Implication Structures

— Draft —

Gunther Schmidt

Institute for Software Technology, Department of Computing Science
Federal Armed Forces University Munich, 85577 Neubiberg
e-Mail: Schmidt@Informatik.UniBw-Muenchen.DE

Abstract

This is a case study using the proposed relational multilevel reference language in order to deal with the interesting logical concept of implication structures, developed already in [SS74,SS76]. In the course of these investigations, the timetable problem is introduced on a componentfree relational level by reworking the papers mentioned before, thus providing an important example of an implication structure.

An implication structure is given when a choice of a subset has to be made from a set of items where an item chosen may imply/forbid some others to be chosen. Also not choosing an item may enforce another one to be chosen. The problem is related to satisfiability. It is, thus, NP-complete, and will normally not admit an efficient algorithm. When studying the implication structure mentioned from the relation-algebraic side, this may result in theoretically sound heuristical approaches.

1 Introduction

We use the proposed relational multilevel reference language [Sch03] to rework [SS74,SS76]. The papers of 1974 had originally been motivated by timetable construction. Timetables in turn constitute models of the more general logical concept of an implication structure.

Considering timetable construction today, one may say that it is NP-complete. There are sub-aspects that may be handled efficiently using assignment procedures. Others are related to satisfiability. While 3-satisfiability is NP-complete, only a 2-hour-timetable corresponds to a 2-satisfiability problem and may be solved efficiently.

Cooperation and communication around this research was partly sponsored by the European COST Action 274: TARSKI (Theory and Applications of Relational Structures as Knowledge Instruments), which is gratefully acknowledged.

In constructing timetables, a lot of implications occur. If a teacher t is assigned to a timeslot h with class c, he should not be assigned for another class c' at the same time. If a professor is about to teach a double lesson, assignment of this lesson to 9h implies that the second part of it be assigned to 10h. If in constructing a timetable, some lecture hall is filled up to a last timeslot, and there is just one of the lectures left planned to take place there, this must now be assigned to the remaining slot, etc.

Finally, sometimes some sort of a complementary enforcing may take place. Not assigning a lesson to some timeslot may enforce that the only other lesson available for that slot must be assigned as otherwise it cannot be accomodated. In total, we find three relations on the assignment possibilities which we call E enforcing, F forbidding, and C counter enforcing.

2 Implication Structures

We assume an implication situation without referring to timetables in this section. Then we have a set of items that may *imply (enforce), forbid, or counterimply* one another. The task is to select a subset such that all the given postulates are satisfied.

In order to model this, let a base set N be given. What we are looking for are subsets $s \subseteq N$ satisfying whatever has been demanded as implication concerning two elements $i, k \in N$:

$$s_i \rightarrow s_k, \qquad s_i \rightarrow \neg s_k, \qquad \neg s_i \rightarrow s_k$$

Subsets s are here conceived as boolean vectors $s \in \mathbb{B}^N$. Therefore, s_i is shorthand for $i \in s$. Enforcing, forbidding, and counter-enforcing are conceived to be given as relations $E, F, C \subseteq N \times N$.

An arbitrary subset may either satisfy the implicational requirements or may not. We are usually not interested in *all* solutions, much in the same way as one timetable satisfying formulated requirements will suffice. For theoretical investigation, we consider the set $S \subseteq \mathcal{P}(N)$ of all subsets fulfilling the given postulates, the possible solutions for the postulated set of implications. They satisfy, thus,

$$\forall s \in S: \quad s_i \rightarrow \quad s_k \qquad \text{if } (i,k) \in E \qquad (*)$$

$$\forall s \in S: \quad s_i \to \neg s_k \qquad \text{if } (i,k) \in F \tag{†}$$
$$\forall s \in S: \neg s_i \to \quad s_k \qquad \text{if } (i,k) \in C \tag{‡}$$

We assume for the moment three relations E, F, C to be arbitrarily given. Our aim is to conceive the relations as some implication structure and to look for the underlying set of solutions. To this end, we define

$$(E,F,C) \mapsto \sigma(E,F,C) := \{v \mid E;\overline{v} \subseteq \overline{v}, \quad F;v \subseteq \overline{v}, \quad C;\overline{v} \subseteq v\}$$

as the transition to the set of solutions of the triple E, F, C.

