]} = 0$ implies $r_{z[y]} = 0$. Hence $r_z \in \mathcal{Q}_\tau(Y)$.
(5) For all $z \in Z$ define a map $\hat{h}_z : Y \to \mathcal{Q}_\tau(W)$ by $\forall y \in Y.\ y\hat{h}_z = zh_y$.
(6) $(\hat{h}_z)_\diamond \sqsubseteq (\nabla_{YI} z\gamma)_\diamond$:

$$\begin{aligned} \hat{h}_z &= \bigsqcup_{y\in Y} y^\sharp z h_y \\ &\sqsubseteq \bigsqcup_{y\in Y} y^\sharp z\gamma && \{h_y \sqsubseteq \gamma\} \\ &= \nabla_{YI} z\gamma && \left\{\bigsqcup_{y\in Y} y^\sharp = \nabla_{YI}\right\}, \end{aligned}$$

which implies $(\hat{h}_z)_\diamond \sqsubseteq (\nabla_{YI} z\gamma)_\diamond$.
(7) Define a map $h : Z \to \mathcal{Q}_\tau(W)$ by

$$\forall z \in Z.\ zh = r_z(\hat{h}_z)_\diamond.$$

(8) $h \sqsubseteq \gamma^\bullet$:

$$\begin{aligned} zh &= r_z(\hat{h}_z)_\diamond && \{(7)\} \\ &\sqsubseteq \nabla_{I\mathcal{Q}_\tau(Y)}(\nabla_{YI} z\gamma)_\diamond && \{(4), (6)\} \\ &\sqsubseteq z\gamma^\bullet && \{\text{Proposition 13(b)}\}. \end{aligned}$$

(9) $pf_\diamond = pg_\diamond h_\diamond$:

$$\begin{aligned} pf_\diamond &= \bigoplus_y p_{[y]} \cdot (yf) \\ &= \bigoplus_y p_{[y]} \cdot \big(yg(h_y)_\diamond\big) && \{(2)\ yf = yg(h_y)_\diamond\} \\ &= \bigoplus_y \big(p_{[y]} \cdot (yg)\big)(h_y)_\diamond && \{\text{Proposition 5(h)}\} \\ &= \bigoplus_y \bigoplus_z \big(p_{[y]}(yg)_{[z]}\big) \cdot (zh_y) \end{aligned}$$

$$
\begin{aligned}
&= \bigoplus_z \bigoplus_y \big((pg_\diamond)_{[z]} r_{z[y]}\big) \cdot (zh_y) && \{(4)\ p_{[y]}(yg)_{[z]} = (pg_\diamond)_{[z]} r_{z[y]}\} \\
&= \bigoplus_z \bigoplus_y \big((pg_\diamond)_{[z]} \cdot r_z\big)_{[y]} \cdot (y\hat{h}_z) && \{(5)\ zh_y = y\hat{h}_z\} \\
&= \bigoplus_z \big((pg_\diamond)_{[z]} \cdot r_z\big)(\hat{h}_z)_\diamond \\
&= \bigoplus_z (pg_\diamond)_{[z]} \cdot \big(r_z(\hat{h}_z)_\diamond\big) && \{\text{Proposition 5(h)}\} \\
&= \bigoplus_z (pg_\diamond)_{[z]} \cdot (zh) && \{(7)\ zh = r_z(\hat{h}_z)_\diamond\} \\
&= (pg_\diamond)h_\diamond.
\end{aligned}
$$

Note that h depends on p and so $f_\diamond = g_\diamond h_\diamond$ may not hold.

(10) $f_\diamond \sqsubseteq \beta_\diamond(\gamma^\bullet)_\diamond$:

For each $p \in \mathcal{Q}_*(Y)$ we have

$$
\begin{aligned}
pf_\diamond &= pg_\diamond h_\diamond && \{(9)\} \\
&\sqsubseteq p\beta_\diamond\big(\gamma^\bullet\big)_\diamond, && \{(1)\ g \sqsubseteq \beta,\ (8)\ h \sqsubseteq \gamma^\bullet\}
\end{aligned}
$$

and hence $f_\diamond \sqsubseteq \beta_\diamond(\gamma^\bullet)_\diamond$. This completes the proof. □

Corollary 2. *Let $\alpha : X \rightharpoonup \mathcal{Q}_\tau(Y)$, $\beta : Y \rightharpoonup \mathcal{Q}_\tau(Z)$ and $\gamma : Z \rightharpoonup \mathcal{Q}_\tau(W)$ be relations. Then*

(a) $\alpha \circ (\beta \circ \gamma) \sqsubseteq (\alpha \circ \beta) \circ \gamma^\bullet$,
(b) *If $\gamma^\bullet = \gamma$, then $\alpha \circ (\beta \circ \gamma) = (\alpha \circ \beta) \circ \gamma$.*

Proof.
(a)

$$
\begin{aligned}
\alpha \circ (\beta \circ \gamma) &= \alpha(\beta \circ \gamma)_\diamond \\
&= \alpha\Big(\bigsqcup_{f \sqsubseteq \beta \circ \gamma} f_\diamond \Big) \\
&\sqsubseteq \alpha\big(\beta_\diamond\big(\gamma^\bullet\big)_\diamond\big) && \{\text{Lemma 1}, f_\diamond \sqsubseteq \beta_\diamond\big(\gamma^\bullet\big)_\diamond\} \\
&= (\alpha\beta_\diamond)\big(\gamma^\bullet\big)_\diamond \\
&= (\alpha \circ \beta) \circ \gamma^\bullet.
\end{aligned}
$$

(b)

$$
\begin{aligned}
\alpha \circ (\beta \circ \gamma) &\sqsubseteq (\alpha \circ \beta) \circ \gamma^\bullet && \{(a)\} \\
&= (\alpha \circ \beta) \circ \gamma && \{\gamma^\bullet = \gamma\} \\
&\sqsubseteq \alpha \circ (\beta \circ \gamma) && \{\text{Proposition 11(d)}\}. \quad \square
\end{aligned}
$$

We proved the associative law of convex composition for convex relations. However, the following example shows that the convex composition $\circ$ need not be associative in general.

Example 4. Consider maps $f, g, h : X \to \mathcal{Q}_*(X)$ which appeared in Example 1. For all $p \in \mathcal{Q}_1(X)$ we have

$$
pf_\diamond = \dot{x}, \qquad pg_\diamond = p_{[x]} \cdot \dot{x}, \quad \text{and} \quad ph_\diamond = p_{[x]} \cdot \dot{y}.
$$

Thus $fg_\diamond = f$, $xfh_\diamond = yfh_\diamond = \dot{y}$, and $p(fh_\diamond)_\diamond = \dot{y}$ for $p \in \mathcal{Q}_1(X)$. Shown in Example 2, the identity $\gamma_\diamond = g_\diamond \sqcup h_\diamond$ holds, and so $pf_\diamond\gamma_\diamond = \dot{x} \sqcup \dot{y}$ for all $p \in \mathcal{Q}_1(X)$. Note that $\gamma^\bullet \neq \gamma$. On the other hand, except for two maps $fg_\diamond$ and $fh_\diamond$ there are just two maps k and k' included in $f\gamma_\diamond$, where $xk = \dot{x}$, $yk = \dot{y}$, $xk' = \dot{y}$ and $yk' = \dot{x}$. Let $p_0 = (\frac{1}{2})_X$, the middle point of $\dot{x}$ and $\dot{y}$. Then we have

$$\begin{aligned} p_0(f\gamma_\diamond)_\diamond &= p_0(fg_\diamond)_\diamond \sqcup p_0(fh_\diamond)_\diamond \sqcup p_0k_\diamond \sqcup p_0k'_\diamond \\ &= \dot{x} \sqcup \dot{y} \sqcup p_0k_\diamond \sqcup p_0k'_\diamond \\ &= \dot{x} \sqcup \dot{y} \sqcup p_0 \\ &\neq \dot{x} \sqcup \dot{y} \\ &= p_0f_\diamond\gamma_\diamond, \end{aligned}$$

which proves that $\alpha f_\diamond\gamma_\diamond \neq \alpha(f\gamma_\diamond)_\diamond$ for a map $\alpha : X \to \mathcal{Q}_1(X)$ such that $x\alpha = y\alpha = p_0$. Therefore $(\alpha \circ f) \circ \gamma = \alpha f_\diamond\gamma_\diamond \neq \alpha(f\gamma_\diamond)_\diamond = \alpha \circ (f \circ \gamma)$.

6. Convex relations and distributivities

Now we discuss the convex relations and the distributive laws of convex composition over the joins.

Proposition 14. *Let $\alpha : X \multimap \mathcal{Q}_*(Y)$ and $\beta : Y \multimap \mathcal{Q}_*(Z)$ be $\mathcal{Q}_*$-convex relations. Then the following holds*:

(a) $\alpha \circ \beta$ *is total,*
(b) $(\alpha \circ \beta)(\xi^*_Z)^\sharp \sqsubseteq \alpha \circ \beta$,
(c) $(\alpha \circ \beta)^\bullet = \alpha \circ \beta$.

Proof.
(a) $\alpha \circ \beta$ is total:
Since α and β are total, $\beta_\diamond$ is total by the definition and so $\alpha \circ \beta = \alpha\beta_\diamond$ is total.
(b) $(\alpha \circ \beta)(\xi^\tau_Z)^\sharp \sqsubseteq \alpha \circ \beta$:

$$\begin{aligned} (\alpha \circ \beta)\big(\xi^\tau_Z\big)^\sharp &= \alpha\beta_\diamond \big(\xi^\tau_Z\big)^\sharp && \{\text{Definition 2}\} \\ &\sqsubseteq \alpha\big(\beta\big(\xi^\tau_Z\big)^\sharp\big)_\diamond && \big\{\text{Proposition 10(c)}\big\} \\ &\sqsubseteq \alpha\beta_\diamond && \big\{\beta\big(\xi^\tau_Z\big)^\sharp = \beta\big\}. \end{aligned}$$

(c) $(\alpha \circ \beta)^\bullet = \alpha \circ \beta$:

$$\begin{aligned} x(\alpha \circ \beta)^\bullet &= \nabla_{I\mathcal{Q}_*(\mathbb{N})} \circ \nabla_{\mathbb{N}I} x(\alpha \circ \beta) \\ &= \nabla_{I\mathcal{Q}_*(\mathbb{N})} \circ (\nabla_{\mathbb{N}I} x\alpha \circ \beta) && \{\alpha \circ \beta = \alpha\beta_\diamond\} \\ &= (\nabla_{I\mathcal{Q}_*(\mathbb{N})} \circ \nabla_{\mathbb{N}I} x\alpha) \circ \beta && \big\{\beta^\bullet = \beta, \text{ Associative law}\big\} \\ &= x\alpha^\bullet \circ \beta \\ &= x\alpha \circ \beta && \big\{\alpha^\bullet = \alpha\big\} \\ &= x(\alpha \circ \beta). && \square \end{aligned}$$

Proposition 15. *If $\alpha : X \multimap \mathcal{Q}_1(Y)$ and $\beta : Y \multimap \mathcal{Q}_1(Z)$ are $\mathcal{Q}_1$-convex relations, then so is the convex composite $\alpha \circ \beta$.*

Proof. The proof is the same as the proof (a) and (c) of Proposition 14. $\square$

In the rest of paper, the subscript τ is one of 1 and $*$. For a set χ of $\mathcal{Q}_\tau$-convex relations $\alpha : X \multimap \mathcal{Q}_\tau(Y)$ define

$$\bigvee \chi = \Big(\bigsqcup \chi\Big)^\bullet.$$

It is trivial that $\bigvee \chi$ gives the join (the least upper bound) of χ.
The following proposition shows the right distributivity over all joins.

Proposition 16. *Let $\alpha : X \multimap \mathcal{Q}_\tau(X)$ and $\beta : X \multimap \mathcal{Q}_\tau(X)$ be relations.*

(a) $\alpha^\bullet \circ \beta \sqsubseteq (\alpha \circ \beta)^\bullet$,
(b) $(\bigvee \chi) \circ \beta = \bigvee(\chi \circ \beta)$.

Proof.
(a) $\alpha^\bullet \circ \beta \sqsubseteq (\alpha \circ \beta)^\bullet$:

$$\begin{aligned}
\forall x \in X.\ x(\alpha^\bullet \circ \beta) &= x\alpha^\bullet\beta_\diamond \\
&= \nabla_{I\mathcal{Q}_\tau(\mathbb{N})}(\nabla_{\mathbb{N}I}x\alpha)_\diamond\beta_\diamond \\
&\sqsubseteq \nabla_{I\mathcal{Q}_\tau(\mathbb{N})}(\nabla_{\mathbb{N}I}x\alpha\beta_\diamond)_\diamond \quad \{\text{Proposition 11(c)}\} \\
&= x(\alpha\beta_\diamond)^\bullet.
\end{aligned}$$

(b) $(\bigvee\chi)\circ\beta = \bigvee(\chi\circ\beta)$:

$$\begin{aligned}
\left(\bigvee\chi\right)\circ\beta &= \left(\bigsqcup\chi\right)^\bullet \circ \beta \\
&\sqsubseteq \left(\left(\bigsqcup\chi\right)\circ\beta\right)^\bullet \quad \{\text{(a)}\} \\
&= \left(\bigsqcup(\chi\circ\beta)\right)^\bullet \\
&= \bigvee(\chi\circ\beta),
\end{aligned}$$

$$\begin{aligned}
\alpha \in \chi \quad &\to \quad \alpha\circ\beta \sqsubseteq \left(\bigvee\chi\right)\circ\beta \qquad \left\{\alpha \sqsubseteq \bigvee\chi\right\} \\
&\to \quad \bigvee(\alpha\circ\beta) \sqsubseteq \left(\bigvee\chi\right)\circ\beta. \qquad \square
\end{aligned}$$

The following example shows that the left distributivity $\alpha\circ\bigvee\chi = \bigvee(\alpha\circ\chi)$ needs not hold in general.

Example 5. Let $\alpha', \beta : X \rightharpoonup \mathcal{Q}_*(X)$ be $\mathcal{Q}_*$-convex relations such that $\alpha' = \alpha(\xi_X^*)^\sharp$, $\beta = h(\xi_X^*)^\sharp$ where X, α and h are appeared in Example 4. Then $\alpha h_\diamond \sqsubseteq \alpha(\xi_X^*)^\sharp = \alpha'$ holds since $x\alpha h_\diamond = y\alpha h_\diamond = p_0 h_\diamond = \frac{1}{2}\cdot\dot{y} \leqslant p_0 = x\alpha = y\alpha$. Shown in Example 2, a relation $h \sqcup e_X$ consists of four maps, that is $h \sqcup e_X = f' \sqcup g \sqcup h \sqcup e_X$. Then we have

$$\begin{aligned}
p_0 f'_\diamond &\sqsubseteq x\alpha'(h\sqcup e_X)_\diamond \qquad \left\{p_0 = x\alpha \sqsubseteq x\alpha',\ f'_\diamond \sqsubseteq (h\sqcup e_X)_\diamond\right\} \\
&\sqsubseteq x\alpha'\left(h(\xi_X^*)^\sharp \sqcup e_X(\xi_X^*)^\sharp\right)_\diamond \\
&= x\alpha'\left(\beta \sqcup \varepsilon_X^*\right)_\diamond \\
&= x\left(\alpha'\circ\left(\beta\sqcup\varepsilon_X^*\right)\right) \\
&\sqsubseteq x\left(\alpha'\circ\left(\beta\vee\varepsilon_X^*\right)\right).
\end{aligned}$$

On the other hand, we have $p_0 f'_\diamond = \dot{y} \not\sqsubseteq x\alpha' = x(\alpha'\circ\beta \vee \alpha'\circ\varepsilon_X^*)$ since

$$\begin{aligned}
\alpha'\circ\beta \sqcup \alpha'\circ\varepsilon_X^* &= \alpha'\beta_\diamond \sqcup \alpha'\left(\varepsilon_X^*\right)_\diamond \\
&= \alpha'\left(h(\xi_X^*)^\sharp\right)_\diamond \sqcup \alpha'\left(e_X(\xi_X^*)^\sharp\right)_\diamond \\
&= \alpha' h_\diamond(\xi_X^*)^\sharp \sqcup \alpha'\,\mathrm{id}_{\mathcal{Q}_*(X)}(\xi_X^*)^\sharp \quad \{\text{Propositions 10(d), 7(c)}\} \\
&= \alpha(\xi_X^*)^\sharp h_\diamond(\xi_X^*)^\sharp \sqcup \alpha'(\xi_X^*)^\sharp \\
&= \alpha h_\diamond(\xi_X^*)^\sharp \sqcup \alpha' \qquad \{\text{Proposition 7(g), } \alpha' : \text{convex}\} \\
&= \alpha' \qquad \{\alpha h_\diamond \sqsubseteq \alpha',\ \alpha' : \text{convex}\}.
\end{aligned}$$

Therefore $\alpha'\circ(\beta'\vee\varepsilon_X^*) \neq \alpha'\circ\beta'\vee\alpha'\circ\varepsilon_X^*$.

Finally, we state several results about directed sets of $\mathcal{Q}_*$-convex relations and their joins.

Lemma 2. *If χ is a directed set of $\mathcal{Q}_*$-convex relations $\alpha : X \rightharpoonup \mathcal{Q}_*(Y)$, then*

$$\left(\bigsqcup\chi\right)_\diamond = \bigsqcup_{\alpha\in\chi}\alpha_\diamond.$$

Proof. The inclusion $\bigsqcup_{\alpha\in\chi}\alpha_\diamond \sqsubseteq (\bigsqcup\chi)_\diamond$ is trivial. We will show that $p(\bigsqcup\chi)_\diamond \sqsubseteq \bigsqcup_{\alpha\in\chi} p\alpha_\diamond$ for all $p\in\mathcal{Q}_*(X)$. For a map $f: X\to\mathcal{Q}_*(Y)$ we have

$$\begin{aligned}
f\sqsubseteq\bigsqcup\chi \quad &\to \quad \forall x\in X.\, xf\sqsubseteq x\Big(\bigsqcup\chi\Big)\\
&\to \quad xf\sqsubseteq \bigsqcup_{\alpha\in\chi} x\alpha\\
&\to \quad \exists\alpha_x\in\chi.\, xf\sqsubseteq x\alpha_x\\
&\xrightarrow{*} \quad \exists\alpha_f\in\chi\ \forall x\in\lfloor p\rfloor.\, xf\sqsubseteq x\alpha_f.
\end{aligned}$$

Note $\xrightarrow{*}$ follows from the assumption that $\lfloor p\rfloor$ is finite and χ is directed. Define a map $f': X\to\mathcal{Q}_*(Y)$ by

$$\forall x\in X.\, xf' = \begin{cases} xf & \text{if } x\in\lfloor p\rfloor,\\ 0_Y & \text{otherwise.}\end{cases}$$

It is clear that $pf_\diamond = pf'_\diamond$. Also $f'\sqsubseteq\alpha_f$ holds, because $\nabla_{XI}0_Y\sqsubseteq\alpha$ (α is total and $\alpha(\xi_Y^\tau)^\sharp=\alpha$). Hence $pf_\diamond = pf'_\diamond\sqsubseteq p\alpha_\diamond$ and so

$$\begin{aligned}
p\Big(\bigsqcup\chi\Big)_\diamond &= p\Big(\bigsqcup_{f\sqsubseteq\bigsqcup\chi} f_\diamond\Big)\\
&= \bigsqcup_{f\sqsubseteq\sqcup\chi} pf_\diamond\\
&\sqsubseteq \bigsqcup_{\alpha\in\chi} p\alpha_\diamond \qquad \{pf_\diamond\sqsubseteq p\alpha_\diamond\}\\
&= p(\bigsqcup_{\alpha\in\chi}\alpha_\diamond). \qquad \square
\end{aligned}$$

The following proposition gives the directed join of $\mathcal{Q}_*$-convex relations.

Proposition 17. *Let χ be a directed set of $\mathcal{Q}_*$-convex relations $\alpha: X\multimap\mathcal{Q}_*(X)$. Then*

$$\bigvee\chi = \bigsqcup\chi.$$

Proof.

$$\begin{aligned}
x\Big(\bigvee\chi\Big) &= x\Big(\bigsqcup\chi\Big)^\bullet\\
&= \nabla_{I\mathcal{Q}_*(\mathbb{N})}\Big(\nabla_{\mathbb{N}I}x\Big(\bigsqcup\chi\Big)\Big)_\diamond\\
&= \nabla_{I\mathcal{Q}_*(\mathbb{N})}\Big(\bigsqcup_{\alpha\in\chi}\nabla_{\mathbb{N}I}x\alpha\Big)_\diamond\\
&= \nabla_{I\mathcal{Q}_*(\mathbb{N})}\bigsqcup_{\alpha\in\chi}(\nabla_{\mathbb{N}I}x\alpha)_\diamond \qquad \{\text{Lemma 2}\}\\
&= \bigsqcup_{\alpha\in\chi}\nabla_{I\mathcal{Q}_*(\mathbb{N})}(\nabla_{\mathbb{N}I}x\alpha)_\diamond\\
&= \bigsqcup_{\alpha\in\chi} x\alpha^\bullet\\
&= \bigsqcup_{\alpha\in\chi} x\alpha\\
&= x\Big(\bigsqcup\chi\Big). \qquad \square
\end{aligned}$$

The composition of $\mathcal{Q}_*$-convex relations distributes all directed joins from the left-hand side.

Proposition 18. *Let $\alpha : X \multimap \mathcal{Q}_*(Y)$ be a $\mathcal{Q}_*$-convex relation and χ a directed set of $\mathcal{Q}_*$-convex relations $\beta : Y \multimap \mathcal{Q}_*(Z)$. Then*

$$\alpha \circ \bigvee \chi = \bigvee (\alpha \circ \chi).$$

Proof.

$$\begin{aligned}
\alpha \circ \bigvee \chi &= \alpha \Big(\bigsqcup \chi \Big)_\diamond && \{\text{Proposition 17}\} \\
&= \alpha \Big(\bigsqcup_{\beta \in \chi} \beta_\diamond \Big) && \{\text{Lemma 2}\} \\
&= \bigsqcup_{\beta \in \chi} \alpha \beta_\diamond \\
&= \bigsqcup_{\beta \in \chi} \alpha \circ \beta \\
&= \bigsqcup (\alpha \circ \chi) \\
&= \bigvee (\alpha \circ \chi). && \square
\end{aligned}$$

7. Conclusion

In this paper we have studied the relations into algebras of probabilistic distributions using relational calculi, although McIver et al. [7] and Tsumagari [16] studied in set-theoretical way. We have shown the following:

- The set of $\mathcal{Q}_\tau$-convex relations forms a category with the convex composition, and the identity morphisms depending on $\tau \in \{*, 1\}$.
- For $\tau \in \{*, 1\}$ the convex composition of $\mathcal{Q}_\tau$-convex relations distributes over all non-empty joins from the right-hand side.
- The convex composition of $\mathcal{Q}_*$-convex relations distributes over all non-empty directed joins even from the left-hand side.

We have proved the associative law of convex composition for $\mathcal{Q}_*$-convex relations and $\mathcal{Q}_1$-convex relations in the same framework, though Tsumagari [16] had studied as their two convex-relations are different. Additionally we have given a counter example for the associative law of the convex composition in the absence of convexity.

The convex composition studied in this paper seems to be a generalization of reachability composition of multirelations. So we might be interested in the another composition of $\mathcal{Q}_\tau$-convex relations, corresponding to the composition of up-closed multirelations studied by Parikh [11,12] and Rewitzky [6,14].

Acknowledgements

First and foremost, we wish to express our deep gratitude to Gunther Schmidt for his leadership and encouragement as a trailblazer in the field of relational methods in computer science. Discussions with Georg Struth have been helpful to improve the paper. We also appreciate anonymous referees for their helpful comments and suggestions. This work was supported in part by Grants-in-Aid for Scientific Research (C) 25330016 from Japan Society for the Promotion of Science (JSPS).

References

[1] R. Berghammer, H. Zierer, Relational algebraic semantics of deterministic and nondeterministic programs, Theor. Comput. Sci. 43 (2–3) (1986) 123–147.
[2] P. Eklund, W. Gähler, Partially ordered monads and powerset Kleene algebras, in: Proc. 10th Information Processing and Management of Uncertainty in Knowledge Based Systems Conference (IPMU 2004), 2004.
[3] R. Goldblatt, Parallel action: concurrent dynamic logic with independent modalities. Logic of action, Stud. Log. 51 (3–4) (1992) 551–578.
[4] H. Jifeng, K. Seidal, A. McIver, Probabilistic models for the guarded command language, Sci. Comput. Program. 28 (1997) 171–192.
[5] D. Kozen, A completeness theorem for Kleene algebras and the algebra of regular events, in: 1991 IEEE Symposium on Logic in Computer Science, Amsterdam, 1991, Inf. Comput. 110 (2) (1994) 366–390.
[6] C.E. Martin, S.A. Curtis, I. Rewitzky, Modelling nondeterminism, in: Mathematics of Program Construction, in: Lecture Notes in Computer Science, vol. 3125, Springer, Berlin, 2004, pp. 228–251.
[7] A.K. McIver, E. Cohen, C.C. Morgan, Using probabilistic Kleene algebra for protocol verification, in: Relations and Kleene Algebra in Computer Science, in: Lecture Notes in Computer Science, vol. 4136, Springer, 2006, pp. 296–310.
[8] A.K. McIver, C.C. Morgan, Abstraction, Refinement and Proof for Probabilistic Systems, Springer, 2005.
[9] B. Möller, Lazy Kleene algebra, in: Mathematics of Program Construction, in: Lecture Notes in Computer Science, vol. 3125, Springer, 2004, pp. 252–273.

[10] K. Nishizawa, N. Tsumagari, H. Furusawa, The cube of Kleene algebras and the triangular prism of multirelations, in: Relations and Kleene Algebra in Computer Science, in: Lecture Notes in Computer Science, vol. 5827, Springer, 2009, pp. 276–290.
[11] R. Parikh, Propositional logics of programs: new directions, in: Foundations of Computation Theory, Borgholm, 1983, in: Lecture Notes in Computer Science, vol. 158, Springer, 1983, pp. 347–359.
[12] M. Pauly, R. Parikh, Game logic—an overview, in: Game Logic and Game Algebra, Helsinki, 2001, Stud. Log. 75 (2) (2003) 165–182.
[13] D. Peleg, Concurrent dynamic logic, J. ACM 34 (2) (1987) 450–479.
[14] I. Rewitzky, Binary multirelations, in: H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments, in: Lecture Notes in Computer Science, vol. 2929, Springer, 2003, pp. 256–271.
[15] G. Schmidt, T. Ströhlein, Relations and Graphs – Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoretical Computer Science, Springer, 1993.
[16] N. Tsumagari, Probabilistic relational models of complete IL-semirings, in: Bulletin of Informatics and Cybernetics, vol. 44, Research Association of Statistical Science, 2012, pp. 87–109.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 284–299

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Relational properties of sequential composition of coalgebras

Michael Winter [a,*,1], Peter Kempf [b]

[a] Department of Computer Science, Brock University, St. Catharines, Canada
[b] Osram GmbH, Munich, Germany

ARTICLE INFO

Article history:
Available online 12 February 2014

Keywords:
Coalgebra
Bisimulation
Sequential composition
Dedekind categories

ABSTRACT

In this paper we define a sequential composition for arbitrary coalgebras in a Dedekind category. We show some basic algebraic properties of this operation up to bisimulation. Furthermore, we consider certain recursive equations and provide an explicit solution, i.e., a solution not based on an iterative process.

1. Introduction

Coalgebras, bisimulation, and coinduction play an important role in mathematics and computer science. One of the first examples of bisimulations appear in process calculi such as CSP or CCS [5,7]. Such processes can be modeled by coalgebras, and the behavioral equivalence is based on bisimulation. These calculi also introduce the basic operations of parallel composition, summation, and prefixing on processes. In a recent paper [2] a coalgebraic logic for deterministic Mealy machines that is sound and complete was presented.

In addition, models of modal logics give naturally rise to coalgebras based on the underlying transition relation of $\Box$ and $\Diamond$ [10]. Bisimulation and coinduction are used to reason about models, property preserving constructions, and relationship to other logics. For example, Van Benthem's characterization theorem shows that modal logic is the fragment of first-order logic that is invariant under bisimulation. Another example is given by the method of filtration that is used to show that the satisfiability problem of certain modal logics is decidable. It has been shown that filtration is based on a bisimilarity relation [13].

Recently there has been great interest in using coinduction and bisimulation to reason about lazy functional programming languages. The first motivation for this approach was given in [1]. In this paper the lazy lambda calculus has been defined and it was shown a bisimulation, called applicative bisimulation, is a congruence on the terms of this calculus. The interest in lazy language also started intensive research on coalgebraic specification. These specifications are based on observation operations instead of constructors, an approach that leads to algebraic specifications. For an intensive study of examples of coalgebraic specifications we refer to [6].

For additional examples and a more detailed overview of methods and applications of coalgebras, bisimulation, and coinduction we refer to [11,12].

In this paper we want to consider coalgebras in the context of Dedekind categories. These categories provide a suitable abstraction to reason about relation [8,9]. Therefore, a coalgebra is a relation $Q : S \to F(S)$ with an appropriate functor F. This generalizes processes, labeled transition systems, or other coalgebraic structures such as Kripke structures, in multiple

* Corresponding author. Tel.: +1 905 688 5550 ext. 3355; fax: +1 905 688 3255.
E-mail addresses: mwinter@brocku.ca (M. Winter), P.Kempf@osram.com (P. Kempf).
[1] The author gratefully acknowledges support from the Natural Sciences and Engineering Research Council of Canada.

http://dx.doi.org/10.1016/j.jlap.2014.02.011

ways. First of all, relations in Dedekind category need not to be relations in the classical sense, i.e., subsets of the cartesian product of sets. Lattice-valued or L-fuzzy relations as well as probabilistic relations are models of Dedekind categories. Among the huge class of additional examples there are even non-representable Dedekind categories, i.e., categories where the morphisms are not equivalent to any notion of a relation based on sets of pairs. The second generalization is given by the functor F. A coalgebra $Q : S \to F(S)$ has two aspects. It provides a transition from a state to a successor state together with an additional effect encoded by F. In this paper we will make no assumption on this additional behavior.

Relational coalgebras have been studied before in [17,18]. These paper mainly studied parallel composition of processes and its relational properties. It was shown, for example, that equivalence classes of certain coalgebras form an ordered category based on parallel composition. In the current paper we want to concentrate on sequential composition.

This paper is organized as follows. Section 2 introduce the basic mathematical notions such as Dedekind categories, relators, coalgebras, rooted coalgebras, and bisimulations. In Section 3 we define the sequential composition of two rooted coalgebras. After investigating some basic properties of this operation we define the sum of rooted coalgebras in Section 4. This is corresponds to the disjoint union of coalgebras by taking their roots into account. Furthermore, some basic property of this operation are shown. The main theorem of this section shows that the sequential composition distributes from the right over the sum. In Section 5 we consider simple recursion on a coalgebra by providing an explicit solution to the equation $X = P.X$ where , denotes the sequential composition. Finally, Section 6 outlines how to solve general equations in a language based on the construction defined in this paper. As in Section 5 the solution is given explicitly.

Due to length restrictions we had to omit several proofs in this paper. The missing parts can be found in the extended version of the paper [19].

2. Mathematical preliminaries

Throughout this paper, we use the following notation. To indicate that a morphism R of a category $\mathcal{R}$ has source A and target B we write $R : A \to B$. The collection of all morphisms $R : A \to B$ is denoted by $\mathcal{R}[A, B]$ and the composition of a morphism $R : A \to B$ followed by a morphism $S : B \to C$ by $R; S$. Last but not least, the identity morphism on A is denoted by $\mathbb{I}_A$.

In this section we recall some fundamentals on Dedekind categories [8,9]. This kind of category is called locally complete division allegories in [4].

Definition 1. A Dedekind category $\mathcal{R}$ is a category satisfying the following:

1. For all objects A and B the collection $\mathcal{R}[A, B]$ of morphisms/relations is a complete distributive lattice. Meet, join, the induced ordering, the least and the greatest element are denoted by $\sqcap$, $\sqcup$, $\sqsubseteq$, $\perp\!\!\!\perp_{AB}$ and $\top\!\!\!\top_{AB}$, respectively.
2. There is a monotone operation $^\smile$ (called converse) mapping a relation $Q : A \to B$ to a relation $Q^\smile : B \to A$ such that $(Q; R)^\smile = R^\smile; Q^\smile$ and $(Q^\smile)^\smile = Q$ for all relations $Q : A \to B$ and $R : B \to C$.
3. For all relations $Q : A \to B$, $R : B \to C$ and $S : A \to C$ the modular law $Q; R \sqcap S \sqsubseteq Q; (R \sqcap Q^\smile; S)$ holds.[2]
4. For all relations $R : B \to C$ and $S : A \to C$ there is a relation $S/R : A \to B$ (called the left residual of S and R) such that for all $Q : A \to B$ the following holds: $Q; R \sqsubseteq S \iff Q \sqsubseteq S/R$.

As already indicated in the definition above we will use morphism and relation interchangeably in the context of Dedekind categories.

Throughout this paper we will use several basic properties of Dedekind categories such as $\mathbb{I}_A^\smile = \mathbb{I}_A$, the monotonicity of composition in both parameters, or the distributivity of ; over $\sqcup$ without mentioning. For details we refer to [3,4,14–16].

An important class of relations within Dedekind categories are mappings.

Definition 2. Let $Q : A \to B$ be a relation. Then we call

1. Q univalent iff $Q^\smile; Q \sqsubseteq \mathbb{I}_B$,
2. Q total iff $\mathbb{I}_A \sqsubseteq Q; Q^\smile$,
3. Q a mapping iff Q is univalent and total.

In the next lemma we recall an important properties of mappings that we will use in this paper.

Lemma 1. *Suppose $Q : A \to B$ and $R : D \to C$ are relations, and $f : B \to C$ is a mapping. Then we have*

$$Q; f \sqsubseteq R \iff Q \sqsubseteq R; f^\smile.$$

[2] By convention the precedence of the operations decreases in the following order $.^\smile$ then ; then $\sqcap$.

A proof may be found in [14–16].

The notion of a unit in a category of relations corresponds to terminal objects in categories of functions.

Definition 3. A unit I of a Dedekind category $\mathcal{R}$ is an object of $\mathcal{R}$ so that $\mathbb{I}_I = \top_{II}$ and $\top_{AI}; \top_{IA} = \top_{AA}$, i.e., $\top_{AI}$ is total, for all objects A.

A unit is a terminal object in the subcategory of mapping, and, therefore, unique up to isomorphism. In the Dedekind category of sets and relations a unit is any singleton set.

A relation $v : I \to A$ is called a vector. Such a relation represents a subset of A in an abstract manner. Elements of A can be modeled by singleton subsets, i.e., by a vector that is a mapping. We call such a relation a point, i.e., $p : I \to A$ is a point iff p is a mapping.

Another important construction is based on forming the disjoint union of sets.

Definition 4. Let A and B be objects of a Dedekind category. An object $A + B$ together with relations $\iota : A \to A + B$ and $\kappa : B \to A + B$ is called a relational sum of A and B iff

$$\iota; \iota^\smile = \mathbb{I}_A, \qquad \kappa; \kappa^\smile = \mathbb{I}_B, \qquad \iota; \kappa^\smile = \mathbb{L}_{AB}, \qquad \iota^\smile; \iota_i \sqcup \kappa^\smile; \kappa = \mathbb{I}_{A+B}.$$

$\mathcal{R}$ has (binary) relational sums iff for every pair of objects the relational sum does exist.

Notice that a relational sum is a biproduct of the Dedekind category. We will use the following abbreviations for relation $Q : A \to C$, $R : B \to C$ and $S : B \to D$:

$$[Q, R] := \iota^\smile; Q \sqcup \kappa^\smile; R,$$
$$Q + S := \iota^\smile; Q; \iota \sqcup \kappa^\smile; S; \kappa.$$

The relations $[Q, R] : A + B \to C$ and $Q + S : A + B \to C + D$ satisfy the following properties:

$$\iota; [Q, R] = Q, \qquad \kappa; [Q, R] = R, \qquad [Q, R]; T = [Q; T, R; T],$$
$$Q + S = [Q; \iota, S; \kappa], \qquad (Q + S); \iota^\smile = \iota; Q, \qquad (Q + S); \kappa^\smile = \kappa; R,$$
$$(Q + S); [U, V] = [Q; U, S; V].$$

It is well-known that morphisms on biproducts can be represented by matrices. Each row (column) of the matrix corresponds to one factor of the relational sum of the source (target) of the relation. For example, if $Q : A \to C$, $R : A \to D$, $S : B \to C$ and $T : B \to D$, then the matrix representation of $X = \iota^\smile; Q; \iota \sqcup \iota^\smile; R; \kappa \sqcup \kappa^\smile; S; \iota \sqcup \kappa^\smile; T; \kappa$ is

$$X = \begin{pmatrix} Q & R \\ S & T \end{pmatrix}.$$

Similar, if $Y : A + B \to C + D$, then we have

$$Y = \begin{pmatrix} \iota; Y; \iota^\smile & \iota; Y; \kappa^\smile \\ \kappa; Y; \iota^\smile & \kappa; Y; \kappa^\smile \end{pmatrix}.$$

In particular, for $[Q, R]$ and $Q + S$ we obtain

$$[Q, R] = \begin{pmatrix} Q \\ P \end{pmatrix}, \qquad Q + S = \begin{pmatrix} Q & \mathbb{L}_{AD} \\ \mathbb{L}_{BC} & S \end{pmatrix}.$$

The operations on relations correspond to matrix operations on the their matrix representation. In most papers, nested biproducts, and, hence, nested matrices, are flattened, i.e., intermediate brackets are ignored due to the associativity of biproducts. We will not adapt this approach here because we want to make the corresponding isomorphisms, and, hence, all bisimulation relations, explicit. However, the relation $\mathrm{assoc} : S_1 + (S_2 + S_3) \to (S_1 + S_2) + S_3$ defined by

$$\mathrm{assoc} = [\iota; \iota, \kappa + \mathbb{I}_{S_3}] = \iota^\smile; \iota; \iota \sqcup \kappa^\smile; \iota^\smile; \kappa; \iota \sqcup \kappa^\smile; \kappa^\smile; \kappa$$

is the isomorphism induced by the associativity of the relational sum. As a consequence we have

$$\mathrm{assoc}; \big((Q + R) + S\big) = \big(Q + (R + S)\big); \mathrm{assoc}$$

for all suitable Q, R and S.

Another important concept is the splitting of a partial equivalence relation. This concept combines two well-known constructions on sets.

Definition 5. Let $\mathcal{R}$ be a Dedekind category, and $Q : A \to A$ be partial equivalence relation, i.e. a symmetric idempotent relation such that $Q^\smile = Q$ and $Q ; Q = Q$. An object B together with a relation $R : B \to A$ is called a splitting of Q (or R splits Q) iff $R ; R^\smile = \mathbb{I}_B$ and $R^\smile ; R = Q$. $\mathcal{R}$ has splittings iff all symmetric idempotent relations in $\mathcal{R}$ split.

A splitting is unique up to isomorphism. If Q is a partial identity, then the object B of the splitting corresponds to the subset given by Q. Analogously, if Q is an equivalence relation, B corresponds to the set of equivalence classes.

A relator $F : \mathcal{R} \to \mathcal{S}$ between two Dedekind categories $\mathcal{R}$ and $\mathcal{S}$ is a functor that is monotonic on relations and preserves converse. Notice that a relator preserves the properties of being total, univalent, and a mapping, i.e., it maps mappings to mappings.

2.1. Coalgebras and bisimulation

As mentioned in the introduction coalgebras play an important role in multiple areas of mathematics and computer science. We are especially interested in coalgebras based on Dedekind categories. Given an endorelator $F : \mathcal{R} \to \mathcal{R}$ a relation $Q : S \to F(S)$ is called an F coalgebra (with states space S).

Definition 6. A relation $\Phi : S_1 \to S_2$ is called a bisimulation between the two F coalgebras $P : S_1 \to F(S_1)$ and $Q : S_2 \to F(S_2)$ iff

$$\Phi ; Q \sqsubseteq P ; F(\Phi) \quad \text{and} \quad \Phi^\smile ; P \sqsubseteq Q ; F(\Phi^\smile).$$

$$\begin{array}{ccc} S_1 & \xrightarrow{\Phi} & S_2 \\ {\scriptstyle P}\downarrow & & \downarrow{\scriptstyle Q} \\ F(S_1) & \xrightarrow[F(\Phi)]{} & F(S_2) \end{array}$$

Notice that by using residuals the two inclusion above can be transformed into a single inclusion

$$\Phi \sqsubseteq \big(P ; F(\Phi)\big)/Q \sqcap P^\smile \backslash \big(F(\Phi) ; Q^\smile\big).$$

In order to call two F coalgebras bisimilar we require that there is a total and surjective bisimulation between them, i.e., each state in one process corresponds to at least one state in the other process. However, notice that we will concentrate on rooted coalgebras in this paper for which it is sufficient that the roots are bisimilar (see Definition 8).

Given two coalgebras it is possible to define a coalgebra on the relational sum of their states. Notice that we do not require any extra properties for the relator F, neither in the following definition nor anywhere else in the paper. In particular, we never require that F is monoidal, i.e., that $F(A + B)$ is isomorphic to $F(A) + F(B)$ for all objects A and B.

Definition 7. Let $P : S_1 \to F(S_1)$ and $Q : S_2 \to F(S_2)$ be coalgebras. Then define

$$P \oplus Q := \big[P ; F(\iota), Q ; F(\kappa)\big] : S_1 + S_2 \to F(S_1 + S_2).$$

In matrix representation we obtain

$$P \oplus Q = \begin{pmatrix} P ; F(\iota) \\ Q ; F(\kappa) \end{pmatrix}.$$

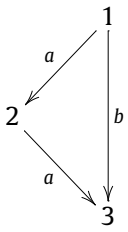
Notice that we also have

$$\begin{aligned} &(Q + R) ; (S \oplus T) ; F(U + V) \\ &\quad = (Q + R) ; \big[S ; F(\iota), T ; F(\kappa)\big] ; F(U + V) \\ &\quad = \big[Q ; S ; F(\iota), R ; T ; F(\kappa)\big] ; F(U + V) \\ &\quad = \iota^\smile ; Q ; S ; F\big(\iota ; (U + V)\big) \sqcup \kappa^\smile ; R ; T ; F\big(\kappa ; ; (U + V)\big) \\ &\quad = \iota^\smile ; Q ; S ; F(U ; \iota) \sqcup \kappa^\smile ; R ; T ; F(V ; \kappa) \\ &\quad = \big(Q ; S ; F(U)\big) \oplus \big(R ; T ; F(V)\big) \end{aligned}$$

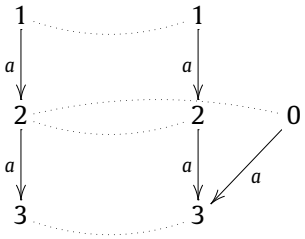
and with respect to the isomorphism assoc

$$\begin{aligned}
\text{assoc}; \big((P \oplus Q) \oplus U\big) &= \big((\iota^\smile; \iota \sqcup \kappa^\smile; \iota^\smile; \kappa); \iota \sqcup \kappa^\smile; \kappa^\smile; \kappa\big); \big((P \oplus Q) \oplus U\big) \\
&= \big(\iota^\smile; \iota \sqcup \kappa^\smile; \iota^\smile; \kappa\big); (P \oplus Q); F(\iota) \sqcup \kappa^\smile; \kappa^\smile; U; F(\kappa) \\
&= \iota^\smile; P; F(\iota; \iota) \sqcup \kappa^\smile; \iota^\smile; Q; F(\kappa; \iota) \sqcup \kappa^\smile; \kappa^\smile; U; F(\kappa) \\
&= \iota^\smile; P; F(\iota; \iota) \sqcup \kappa^\smile; (Q \oplus U); F\big(\iota^\smile; \kappa; \iota \sqcup \kappa^\smile; \kappa\big) \\
&= \big(P \oplus (Q \oplus U)\big); F\big(\iota^\smile; \iota; \iota \sqcup \kappa^\smile; \big(\iota^\smile; \kappa; \iota \sqcup \kappa^\smile; \kappa\big)\big) \\
&= \big(P \oplus (Q \oplus U)\big); F(\text{assoc}).
\end{aligned}$$

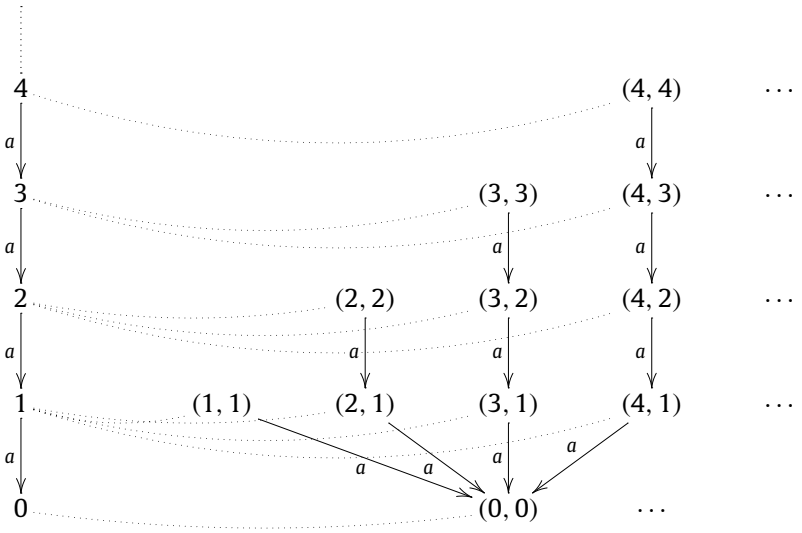
For the examples in this paper we will always use the Dedekind category of sets and relations together with the functor $F(S) = L \times S$. We will chose a particular representation of the unit I as the set that contains the element $*$. A coalgebra $Q : S \to F(S)$ in this category is a labeled transition system with state space S and labels from the set L. For example, if $S = \{1, 2, 3\}$, $L = \{a, b\}$ and $Q = \{(1, (a, 2)), (1, (b, 3)), (2, (a, 3))\}$, then this system can be visualized as a labeled graph by



Notice that labeled transition systems as F coalgebras with F defined above do not have an explicit initial state. Even though it seems natural to claim that a source node of the graph is an initial state of the labeled transition system, this is not the case. Actually any state may serve as the initial (or current) state. All of this can be illustrated by the fact that the following two labeled graphs are bisimilar in the sense defined after Definition 6, i.e., each state of each coalgebra is related by a bisimulation to at least one state of the other coalgebra. The dotted lines indicate the largest bisimulation between the graphs which is indeed total and surjective.