We may, however, also start with any set S of subsets of N and ask whether it is a solution of some triple of implication relations. Then we define as transition to the triple of implication relations of S

$$S \mapsto \pi(S) := (\mathtt{inf}_{s\in S}\{\overline{s;\overline{s^\mathsf{T}}}\}, \mathtt{inf}_{s\in S}\{\overline{s;s^\mathsf{T}}\}, \mathtt{inf}_{s\in S}\{\overline{\overline{s};\overline{s^\mathsf{T}}}\})$$

2.1 Theorem. The two functionals σ, π form a Galois connection between subsets $S \subseteq \mathcal{P}(N)$ and relation triples E, F, C on N.

Proof: We exhibit that

$$\left\{\begin{array}{lll} E & \subseteq & \pi(S)_1 \text{ and} \\ F & \subseteq & \pi(S)_2 \text{ and} \\ C & \subseteq & \pi(S)_3 \end{array}\right\} \iff \quad S \subseteq \sigma(E,F,C)$$

We start from $E \subseteq \pi(S)_1 = \mathtt{inf}_{s\in S}\,\overline{s;\overline{s^\mathsf{T}}}$ which implies that we have $E \subseteq \overline{s;\overline{s^\mathsf{T}}}$ for all $s \in S$. Negating results in $s;\overline{s^\mathsf{T}} \subseteq \overline{E}$ for all s. Using Schröder's rule, we get $E;\overline{s} \subseteq \overline{s}$ for all s and, thus, the first condition in forming $\sigma(E,F,C)$. The other two cases are handled in the same way.

Now, we work in the reverse direction, assuming
$S \subseteq \sigma(E,F,C) = \{v \mid E;\overline{v} \subseteq \overline{v}, \quad F;v \subseteq \overline{v}, \quad C;\overline{v} \subseteq v\}$.
This means that we have $E;\overline{s} \subseteq \overline{s}, F;s \subseteq \overline{s}, C;\overline{s} \subseteq s$ for every $s \in S$. The negations and the Schröder steps taken before, had been equivalences, and may, thus, be reversed. This means that for all $s \in S$ we have $E \subseteq \overline{s;\overline{s^\mathsf{T}}}$, $F \subseteq \overline{s;s^\mathsf{T}}$, $C \subseteq \overline{\overline{s};\overline{s^\mathsf{T}}}$. In this way, we see that E, F, C stay below the infima. □

It is then straightforward to prove all the results that follow simply by Galois folklore. In particular we study $\varphi(S) := \sigma(\pi(S))$ and $\rho(E,F,C) := \pi(\sigma(E,F,C))$.

- φ and ρ are expanding, i.e., $E \subseteq \pi_1(\sigma(E,F,C))$, $F \subseteq \pi_2(\sigma(E,F,C))$, $C \subseteq \pi_3(\sigma(E,F,C))$, and in addition $S \subseteq \sigma(\pi(S))$ for all E, F, C and S
- φ and ρ are idempotent, i.e., $(E,F,C) = \rho(\rho(E,F,C))$, and in addition $S = \varphi(\varphi(S))$ for all E, F, C and S.
- ρ, φ are monotonic and, thus, closure operations.
- There exist fixedpoints for ρ, φ.
- The fixedpoint sets with regard to ρ, φ are mapped antitonely onto one another.

2.2 Theorem. All the fixedpoints of the Galois connection satisfy
i) $F = F^{\intercal\intercal}, \quad C = C^{\intercal\intercal}$
ii) $\mathbb{I} \subseteq E = E^2$
iii) $E ; F = F, \quad C ; E = C$
iv) $F ; C \subseteq E$

Proof: We immediately see that the second as well as the third component of $\pi(S)$ are symmetric by definition. $\mathbb{I} \subseteq E$ as obviously $\mathbb{I} \subseteq \overline{s ; \overline{s^\intercal}}$ for all s.

Transitivity of E follows as $\overline{s ; \overline{s^\intercal}} ; \overline{s ; \overline{s^\intercal}} \subseteq \overline{s ; \overline{s^\intercal}}$ and since transitivity is $\cap$-hereditary. Similarly, for F and C in (iii) and for (iv). □

The enforcing relation E is, thus, a preorder. While these properties concerned non-negated implications relations, there are others including also negated ones.

2.3 Corollary. The fixedpoints of the Galois connection satisfy in addition
i) $E ; \overline{C} = \overline{C}, \quad \overline{F} ; E = \overline{F}$
ii) $C ; \overline{C} \subseteq \overline{E}, \quad \overline{E} ; C \subseteq \overline{F}$
iii) $\overline{F} ; F \subseteq \overline{E}, \quad F ; \overline{E} \subseteq \overline{C}$ □

Now we consider closure forming

$S \mapsto \varphi(S)$ and $(E,F,C) \mapsto \rho(E,F,C)$

from a computational point of view. While it seems possible to handle three $n \times n$-matrices E, F, C for rather big numbers n in a relation-algebraic way with RELVIEW, it will soon become extremely difficult to determine vectors of length n satisfying certain given implications. While we do not see enough mathematical structure on the S-side, the formulae just proved on the (E,F,C)-side may be helpful.