The state 2 in the left graph is not a source of the graph. However, 2 is bisimilar to the state 0 of the second graph which is a source of that graph. As a second example consider the following two infinite graphs:



The left graph does not have any sources but each state is bisimilar to a source in the right graph. Notice that those graphs are not bisimilar to $\{(0, (a, 0)\}$, i.e., the graph:

Since this paper studies sequential composition of F coalgebras we need to make an initial state explicit. Therefore, we will consider rooted F coalgebras. A rooted coalgebra is a pair (P, i) consisting of a F coalgebra $P : S \to F(S)$ and a point $i : I \to S$; its initial state. We will use the notation $(P, i) : S \to F(S)$ to indicate that (P, i) is a rooted F coalgebra with states from S.

To call two rooted coalgebras bisimilar it is not longer necessary that all states of one coalgebra correspond to at least one state of the other coalgebra. It is sufficient that the all the states that are reachable from the initial state are covered, i.e., that there is a bisimulation that is total and surjective with respect to the reachable states. Equivalently, we may require that the initial states are related.

Definition 8. Two rooted F coalgebras $(P, i_P) : S_1 \to F(S_1)$ and $(Q, i_Q) : S_2 \to F(S_2)$ are called bisimilar, denoted by $(P, i_P) \sim (Q, i_Q)$, iff there is a bisimulation $\Phi : S_1 \to S_2$ so that $i_P^\smile ; i_Q \sqsubseteq \Phi$.

Notice that the property above is equivalent by Lemma 1 to either of the two inclusions

$$i_Q \sqsubseteq i_P ; \Phi, \qquad i_P \sqsubseteq i_Q ; \Phi^\smile.$$

On the other hand, an arbitrary coalgebra P may have terminal states or nodes. These are described by the vector $\perp\!\!\!\perp_{IF(S)}/P$ since

$$v \sqsubseteq \perp\!\!\!\perp_{IF(S)}/P \quad \Longleftrightarrow \quad v ; P \sqsubseteq \perp\!\!\!\perp_{IF(S)},$$

i.e., this vector describes those states from which no transition is possible. We will denote this vector also by t_P, i.e., $t_P = \perp\!\!\!\perp_{IF(S)}/P$. Notice that we have $\top\!\!\!\top_{AI} ; t_P = \perp\!\!\!\perp_{AF(S)}/P$ for every object A and $t_P^\smile = P^\smile \backslash \perp\!\!\!\perp_{F(S)I}$.

Lemma 2. *Suppose $\Phi : S_1 \to S_2$ is a bisimulation between $P_1 : S_1 \to F(S_1)$ and $P_2 : S_2 \to F(S_2)$. Then we have $t_{P_1} ; \Phi \sqsubseteq t_{P_2}$ and $t_{P_2} ; \Phi^\smile \sqsubseteq t_{P_1}$.*

Proof. From the computation

$$\begin{aligned} (\perp\!\!\!\perp_{AF(S_1)}/P_1) ; \Phi ; P_2 &\sqsubseteq (\perp\!\!\!\perp_{AF(S_1)}/P_1) ; P_1 ; F(\Phi) && \Phi \text{ bisimulation} \\ &\sqsubseteq \perp\!\!\!\perp_{AF(S_2)} && \text{Def. residual} \end{aligned}$$

we immediately conclude the first inclusion. The second inclusion follows analogously using the second property of bisimulations. □

3. Sequential composition of coalgebras

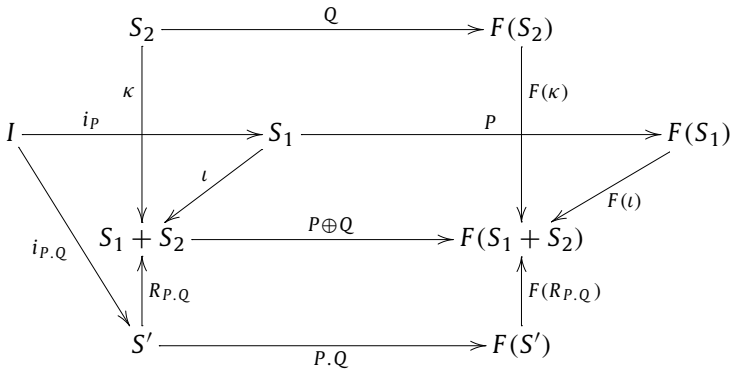
On of the main purposes of this paper is to define a sequential composition of coalgebras and show its basic properties. The idea of the sequential composition $P.Q$ is that if we reach a terminal state of P, then we continue at the initial state of Q. Notice that the sequential composition is a generalization of prefixing known from process calculi and similar languages. Prefixing is a sequential composition in which the first process consists of a single action only. The logic of coalgebras with prefixing was investigated in [2].

Definition 9. Let $\mathcal{R}$ be a Dedekind category with (binary) relational sums and subobjects. Furthermore, let $F : \mathcal{R} \to \mathcal{R}$ be an endorelator, $(P, i_P) : S_1 \to F(S_1)$ and $(Q, i_Q) : S_2 \to F(S_2)$ be rooted coalgebras, and $R_{P.Q} : S_{P.Q} \to S_1 + S_2$ the splitting of the relation $\Xi_{P.Q} = \mathbb{I}_{S_1+S_2} \sqcup (t_P ; \iota \sqcup i_Q ; \kappa)^\smile ; (t_P ; \iota \sqcup i_Q ; \kappa)$. Then we define a rooted coalgebra $(P.Q, i_{P.Q}) : S' \to F(S')$ by

$$P.Q := R_{P.Q} ; (P \oplus Q) ; F(R_{P.Q}^\smile),$$

$$i_{P.Q} := i_P ; \iota ; R_{P.Q}^\smile.$$

The typing of the relations in the previous definition can be visualized by:



It is easy to verify that $i_{P.Q}$ is indeed a point. This follows immediately from the fact that i_P, ι and $R^{\smile}_{P.Q}$ are all mappings.

The equivalence relation $\Xi_{P.Q}$ identifies the terminal states of P with the initial state of Q. The following computation presents this relation in matrix form:

$$\begin{aligned}
\Xi_{P.Q} &= \mathbb{I}_{S_1+S_2} \sqcup (t_P; \iota \sqcup i_Q; \kappa)^{\smile}; (t_P; \iota \sqcup i_Q; \kappa) \\
&= \iota^{\smile}; \iota \sqcup \kappa^{\smile}; \kappa \sqcup \iota^{\smile}; t_P^{\smile}; t_P; \iota \sqcup \iota^{\smile}; t_P^{\smile}; i_Q; \kappa \sqcup \kappa^{\smile}; i_Q^{\smile}; t_P; \iota \sqcup \kappa^{\smile}; i_Q^{\smile}; i_Q; \kappa \\
&= \iota^{\smile}; \left(\mathbb{I}_{S_1} \sqcup t_P^{\smile}; t_P\right); \iota \sqcup \iota^{\smile}; t_P^{\smile}; i_Q; \kappa \sqcup \kappa^{\smile}; i_Q^{\smile}; t_P; \iota \sqcup \kappa^{\smile}; \left(\mathbb{I}_{S_2} \sqcup i_Q^{\smile}; i_Q\right); \kappa \\
&= \iota^{\smile}; \left(\mathbb{I}_{S_1} \sqcup t_P^{\smile}; t_P\right); \iota \sqcup \iota^{\smile}; t_P^{\smile}; i_Q; \kappa \sqcup \kappa^{\smile}; i_Q^{\smile}; t_P; \iota \sqcup \kappa^{\smile}; \kappa \qquad i_Q \text{ univalent} \\
&= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^{\smile}; t_P & t_P^{\smile}; i_Q \\ i_Q^{\smile}; t_P & \mathbb{I}_{S_2} \end{pmatrix}.
\end{aligned}$$

Notice that $R^{\smile}_{P.Q}$ is a surjective mapping because it splits an equivalence relations.

Before we show properties about sequential composition we want to mention that one can define the sequential composition with respect of certain terminal states of P only, i.e., with respect to a vector $r \sqsubseteq t_P$. This composition continues with the initial state in Q only if a terminal state in r is reached. We denote this composition by $P \overset{r}{\cdot} Q$. Formally, this definition only requires to replace t_p in the definition of Ξ by r. All properties (or their obvious generalization) in this paper remain true because their proofs only use that terminal states are terminal, i.e., $t_P; P = \bot\!\!\!\bot_{SF(S)}$.

As a first property we want to show that the definition of a sequential composition respects the equivalence relation on rooted coalgebras induced by the notion of bisimilarity, i.e., it defines an operation on equivalence classes of bisimilar coalgebras.

Lemma 3. *Let $(P_1, i_{P_1}) : S_1 \to F(S_1), (P_2, i_{P_2}) : S_2 \to F(S_2), (Q_1, i_{Q_1}) : T_1 \to F(T_1)$ and $(Q_2, i_{Q_2}) : T_2 \to F(T_2)$ be rooted F coalgebras with $(P_1, i_{P_1}) \sim (P_2, i_{P_2})$ and $(Q_1, i_{Q_1}) \sim (Q_2, i_{Q_2})$. Then we have*

$$(P_1.Q_1, i_{P_1.Q_1}) \sim (P_2.Q_2, i_{P_2.Q_2}).$$

Proof sketch. Suppose $\Phi : S_1 \to S_2$ and $\Psi : T_1 \to T_2$ are bisimulations with $i_{P_1}^{\smile}; i_{P_2} \sqsubseteq \Phi$ and $i_{Q_1}^{\smile}; i_{Q_2} \sqsubseteq \Psi$. Then the relation $R_{P_1.Q_1}; (\Phi + \Psi); R^{\smile}_{P_2.Q_2} : S' \to T'$ is the required bisimulation from $P_1.Q_1$ to $P_2.Q_2$. The details of the proof can be found in [19]. □

As a second property we want to investigate the terminal states of $P.Q$. They are essentially given by the terminal states of Q modulo the equivalence relation $\Xi_{P.Q}$.

Lemma 4. *Let $(P, i_P) : S_1 \to F(S_1)$ and $(Q, i_Q) : S_2 \to F(S_2)$ be rooted F coalgebras. Then $t_{P.Q} = t_Q; \kappa; R^{\smile}_{P.Q}$.*

Proof. From the computation

$$\begin{aligned}
& t_Q; \kappa; R^{\smile}_{P.Q}; (P.Q) \\
&= t_Q; \kappa; R^{\smile}_{P.Q}; R_{P.Q}; (P \oplus Q); F\left(R^{\smile}_{P.Q}\right) \\
&= t_Q; \kappa; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^{\smile}; t_P & t_P^{\smile}; i_Q \\ i_Q^{\smile}; t_P & \mathbb{I}_{S_2} \end{pmatrix}; \begin{pmatrix} P; F(\iota) \\ Q; F(\kappa) \end{pmatrix}; F\left(R^{\smile}_{P.Q}\right)
\end{aligned}$$

$$
\begin{aligned}
&= t_Q ; \begin{pmatrix} i_Q^\smile ; t_P & \mathbb{I}_{S_2} \end{pmatrix} ; \begin{pmatrix} P ; F(\iota) \\ Q ; F(\kappa) \end{pmatrix} ; F\big(R_{P.Q}^\smile\big) \\
&= t_Q ; \big(i_Q^\smile ; t_P ; P ; F(\iota) \sqcup Q ; F(\kappa)\big) ; F\big(R_{P.Q}^\smile\big) \\
&= t_Q ; Q ; F(\kappa) ; F(R_{P.Q}^\smile) && t_P ; P = \mathbb{L}_{IF(S_1)} \\
&= \mathbb{L}_I && t_Q ; Q = \mathbb{L}_{IF(S_2)}
\end{aligned}
$$

we immediately conclude the inclusion $t_Q ; \kappa ; R_{P.Q}^\smile \sqsubseteq t_{P.Q}$. For the other inclusion we first compute

$$
\begin{aligned}
&\iota ; R_{P.Q}^\smile ; R_{P.Q} ; (P \oplus Q) \\
&= \iota ; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q \\ i_Q^\smile ; t_P & \mathbb{I}_{S_2} \end{pmatrix} ; \begin{pmatrix} P ; F(\iota) \\ Q ; F(\kappa) \end{pmatrix} \\
&= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q \end{pmatrix} ; \begin{pmatrix} P ; F(\iota) \\ Q ; F(\kappa) \end{pmatrix} \\
&= \big(\mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P\big) P ; F(\iota) \sqcup t_P^\smile ; i_Q ; Q ; F(\kappa) \\
&= P ; F(\iota) \sqcup t_P^\smile ; i_Q ; Q ; F(\kappa) && t_P ; P = \mathbb{L}_{IS_1}
\end{aligned}
$$

from which $\iota ; R_{P.Q}^\smile ; (P.Q) = (P ; F(\iota) \sqcup t_P^\smile ; i_Q ; Q ; F(\kappa)) ; F(R_{P.Q}^\smile)$ follows. Now, suppose $v : I \to S_1$ is an arbitrary relation and compute

$$
\begin{aligned}
& v \sqsubseteq t_{P.Q} ; R_{P.Q} ; \iota^\smile \\
\iff\ & v ; \iota \sqsubseteq t_{P.Q} ; R_{P.Q} && \text{Lemma 1} \\
\iff\ & v ; \iota ; R_{P.Q}^\smile \sqsubseteq t_{P.Q} && \text{Lemma 1} \\
\iff\ & v ; \iota ; R_{P.Q}^\smile ; (P.Q) \sqsubseteq \mathbb{L}_{IF(S_{P.Q})} \\
\iff\ & v ; \big(P ; F(\iota) \sqcup t_P^\smile ; i_Q ; Q ; F(\kappa)\big) ; F\big(R_{P.Q}^\smile\big) \sqsubseteq \mathbb{L}_{IF(S_{P.Q})} && \text{see above} \\
\iff\ & v ; \big(P ; F(\iota) \sqcup t_P^\smile ; i_Q ; Q ; F(\kappa)\big) \sqsubseteq \mathbb{L}_{IF(S_1+S_2)} && \text{Lemma 1} \\
\iff\ & v ; P ; F(\iota) \sqsubseteq \mathbb{L}_{IF(S_1+S_2)} \text{ and } v ; t_P^\smile ; i_Q ; Q ; F(\kappa) \sqsubseteq \mathbb{L}_{IF(S_1+S_2)} \\
\iff\ & v ; P \sqsubseteq \mathbb{L}_{IF(S_1)} \quad \text{and} \quad v ; t_P^\smile ; i_Q ; Q \sqsubseteq \mathbb{L}_{IF(S_2)} && \text{Lemma 1} \\
\iff\ & v \sqsubseteq t_P \text{ and } v ; t_P^\smile ; i_Q \sqsubseteq t_Q \\
\iff\ & v \sqsubseteq t_P \text{ and } v ; t_P^\smile \sqsubseteq t_Q ; i_Q^\smile . && \text{Lemma 1}
\end{aligned}
$$

The last two properties imply $v \sqsubseteq v ; v^\smile ; v \sqsubseteq v ; t_P^\smile ; t_P \sqsubseteq t_Q ; i_Q^\smile ; t_P$ so that we conclude $t_{P.Q} ; R_{P.Q} ; \iota^\smile \sqsubseteq t_Q ; i_Q^\smile ; t_P$. In addition, we obtain

$$
\begin{aligned}
& t_{P.Q} ; (P.Q) \sqsubseteq \mathbb{L}_{IF(S_{P.Q})} \\
\iff\ & t_{P.Q} ; R_{P.Q} ; (P \oplus Q) ; F\big(R_{P.Q}^\smile\big) \sqsubseteq \mathbb{L}_{IF(S_{P.Q})} \\
\iff\ & t_{P.Q} ; R_{P.Q} ; (P \oplus Q) \sqsubseteq \mathbb{L}_{IF(S_1+S_2)} && \text{Lemma 1} \\
\implies\ & t_{P.Q} ; R_{P.Q} ; \kappa^\smile ; Q ; F\big(\kappa^\smile\big) \sqsubseteq \mathbb{L}_{IF(S_1+S_2)} && \kappa^\smile Q ; F\big(\kappa^\smile\big) \sqsubseteq P \oplus Q \\
\iff\ & t_{P.Q} ; R_{P.Q} ; \kappa^\smile ; Q \sqsubseteq \mathbb{L}_{IF(S_2)} && \text{Lemma 1} \\
\iff\ & t_{P.Q} ; R_{P.Q} ; \kappa^\smile \sqsubseteq t_Q .
\end{aligned}
$$

The last two properties together imply

$$
\begin{aligned}
t_{P.Q} &= t_{P.Q} ; R_{P.Q} ; R_{P.Q}^\smile \\
&= t_{P.Q} ; R_{P.Q} ; \big(\iota^\smile ; \iota \sqcup \kappa^\smile ; \kappa\big) ; R_{P.Q}^\smile \\
&= (t_{P.Q} ; R_{P.Q} ; \iota^\smile ; \iota \sqcup t_{P.Q} ; R_{P.Q} ; \kappa^\smile ; \kappa) ; R_{P.Q}^\smile \\
&\sqsubseteq \big(t_Q ; i_Q^\smile ; t_P ; \iota \sqcup t_Q ; \kappa\big) ; R_{P.Q}^\smile && \text{see above}
\end{aligned}
$$

$$
\begin{aligned}
&= t_Q ; \kappa ; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^{\smile} ; t_P & t_P^{\smile} ; i_Q \\ i_Q^{\smile} ; t_P & \mathbb{I}_{S_2} \end{pmatrix} ; R_{P.Q}^{\smile} \\
&= t_Q ; \kappa ; R_{P.Q}^{\smile} ; R_{P.Q} ; R_{P.Q}^{\smile} \\
&= t_Q ; \kappa ; R_{P.Q}^{\smile} .
\end{aligned}
$$

This completes the proof. □

The following lemma gives sufficient properties when the sequential composition of two coalgebras result in just the first coalgebra.

Lemma 5. *Let $(P, i_P) : S_1 \to F(S_1)$ and $(Q, i_Q) : S_2 \to F(S_2)$ be rooted F coalgebras. Then we have*

1. *If $(Q, i_Q) \sim (\perp\!\!\!\perp_{IF(I)}, \mathbb{I}_I)$ or $(P, i_P) \sim (P', i_{P'})$ with $t_{P'} = \perp\!\!\!\perp_{IS_1'}$, then $(P, i_P) \sim (P.Q, i_{P.Q})$.*
2. *If $(P, i_P) \sim (\perp\!\!\!\perp_{IF(I)}, \mathbb{I}_I)$, then $(Q, i_Q) \sim (P.Q, i_{P.Q})$.*

The proof can be found in [19].

The main theorem of this section shows that the sequential composition of coalgebras is associative (up to bisimilarity).

Theorem 6. *Let $(P, i_P) : S_1 \to F(S_1), (Q, i_Q) : S_2 \to F(S_2)$ and $(U, i_U) : S_3 \to F(S_3)$ be rooted F coalgebras. Then $(P.(Q.U), i_{P.(Q.U)}) \sim ((P.Q).U, i_{(P.Q).U})$.*

Proof. We want to show that

$$\Phi = R_{P.(Q.U)} ; (\mathbb{I}_{S_1} + R_{Q.U}) ; \text{assoc} ; \left(R_{P.Q}^{\smile} + \mathbb{I}_{S_3}\right) ; R_{(P.Q).U}^{\smile}$$

is the required bisimulation from $P.(Q.U)$ to $(P.Q).U$. First we compute

$$
\begin{aligned}
P.(Q.U) &= R_{P.(Q.U)} ; \left(P \oplus (Q.U)\right) ; F\left(R_{P.(Q.U)}^{\smile}\right) \\
&= R_{P.(Q.U)} ; \left(P \oplus \left(R_{Q.U} ; (Q \oplus U) ; F\left(R_{Q.U}^{\smile}\right)\right)\right) ; F\left(R_{P.(Q.U)}^{\smile}\right) \\
&= R_{P.(Q.U)} ; (\mathbb{I}_{S_1} + R_{Q.U}) ; \left(P \oplus (Q \oplus U)\right) ; F\left(\left(\mathbb{I}_{S_1} + R_{Q.U}^{\smile}\right) ; R_{P.(Q.U)}^{\smile}\right).
\end{aligned}
$$

Analogously we obtain

$$(P.Q).U = R_{(P.Q).U} ; (R_{P.Q} + \mathbb{I}_{S_3}) ; \left((P \oplus Q) \oplus U\right) ; F\left(\left(R_{P.Q}^{\smile} + \mathbb{I}_{S_3}\right) ; R_{(P.Q).U}^{\smile}\right).$$

Now we want to analyze the structure of the equivalence relation Ξ_1 on $S_1 + (S_2 + S_3)$ induced by the two equivalence relations $\Xi_{Q.U}$ and $\Xi_{P.(Q.U)}$.

$$
\begin{aligned}
&\left(\mathbb{I}_{S_1} + R_{Q.U}^{\smile}\right) ; R_{P.(Q.U)}^{\smile} ; R_{P.(Q.U)} ; (\mathbb{I}_{S_1} + R_{Q.U}) \\
&= \begin{pmatrix} \mathbb{I}_{S_1} & \perp\!\!\!\perp_{S_1 S_{Q.U}} \\ \perp\!\!\!\perp_{S_2+S_3 S_1} & R_{Q.U}^{\smile} \end{pmatrix} ; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^{\smile} ; t_P & t_P^{\smile} ; i_{Q.U} \\ i_{Q.U}^{\smile} ; t_P & \mathbb{I}_{S_{Q.U}} \end{pmatrix} ; \begin{pmatrix} \mathbb{I}_{S_1} & \perp\!\!\!\perp_{S_1 S_2+S_3} \\ \perp\!\!\!\perp_{S_{Q.U} S_1} & R_{Q.U} \end{pmatrix} \\
&= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^{\smile} ; t_P & t_P^{\smile} ; i_{Q.U} \\ R_{Q.U}^{\smile} ; i_{Q.U}^{\smile} ; t_P & R_{Q.U}^{\smile} \end{pmatrix} ; \begin{pmatrix} \mathbb{I}_{S_1} & \perp\!\!\!\perp_{S_1 S_2+S_3} \\ \perp\!\!\!\perp_{S_{Q.U} S_1} & R_{Q.U} \end{pmatrix} \\
&= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^{\smile} ; t_P & t_P^{\smile} ; i_{Q.U} ; R_{Q.U} \\ R_{Q.U}^{\smile} ; i_{Q.U}^{\smile} ; t_P & R_{Q.U}^{\smile} ; R_{Q.U} \end{pmatrix} \\
&= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^{\smile} ; t_P & t_P^{\smile} ; i_Q ; \iota ; R_{Q.U}^{\smile} ; R_{Q.U} \\ R_{Q.U}^{\smile} ; R_{Q.U} ; \iota^{\smile} ; i_Q^{\smile} ; t_P & R_{Q.U}^{\smile} ; R_{Q.U} \end{pmatrix} \\
&= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^{\smile} ; t_P & t_P^{\smile} ; i_Q ; \iota ; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_Q^{\smile} ; t_Q & t_Q^{\smile} ; i_U \\ i_U^{\smile} ; t_Q & \mathbb{I}_{S_U} \end{pmatrix} \\ \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_Q^{\smile} ; t_Q & t_Q^{\smile} ; i_U \\ i_U^{\smile} ; t_Q & \mathbb{I}_{S_U} \end{pmatrix} ; \iota^{\smile} ; i_Q^{\smile} ; t_P & \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_Q^{\smile} ; t_Q & t_Q^{\smile} ; i_U \\ i_U^{\smile} ; t_Q & \mathbb{I}_{S_U} \end{pmatrix} \end{pmatrix}
\end{aligned}
$$

$$= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q ; (\mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q \quad t_Q^\smile ; i_U) \\ \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q \\ i_U^\smile ; t_Q \end{pmatrix} ; i_Q^\smile ; t_P & \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q & t_Q^\smile ; i_U \\ i_U^\smile ; t_Q & \mathbb{I}_{S_U} \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & (t_P^\smile ; i_Q ; (\mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q) \quad t_P^\smile ; i_Q ; t_Q^\smile ; i_U) \\ ((\mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q) ; i_Q^\smile ; t_P & \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q & t_Q^\smile ; i_U \\ i_U^\smile ; t_Q & \mathbb{I}_{S_U} \end{pmatrix} \\ i_U^\smile ; t_Q ; i_Q^\smile ; t_P & \end{pmatrix}.$$

Next we focus on the equivalence relation Ξ_2 on $(S_1 + S_2) + S_3$ induced by $\Xi_{P.Q}$ and $\Xi_{(P.Q).U}$. We obtain

$$\begin{aligned}
&\left(R_{P.Q}^\smile + \mathbb{I}_{S_3}\right) ; R_{(P.Q).U}^\smile ; R_{(P.Q).U} ; (R_{P.Q} + \mathbb{I}_{S_3}) \\
&= \begin{pmatrix} R_{P.Q}^\smile & \perp\!\!\!\perp_{S_1+S_2 S_3} \\ \perp\!\!\!\perp_{S_3 S_{P.Q}} & \mathbb{I}_{S_3} \end{pmatrix} ; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q} & t_{P.Q}^\smile ; i_U \\ i_U^\smile ; t_{P.Q} & \mathbb{I}_{S_3} \end{pmatrix} ; \begin{pmatrix} R_{P.Q} & \perp\!\!\!\perp_{S_{P.Q} S_3} \\ \perp\!\!\!\perp_{S_3 S_1+S_2} & \mathbb{I}_{S_3} \end{pmatrix} \\
&= \begin{pmatrix} R_{P.Q}^\smile ; (\mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q}) & R_{P.Q}^\smile ; t_{P.Q}^\smile ; i_U \\ i_U^\smile ; t_{P.Q} & \mathbb{I}_{S_3} \end{pmatrix} ; \begin{pmatrix} R_{P.Q} & \perp\!\!\!\perp_{S_{P.Q} S_3} \\ \perp\!\!\!\perp_{S_3 S_1+S_2} & \mathbb{I}_{S_3} \end{pmatrix} \\
&= \begin{pmatrix} R_{P.Q}^\smile ; (\mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q}) ; R_{P.Q} & R_{P.Q}^\smile ; t_{P.Q}^\smile ; i_U \\ i_U^\smile ; t_{P.Q} ; R_{P.Q} & \mathbb{I}_{S_3} \end{pmatrix} \\
&= \begin{pmatrix} R_{P.Q}^\smile ; (\mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q}) ; R_{P.Q} & R_{P.Q}^\smile ; R_{P.Q} ; \kappa^\smile ; t_Q^\smile ; i_U \\ i_U^\smile ; t_Q ; \kappa ; R_{P.Q}^\smile ; R_{P.Q} & \mathbb{I}_{S_3} \end{pmatrix} \qquad \text{Lemma 4} \\
&= \begin{pmatrix} R_{P.Q}^\smile ; (\mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q}) ; R_{P.Q} & \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q \\ i_Q^\smile ; t_P & \mathbb{I}_{S_2} \end{pmatrix} ; \kappa^\smile ; t_Q^\smile ; i_U \\ i_U^\smile ; t_Q ; \kappa ; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q \\ i_Q^\smile ; t_P & \mathbb{I}_{S_2} \end{pmatrix} & \mathbb{I}_{S_3} \end{pmatrix} \\
&= \begin{pmatrix} R_{P.Q}^\smile ; (\mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q}) ; R_{P.Q} & \begin{pmatrix} t_P^\smile ; i_Q \\ \mathbb{I}_{S_2} \end{pmatrix} ; t_Q^\smile ; i_U \\ i_U^\smile ; t_Q ; \begin{pmatrix} i_Q^\smile ; t_P & \mathbb{I}_{S_2} \end{pmatrix} & \mathbb{I}_{S_3} \end{pmatrix} \\
&= \begin{pmatrix} R_{P.Q}^\smile ; (\mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q}) ; R_{P.Q} & \begin{pmatrix} t_P^\smile ; i_Q ; t_Q^\smile ; i_U \\ t_Q^\smile ; i_U \end{pmatrix} \\ \begin{pmatrix} i_U^\smile ; t_Q ; i_Q^\smile ; t_P & i_U^\smile ; t_Q \end{pmatrix} & \mathbb{I}_{S_3} \end{pmatrix}.
\end{aligned}$$

In order to verify that the matrix representation of Ξ_2 has the same entries (but a different hierarchy of matrices) as the matrix representation of Ξ_1 we have to show that

$$R_{P.Q}^\smile ; \left(\mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q}\right) ; R_{P.Q} = \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q ; (\mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q) \\ (\mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q) ; i_Q^\smile ; t_P & \mathbb{I}_{S_1} \sqcup t_Q^\smile ; t_Q \end{pmatrix}.$$

Therefore, we investigate the term on the left-hand side as follows

$$\begin{aligned}
&R_{P.Q}^\smile ; \left(\mathbb{I}_{S_1} \sqcup t_{P.Q}^\smile ; t_{P.Q}\right) ; R_{P.Q} \\
&= R_{P.Q}^\smile ; \left(\mathbb{I}_{S_1} \sqcup R_{P.Q} ; \kappa^\smile ; t_Q^\smile ; t_Q ; \kappa ; R_{P.Q}^\smile\right) ; R_{P.Q} \qquad \text{Lemma 4} \\
&= R_{P.Q}^\smile ; \left(R_{P.Q} ; R_{P.Q}^\smile \sqcup R_{P.Q} ; \kappa^\smile ; t_Q^\smile ; t_Q ; \kappa ; R_{P.Q}^\smile\right) ; R_{P.Q} \\
&= R_{P.Q}^\smile ; R_{P.Q} ; \left(\mathbb{I}_{S_1+S_2} \sqcup \kappa^\smile ; t_Q^\smile ; t_Q ; \kappa\right) ; R_{P.Q}^\smile ; R_{P.Q} \\
&= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q \\ i_Q^\smile ; t_P & \mathbb{I}_{S_2} \end{pmatrix} ; \begin{pmatrix} \mathbb{I}_{S_1} & \perp\!\!\!\perp_{S_1 S_2} \\ \perp\!\!\!\perp_{S_2 S_1} & \mathbb{I}_{S_2} \sqcup t_Q^\smile ; t_Q \end{pmatrix} ; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q \\ i_Q^\smile ; t_P & \mathbb{I}_{S_2} \end{pmatrix} \\
&= \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q ; (\mathbb{I}_{S_2} \sqcup t_Q^\smile ; t_Q) \\ i_Q^\smile ; t_P & \mathbb{I}_{S_2} \sqcup t_Q^\smile ; t_Q \end{pmatrix} ; \begin{pmatrix} \mathbb{I}_{S_1} \sqcup t_P^\smile ; t_P & t_P^\smile ; i_Q \\ i_Q^\smile ; t_P & \mathbb{I}_{S_2} \end{pmatrix}.
\end{aligned}$$

Starting with the term in the left-upper corner of this composition we get

$$
\begin{aligned}
&\big(\mathbb{I}_{S_1} \sqcup t_P^\smile; t_P\big); \big(\mathbb{I}_{S_1} \sqcup t_P^\smile; t_P\big) \sqcup t_P^\smile; i_Q; \big(\mathbb{I}_{S_2} \sqcup t_Q^\smile; t_Q\big); i_Q^\smile; t_P \\
&\quad = \mathbb{I}_{S_1} \sqcup t_P^\smile; t_P \sqcup t_P^\smile; i_Q; \big(\mathbb{I}_{S_2} \sqcup t_Q^\smile; t_Q\big); i_Q^\smile; t_P \\
&\quad = \mathbb{I}_{S_1} \sqcup t_P^\smile; t_P,
\end{aligned}
$$

where the last equation follows because $i_Q; (\mathbb{I}_{S_2} \sqcup t_Q^\smile; t_Q); i_Q^\smile \sqsubseteq \mathbb{T}_{II} = \mathbb{I}_I$. The term in the right-upper corner computes to

$$
\begin{aligned}
&\big(\mathbb{I}_{S_1} \sqcup t_P^\smile; t_P\big); t_P^\smile; i_Q \sqcup t_P^\smile; i_Q; \big(\mathbb{I}_{S_2} \sqcup t_Q^\smile; t_Q\big) \\
&\quad = t_P^\smile; i_Q \sqcup t_P^\smile; i_Q; \big(\mathbb{I}_{S_2} \sqcup t_Q^\smile; t_Q\big) \qquad \text{since } t_P^\smile; t_P; t_P^\smile; i_Q \sqsubseteq t_P^\smile; \mathbb{T}_{II}; i_Q = t_P^\smile; i_Q \\
&\quad = t_P^\smile; i_Q; \big(\mathbb{I}_{S_2} \sqcup t_Q^\smile; t_Q\big).
\end{aligned}
$$

Analogously, we obtain $(\mathbb{I}_{S_1} \sqcup t_Q^\smile; t_Q); i_Q^\smile; t_P$ in the left-lower corner. Finally, we obtain in the right-lower corner

$$
\begin{aligned}
&i_Q{}^\smile; t_P; t_P^\smile; i_Q \sqcup \mathbb{I}_{S_1} \sqcup t_Q^\smile; t_Q \\
&\quad = \mathbb{I}_{S_1} \sqcup t_Q^\smile; t_Q \qquad \text{since } i_Q^\smile; t_P; t_P^\smile; i_Q \sqsubseteq i_Q^\smile; \mathbb{T}_{II}; i_Q = i_Q^\smile; i_Q \sqsubseteq \mathbb{I}_{S_2}.
\end{aligned}
$$

From the fact that all entries in Ξ_1 and Ξ_2 are identical and the definition of the isomorphism assoc we conclude that $\text{assoc}^\smile; \Xi_1; \text{assoc} = \Xi_2$ or, equivalently, $\Xi_1; \text{assoc} = \text{assoc}; \Xi_2$. Using this equation and the additional properties of assoc shown earlier we obtain

$$
\begin{aligned}
&\Phi; \big((P.Q).U\big) \\
&\quad = R_{P.(Q.U)}; (\mathbb{I}_{S_1} + R_{Q.U}); \text{assoc}; \Xi_2; \big((P \oplus Q) \oplus U\big); F\big(\big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile\big) \\
&\quad = R_{P.(Q.U)}; (\mathbb{I}_{S_1} + R_{Q.U}); \Xi_1; \text{assoc}; \big((P \oplus Q) \oplus U\big); F\big(\big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile\big) \\
&\quad = R_{P.(Q.U)}; (\mathbb{I}_{S_1} + R_{Q.U}); \Xi_1; \big(P \oplus (Q \oplus U)\big); F\big(\text{assoc}; \big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile\big) \\
&\quad = R_{P.(Q.U)}; (\mathbb{I}_{S_1} + R_{Q.U}); \big(P \oplus (Q \oplus U)\big); F\big(\text{assoc}; \Xi_2; \big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile\big) \\
&\quad = R_{P.(Q.U)}; (\mathbb{I}_{S_1} + R_{Q.U}); \big(P \oplus (Q \oplus U)\big); F\big(\Xi_1; \text{assoc}; \big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile\big) \\
&\quad = \big(P.(Q.U)\big); F(\Phi).
\end{aligned}
$$

The inclusion $\Phi^\smile; (P.(Q.U)) \sqsubseteq ((P.Q).U); F(\Phi^\smile)$ follows analogously. Finally, we obtain

$$
\begin{aligned}
&i_{P.(Q.U)}; \Phi \\
&\quad = i_P; \iota; R_{P.(Q.U)}^\smile; R_{P.(Q.U)}; (\mathbb{I}_{S_1} + R_{Q.U}); \text{assoc}; \big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile \\
&\quad = i_P; \iota; \Xi_1; \text{assoc}; \big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile \qquad \text{since } \iota = \iota; \big(\mathbb{I}_{S_1} + R_{Q.U}^\smile\big) \\
&\quad = i_P; \iota; \text{assoc}; \Xi_2; \big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile \\
&\quad = i_P; \iota; \text{assoc}; \big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile \\
&\quad = i_P; \iota; \iota; \big(R_{P.Q}^\smile + \mathbb{I}_{S_3}\big); R_{(P.Q).U}^\smile \\
&\quad = i_P; \iota; R_{P.Q}^\smile; \iota; R_{(P.Q).U}^\smile \\
&\quad = i_{P.Q}; \iota; R_{(P.Q).U}^\smile \\
&\quad = i_{(P.Q).U}.
\end{aligned}
$$

This completes the proof. □

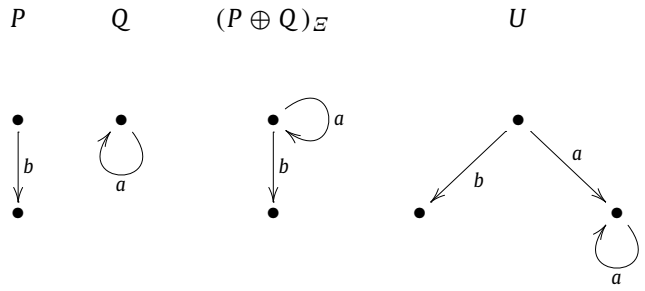
4. Sum of rooted coalgebras

The operation $\oplus$ defines a coalgebra on the sum of the state spaces. In terms of labeled transition systems this coalgebra corresponds to two alternative branches of execution. However, $P \oplus Q$ does not provide sufficient means to define an initial state. This can be done by adding a new state essentially combining the initial state of P and Q.

Definition 10. Let $(P, i_P) : S_1 \to F(S_1)$ and $(Q, i_Q) : S_2 \to F(S_2)$ be rooted F coalgebras. Then we define a rooted coalgebra $(P \boxplus Q, i_{P \boxplus Q}) : I + (S_1 + S_2) \to F(I + (S_1 + S2))$ by

$$P \boxplus Q = \big[(i_P ; \iota \sqcup i_Q ; \kappa), \mathbb{I}_{S_1+S_2}\big] ; (P \oplus Q) ; F(\kappa),$$
$$i_{P \boxplus Q} := \iota.$$

Notice that simply identifying the initial state of P and Q in $P \oplus Q$ would not work because any initial state can also be state reachable state later, i.e., there might be a loop in the coalgebra leading back to the initial state. For example, if we denote by $(P \oplus Q)_{\Xi}$ the coalgebra that is obtained from $P \oplus Q$ by identifying the initial states we get



The sequence ab is possible in the labeled transition system $(P \oplus Q)_{\Xi}$ but neither possible in P nor Q. Notice that $P \boxplus Q$ is bisimilar to U.

Lemma 7. *Let $(P_1, i_{P_1}) : S_1 \to F(S_1)$, $(P_2, i_{P_2}) : S_2 \to F(S_2)$, $(Q_1, i_{Q_1}) : T_1 \to F(T_1)$ and $(Q_2, i_{Q_2}) : T_2 \to F(T_2)$ be rooted F coalgebras with $(P_1, i_{P_1}) \sim (P_2, i_{P_2})$ and $(Q_1, i_{Q_1}) \sim (Q_2, i_{Q_2})$. Then we have*

$$(P_1 \boxplus Q_1, i_{P_1 \boxplus Q_1}) \sim (P_2 \boxplus Q_2, i_{P_2 \boxplus Q_2}).$$

Proof sketch. Suppose $\Phi : S_1 \to S_2$ and $\Psi : T_1 \to T_2$ are bisimulations with $i_{P_1}^{\smile} ; i_{P_2} \sqsubseteq \Phi$ and $i_{Q_1}^{\smile} ; i_{Q_2} \sqsubseteq \Psi$. It is straightforward to verify that $\mathbb{I}_I + (\Phi + \Psi)$ is the required bisimulation. The details can be found in [19]. □

In the next lemma we want to investigate the terminal states of $P \boxplus Q$. Obviously, the terminal states of P as well as the terminal states of Q are such states. In addition, the new initial state of $P \boxplus Q$ is also a terminal state if the initial states of P and Q are also terminal.