Therefore, we try to determine for given elementary implication matrices (E,F,C) their implication closure $\rho(E,F,C)$. In case there exist very long chains of implications, it may well be the case that the closure $\rho(E,F,C)$ makes it easier to determine a solution S by applying the following concepts.

The structure of the definition of implication relations leads us to call $i \in N$

a *tight* element with respect to S if $C_S(i,i) = \mathbf{1}$,
a *pseudo* element if $F_S(i,i) = \mathbf{1}$,
otherwise it is called a *flexible* element.

The names are derived from the instantiations of $(\dagger, \ddagger)$ for $k := i$:

$\forall s \in S: \quad s_i \to \neg s_i \quad$ if $(i,i) \in F_S$, meaning $\quad \forall s \in S : \neg s_i$
$\forall s \in S: \neg s_i \to \quad s_i \quad$ if $(i,i) \in C_S$, meaning $\quad \forall s \in S : s_i$

So, for every $\mathbf{1}$ in the diagonal of C, the corresponding element must and for every $\mathbf{1}$ in the diagonal of F, it must not belong to *any* solution S. These facts *together with all their implications* according to E, F, C may be helpful in looking for solutions S.

Implication matrices are only interesting modulo the equivalence $E \cap E^\mathsf{T}$ derived from the preorder E. It is an easy consequence that by simultaneous permutation of rows and columns every triple of implication matrices may be arranged in the following standard form:

$$E = \begin{pmatrix} \top & \top & \top \\ \bot & E_0 & \top \\ \bot & \bot & \top \end{pmatrix}, \; F = \begin{pmatrix} \top & \top & \top \\ \top & F_0 & \bot \\ \top & \bot & \bot \end{pmatrix}, \; C = \begin{pmatrix} \bot & \bot & \top \\ \bot & C_0 & \top \\ \top & \top & \top \end{pmatrix}$$

Tight elements are here positioned in the first group of rows, followed by flexible ones, and pseudo elements. E_0, F_0, C_0 satisfy some additional rules.

There is one further idea. E is a preorder according to Thm. 2.2. One may ask which influence it has for the necessarily heuristic algorithm when one chooses minimal/maximal elements to be handled first. Selecting minimal elements with regard to E first, makes fundamental decisions early in a backtracking algorithm. It may, however, also be wise, to assign maximal elements first. Then some freedom is still left to assign the others — and to fit to criteria not yet formalized.

Considering Thm. 2.2, one will easily suggest to apply round-robin-wise the following steps until a stable situation is reached:

- determine the reflexive-transive closure of E
- determine the symmetric closure of F and C
- expand F to $E;F$ and C to $C;E$
- add $F;C$ to E

3 Implication Structures in Haskell

The following is written using the relational language proposed in [Sch03].

```
implicationTheory =
   let baseSet   = OC $ CstO "BaseSet"
       subsetSet = OC $ CstO "SubsetSet"
       eMat  = Rela "Given E" baseSet baseSet
       fMat  = Rela "Given F" baseSet baseSet
       cMat  = Rela "Given C" baseSet baseSet
       vectorSet = Rela "Vectors as relation columns" baseSet subsetSet
       matr = vectSetToEFC (RC vectorSet)
       form1 = RF $ fst3 matr :>==: (RC eMat)
       form2 = RF $ snd3 matr :>==: (RC fMat)
       form3 = RF $ thd3 matr :>==: (RC cMat)
       form4 = RF $ matrixToVectorSet (RC eMat) (RC fMat) (RC cMat)
                          :===: (RC vectorSet)
   in  TH "Implication Theory"
          [baseSet,subsetSet]         -- objects
          [] []                       -- element and vector constants
          [eMat,fMat,cMat,vectorSet]  -- relation constants
          [] [] [] [form1,form2,form3,form4]
```