Lemma 8. *Let $(P, i_P) : S_1 \to F(S_1)$ and $(Q, i_Q) : S_2 \to F(S_2)$ be rooted F coalgebras. Then we have $t_{P \boxplus Q} = (t_P ; i_P^{\smile} \sqcap t_Q ; i_Q^{\smile}) ; \iota \sqcup (t_P ; \iota \sqcup t_Q ; \kappa) ; \kappa$.*

Proof. First of all, from the computation

$$\begin{aligned}
&\big(\big(t_P ; i_P^{\smile} \sqcap t_Q ; i_Q^{\smile}\big) ; \iota \sqcup (t_P ; \iota \sqcup t_Q ; \kappa) ; \kappa\big) ; (P \boxplus Q) \\
&\quad = \big(\big(t_P ; i_P^{\smile} \sqcap t_Q ; i_Q^{\smile}\big) ; (i_P ; \iota \sqcup i_Q ; \kappa) ; (P \oplus Q) \sqcup (t_P ; \iota \sqcup t_Q ; \kappa) ; (P \oplus Q)\big) ; F(\kappa) \\
&\quad \sqsubseteq \big(\big(t_P ; i_P^{\smile} ; i_P ; P ; F(\iota) \sqcup t_Q ; i_Q^{\smile} ; i_Q ; Q ; F(\kappa)\big) \sqcup \big(t_P ; P ; F(\iota) \sqcup t_Q ; Q ; F(\kappa)\big)\big) ; F(\kappa) \\
&\quad \sqsubseteq \big(\big(t_P ; P ; F(\iota) \sqcup t_Q ; Q ; F(\kappa)\big) \sqcup \big(t_P ; P ; F(\iota) \sqcup t_Q ; Q ; F(\kappa)\big)\big) ; F(\kappa) \\
&\quad = \mathbb{\perp}_{IF(I+(S_1+S_2))} \qquad t_P ; P = \mathbb{\perp}_{IF(S_1)} \text{ and } t_Q ; Q = \mathbb{\perp}_{IF(S_2)}
\end{aligned}$$

we conclude $(t_P ; i_P^{\smile} \sqcap t_Q ; i_Q^{\smile}) ; \iota \sqcup (t_P ; \iota \sqcup t_Q ; \kappa) ; \kappa \sqsubseteq t_{P \boxplus Q}$. Now consider

$$\begin{aligned}
& v \sqsubseteq t_{P \boxplus Q} \\
\iff\quad & v ; (P \boxplus Q) \sqsubseteq \mathbb{\perp}_{IF(I+(S_1+S_2))} \\
\iff\quad & v ; \big[(i_P ; \iota \sqcup i_Q ; \kappa), \mathbb{I}_{S_1+S_2}\big] ; (P \oplus Q) ; F(\kappa) \sqsubseteq \mathbb{\perp}_{IF(I+(S_1+S_2))} \\
\iff\quad & v ; \big[(i_P ; \iota \sqcup i_Q ; \kappa), \mathbb{I}_{S_1+S_2}\big] ; (P \oplus Q) \sqsubseteq \mathbb{\perp}_{IF(S_1+S_2)} \qquad \text{Lemma 1} \\
\iff\quad & v ; \iota^{\smile} ; (i_P ; \iota \sqcup i_Q ; \kappa) ; (P \oplus Q) \sqsubseteq \mathbb{\perp}_{IF(S_1+S_2)} \\
& \qquad \text{and } v ; \kappa^{\smile} ; (P \oplus Q) \sqsubseteq \mathbb{\perp}_{IF(S_1+S_2)}
\end{aligned}$$

$$\begin{aligned}
&\iff && v;\iota^{\smile};\big(i_P;P;F(\iota)\sqcup i_Q;Q;F(\kappa)\big)\sqsubseteq \perp\!\!\!\perp_{IF(S_1+S_2)}\\
&&&\text{and } v;\kappa^{\smile};\big(\iota^{\smile};P;F(\iota)\sqcup \kappa^{\smile};Q;F(\kappa)\big)\sqsubseteq \perp\!\!\!\perp_{IF(S_1+S_2)}\\
&\iff && v;\iota^{\smile};i_P;P;F(\iota)\sqsubseteq \perp\!\!\!\perp_{IF(S_1+S_2)}\\
&&&\text{and } v;\iota^{\smile};i_Q;Q;F(\kappa)\sqsubseteq \perp\!\!\!\perp_{IF(S_1+S_2)}\\
&&&\text{and } v;\kappa^{\smile};\iota^{\smile};P;F(\iota)\sqsubseteq \perp\!\!\!\perp_{IF(S_1+S_2)}\\
&&&\text{and } v;\kappa^{\smile};\kappa^{\smile};Q;F(\kappa)\sqsubseteq \perp\!\!\!\perp_{IF(S_1+S_2)}.
\end{aligned}$$

If we use the first inclusion, we obtain

$$\begin{aligned}
v;\iota^{\smile};i_P;P;F(\iota)\sqsubseteq \perp\!\!\!\perp_{IF(S_1+S_2)} &\iff v;\iota^{\smile};i_P;P\sqsubseteq \perp\!\!\!\perp_{IF(S_1)} && \text{Lemma 1}\\
&\iff v;\iota^{\smile};i_P\sqsubseteq t_P\\
&\iff v;\iota^{\smile}\sqsubseteq t_P;i_P^{\smile}. && \text{Lemma 1}
\end{aligned}$$

2. Analogously, the second inclusion is equivalent to $v;\iota^{\smile}\sqsubseteq t_Q;i_Q^{\smile}$ so that $v;\iota^{\smile}\sqsubseteq t_P;i_P^{\smile}\sqcap t_Q;i_Q^{\smile}$ follows. Now we consider the third inclusion, and we get

$$\begin{aligned}
v;\kappa^{\smile};\iota^{\smile};P;F(\iota)\sqsubseteq \perp\!\!\!\perp_{IF(S_1+S_2)} &\iff v;\kappa^{\smile};\iota^{\smile};P\sqsubseteq \perp\!\!\!\perp_{IF(S_1)} && \text{Lemma 1}\\
&\iff v;\kappa^{\smile};\iota^{\smile}\sqsubseteq t_P.
\end{aligned}$$

The fourth inclusion is equivalent to $v;\kappa^{\smile};\kappa^{\smile}\sqsubseteq t_Q$ by a similar computation. The two properties together imply $v;\kappa^{\smile}=v;\kappa^{\smile};(\iota^{\smile};\iota\sqcup\kappa^{\smile};\kappa)\sqsubseteq t_P;\iota\sqcup t_Q;\kappa$. Together with the previous result we get

$$v=v;(\iota^{\smile};\iota\sqcup\kappa^{\smile};\kappa)\sqsubseteq(t_P;i_P^{\smile}\sqcap t_Q;i_Q^{\smile});\iota\sqcup(t_P;\iota\sqcup t_Q;\kappa);\kappa.$$

This finally implies $t_{P\boxplus Q}\sqsubseteq(t_P;i_P^{\smile}\sqcap t_Q;i_Q^{\smile});\iota\sqcup(t_P;\iota\sqcup t_Q;\kappa);\kappa$. □

The next lemma is of particular interest. It shows that the sequential composition with a coalgebra U distributes from the right over $\boxplus$.

Theorem 9. *Let $(P,i_P):S_1\to F(S_1),(Q,i_Q):S_2\to F(S_2)$ and $(U,i_U):S_3\to F(S_3)$ be rooted F coalgebras. Then we have*

$$\big((P.U)\boxplus(Q.U),i_{(P.U)\boxplus(Q.U)}\big)\sim\big((P\boxplus Q).U,i_{(P\boxplus Q).U}\big).$$

Proof sketch. First we define $\alpha:I+((S_1+S_3)+(S_2+S_3))\to(I+(S_1+S_2))+S_3$ by

$$\begin{aligned}
\alpha&=\big[\iota;\iota,\big[[\iota;\kappa;\iota,\kappa],[\kappa;\kappa;\iota,\kappa]\big]\big]\\
&=\iota^{\smile};\iota;\iota\sqcup\kappa^{\smile};\iota^{\smile};\iota^{\smile};\iota;\kappa;\iota\sqcup\kappa^{\smile};\iota^{\smile};\kappa^{\smile};\kappa\sqcup\kappa^{\smile};\kappa^{\smile};\iota^{\smile};\kappa;\kappa;\iota\sqcup\kappa^{\smile};\kappa^{\smile};\kappa^{\smile};\kappa.
\end{aligned}$$

In matrix form we have

$$\alpha=\begin{pmatrix}
\begin{pmatrix}\mathbb{I}_I\\ \begin{pmatrix}\perp\!\!\!\perp_{S_1I}\\ \perp\!\!\!\perp_{S_3I}\end{pmatrix}\\ \begin{pmatrix}\perp\!\!\!\perp_{S_2I}\\ \perp\!\!\!\perp_{S_3I}\end{pmatrix}\end{pmatrix} &
\begin{pmatrix}\begin{pmatrix}\perp\!\!\!\perp_{IS_1} & \perp\!\!\!\perp_{IS_2}\end{pmatrix}\\ \begin{pmatrix}\mathbb{I}_{S_1} & \perp\!\!\!\perp_{S_1S_2}\\ \perp\!\!\!\perp_{S_3S_1} & \perp\!\!\!\perp_{S_3S_2}\end{pmatrix} & \begin{pmatrix}\perp\!\!\!\perp_{S_1S_3}\\ \mathbb{I}_{S_3}\end{pmatrix}\end{pmatrix}\\
\begin{pmatrix}\begin{pmatrix}\perp\!\!\!\perp_{S_1S_3}\\ \mathbb{I}_{S_3}\end{pmatrix}\\ \begin{pmatrix}\perp\!\!\!\perp_{S_2S_1} & \mathbb{I}_{S_2}\\ \perp\!\!\!\perp_{S_3S_1} & \perp\!\!\!\perp_{S_3S_2}\end{pmatrix} & \begin{pmatrix}\perp\!\!\!\perp_{S_2S_3}\\ \mathbb{I}_{S_3}\end{pmatrix}\end{pmatrix} & \perp\!\!\!\perp_{S_3S_3}
\end{pmatrix}.$$

The theorem follows by showing $\Phi=(\mathbb{I}_I+(R_{P.U}+R_{Q.U}));\alpha;R^{\smile}_{(P\boxplus Q).U}$ is the required bisimulation from $(P.U)\boxplus(Q.U)$ to $(P\boxplus Q).U$. The details can be found in [19]. □

5. Recursion on coalgebras

In this section we want to investigate an operation closely related to iteration or recursion. The coalgebra $\circlearrowleft P$ identifies the terminal states P with its initial state, i.e. introduces a loop in the graph of P. Therefore, $\circlearrowleft$ is a kind of a Kleene star operator.

Definition 11. Let $(P, i_P) : S \to F(S)$. Furthermore, let $R_{\circlearrowleft P} : S_{\circlearrowleft P} \to S$ be the splitting of the relation $\Xi_{\circlearrowleft P} = \mathbb{I}_{S_1} \sqcup (t_P \sqcup i_P)^\smile ; (t_P \sqcup i_P)$. Then we define a rooted coalgebra $(\circlearrowleft P, i_{\circlearrowleft P}) : S_{\circlearrowleft P} \to F(S_{\circlearrowleft P})$ by

$$\circlearrowleft P := R_{\circlearrowleft P} ; P ; F(R_{\circlearrowleft P}^\smile),$$
$$i_{\circlearrowleft P} := i_P ; R_{\circlearrowleft P}^\smile .$$

Similar to the sequential composition we can define recursion for certain terminal states of P only, i.e., with respect to a vector $r \sqsubseteq t_P$. This recursion operation goes back to the initial state only if a terminal state in r is reached. We denote this operation by $\overset{r}{\circlearrowleft} P$. Formally, the definition only requires to replace t_p in the definition of Ξ by r. As before in all proofs we only use that terminal states are terminal, i.e., $t_P ; P = \mathbb{\bot}_{SF(S)}$, so that the corresponding properties (or their obvious generalization) remain true.

First, we want to show that the loop operation respects the notion of bisimilarity.

Lemma 10. *Let $(P_1, i_{P_1}) : S_1 \to F(S_1)$ and $(P_2, i_{P_2}) : S_2 \to F(S_2)$ be rooted coalgebras with $(P_1, i_{P_1}) \sim (P_2, i_{P_2})$. Then we have $(\circlearrowleft P_1, i_{\circlearrowleft P_1}) \sim (\circlearrowleft P_2, i_{\circlearrowleft P_2})$.*

Proof. Suppose $\Phi : S_1 \to S_2$ is a bisimulation from P_1 to P_2 with $i_{P_2} \sqsubseteq i_{P_1} ; \Phi$. We want to show that $\Psi = R_{\circlearrowleft P_1} ; \Phi ; R_{\circlearrowleft P_2}^\smile$ is the required bisimulation from $\circlearrowleft P_1$ to $\circlearrowleft P_2$. The first inclusion follows from

$$\begin{aligned}
&\Psi ; \circlearrowleft P_2 \\
&= R_{\circlearrowleft P_1} ; \Phi ; R_{\circlearrowleft P_2}^\smile ; R_{\circlearrowleft P_2} ; P_2 ; F\big(R_{\circlearrowleft P_2}^\smile\big) \\
&= R_{\circlearrowleft P_1} ; \Phi ; \Xi_{\circlearrowleft P_2} ; P_2 ; F\big(R_{\circlearrowleft P_2}^\smile\big) \\
&= R_{\circlearrowleft P_1} ; \Phi ; \big(\mathbb{I}_{S_1} \sqcup t_{P_2}^\smile ; t_{P_2} \sqcup t_{P_2}^\smile ; i_{P_2} \sqcup i_{P_2}^\smile ; t_{P_2} \sqcup i_{P_2}^\smile ; i_{P_2}\big) ; P_2 ; F\big(R_{\circlearrowleft P_2}^\smile\big) \\
&= R_{\circlearrowleft P_1} ; \Phi ; \big(\mathbb{I}_{S_1} \sqcup t_{P_2}^\smile ; t_{P_2} \sqcup t_{P_2}^\smile ; i_{P_2} \sqcup i_{P_2}^\smile ; t_{P_2}\big) ; P_2 ; F\big(R_{\circlearrowleft P_2}^\smile\big) && i_{P_2} \text{ univalent} \\
&= R_{\circlearrowleft P_1} ; \big(\Phi ; P_2 \sqcup \Phi ; t_{P_2}^\smile ; i_{P_2} ; P_2\big) ; F\big(R_{\circlearrowleft P_2}^\smile\big) && t_{P_2} ; P_2 = \mathbb{\bot}_{IF(S_2)} \\
&\sqsubseteq R_{\circlearrowleft P_1} ; \big(\Phi ; P_2 \sqcup t_{P_1}^\smile ; i_{P_2} ; P_2\big) ; F\big(R_{\circlearrowleft P_2}^\smile\big) && \text{Lemma 2} \\
&\sqsubseteq R_{\circlearrowleft P_1} ; \big(\Phi ; P_2 \sqcup t_{P_1}^\smile ; i_{P_1} ; \Phi ; P_2\big) ; F\big(R_{\circlearrowleft P_2}^\smile\big) \\
&\sqsubseteq R_{\circlearrowleft P_1} ; \big(P_1 \sqcup t_{P_1}^\smile ; i_{P_1} ; P_1\big) ; F\big(\Phi ; R_{\circlearrowleft P_2}^\smile\big) \\
&= R_{\circlearrowleft P_1} ; \big(\mathbb{I}_{S_1} \sqcup t_{P_1}^\smile ; t_{P_1} \sqcup t_{P_1}^\smile ; i_{P_1} \sqcup i_{P_1}^\smile ; t_{P_1}\big) ; P_1 ; F\big(\Phi ; R_{\circlearrowleft P_2}^\smile\big) && t_{P_1} ; P_1 = \mathbb{\bot}_{IF(S_1)} \\
&= R_{\circlearrowleft P_1} ; \Xi_{\circlearrowleft P_1} ; P_1 ; F\big(\Phi ; R_{\circlearrowleft P_2}^\smile\big) && i_{P_1} \text{ univalent} \\
&= R_{\circlearrowleft P_1} ; P_1 ; F\big(\Phi ; R_{\circlearrowleft P_2}^\smile\big) \\
&\sqsubseteq R_{\circlearrowleft P_1} ; P_1 ; F\big(R_{\circlearrowleft P_1}^\smile ; R_{\circlearrowleft P_1} \Phi ; R_{\circlearrowleft P_2}^\smile\big) \\
&= \circlearrowleft P_1 ; F(\Psi).
\end{aligned}$$

The second inclusion is shown analogously. Finally, the computation

$$\begin{aligned}
i_{\circlearrowleft P_2} &= i_{P_2} ; R_{\circlearrowleft P_2}^\smile \\
&\sqsubseteq i_{P_1} ; \Phi ; R_{\circlearrowleft P_2}^\smile \\
&\sqsubseteq i_{P_1} ; \Xi_{\circlearrowleft P_1} ; \Phi ; R_{\circlearrowleft P_2}^\smile \\
&= i_{P_1} ; R_{\circlearrowleft P_1}^\smile ; R_{\circlearrowleft P_1} ; \Phi ; R_{\circlearrowleft P_2}^\smile \\
&= i_{\circlearrowleft P_1} ; \Psi
\end{aligned}$$

shows that Ψ is the required bisimulation. □

The main theorem of this section shows that $\circlearrowleft P$ actually solves a recursive equation.

Theorem 11. *Let $(P, i_P) : S \to F(S)$. Then $(\circlearrowleft P, i_{\circlearrowleft P}) \sim (P.\circlearrowleft P, i_{P.\circlearrowleft P})$.*

Proof sketch. The theorem is shown by proving that $\Phi = R_{\circlearrowleft P}; (\iota \sqcup \kappa); (\mathbb{I}_S + R^{\smile}_{\circlearrowleft P}); R^{\smile}_{P.\circlearrowleft P}$ is the required bisimulation from $\circlearrowleft P$ to $P.\circlearrowleft P$. Again, the details can be found in [19]. □

6. Recursive equations

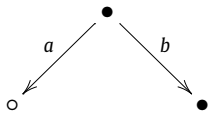
In this section we want to show how to solve certain recursive equations using the operator from the previous section. Therefore, we consider a language that is based on variables $X, Y, \ldots$ for coalgebras and some constants $P, Q, \ldots$. A grammar for this language is given by

$$Exp ::= X \mid P \mid Exp + Exp \mid P.Exp$$

If $t(X)$ is an expression in the language above with free variable X, then we are interested in a coalgebra P that satisfies $P \sim t(P)$. In order to solve this equation we assign a coalgebra P_t and a vector $r_t \sqsubseteq t_P$ to the expression $t(X)$ recursively. The vector r_t describes those terminal states that should loop.

1. If $t(X) = X$, then $P_t = (\perp\!\!\!\perp_{IF(I)}, \mathbb{I}_I)$ and $r_t = \mathbb{I}_I$.
2. If $t(X) = Q$, then $P_t = Q$ and $r_t = \perp\!\!\!\perp_{IS}$.
3. If $t(X) = t_1(X) \boxplus t_2(X)$, then $P_t = P_{t_1} \boxplus P_{t_2}$ and $r_t = (r_{t_1}; i^{\smile}_{P_{t_1}} \sqcap r_{t_2}; i^{\smile}_{P_{t_2}}); \iota \sqcup (r_{t_1}; \iota \sqcup r_{t_2}; \kappa); \kappa$.
4. If $t(X) = Q.t'(X)$, then $P_t = Q.P_{t'}$ and $r_t = r_{t'}$.

As an example consider the expression $t(X) = (a.X) + b$ where a and b are the labeled transition systems with exactly one transition from the initial state to a terminal state labeled a and b, respectively. The labeled transition system P_t can be found in the following figure where the states in r_t and indicated by a hollow bullet.



From the example we see that a recursion on r_t leads to the intended result.

Theorem 12. *We have* $\overset{r_t}{\circlearrowleft} P_t \sim t(\circlearrowleft r_t P_t)$.

Proof. By Theorem 11 it is sufficient to show that $t(\circlearrowleft r_t P_t) \sim P_t \overset{r_t}{:} \overset{r_t}{\circlearrowleft} P_t$. Therefore we show that $t(P) = P_t \overset{r_t}{:} P$ by induction on the structure of $t(X)$ for all coalgebras P.

$t(X) = X$: In this case $t(P) = \perp\!\!\!\perp P \sim P$ by Lemma 5(2).
$t(X) = Q$: We have $t(P) = Q = Q \overset{\perp\!\!\!\perp}{\cdot} P$ by Lemma 5(1).
$t(X) = t_1(X) + t_2(X)$: Now, we get

$$\begin{aligned} t(P) &= t_1(P) \boxplus t_2(P) \\ &= \left(P_{t_1} \overset{r_{t_1}}{:} P\right) \boxplus \left(P_{t_2} \overset{r_{t_1}}{:} P\right) && \text{induct. hyp.} \\ &= (P_{t_1} \boxplus P_{t_2}) \overset{r_t}{:} P. && \text{Theorem 6} \end{aligned}$$

$t(X) = Q.t'(X)$: again, we have

$$\begin{aligned} t(P) &= Q.t'(P) \\ &= Q.\left(P_{t'} \overset{r_{t'}}{:} P\right) && \text{induct. hyp.} \\ &= (Q.P_{t'}) \overset{r_{t'}}{:} P && \text{Theorem 9} \\ &= P_t \overset{r_t}{:} P. \end{aligned}$$

This completes the proof. □

7. Conclusion and future work

In this paper we have defined a sequential composition for rooted coalgebras. In addition, we have shown how recursive equations based on sequential composition and summation can be solved explicitly. From process calculi we know that certain equation, so-called guarded equations, have a unique fixed point. Future research will investigate whether this

remains true in the more general context of rooted coalgebras. This requires to generalize the notion of a guarded expression to relators of some kind, of course.

Another area of study is the relationship between parallel and sequential composition. In particular, the combination of the two categories – the ordered category of processes based on parallel composition, and the category of rooted coalgebras based on sequential composition. This kind of work might lead to completely algebraic treatment of processes and/or coalgebras.

References

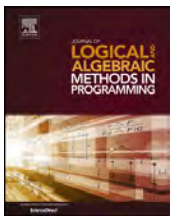
[1] S. Abramsky, The lazy lambda calculus, in: D. Turner (Ed.), Research Topics in Functional Programming, Addison Wesley, 1990, pp. 65–117.
[2] M.M. Bonsangue, J.J.M.M. Rutten, A. Silva, Coalgebraic logic and synthesis of Mealy machines, in: Proc. Foundations of Software Science and Computation Structures (FOSSACS), in: LNCS, vol. 4962, 2008, pp. 231–245.
[3] L.H. Chin, A. Tarski, Distributive and Modular Laws in the Arithmetic of Relation Algebras, University of California Press, Berkley, Los Angeles, 1951.
[4] P. Freyd, A. Scedrov, Categories, Allegories, North-Holland, 1990.
[5] C.A.R. Hoare, Communicating sequential processes, Commun. ACM 21 (8) (1978) 666–677.
[6] B. Jacobs, Exercises in coalgebraic specification. Algebraic and coalgebraic methods in the mathematics of program construction, in: LNCS, vol. 2297, 2002, pp. 237–281.
[7] R. Milner, Communication and Concurrency, Prentice Hall, 1989.
[8] J.P. Olivier, D. Serrato, Catégories de Dedekind. Morphismes dans les catégories de Schröder, C. R. Acad. Sci. Paris 290 (1980) 939–941.
[9] J.P. Olivier, D. Serrato, Squares and rectangles in relational categories – Three cases: semilattice, distributive lattice and boolean non-unitary, Fuzzy Sets Syst. 72 (1995) 167–178.
[10] M. Rößiger, Coalgebras and modal logic. Coalgebraic methods in computer science, Electron. Notes Theor. Comput. Sci. 33 (2000) 299–320.
[11] D. Sangiori, J. Rutten (Eds.), Advanced Topics in Bisimulation and Coinduction, Cambridge Tracts in Theoretical Computer Science, vol. 52, 2012.
[12] D. Sangiori, Introduction to Bisimulation and Coinduction, Cambridge University Press, 2012.
[13] V. Shehtman, Filtration via bisimulation, in: R. Schmidt, I. Pratt-Hartmann, M. Reynolds, H. Wansing (Eds.), Advances in Modal Logic, vol. 5, 2005, pp. 289–308.
[14] G. Schmidt, T. Ströhlein, Relationen und Graphen, Springer, 1989; English version: Relations and Graphs, Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoret. Comput. Sci., Springer, 1993.
[15] G. Schmidt, C. Hattensperger, M. Winter, Heterogeneous relation algebras, in: C. Brink, W. Kahl, G. Schmidt (Eds.), Relational Methods in Computer Science, Advances in Computer Science, Springer, Vienna, 1997.
[16] M. Winter, Strukturtheorie heterogener Relationenalgebren mit Anwendung auf Nichtdetermismus in Programmiersprachen, Dissertationsverlag NG Kopierladen GmbH, München, 1998.
[17] M. Winter, An ordered category of processes, in: R. Berghammer, B. Möller, G. Struth (Eds.), Relations and Kleene Algebra in Computer Science RelMiCS/AKA 2008, in: LNCS, vol. 4988, 2008, pp. 367–381.
[18] M. Winter, P. Kempf, Relational semantics for processes, in: Relational Methods for Computer Science Applications. Studies in Fuzzyness and Soft Computing, Physika Verlag, Heidelberg, 2001, pp. 59–73.
[19] M. Winter, P. Kempf, Relational properties of sequential composition of coalgebras, 2013 (long version). Brock University CS-13-09.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 300–308

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp



Gunther Schmidt's life as a mathematician and computer scientist

Rudolf Berghammer [a,*], Michael Winter [b]

[a] Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24098 Kiel, Germany
[b] Computer Science Department, Brock University, St. Catharines, Ontario L2S 3A1, Canada

ARTICLE INFO

ABSTRACT

We provide a synopsis of Gunther Schmidt's academic life, ranging from his early days as a student at the Georg-Augustus-Universität Göttingen and the Ludwig-Maximilians-Universität München over his period at the Technische Universität München to his time at the Universität der Bundeswehr München. We also highlight his efforts in providing machine support for mathematical reasoning with relations and for program development, his activities concerning collaborations and projects, and finally some of Gunther's interests beside work and science.

1. Introduction

In this paper we provide a synopsis of Gunther Schmidt's academic life to our best knowledge and interpretation. We had the pleasure to be Gunther's students and assistants for a period of 27 years in total. Moreover, we have always been, and still are, collaborating with him on various projects. Gunther has been very influential in the modern development of relation algebra, categories of relations as well as concerning relational methods in mathematics, computer science, and engineering. It is our firm belief that relational methods would not be at their current stage without Gunther. In addition to these achievements, Gunther is a person who brings together people from different areas of mathematics, computer science, engineering, social sciences and other fields. This ability has led to several national and international initiatives and collaborations, all of them extremely successful. Last but not least, the two authors bear testimony that Gunther is a superb mentor of his students. The second author can definitely state that he would not have been able to pursue an academic career if it were not for Gunther's perseverance in obtaining and providing a Ph.D. position for him at the University of the German Armed Forces Munich while he still was an officer in the German military.

We recount Gunther's academic work around his way from Göttingen to Munich and from complex analysis to computer science and relation algebra. In addition, we highlight his efforts in providing machine support for mathematical reasoning with relations and program development as well as his activities concerning national and international collaborations and projects. Last but not least, we also want to highlight some of Gunther's interests beside work and science.

In the preparation of this paper we also relied on information and material provided by Mrs. Natalia Schmidt, several companions of Gunther and some of his and our colleagues. We want to thank all of them for their help. Especially, we are grateful for Peter Jipsen's, Hans Langmaack's and Bernhard Möller's support.

* Corresponding author. Tel.: +49 431 7272; fax: +49 431 7613.
E-mail address: rub@informatik.uni-kiel.de (R. Berghammer).

http://dx.doi.org/10.1016/j.jlap.2014.02.015
2352-2208/2014 Published by Elsevier Inc.

2. Education and occupational history

Gunther's childhood and early days were determined by the horror of the Second World War and the troubles of the post-war period. A university career was unforeseeable for him at the start of his primary school in September 1945 in Rüdersdorf (near Berlin). Several times he has mentioned that his secondary school period in Nienburg and Bremen was difficult. For example, because of adverse conditions, his schooling did not include a systematic treatment of fractions. Nevertheless, having filled such gaps on his own, he passed the German Abitur in 1957 at the age of 17 years and went to Göttingen to study mathematics and physics.

2.1. Georg-Augustus-Universität Göttingen

Gunther started the study of mathematics and physics at the Georg-Augustus-Universität Göttingen in the summer semester 1957. Before the Second World War, Göttingen was widely viewed as the world center of mathematics. But this leading position was destroyed by the antisemitism of the Nazi regime, and almost all leading mathematicians of Göttingen fled from Nazi Germany. Some of them returned after the war, such as Carl Ludwig Siegel, one of the most famous mathematicians in number theory. Gunther attended courses by Carl Ludwig Siegel and also participated in a seminar offered by him.

After his Vordiplom examination, which he passed during the winter semester 1959/60, Gunther decided to leave Göttingen for one or two semesters to gain experience at other universities. He decided to go to Munich to study at the Ludwig-Maximilians-Universität München – and would never return to the Georg-Augustus-Universität.

2.2. Ludwig-Maximilians-Universität München

Already in his first semester in Munich, the summer semester 1960, Gunther attended a lecture by Karl Stein and also participated in a seminar of this well-known Munich mathematician. Several further courses by Karl Stein followed, including two courses on n-dimensional complex analysis in the summer semester 1961 and the winter semester 1961/62 (in German: Vorlesungen über die Funktionentheorie mehrerer Variabler I und II). The subsequent course on selected topics of complex analysis in Gunther's last semester of studies led to a Diploma thesis [17] supervised by Karl Stein. Following an approach of the French mathematician Henri Cartan in [17], a criterion concerning the compactification of normal complex spaces was investigated.

Having passed his Diploma examinations in the summer semester 1962, Gunther started working on a Ph.D. thesis, again on n-dimensional complex analysis and again supervised by Karl Stein. He concentrated on the continuation of functions of several complex variables. Continuation is an area of mathematics that is concerned with extending a function f to values for which f was previously not defined. In the context of complex analysis this is usually applied to so-called holomorphic functions. In [18] Gunther develops techniques that allow the continuation of holomorphic functions of several complex variables by extending their ranges provided the underlying complex space is normal. The Ph.D. thesis was finished in November 1965 and still during the winter semester 1965/66 the successful Ph.D. defense took place.

2.3. Technische Universität München

During his work on the Ph.D. thesis Gunther was informed by a fellow student, Rudolf Peters, about an open position at the Technische Universität München, at that time still called Technische Hochschule München. He applied for the position, was accepted and started on October 1, 1962. He remained at the Technische Universität until May 30, 1988. During this long period of time he had many different positions, leading from a 'Verwalter der Dienstgeschäfte eines Wissenschaftlichen Assistenten' in the group of Robert Sauer to a professorship. He also had a lot of different duties. At the beginning he was mainly concerned with the labs and tutorials accompanying the courses of Robert Sauer for engineering students, then he worked on staff administration for the department for a long time, and finally he took to care of his own group of Ph.D. and Diploma students.

Having completed his Ph.D. thesis, Gunther also changed his scientific interests from pure to applied mathematics, especially towards algorithms and computer science. Later on he sometimes mentioned that the collaboration with Hans Langmaack on Emil Artin's braid group and its relationship to formal languages and rewriting, with the results presented in [13,14], was the first step en route from ordinary mathematics to applications of mathematics in computer science and, finally, to computer science. In collaboration with Thomas Ströhlein and some others then he worked in graph theory and combinatorics, chess problems, the construction of timetables etc. Around 1974 he finally became a member of the CIP programming methodology group of Friedrich L. Bauer and Klaus Samelson [3]. As we will show in the next section, the first-mentioned topics led to his interest in relation algebra. In addition, the CIP project was of high importance for his further scientific life. In his Habilitation thesis [21], completed in 1977, he used relation algebra the first time as the conceptual and methodical tool for solving some fundamental questions concerning programming – there concretely the semantics of programming languages.

During his time at the Technische Universität München Gunther gave numerous courses on different topics, ranging from Mathematical Logic to VLSI Design. He supervised two Ph.D. theses (thesis of Hans Zierer and of the first author) and 39 Diploma theses.

2.4. Universität der Bundeswehr München

Gunther changed to the University of the German Armed Forces Munich on May 31, 1988, as a full professor and together with Hans Zierer and the first author as his coworkers. Some time later, Peter Kempf, Ludwig Bayer and Wolfram Kahl joined his group.

Concerning lecturing, during the first years Gunther concentrated on the redesign of the introductory courses in programming. Later he also offered many courses on advanced topics, including one on semantics of programming languages and domain constructions, and another on relational methods in computer science. Scientifically, Gunther first concentrated on the translation of his and Thomas Ströhlein's German book 'Relationen und Graphen' into English [26]. This book attracted attention in the international relational community, and he became finally one of its leading members. We will elaborate more on this in Section 5. The contact with many scientists from quite different areas, e.g., from logic, pure mathematics, computer science, engineering, economics and social sciences, which all have been interested in relations and their use for problem solving, led to many new ideas and, of course, publications. All that, finally, culminated in Gunter's book [32]. Besides teaching and research, Gunther also was very active in the academic administration of his university and a member of some influential committees. From 1999 to 2002 he was the Dean of the Faculty of Computer Science. He retired in the year 2004.

During his time at the University of the German Armed Forces Gunther supervised three Habilitation theses (that of Wolfram Kahl and those of the two authors), nine Ph.D. theses (among them those of Wolfram Kahl and the second author) and 38 Diploma theses.

3. From the early projects to relational mathematics

Nowadays, in the scientific community Gunther is mostly known for his outstanding work on relation algebra and its applications to problem solving in mathematics, computer science, and related disciplines. It was a long way from n-dimensional complex analysis to that topic. In this section we want to sketch some of the decisive milestones. Thereby, we also hope to demonstrate that Gunther's scientific interests were not pre-determined but the result of his constant search for solutions to problems that are both mathematically sound and as simple and elegant as possible.

3.1. Boolean matrices and construction of timetables

In the year 1971, the Department of Mathematics at the Technische Universität München was asked by the Bavarian Ministry of Education to develop a computer system for the solution of a timetabling problem for high schools. Gunther was one of the leaders of this large project, called MASTER. During the next three years the MASTER project group studied the relevant literature, developed additional novel solutions and concepts, and implemented a computer system for timetabling. However, most of the solutions it found for concrete application cases were not satisfactory.

As a consequence, in the year 1974 the timetabling problem was reconsidered, this time emphasizing the theoretical aspects in more detail. Gunther and his colleague Thomas Ströhlein developed a theoretical model of timetabling. It is based on three sets: the set of participants, the set of meetings (called meets in [19]) and the set of available time slots (called hours in [19]). The relationships between the meets and participants are specified via a Boolean matrix and the availability of the meets and the requirements on the hours for the meets are specified via two Boolean vectors. Using this approach, they were able to specify a solution of the timetabling problem as a Boolean vector, and to develop a Boolean matrix iteration technique that helps to find solutions [19].

In [19] all calculations are done with Boolean matrices and Boolean vectors. Many of these calculations were already done without referring to indices, i.e., without referring to individual matrix elements. Instead they used matrix operations such as multiplication, transposition and (component-wise) negation and their 'obvious' laws. The usage of index-free calculations is very similar to today's calculations within the calculus of relations and, in fact, [19] and some other papers from that time constitute a very important first step of Gunther's way to relation algebra.

3.2. A relation-algebraic approach to graph theory

A common way to specify directed graphs is to define them as pairs $G = (V, E)$, where V is the set of vertices, and E is the set of edges. Each edge is a pair (x, y) of vertices such that, when drawn as an arrow, the edge starts at x end ends at y. Consequently, the set of edges E is in fact a relation on the set of vertices V, or, in other words, directed graphs and relations on a carrier set are essentially the same thing.

Gunther was always interested in solving problems on certain classes of graphs, e.g., in the project METHUSALEM optimal assignments for secondary school teachers were constructed via maximum flows. Even in [19] graphs were used to explain and visualize timetables. In the late 80's and early 90's Gunther was involved in the conference series 'Graph-Theoretic Concepts in Computer Science', and he organized this conference in the year 1991, together with the first author [15,16,39].

During his investigation of graph-theoretic problems, in the middle of the 70's Gunther noticed that viewing directed graphs as relations and these in turn as elements of a relation algebra allows formulating many graph-theoretic properties and problems in a purely relation-algebraic way. An essential prerequisite for this was an algebraic treatment of sets and

their elements. This led to the notions of vectors and points. In the Boolean matrix model vectors are relations in which the rows consist of only 1's or only 0's, and points are vectors with precisely one row with only 1's. Since, therefore, for vectors and points in this model the number of columns is irrelevant, for practical reasons it should be set to one. Transferring this to abstract algebra led to the concept of typed relations and hence to what is now called a heterogeneous relation algebra.

Originally Gunther presented his ideas in [20], already using the notion of a relation algebra and important laws like the Schröder rule. Later, he extended the approach to other classes of graphs and solved a lot of graph-theoretic problems relation-algebraically. Most solutions are documented in his books [26,32] and in a series of conference and journal articles. Members of his former groups and other scientists continued the work by considering more sophisticated graph-theoretic problems (like algorithms for graph searching and the computation of Hamiltonian cycles), other problem areas (e.g., order and lattice theory, Petri nets, games, social choice theory) and more general algebraic structures (like semirings, Kleene algebras and sequential algebras). In algorithm development this led to many interesting problem solutions, especially, by combining the relational approach with a formal program development technique like the Dijkstra–Gries method or transformational programming.

3.3. Relational semantics of programming languages

Programming languages and programs lie at the basis of computer science. As a mathematician Gunther always believed that a precise mathematical description of the behavior of programs is essential to understand, to reason about and to develop them. Especially the flow and gain of information during the execution of a program has always been a major part of his studies.

In his Habilitation thesis [21] Gunther for the first time formally introduces the notion of a heterogeneous relation algebra and proves in the first part many properties from its axioms. In the remainder of [21] he then summarizes his work on the relation-algebraic description of programs as partial graphs. This work was extended and published in a series of papers [21–24] during the late 70's and the early 80's. The basic ideas and results were also included in his book [26]. Gunther's approach to program semantics using graphs is based on an operational description. Each program construction is interpreted as a simple flow graph represented as a relation together with another relation that describes the effect of the program on the data. Further studies include partial and total correctness and program transformations.

Gunther also contributed to the so-called denotational semantics of programming languages. He strongly believes that all mathematical constructions for program semantics must be generated from finite instances using a constructive process. This idea led to 'Domains with Sprouts' – domains that are generated by their finite approximations [25]. Later it turned out that his approach is equivalent to the so-called profinite domains of Carl Gunter [8]. Following the basic theory of domains with sprouts, Gunther also investigated non-standard constructions such as congruences and specific isomorphisms. In [25] domains with Sprouts are treated using ordinary set theory. Later, in the Ph.D. thesis [41], Hans Zierer (supervised by Gunther) presented a purely relation-algebraic description of the approach.

When writing his Diploma thesis [7], Rodrigo Cardoso, supervised by Gunther, encountered a problem in proving a specific equation in the theory of relation algebras with relational products. If we identify programs with relations and parallel composition with products, his problem can be formulated as follows: We consider two different systems of processes constructed from basic processes P, Q, R, and S. It is assumed that P and R receive and send information on channel 1 while R and S use channel 2. Does the system in which P runs in parallel with Q followed by R in parallel with S behave the same way as the system where P followed by R runs in parallel to Q followed by S? Graphically the two systems look as follows:

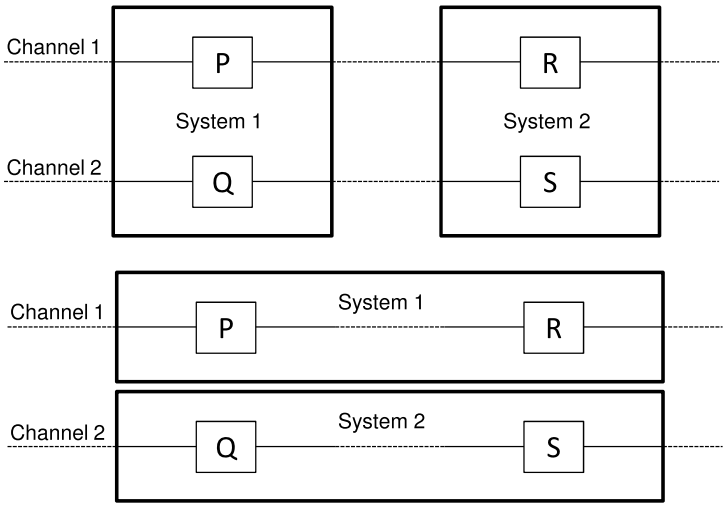


Fig. 1. Two systems of processes.

This question is of particular interest if only local observation is possible or information about the system is incomplete. The problem led to a series of papers by Gunther investigating the flow of information and partialities in the context of parallel composition and relation algebras [28,34,35].

3.4. Relational mathematics

During Gunther's academic life the calculus of relations became more and more fascinating for him. In recent years he focused all his research efforts completely on this topic. In retrospect, this seems to have mainly four reasons.

First, a lot of interesting properties can be expressed relationally in a compact and variable-free manner. This makes it possible to formulate quite large problems in fully algebraic form without relying on descriptions in natural language. Next, relation-algebraic reasoning is based on equations or inclusions without complicated quantifiers. This makes proofs and computations easy to verify, especially by supporting tools, and allows the direct execution of the constructions for finite examples. But, of course, all this does not imply that proofs are easier to find. Actually this can even be harder due to the reason we will consider next. Beside the standard model of relations as sets of pairs, relation algebras cover additional models such as fuzzy relations and non-representable relation algebras. Therefore theorems derived in the theory of relation algebras are more general than the corresponding ones in regular logic. And, as a final reason, in the standard model of relation algebras examples for an expression or a property can easily be visualized using Boolean matrices. Similar to Linear Algebra, all computations can be done using simple operations on matrices.

As a consequence, in recent years Gunther invested the main part of his scientific work into the effort to 'relationalize' mathematics, i.e., to rephrase and to generalize mathematical theories within the calculus of relations, and to apply the results to the solution of important problems of mathematics, computer science, and related disciplines. After his very early projects (we only have mentioned two of them, viz. Master and Methusalem) he started with problems of graph theory and the semantics of programming languages as already mentioned in the previous two sections. Later investigations considered, for instance, a relation-algebraic approach to social choice theory, preference modeling, and related topics. This includes work on relation-algebraic formulations of problems in that area, computational aspects, and general studies of order, measures and integrals [29–31,36].

All this work on 'relational mathematics' is contained in the book [32], published by Gunther in the year 2010. This book can be considered as his *opus magnum*, a summation and consolidation of a great part of his results. But this book is not the end of the story. Gunther still works on relational mathematics and new applications. For example, recently he started investigating a relation-algebraic version of general topology [33]. To our knowledge he is still working on a large paper covering that topic in more generality.

4. Mechanization and tool support

When relation algebra is used as the basic principle for problem solving, all specifications, derivations, and calculations are usually done in a very precise and formal manner. Furthermore, as mentioned above, in the standard model of relation algebra Boolean matrices can be used for visualization and implementation. Because of these reasons, Gunther came to the conclusion that computer assistance is not only possible but essential while working with relations. For example, systems can be used to reduce potential errors in proofs, to support calculations, to test hypotheses, and to get new scientific insights by systematic experiments. Already during the 80's he initiated the development of appropriate computer systems, and a series of relation-algebraic tools with quite distinct areas of applications. The most important ones are sketched in the following; some of them are still in use.

4.1. RelView

RelView is an interactive and graphics-oriented computer system for calculating with relations and relational programming. Almost all data in it are represented as relations, which the system visualizes as Boolean matrices or directed graphs. For the latter it includes sophisticated algorithms for drawing graphs nicely. The tool allows the user to manipulate and analyze relations by a lot of predefined operations and tests, which can interactively be accessed through command buttons and simple mouse-clicks. They are also available in the tool's programming language and the usual way of manipulating relations is via user-defined relational functions and programs.

Gunther initiated the development of a tool for the manipulation of relations while still at the Technische Universität München. A first prototypic implementation was done in the course of a Diploma thesis supervised by him [2]. After his move to the University of the German Armed Forces, the development of a much more ambitious tool was started, implemented by Hilde Abold-Thalmann, and the first description of the result, now called RelView, appeared in 1989 [1]. In this tool relations were implemented via lists of bits. But experience subsequently showed that many RelView applications and computations use very large relations, for instance, membership and size comparison relations on powersets. Gunther proposed the use of reduced ordered binary decision diagrams (ROBDDs) instead of lists of bits for implementing relations to cope with such applications. Some time around the year 2000, this suggestion was implemented at the University of Kiel by the research group of the first author of this paper in the course of two Ph.D. theses and a series of Diploma theses. Since that time efficient ROBDD-algorithms constitute the kernel of the RelView tool.

4.2. RALF

In contrast to RELVIEW, the RALF system is an interactive proof assistant for the calculus of relations [10]. It allows the user to enter relation-algebraic expressions and to manipulate them using axioms and theorems of relation algebra as well as higher-order logical rules. The system uses a graphical user interface that represents expressions and formulas as term graphs. This makes it easier to identify the structure of the objects in question and the places where rule applications are advantageous. RALF is a multi-user system. Users are authorized by a password mechanism and there is a superuser having special rights, like defining theorems, meta-rules and definitions that may be used by everyone. So, all users have common theorems and definitions to work with in addition to their own ones.

Like the development of RELVIEW, the development of RALF was also initiated by Gunther in the course of a Diploma thesis [5]. The completion of this prototype to the current state then was done by Claudia Hattensperger during her Ph.D. thesis [9], supervised by Gunther, and a series of accompanying Diploma theses.

4.3. RATH

The RATH system presents a library of Haskell modules. Its main purpose is to explore and investigate relation algebras and several weaker structures, such as categories, allegories, distributive allegories, division allegories, and Dedekind categories, by providing possibilities to construct and experiment with such structures [4]. The tool constitutes a common framework for calculational work with all the structures mentioned. It takes into account that they share concepts and properties, so as, for instance, to introduce the idea of division only once for division allegories and to directly reuse it for the more specific Dedekind allegories as well as for relation algebras. When using RATH, it is also possible to execute relation-algebraic expressions. Usually, executions are not as efficient as in RELVIEW. But in contrast with RELVIEW, the system provides the possibility to switch to non-standard relation algebras, i.e., to exchange the underlying model of the relational category in question. RATH also provides means to construct new algebras from given ones as product algebras, subalgebras, and matrix algebras. Last but not least, it is possible to generate a specific relational category by defining the corresponding operations on the set of atoms and taking the complex algebra over this atom structure. This reduces the size of the algebra and the complexity of the operations.

The RATH tool has also been initiated by Gunther. It was implemented by members of his group at the University of the German Armed Forces. Based on a result of Roger Maddux, as one of the most important applications of RATH by the second author of this paper, it was verified that Rodrigo Cardoso's problem mentioned above has a negative answer, that is, there exists a non-standard model of relation algebra where the two systems of Fig. 1 do not behave the same way.

4.4. TITUREL

TITUREL is a relation-algebraic description language that is based on and completely implemented in Haskell [27]. It allows one to compute and visualize relation-algebraic expressions for finite examples in the standard model of relation algebra. In this respect, the tool has many similarities with the RELVIEW tool. However, in addition to the relation-algebraic constructions, in TITUREL all features of Haskell are available, in particular an established static type inference system. This makes the tool much more type-safe and also expressive than RELVIEW, since in the latter the correct typing is only dynamically controlled and the programming language is, with only few exceptions, the language of while-programs over a data type of relations, i.e., rather restrictive. Again compared with RELVIEW, the drawback of TITUREL is the simple and, therefore, frequently very inefficient implementation of relations.