For small examples one may calculate as follows:

```
vectSetToEFC vs =
   let src = domRT vs
       tgt = codRT vs
```

```
       ev = VarE "x" tgt
       v  = RC vectorSet :****: (PointVect $ EV ev)
       fctlElemToRelaE = RFCT (EVar ev) (       v  :||--: (NegaV v))
       eeeRelaSET = RT fctlElemToRelaE (ET tgt)
       fctlElemToRelaF = RFCT (EVar ev) (       v  :||--:        v)
       fffRelaSET = RT fctlElemToRelaF (ET tgt)
       fctlElemToRelaC = RFCT (EVar ev) ((NegaV v) :||--: (NegaV v))
       cccRelaSET = RT fctlElemToRelaC (ET tgt)
       e = NegaR $ SupRela eeeRelaSET
       f = NegaR $ SupRela fffRelaSET
       c = NegaR $ SupRela cccRelaSET
   in  (e,f,c)
matrixToVectorSet e f c =
   let epsi     = Epsi (dom e)
       epsiNeg  = NegaR epsi
       unit     = UnivR UnitOb (dom e)
       unitVect = UnivV UnitOb
       filterMatr =
          (unit :***: (e :***: epsiNeg :&&&: epsi   ))  :|||:
          (unit :***: (f :***:   epsi  :&&&: epsi   ))  :|||:
          (unit :***: (c :***: epsiNeg :&&&: epsiNeg))
       filterVect = Convs (NegaR filterMatr) :****: unitVect
       subsets = InjTerm filterVect :***: (Convs epsi)
   in Convs subsets
```

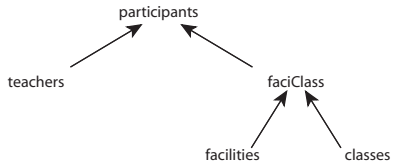
4 Theory of Timetables

We define a timetable problem using the relational language directly. There are given participants, which may be persons, school classes, lecture rooms, technical facilities, etc.

```
teachers     = OC $ CstO "Teaching personal"
classes      = OC $ CstO "School classes"
facilities   = OC $ CstO "Facilities"
faciClass    = DirSum facilities classes
participants = DirSum teachers   faciClass
```

Using the direct sum construct, we will be able to interpret with small matrices via the injections we introduce here.

```
iotaFC   = Iota  facilities classes
kappaFC  = Kappa facilities classes
iotaTFC  = Iota  teachers   faciClass
kappaTFC = Kappa teachers   faciClass
```



It must be prescribed who has to join for a lesson, e.g. We call such a get-together a *meet* and list which set of participants has to meet. There exists a set of hours or time slots at which participants may be available and to which meets shall be assigned.

```
meets = OC $ CstO "Meets to be assigned"
hours = OC $ CstO "Possible hours"

meetTeacher    = Rela "Who teaches this lesson"          meets teachers
meetClass      = Rela "Which classes attend this lesson" meets classes
meetFacilities = Rela "Which beamer for this lesson"     meets facilities
```

The relation between meets and participants may again be constructed with injections.

```
meetParticipants =
   RC meetTeacher    :***: iotaTFC                   :|||:
  (RC meetFacilities :***: iotaFC  :***: kappaTFC)   :|||:
  (RC meetClass      :***: kappaFC :***: kappaTFC)
```

Now it is discussed who is available at which hour. First a relation constant is provided that may be interpreted in a model. When working with it, we will also have the relational term consisting of that constant. From the separate availabilities we build the combined ones.

```
availTeachConst = Rela "Free hours of teachers"   teachers   hours
availTeach      = RC availTeachConst
availClassConst = Rela "Free hours of classes"    classes    hours
availClass      = RC availClassConst
availFaciConst  = Rela "Free hours of facilities" facilities hours
availFaci       = RC availFaciConst
availFaciClass  = Convs iotaFC   :***: availFaci       :|||:
                  (Convs kappaFC  :***: availClass)
availPart       = Convs iotaTFC  :***: availTeach      :|||:
                  (Convs kappaTFC :***: availFaciClass)
```

From school life it is known that some lessons should not be assigned to certain hours regardless of whether the participants are available — no math course at noon, no sports on Monday morning, e.g. Therefore, also meets themselves may be given an availability.

```
availMeetConst    = Rela "free hours of meet" meets hours
availMeet         = RC availMeetConst
commonAvailOfMeet = availMeet :&&&:
   (NegaR $ meetParticipants :***: (NegaR availPart))
```

To solve a timetable problem means to assign meets to hours in a certain way. To this end, we introduce these possible associations as a direct product together with the two projections, their converses and formulate the criteria to be followed.

```
meetHour = DirPro meets hours
piMH     = Pi     meets hours
rhoMH    = Rho    meets hours
piMHT    = Convs  piMH
rhoMHT   = Convs  rhoMH
```