Gunther has developed the TITUREL library for his own purpose in parallel with the writing of his book [32]. Its language serves as the reference language throughout the book and each of the numerous examples (including the labeling) has been produced with its help. For such applications and similar ones the efficiency of TITUREL proved to be sufficient.

4.5. HOPS

A great deal of Gunther's scientific investigations were motivated by the goal to find sound mathematical techniques which allow the development of correct programs from formal problem specifications. As already mentioned, in the mid 1970's he became a member of the well-known CIP project [3]. But after some time he began to doubt that the very ambitious goals of the project concerning the transformation of programs are reachable if programs are considered as strings and many of the transformations are done at a rather low level. In close cooperation with Hans Zierer and Wolfram Kahl he started HOPS, his own project on a 'Higher Order Programming System' [40]. At first glance the HOPS tool does not look like a relational tool but like a graphical system for transformational program development, visualization, and debugging [12]. But a detailed analysis shows that there are deep connections between HOPS and relation algebra, since all concepts underlying the tool are specified relation-algebraically.

In contrast with the usual string-oriented approach Gunther was critical of during the CIP project, in the HOPS tool all programs are represented as second-order term graphs and, program transformation is done via graph transformation [11]. The term graph representation is used to make explicit the structure of programs, which usually is encoded via names. Besides this, it is also used to make the structure of the program typing and the scores of nameless variables explicit.

Graph edges determine which node binds which variable. This approach avoids many problems which in a string-oriented approach are connected with name clashes and variable renaming. Furthermore, the term graph representation easily allows syntax-directed editing with strong online typing. As a consequence, in the Hops tool the user is only able to construct syntactically correct and well-typed programs.

5. Collaborative research and networking

Collaboration of researchers within scientific networks is one of the most effective ways to strengthen science and to produce new and innovative results. Throughout his scientific life, Gunther actively communicated with many scientists. He also was part of some national and international networks and initiatives. We already have mentioned his role in the conference series on graph-theoretic concepts in computer science. In view of scientific cooperations his RelMiCS initiative starting in 1994 is surely his most influential one. It not only led to new and intensive cooperations within the relational community and a now well-established conference series, but also to some important descendants.

5.1. Relational methods in computer science

Because of their work on relation algebra, in the year 1991 Gunther and Thomas Ströhlein were invited to a semester seminar on algebraic logic, held at the Banach Mathematical Center in Warsaw. The participants of the meeting took this occasion to discuss whether and how to establish a new conference series on the use of algebraic techniques in software technology (now known as the Amast conference series). Influenced by this idea, Gunther decided to apply for a Dagstuhl seminar to bring together researchers from the different groups (spread over almost all continents) which use relation algebra and related structures as the conceptual and methodical base of their work. Since such a seminar requires a second foreign chairperson, Chris Brink from Cape Town University was asked to join the initiative. He agreed and suggested 'Relational Methods in Computer Science' as the name of the seminar, abbreviated as RelMiCS. The first RelMiCS seminar took place in January 1994. At this seminar there originated the idea to produce a common book on this topic, which appeared three years later [6].

As already mentioned, Gunther's initiative led to a well-established conference series. In view of its success and adaptability we think that the following list speaks for itself: Until October 2001 five further RelMiCS meetings took place: 1995 in Parati (Brazil), 1997 in Hammamet (Tunisia), 1998 in Warsaw (Poland), 2000 in Québec (Canada), and 2001 in Oisterwijk (The Netherlands). During the preparation of the seventh meeting it was decided to join RelMiCS with the AKA workshop series on Applications of Kleene algebra. This was mainly motivated by the substantial common interests and overlap of the two communities. After RelMiCS/AKA 2003 in Malente (Germany) the next meetings were: 2005 in St. Catharines (Canada), 2006 in Manchester (UK), 2008 in Frauenwörth (Germany), and 2009 in Doha (Qatar). Over time it became clear that, besides relation algebra and Kleene algebra, there are many similar algebraic structures which have their own advantages in case of specific applications. To stay abreast with this development, since the 2011 meeting in Rotterdam (The Netherlands) the series was renamed to 'Relational and Algebraic Methods in Computer Science', abbreviated as RAMiCS. In 2012 RAMiCS was held in Cambridge (UK) and 2014 RAMiCS will be held in Marienstatt (Germany).

5.2. *The* Cost *Action 274* – Tarski

Cost is a framework of the European Science Foundation for the support of cooperation in science and technology in Europe. As an active member of the relational community and encouraged by the great success of RelMiCS, in the year 2000 Ivo Düntsch (at that time at the University of Ulster) took the initiative to apply for a Cost action. This Cost Action 274, with the name 'Theory and Application of Relational Structures as Knowledge Instruments' and abbreviated as Tarski, started in June 2001. After a short time, however, its founding chairman Ivo Düntsch moved to Brock University (Canada) and the 30 members of the Tarski management committee (from 17 European countries and representing researchers from 50 different European research institutions) were confronted with the problem of searching for a new (European) chairperson of this very large project.

Gunther agreed to act as the new chairman of Tarski, supported by Ewa Orlowska, Marc Roubens and Harrie de Swart. This turned out to be a very demanding job with a lot of committee meetings and other time-consuming administrative duties, until the end of the action in June 2005. His engagement and enthusiasm were one of the reasons for the great success of Tarski. The latter is documented, among other things, by the 196 publications listed in the bibliography of the Tarski final report, including the start-up volume [37] and the final volume [38], and the large number of meetings of the working groups and mutual visits of individual Tarski members.

6. There is a life beside science

In the year 1976 the first author of the present paper attended his first lecture by Gunther on graphs and relations, still as a third-year student of mathematics and computer science at the Technische Universität München. Later he was a member of Gunther's group from 1979 to 1993. The second author attended his first lecture in the year 1989 at the University of the German Armed Forces and was a member of Gunther's group at that university from 1993 to 2003. During

this time we both noticed that science constitutes an import part of Gunther's life. But we also observed that science is not the sole purpose of his life and that there are a lot of other things he enjoys. Some of them we want to mention at the end of this appreciation.

Gunther has always been a very sociable person. He definitely enjoys a nice conversation among colleagues and friends, while having a glass of wine or beer. This could be the case in a good restaurant, a Bavarian inn or beer garden, e.g., after a seminar talk or a business meeting, or in his domicile in the Tituralstraße in Munich. From his time at the University of the Federal Armed Forces Munich certain occasions deserve to be mentioned. Every year the department spent an afternoon at the Oktoberfest. Gunther was one of the few faculty members who regularly participated. Even though verbal communication in a beer hall at the Oktoberfest is difficult, if not impossible, these 'departmental meetings' were very fruitful. In this context also the parties after a successful Ph.D. defense in the Department of Computer Science at the University of the Federal Armed Forces Munich should be mentioned, since they are legendary. They normally lasted until long after midnight – and it goes without saying that Gunther was never the first to leave. During one of these events Gunther even engaged in baby-sitting the son Felix Winter (Gunther always calls him Felix Klein) of the second author. Felix was less than one year old at the time so that this experience provided a very good training for Gunther's becoming grandfather a few years later.

Sociable people are often interested in sports. Therefore, it is not surprising that sports always played an important role in Gunther's non-scientific life. He and his wife Natalia are very active long-time members of a fitness group in their sports club. Skiing and hiking with friends are other kinds of sports they enjoy. And, finally, swimming must be mentioned. Natalia and Gunther like to swim, and in his youth Gunther was an excellent swimmer. This is proved by the following little story. During the late summer 1964 a secretary of the department, Ingrid Pauli, enthusiastically told him about a flight in a glider across the Schliersee, a lake 55 km south-east of Munich. During the conversation Gunther claimed that the Schliersee is a very small lake and he would be able – using flippers – to swim across it from north to south (2.3 km) within 2 hours. A bet was made and Gunther won it on September 17, 1964 with a time of only 57 minutes!

Last but not least, the authors wish Gunther a long life in the best of health, together with his wife and the family. We also wish him many more interesting scientific ideas and corresponding publications, thank him for his support and friendship over all the years, and hope that he remains the competent and cooperative partner in the future.

References

[1] H. Abold-Thalmann, R. Berghammer, G. Schmidt, Manipulation of concrete relations: The RelView-system, Bericht Nr. 8905, Fakultät für Informatik, Universität der Bundeswehr München, 1989.
[2] L. Alami, Dialogsystem für die Überprüfung von Behauptungen der Relationenalgebra, Diplomarbeit, Technische Universität München, 1985.
[3] F.L. Bauer, et al., The Munich Project CIP, vol. I: The Wide Spectrum Language CIP-L, Lect. Notes Comput. Sci., vol. 183, Springer, 1985.
[4] R. Berghammer, G. Schmidt, M. Winter, RelView and Rath – two systems for dealing with relations, in: H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments, in: Lect. Notes Comput. Sci., vol. 2929, Springer, 2003, pp. 1–16.
[5] R. Brethauer, Ein Formelmanipulationssystem zur computergestützten Beweisführung in der Relationenalgebra, Diplomarbeit, Universität der Bundeswehr München, 1991.
[6] C. Brink, W. Kahl, G. Schmidt (Eds.), Relational Methods in Computer Science, Advances in Computing, Springer, 1997.
[7] R. Cardoso, Untersuchung von parallelen Programmen mit relationenalgebraischen Methoden, Diplomarbeit, Technische Universität München, 1982.
[8] C.A. Gunter, A universal domain technique for profinite posets, in: W. Brauer (Ed.), Automata, Languages and Programming, in: Lect. Notes Comput. Sci., vol. 194, 1985, pp. 232–243.
[9] C. Hattensperger, Rechnergestütztes Beweisen in heterogenen Relationenalgebren, Dissertation, Universität der Bundeswehr München, 1997.
[10] C. Hattensperger, R. Berghammer, G. Schmidt, Ralf – a relation-algebraic formula manipulation system and proof checker, in: M. Nivat, C. Rattray, T. Rus, G. Scollo (Eds.), Algebraic Methodology and Software Technology, in: Workshops in Computing, Springer, 1993, pp. 407–408.
[11] W. Kahl, Algebraische Termersetzung mit gebundenen Variablen, Dissertation, Universität der Bundeswehr München, 1995.
[12] W. Kahl, The term graph programming system Hops, in: R. Berghammer, Y. Lakhnech (Eds.), Tool Support for System Specification, Development and Verification, Springer, 1999, pp. 136–149.
[13] H. Langmaack, G. Schmidt, Der Verband der Zöpfe, Bericht 6914, Abteilung Mathematik, Technische Hochschule München, 1969.
[14] H. Langmaack, G. Schmidt, Klassen unwesentlich verschiedener Ableitungen als Verbände, in: J. Dörr, G. Hotz (Eds.), Automatentheorie und Formale Sprachen, Bericht 3, Mathematisches Forschungsinstitut Oberwolfach, 1970, pp. 169–172.
[15] E. Mayr, G. Schmidt, G. Tinhofer (Eds.), Graph-Theoretic Concepts in Computer Science, Lect. Notes Comput. Sci., vol. 903, Springer, 1994.
[16] G. Schmidt, R. Berghammer (Eds.), Graph-Theoretic Concepts in Computer Science, Lect. Notes Comput. Sci., vol. 570, Springer, 1992.
[17] G. Schmidt, Kompaktifizierung normaler komplexer Räume, Diplomarbeit, Ludwig-Maximilians-Universität München, 1962.
[18] G. Schmidt, Fortsetzung holomorpher Abbildungen unter Erweiterung des Bildraums, Dissertation, Ludwig-Maximilians-Universität München, 1965.
[19] G. Schmidt, T. Ströhlein, A Boolean matrix iteration in timetable construction, Linear Algebra Appl. 15 (1976) 27–51.
[20] G. Schmidt, Eine relationenalgebraische Auffassung der Graphentheorie, in: H. Noltemeier (Ed.), Graphen, Algorithmen und Datenstrukturen, in: Applied Computer Science – Berichte zur Praktischen Informatik, Hanser, 1976, pp. 315–325.
[21] G. Schmidt, Programme als partielle Graphen, Habilitationsschrift, Technische Universität München, 1977.
[22] G. Schmidt, Investigating programs in terms of partial graphs, in: H.A. Maurer (Ed.), Automata, Languages and Programming, in: Lect. Notes Comput. Sci., vol. 71, 1979, pp. 505–519.
[23] G. Schmidt, Programs as partial graphs I: Flow equivalence and correctness, Theor. Comput. Sci. 15 (1981) 1–25.
[24] G. Schmidt, Programs as partial graphs II: Recursion, Theor. Comput. Sci. 15 (1981) 159–179.
[25] G. Schmidt, R. Berghammer, H. Zierer, Describing semantic domains with sprouts, Acta Inform. 27 (3) (1989) 217–245.
[26] G. Schmidt, T. Ströhlein, Relationen und Graphen, Springer, 1989;
English translation: G. Schmidt, T. Ströhlein, Relations and Graphs, Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoretical Computer Science, Springer, 1993.
[27] G. Schmidt, Proposal for a multilevel relational reference language, J. Relat. Methods Comput. Sci. 1 (2004) 314–338.

[28] G. Schmidt, Partiality I: Embedding relation algebras, J. Log. Algebr. Program. 66 (2) (2006) 212–238.
[29] G. Schmidt, Relational measures and integration, in: R.A. Schmidt (Ed.), Relations and Kleene Algebra in Computer Science, in: Lect. Notes Comput. Sci., vol. 4136, 2006, pp. 343–357.
[30] G. Schmidt, Rectangles, fringes, and inverses, in: R. Berghammer, B. Möller, G. Struth (Eds.), Relations and Kleene Algebra in Computer Science, in: Lect. Notes Comput. Sci., vol. 4988, 2008, pp. 352–366.
[31] G. Schmidt, R. Berghammer, Relational measures and integration in preference modeling, J. Log. Algebr. Program. 76 (1) (2008) 112–129.
[32] G. Schmidt, Relational Mathematics, Encyclopedia of Mathematics and Its Applications, vol. 132, Cambridge University Press, 2010.
[33] G. Schmidt, R. Berghammer, Contact, closure, topology, and the linking of row and column types of relations, J. Log. Algebr. Program. 80 (6) (2011) 339–361.
[34] G. Schmidt, Constructions around partialities, in: H.C.M. de Swart (Ed.), Relational and Algebraic Methods in Computer Science, in: Lect. Notes Comput. Sci., vol. 6663, 2011, pp. 314–330.
[35] G. Schmidt, Partiality II: Constructed relation algebras, J. Log. Algebr. Program. 81 (6) (2012) 660–679.
[36] G. Schmidt, Relational concepts in social choice, in: W. Kahl, T.G. Griffin (Eds.), Relational and Algebraic Methods in Computer Science, in: Lect. Notes Comput. Sci., vol. 7560, 2012, pp. 278–293.
[37] H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments, Lect. Notes Comput. Sci., vol. 2929, Springer, 2003.
[38] H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments II, Lect. Notes Artif. Intell., vol. 4342, Springer, 2006.
[39] G. Tinhofer, G. Schmidt (Eds.), Graph-Theoretic Concepts in Computer Science, Lect. Notes Comput. Sci., vol. 246, Springer, 1987.
[40] H. Zierer, G. Schmidt, R. Berghammer, An interactive graphical manipulation system for higher objects based on relation algebra, in: G. Tinhofer, G. Schmidt (Eds.), Graph-Theoretic Concepts in Computer Science, in: Lect. Notes Comput. Sci., vol. 246, Springer, 1987, pp. 68–81.
[41] H. Zierer, Programmierung mit Funktionsobjekten: Konstruktive Erzeugung semantischer Bereiche und Anwendung auf die partielle Auswertung, Dissertation, Technische Universität München, 1988.

Journal of Logical and Algebraic Methods in Programming 83 (2014) 309–317

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Editorial

On nothing

Martin E. Müller *, Bernhard Möller

Unstitute for Unformatics, Un-Diversity of Sdnegrin

ARTICLE INFO

Article history:
Available online 12 February 2014

ABSTRACT

This article is the posthumous publication of a fundamental work of the late genius M.U. Newtral. Little is known of his life, upbringing, education and his golf handicap. Whether he enjoyed flyfishing is still a matter of scientific controversy. From his style of writing and the locations where his manuscripts were discovered one might conclude that he spent a reasonable time of his life in India and the United Kingdom, where he presumably worked under the influence of many such famous people as Luke Withstone, Rudinch Kipling, Morten Haydagger and, last but not least, Petula Farnsbath-Wellworth, 31 Rosebud Drive, Eightashgreen, Devonshire (ring and knock thrice to be admitted). Newtral disappeared under nebulous circumstances somewhere in South-America. Some of his belongings were discovered by Leumas Reltub on the Island of Erewhon, and the here partially reprinted manuscript was not discovered by a nihilist. Yet, the editors wish to keep the exact location of the original a secret. Newtral probably received the impetus for writing his treatise upon the following occasion.

In his 2011 paper on "Constructions Around Partialities" Gunther Schmidt discussed a partiality of quite unique character. It is known as the smallest part or piece of a whole, and it is unique in the way that it seems there is only exactly one instance and interpretation of it: nothing. *When the editors became involved in the production of a Festschrift for Gunther Schmidt they decided to finally publish Newtral's work, which happened to have been in their hands for quite some time without them knowing what to do with it, as a tribute to Gunther in the hope that it will further or at least subtract nothing from the growth of relational mathematics. The original manuscript (partially handwritten) has been typeset with greatest care; so for all the mistakes, only M.U. Newtral himself is to blame. Helpful comments by R. Berghammer, P. Höfner and M. Winter are gratefully acknowledged.*

$\mathcal{O}$. Introduction

[...] however, the question coming to mind of a mathematician, when it comes down to sharing things among friends, is whether *nothing* actually is *some-thing* or if it is not. Or, if nothing *is not*, anyway. Or, finally, if *no-thing* is not nothing but not something, either.

By x, we shall refer to an arbitrary *thing* that lives (or *is*) within our world W. For example, x could be an apple. x could also be a heap of apples. But could x be *no-thing*, *nothing*, or does the usage of x imply that it *is something*?

Since we all did our homework, we shall keep in mind that there might be a few traps there. But these are traps that we can carelessly yet safely neglect for now since we care for nothing. And, also, since we want to make a point here, the question of whether having not-a-point or not having a point is anything but pointless.

* Corresponding author.

http://dx.doi.org/10.1016/j.jlap.2014.02.016
2352-2208/

[Obviously, as we shall see later on, Newtral was an impressively literate person. For this reason it seems quite astounding that, especially in questions concerning apples and bags thereof, he did not refer to the problem of types as it has been thoroughly studied in the last century. The reason is unknown to the editors—it is hard to believe that Newtral did not know about it. The only thing we can imagine is that this paradox has been dealt with on the 357 pages that were attached to this manuscript. Another, although not as obvious explanation, could be that it appears irrelevant to him to assign a type to nothing. In the Newtral research community it is currently discussed whether the colloquial term "nope" actually was introduced by Newtral to denote "no-*thing-type*"*. —The editors.]*

|{{}}|. Foundations

... in which we discover the perils of playing with knives and apples.

By /, we shall denote the operation of *slashing*. Let x be an apple. We slash an apple and get

one piece a and another piece b.

Obviously, the apple has been divided into two parts and no part intersects with the other and, using a reverse operation (\, "glue") would yield the original apple $x = a\backslash b$ (and no more and no less). Also, a is what remains when slashing b away from x and vice versa: $a = x/b$ and $b = x/a$. At the same time, slashing an apple results in

one piece a and another piece b *and no-thing* c *else*.

This means that slashing an apple delivers *three* things: the first piece, the second piece, and one piece that *is-not-there*.

Let us now put a and b on a balance. Of course, the added weights result in the weight of the entire apple x. Supposing an ideal cut and denoting the weight of an object by $|\cdot|$, we have $|a| + |b| = |x|$. Again, we also have $|a| + |b| = |x| = |a| + |b| + |c| = |x| + |c|$. Obviously, there appears to be no difference whether an apple is slashed into two or three pieces. Good to know if you are expecting a dozen of guests for dinner and you have only one apple in your fridge. On the other hand: You don't have to slash something at all to get nothing from it. (Good to know, if you don't even have an apple in your fridge with your friends showing up unannounced). Also, we derive a very important lemma from this observation:

Nothing is for free.

Let x be a bag containing n apples $a_1, \ldots, a_n$. Obviously, we have $n+1$ objects, where n objects reside in the $(n+1)$-st one. We slash the bag carefully (without slashing any apples in that process). We get: a heap of n apples on the floor and *two* torn pieces of cloth. So one bag became two non-bags y_1 and y_2, and n apples. Does this make $n+2$ or n things? Let us try the scales experiment:

$$|x| + \sum_{i \in \mathbf{n}} |a_i| = |y_1| + |y_2| + \sum_{i \in \mathbf{n}} |a_i|$$

But since neither y_1 and y_2 are bags, they are *no-bags*. They are *not* bags. Somehow, the bag disappeared. And if something *is-not*, how can it be of any mass? Therefore, y_1 and y_2 have no bag-weight:

$$|x| - |y_1| = |x| = |x| - |y_2| \text{ which requires that } |y_1| = |y_2|.$$

So if the two parts of the bag had a weight then they would have the same weight—which means that there can't be slashes that separate a bag into two pieces of different weight. On our table we now have objects that in sum have the weight

$$|x| + \sum_{i \in \mathbf{n}} |a_i| = |y_1| + |y_2| + \sum_{i \in \mathbf{n}} |a_i| = 2|y_2| + \sum_{i \in \mathbf{n}} |a_i| = \sum_{i \in \mathbf{n}} |a_i|.$$

So while we first got *one* bag, we now have $0 + 0 = 0$ bags plus apples

$$a_1, a_2, \ldots, a_n$$

which gives n things plus two pieces of cloth (and no bag). This result supports the intuitive solution that comes to mind when wondering about the fact that any cut through a bag always results in two pieces of same weight: It is hardly ever the case that we can evenly cut any object into exactly similar (volume, weight or taste) pieces. In the case of the bags the immediate consequence is the following corollary:

Bags have no weight.

The interesting thing is that if the bag was empty, we are left with *two* pieces—which seems perfectly all right: slashing something results in more things. But if we now take the two non-bags and slash each of them again, we get an arbitrary number of pieces (theoretically) but not a single bag.

We conclude: Under certain, wise circumstances (preservation of things and least effort Ockham-razors) slashing turns out to be monotone and idempotent. Together, it means that slashing is rather pointless: we can cut as often as we want or we can not cut as often as we want—we always yield something: nothing.

But somehow, there ought to be a difference about not cutting something or cutting off nothing from something. Otherwise, we'd all be living a life of ease without worrying that nothing might be a not big enough piece of something.

|{{}, {{}}}|. Section 3

... which rather should be called "Section Naught".

Talking about apples does not require the existence of apples—all it takes is to have a symbol or word or sign that we agree upon to denote (the idea of) an apple. So if we agree that a denotes an apple, "a is rotten" is a proposition that we can work with.

Since actual existence (whatever that may be) is *not* required, we can talk about things that are not there—as long as we can name them.[1] It seems perfectly alright to assume there are many no-things out there (there is neither Santa nor a unicorn but there are No-Santa and No-Unicorn). But why do we only speak of "nothing" instead of "nothings"? Is it that not-being is some kind of unique property that makes No-Santa the same as No-Unicorn?

Obviously, the term "nothing" also denotes a *generalisation* over all those things that share the common property of not being there. That makes "nothing" also the set of all "nothings". Which in turn means that "nothings" are not there. The immediate and infallible consequence would be that there are no nothings!

No apples

Counting is a useful *measure* for things or no-things. An apple and a pear together in a bag {} make two things in the bag:

$$|\{a, p\}| = 2$$

Not adding anything else to this set of things leaves the set unchanged and thus its cardinality, too. Adding nothing to this set means to put something non-existent into the bag, for example the present king k of France[2]:

$$\begin{aligned} |\text{putinto}(k, \{a, b\})| &= |\{a, b, k\}| \\ &= \begin{cases} 3, & \text{if the thing referred to by } k \text{ exists} \\ 2, & \text{if the thing referred to by } k \text{ does not exist} \end{cases} \end{aligned}$$

This is somehow unsatisfactory: repeatedly putting things into the bag (may they exist or not) results in a repeated (and exponentially growing!) number of possible cardinalities that we can only resolve by checking for the very existence of every thing that we put into the bag.

We could try a trick: put everything into an empty bag of its own before putting it into the larger bag:

$$\begin{aligned} |\text{putinto}(\text{putinto}(k, \{\}), \{a, b\})| &= |\text{putinto}(\{k\}, \{a, b\})| \\ &= |\{a, b, \{k\}\}| \\ &= 3 \end{aligned}$$

The problem is: What is in {}? What does "*empty*" mean? Either, there exist no things that are in this bag. But again, there is a thing (e.g. "foob", see footnote 1). To be precise, there are as many non-existent things as we can find non-equivalent names for them. Or, since we agreed that "nothing" is a generic concept (i.e. a bag of all nothings...), the empty bag is the bag that contains the bag containing all nothings. Which in turn makes the empty bag not really empty. The next method would be to deconstruct an empty bag from an arbitrary bag by taking away from it everything that is in it. This appears to be a pretty nice method: We can take away things that exist *and* we can also take away things that do not exist. We cannot take away anything from an empty bag, which is good, too, as it leaves us with a fixed point: an empty bag from which, when we try to take anything away from it, remains an empty bag. As an example, removing all barbers from the bag containing all barbers that do not shave themselves simply results in the empty bag. This could have saved Profs. Ruse-Sell and Blackfoot many sleepless nights.

[1] One scary corollary is that any name therefore denotes something: either something that is or something that isn't. But it means that any sentence of correct grammatical structure becomes a "meaningful" sentence, since we can assume any word in it to be grounded in the real world. Colorless green ideas sleep furiously!

[2] We make a very important assumption here: Either a thing is in a bag or it is not: We cannot put something into a set if it's already in there. Schrödinger's cat is not both in the set of living things and not. It is just that Schrödingers cat and Not-Schrödingers-cat are in both the sets of living things and not-living things—until one finds out that either the living cat exists or the dead one.

Interlude: *a* or *b* or neither and {*F*|*K*|*M*} *oo*

Yet, there is a big problem: Consider the bag

$$\{a, b\}.$$

What does it mean to take away b? Or, what is *left* after we took away b?[3] There are two possible answers:

$$\text{takeaway}(b, \{a, b\}) = \begin{cases} \{a\}, & \text{if removal implies non-existence} \\ \{a, \ \}, & \text{if removal implies disappearance.} \end{cases} \tag{1}$$

But what happens if we want to take away something that is not in there—may it exist or not? The dual method is to postulate that an empty bag is a bag where no-one ever has put anything into it. It requires us to postulate that all bags are initially empty.

In order to avoid confusion for the remainder of the paper, we shall use the following terms: For any set x, we say that $\texttt{koo}(x)$ is true, iff x is empty:

$$\texttt{koo}(x) = \mathbf{1} \quad :\Longleftrightarrow \quad x = \{\}. \tag{2}$$

In other words, "koo" denotes the unique property of empty sets and is called "emptiness".

In contrast to this, "moo" is a property of things that do not exist. A thing is "moo", if it is not there—yet we can speak of it as we can name it:

$$\texttt{moo}(x) = \mathbf{1} \quad :\Longleftrightarrow \quad |\text{putinto}(x, y)| = |y| \tag{3}$$

Mooish things do not exist, but they do well exist in our universe of reference. They all are "no-things" and the property they share is no-thingness.

Finally, "foo" is a function that for a given thing x delivers its counterpart: for an existing thing x it delivers it's mooish version and vice versa:

$$\texttt{foo}(x) \quad :\longleftrightarrow \quad \begin{cases} x' \text{ with } \texttt{moo}(x') = \mathbf{1}, & \text{if } \texttt{moo}(x) = \mathbf{0} \\ x' \text{ with } \texttt{moo}(x') = \mathbf{0}, & \text{if } \texttt{moo}(x) = \mathbf{1} \end{cases} \tag{4}$$

From our definitions it follows immediately that

$$|\mathit{Koo}| \ = |\{x : \texttt{koo}(x) = \mathbf{1}\}| = |\{\{\}\}| = 1 \tag{5}$$

$$|\mathit{Moo}| = |\{x : \texttt{moo}(x) = \mathbf{1}\}| = \infty \tag{6}$$

$$|\mathit{Foo}_s| = |\{x : \forall x : x \in s \longrightarrow \texttt{foo}(x)\}| = \infty. \tag{7}$$

This means that there is exactly one empty set in our world: there is something that we call empty and there is only one such thing. From this we gain the definition of "1": it is the property of all sets that have the same number of elements as the set *Koo*. It also gives us an idea of the meaning of "∞": it is the number of elements of a set that remains the same when putting something (truly) additional into it. Since there is just one thing of which we know that it exists (namely {}), and since the bag containing all such things therefore has exactly one element (*Koo*), we can safely add it to the set thereby increasing its cardinality by 1:

$$\infty + 1 = \infty \tag{8}$$

[We spent a lot of research to resolve the question whether there is a corresponding `boo`*. It was when we discovered that a Japanese Kanji writing of "Moo" is pronounced "Boo" in some Chinese dialects (but also as "Woo") that we wondered why Newtral chose these function and set names. The Kanji symbol for "Moo" (or Boo or Woo) itself has, surprisingly, the meaning of "nothing" or a "negligible set of things". The fact that "Foo" and "Koo" phonetically resemble the meanings of a negating prefix ("non-...") and void or vacuum is, of course, pure incident.—The editors.]*

Between $[-\infty, 3]$ and $[3, \infty]$.

Nothing is what is contained in *Koo*. This is what reminds us of the definition of a hole:

A *hole* is nothing with something around it.

[3] To avoid confusion please recall that "," is just a syntactical means to distinguish one thing *ab* from two things a, b.

Nothing is trivially written as , and something around (and enclosing) it means (to draw) something around it (i.e., something around). Traditionally, in a nicely curved and rounded style, this is depicted as 0 rather than . This is perfectly right, since 0 is a symbol that we talk about and, therefore, it is there—no matter whether its meaning actually exists ($\mathtt{moo}(0) = \mathbf{1}$ or $\mathtt{moo}(0) = \mathbf{0}$ both imply the existence of their counterparts $\mathtt{foo}(0) = 0'$). This, finally, justifies the notation of

$$\{\}$$

as a hole and therefore the set with nothing in it.[4]

The symbol "0" is commonly pronounced as "zero", sometimes also as "nil" or "nul" or "naught". Whether this explains the common sense meaning of the adjective "naughty" for a wicked problem arising from talking about nothing or making much ado about nothing (i.e. common sense romantic naughtiness) is left as an exercise for the interested reader.

`NULL` is also used as a name for a pointer into the void where a pointer is nothing else than the interpretation of the value of a value: the assignment

```
p=NULL;
```

means that henceforth `p` means nothing: evaluating the pointer (i.e. referring to the object to which it points to) yields , which, in turn is written as "0"; in other words, `p` is now a pointless pointer. `NIL` also denotes the *empty list of things*:[5].

$$[\quad\quad] = (\quad\quad).$$

Clearly, $|[\quad]| = |[\quad\quad]| = 0$, but is $[\quad] = [\quad\quad]$?

Foor. Talking about "”

The word "zero" has an interesting etymology: It originates from "zefiro". It was one of Fibonacci's great accomplishments that he brought nothing with him from Africa, where he was raised in his early childhood: "zefiro" is just the medieval attempt of using an already known word meaning "a wind blowing from the west" whose pronunciation came closest to the Arabic word for just nothing (transliterated "*sfr*"). First, it is noteworthy that the west wind is dominant for the spring climate in the Mediterranean and that it is the wind of *least* force bringing about the blooming and, hence, all life. The wind was called after the Greek god $Ζεφυρος$ who, in turn, is known for his rather naught–y lifestyle.

The word (when written *sfr*) has several meanings according to the various forms it may appear in: As a noun, it refers to the second month of the lunar calendar which has the property of beginning with *new moon*—i.e., the very day on which the calendar itself is based is not present in the sense that it is not visible and therefore has no sign. It, too, could be written as , since we can't see it. As a verb, *sfr* has several similar meanings with slightly different connotations: "to empty" means to take something out of something, "to vacate" means to leave, "to (e)vacuate" means to remove everything leaving nothing, i.e. a void or vacuum as a state of deprivement, and finally, "to free" means to get rid of something. All of these meanings have their counterparts when constructing the perfect form of the verb: we have the emptied, the left (over), the evacuated and the freed.

Yet it is not as simple to speak about the void as it is to speak about a hole: The latter is nothing with something around it, but the former is simply nothing. So evacuating a bottle results in an empty bottle—but not in a bottle containing a vacuum (for the vacuum does not exist: it is what is *inside* the empty set):

$$\mathtt{woo}(x) = \mathbf{1} \quad :\Longleftrightarrow \quad \forall s : x \in s \longrightarrow \mathtt{koo}(s). \tag{9}$$

The vacuum, so to say, is the only thing that does not exist, it is a thing that negates existence in the sense that one cannot even say "there is a vacuum inside the bottle". It has been evacuated and now there is nothing left over so there is nothing in it any more so it's empty. It is interesting to spend another thought on the meaning of "freedom": it is the former content of the bottle that is freed from its surroundings. Also, the volume of the bottle is free from some kind of physical restriction. A bottle of wine is not as much a bottle as an empty bottle because it is determined by its content whereas an empty bottle is a bottle with the potential of keeping anything (as long as it fits through the neck and into the volume).

Sfr itself was imported to the African-Arabic world from Asia. In Sanskrit, the adjective *sunya* means "nothing". Its corresponding noun *sunyata* means emptiness or voidness on the one hand but also describes ultimate openness or unlimitedness. Now we find two seemingly dual concepts to be covered under one and the same name. But again, emptiness on the one hand and unlimitedness on the other nicely correspond to the ideas of vacuum and ultimate freedom.

[4] An interesting point beyond the scope of this article is how to measure different kinds of holes. Undoubtedly, there are small holes, big holes, deep holes and shallow ones, or even black holes. One might, for example, ask, whether a big hole is just the same as a small hole with less around or whether a small hole is a big hole with less nothing or even less than nothing in it.

[5] Both in Prolog and LISP notation.

Sunyata is a fundamental term in Buddhism. Buddhism spread all over Asia, first to China and then, via Korea, to Japan. Over time many flavours emerged but they all share the fundamental concept of nothing.

From many sutras we can learn that we can't tell the difference between a thing that is not there and a thing that we can't perceive. The problem (or rather *the* problem that causes all the trouble) is that nothing is just a thing that is not there and even if it were, we couldn't perceive it. Just look at the following . Is there a or is it that you just can't perceive the gap? Either way we seem to be speaking about it, so it is moo. But if we put into a bag, we have

$$\{\quad\} \stackrel{?}{=} \{\}.$$

Imagine $\{\quad\} = \{\}$. Then, $\{\quad\} \in \mathit{Koo}$, and, hence, $\mathtt{koo}(\{\quad\}) = \mathtt{koo}(\{\})$. So far, so good, but let us take a copy of each set and put it into the other one:

$$A := \mathrm{putinto}(\{\}, \{\quad\}) = \{\quad, \{\}\} \tag{10}$$

$$B := \mathrm{putinto}(\{\quad\}, \{\}) = \{\{\quad\}\} \tag{11}$$

They do not only look different, they are: Since we do not know anything about the existence of , we do not know the cardinality of $\{\quad\}$ (except that it is less than two). Hence the set A may contain 1 or two elements whereas B contains exactly one element and no more and no less. The only difference is whether the set contained in B contains something—but that is not the question. The consequence is that gaps like are something different than gaps which are not there (it is easy to see that $\neq$). As a result, we have a contradiction to our initial assumption. We conclude that

$$\{\quad\} \neq \{\}. \tag{12}$$

V. Void

So not-being is different from being-nothing: it's just that we cannot perceive the difference between an absent being and a (present) not-being. This problem arises from the simple duality inherent to our understanding of the world: *Something is or it is not*. Obviously, this is *wrong*.

Let us now take a look at *Foo*, *Moo* and *Koo* from a philosophical point of view: Just as we described the difference between the an empty bottle and a bottle of wine regarding their respective potential of representing the "ideal bottle", *Sunyata* interpreted as "emptiness" refers to a state of being or perceiving an entity, whereas *sunya* is simply nothing. For an empty mind (or, say, an empty bottle) there is no difference in not being able to perceive its own content (it contains) or postulating that there is no-thing in it (it "contains" a *vacuum*) or that it does not contain anything (all it contains are *moo*-ish things).

The problem of discriminating between all these different concepts only arises by the mere attempt to talk and reason about it.

IIIIX. Independence and relating nothing

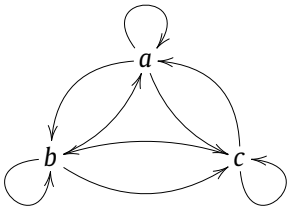
We now come to one interesting part that I do hope will be topic of future research as it concerns not only the very core of mathematics but also every science and, if I might add, the entire life of beings altogether. It is the question of whether nothing can be related to nothing else and, if so, whether it is the same as not relating some-things or no-things.

[The author of the manuscript has found posthumous appreciation—even if not directly influenced by his manuscript as it is presented here, but by the individual research of such great mathematicians as Frank Harary, Ronald Read and, of course, Gunther Schmidt.—The editors.]

Let us consider two sets A, B and C. Imagine that $\mathtt{koo}(C) = \mathbf{1}$, i.e. $C = \{\}$. Similarly, imagine that $A = \{a, b, c\}$ with all a, b, c not being moo. And finally, assume that $B = \{\mathtt{foo}(x) : x \in A\}$.

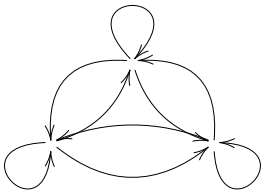
We now consider the relation $\top\!\!\top_A := A \times A$. Of course, it is the universal relation on A:

	a	b	c
a	1	1	1
b	1	1	1
c	1	1	1



What does $\top\!\!\top_B = B \times B$ look like? It relates for all x in A the non-existing counterparts $\mathtt{foo}$. It becomes what is known as a rather pointless argument: It relates nothings and makes a whole lot ado about nothing:

1	1	1
1	1	1
1	1	1



As one can see, the graph of $\top\!\!\top_B$ is *not empty*; it is not $\mathtt{koo}$-ish—it's just that is seems not to have any nodes. This is hard to be put into a formula, because $G_{\top\!\!\top_B} = \langle B, \top\!\!\top_B \rangle$ appears to be

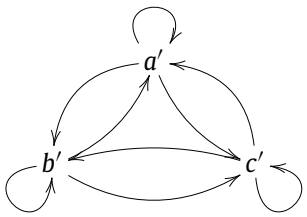
$$\langle \{\}, \{\langle\rangle, \langle\rangle, \langle\rangle, \langle\rangle, \langle\rangle, \langle\rangle, \langle\rangle, \langle\rangle, \langle\rangle\} \rangle. \tag{13}$$

Since elements of a set occur only once, this would be the same as

$$\langle \{\}, \{\langle\rangle\} \rangle. \tag{14}$$

So, a relation on an empty domain has at least one element! On the other hand, $\top\!\!\top_B$ is a subset of $B \times B$ and B appears to be quite empty. So how can a set with at least one element in it be a subset of an empty set? The answer is simple: It can well be, if all the elements do not exist. Therefore, let us formally precisely redefine $\top\!\!\top_B$, and we shall see that all cumbersome side-effects will disappear:

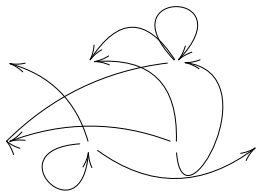
	$\mathtt{foo}(a)$	$\mathtt{foo}(b)$	$\mathtt{foo}(c)$
$\mathtt{foo}(a)$	1	1	1
$\mathtt{foo}(b)$	1	1	1
$\mathtt{foo}(c)$	1	1	1



where all $x' \in \mathtt{Moo}$ and $x' = \mathtt{foo}(x)$.

Obviously, the graph that we achieve by $\mathtt{foo}$ing $\top\!\!\top_B$, is isomorphic to the original one, and we even have that $x'R'y' \iff xRy$. Also, by using $\mathtt{oof} := \mathtt{foo}^{\smile}$ we call the result of applying the existential negation $\mathtt{foo}$ the graph G_{oof}. The problem is that by definition of $\mathtt{foo}$ and $\mathtt{moo}$, the set G_{oof} contains only non-existing things which means that B's cardinality is 0.

To be precise, the according matrix and graph representations then are



where the wild arrangement of arrows is just to indicate that there are many arrows with unknown domain and codomain. But still, everyone would agree that this graph is different from the graph

.

Hence, a relation between non-existing things is not necessarily an empty relation.

Let us, finally, consider $\top_C$. It is $\{\} \times \{\}$ and the arrows in there are the arrows connecting x and y for which $\mathtt{koo}(x) = \mathtt{koo}(y) = \mathbf{1}$, i.e. $x, y \in \{\}$. Trivially, there are no such arrows—and, obviously, $\{\}$ does not contain anything either. So its proper graph representation actually is

.

But since $\top_C$ connects *all* such entities that are in $\{\}$ the relation matrix becomes

$$\begin{array}{c|c} & \cdots \\ \hline 0 & \cdots \\ \vdots & \ddots \\ \vdots & \end{array} .$$

which is a possibly infinite matrix with 0s denoting that nothing is connected to anything but height and width, since there is just one koo.

We conclude:

$$G_A = \langle A, A \times A \rangle = \langle \{a, b, c\}, \{\langle x, y \rangle : x, y \in \{a, b, c\}\} \rangle \tag{15}$$

$$G_B = \langle \{\}, \{\} \times \{\} \rangle = \langle \{\}, \{\langle x, y \rangle : x, y \notin A\} \rangle \tag{16}$$

$$G_C = \langle \{\}, \{\} \rangle \tag{17}$$

Therefore, relating nothing is similar to speaking about nothing: A comes into existence by its naming. Hence, a relation between nothings is *different* from all the ways one *could* relate nothing. So there is a difference between a graph without nodes, a graph without edges and a graph without both and an empty graph: The latter one is the "purest" graph in the sense that it does not contain points, no edges and, most importantly, no intention of relating anything with anything else. The only consequence of this is the following: The distinction becomes pointless as soon as we do not distinguish between non-existent and non-perceivable points and the lesson learned is that all those problems are simply a-void-able by not trying to distinguish.

[With these remarks, Newtral unknowingly also sheds some further light on the notion of an unsharp relational product, a topic that has been tackled by Gunter Schmidt from various angles: relating nothing to something that itself is related to nothing seems to be properly more than relating nothing to nothing.—The editors.]

|World miracles|. Not the end.

... in which we shall deal with nothing
and the question whether this is would be the same
as not dealing with something.

Having an "empty mind" (Mu shin) means to have no self (for the self is just a creation of something having not an empty mind: "it" starts thinking and as such is able to recognise itself).

Cogito ergo sum

is one common phrase trying to grasp this idea. Another one is that

Being no-one

is the only way to perceive the world as it really is: If we are, there is also some "I" that perceives the world. However, how should "I" perceive myself without having my perception of myself being influenced by I's perception process? Therefore, it is impossible to have a reliable image of the real world unless the perception is *transparent*. But then, again, if it *really* is (absolutely) transparent, it is not perceivable. And that means, that some "I" would be incapable of perceiving itself. "I" is always deceived by its being. And a true "I" is there only if it is empty—just as an empty bottle is a true bottle (in contrast to a bottle of wine (or a bottle of Klein, for which the emptiness problem is even more tricky)).

Trying to give "I" a meaning means to put some sense into it. This again leads us to two famous western philosophers: Heidegger and Wittgenstein.

Heidegger stated that "Dasein" is the state of mind of a being that realizes its thrownness into the world. The only thing one can say for sure about conscious beings is that they know about their existence as a result of being thrown into the world. From this there follows the fear of being taken away from the world again—i.e., to die. As an immediate consequence, every such being tries to find some comfort and it does so by trying to find a sense or reason or justification for its being there. And this is, where, by trying to make life bearable, all the trouble actually starts: Bottles try to give themselves (among all others) a justification by holding something, e.g. some wine, and in this process becoming a bottle-of-wine. This

is a strong limitation (and alienation) from a bottle's true nature as being something that *can* hold many different things (unless it is filled).

Wittgenstein made many attempts to understand Human language and its meaning. It would be far beyond the scope of this small paper to give a satisfying summary. Therefore, we shall just conclude with one famous quote by him:

The meaning of a word is its use.

So, the meaning of "nothing" is: . And since we cannot speak of nothing without speaking, the story about nothing better remains untold:

[At this point the original manuscript abruptly ends. However, it continues in the sense that there are 357 more sheets of paper attached. After several years of intensive research, the editors are still unable to determine whether these pages are empty or whether nothing has been written on them.]