We allow to consider assignment of meets to hours also from another point of view by converting from relation to a vector on the direct product of meets and hours.

```
assignAsVector   as = RelaToVect     as
assignAsRelation v  = ProdVectToRela v
```

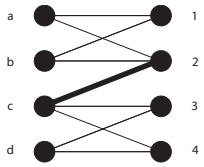
The criteria are a conjunction of three properties. A meet is only assigned if all of its participants are available; every meet is assigned precisely once, and no participant is assigned twice in an hour.

```
isAdmissibleAssignment as = Conjunct
   (Conjunct (RF $ as :<==: commonAvailOfMeet)
             (RF $ Convs as :***: as          :<==: (Ident hours)))
   (RF $ meetParticipants   :***: (Convs meetParticipants)
             :&&&:      (as :***: (Convs as)) :<==: (Ident meets))

isSolution as = Conjunct
   (isAdmissibleAssignment as)
   (RF $ as :***: (UnivR hours UnitOb) :===: (UnivR meets UnitOb))
```

In the process of solving a concrete timetable problem, one will assign meets to hours, i.e., fill the relation `as`. In order that this NP-complete problem be solved, one will have to apply heuristics. These heuristics will mainly consist in some sort of accounting. That is, one will try to immediately update availabilities which are reduced when an assignment has been made. One will also immediately apply the highly efficient mincut analysis. This means that for every

participant all the meets he is involved in will be assigned testwise regardless of other participants. It will efficiently detect, e.g., that the fat assignment must not be used, as then not all of $\{a, b, c, d\}$ can be accomodated.



The result may also be that a full assignment is no longer possible. Then backtracking is necessary. The result may finally be that from mincut analysis alone, it follows that some meet is already tight with regard to the possibilities to assign it. This meet will then be taken to be formally assigned.

The question is, whether some sort of algebraic manipulation can help in a similar way. To this end we prepare formalization of elementary implications. Relation `phi` relates a combination $(meet, hour)$ to another one with a different hour, while `phi0MH` relates it with another meet-hour combination at the same time.

```
phi =    piMH  :***: (Ident meets)         :***: piMHT    :&&&:
        (rhoMH :***: (NegaR $ Ident hours) :***: rhoMHT)
phi0MH = piMH  :***: (NegaR $ Ident meets) :***: piMHT    :&&&:
        (rhoMH :***: (Ident hours)         :***: rhoMHT)
```

The following is a predicate on $(meet, hour)$-combinations saying nothing more than that participant `p` is not involved in the second meet.

```
capitalA p =
   let point = PointVect p
       mask  = Convs $ (NegaV $ meetParticipants :****: point)
                 :||--: UnivV meets
   in  piMH :***: mask :***: piMHT
```

This is now used to formalize basics of forbidding in otrder to start an iteration to obtain an (E, F, C)-closure.

```
phi0 p = phi0MH :&&&: (piMH :***: meetParticipants :***:
   (PointDiag p) :***: (Convs meetParticipants) :***: piMHT)
capitalPhi0 = let phi0Map = RT (RFCT (EVar $ VarE "x" participants)
```

```
                                (phi0 $ EV (VarE "x" participants)))
                                (ET participants)
              in  phi :|||: (SupRela phi0Map)
phiP p = phi0 p :|||: (phi :&&&: (capitalA p))
```

Building on these relations, the timetable theory may now be formulated with `p` running over all participants. It can be proved that an implication structure as been established. Several practical examples show that the implication structure may deliver results even when mincut analysis does no longer produce any. The effort should, however, only be made when the construction comes close to the end, when just a few availabilities are left. Then in particular will the technique produce non-trivial results.

5 Outlook

We have used the multilevel relational reference language to write two concepts down, implication theory and timetable theory. It was then immediately operational, i.e., could be run for moderately sized examples. During this endeavour additional improvements became necessary, not least the possibility to run through the elements of a category object. One should be able to formulate it, but cannot do it prior to some interpretation chosen. So some generic form has been introduced.

References

[Sch03] Gunther Schmidt. Relational Language. Technical Report 2003-05, Fakultät für Informatik, Universität der Bundeswehr München, 2003. 101 pages, `http://ist.unibw-muenchen.de/Inst2/People/schmidt/RelLangHomePage.html`

[SS74] Gunther Schmidt and Thomas Ströhlein. A boolean matrix iteration in timetable construction. Technical Report 7406, Abteilung Mathematik der Technischen Universität München, 1974.

[SS76] Gunther Schmidt and Thomas Ströhlein. A boolean matrix iteration in timetable construction. *Linear Algebra and Its Applications*, 15:27–51, 1976.