Representing School Timetabling in a Disjunctive Logic Programming Language*

Wolfgang Faber† **Nicola Leone** **Gerald Pfeifer**

Institut für Informationssysteme, TU Wien
Paniglgasse 16, A-1040 Wien, Austria
{faber,leone,pfeifer}@dbai.tuwien.ac.at

1 Introduction

Almost everybody has come across school timetabling during his or her life. For a long time almost all school timetables were created manually, which usually took rather long, depending on the size of the school. In the last thirty years, however, systems have been designed which automate timetable creation.

Timetabling problems in general are defined as having to find suitable combinations of two or more types of resources which have to take place during a fixed period of time and which have to satisfy various constraints. Some of these constraints are strict while some are not (the latter express preferences or desiderata).

In the case of school timetabling problems the resources are teachers and classes. These have to meet ("combination") during several fixed time-slots, which are usually called "periods", typically in the scope of one week. The non-strict constraints in this problem are usually preferences which originate from didactical, organisational and personal considerations.

Timetabling problems have been studied in depth for the last few years, mainly in the Artificial Intelligence and Operations Research communities [CDM98, CK96, HW96b, HW96a, Sch95, YKNW94]. A special conference on this topic has been installed [BR96], and one can observe a considerable increase of research papers [Bar96]. [Sch95] gives a comprehensive survey of the developments in this area.

Existing timetabling systems are mostly highly specialised and efficient, but usually the problem encoding is not declarative. Other aproaches use

*Supported by *FWF (Austrian Science Funds)* under the project P11580-MAT "A Query System for Disjunctive Deductive Databases"

†Please address correspondence to this author.

general purpose constraint logic programming systems [HW96b, HW96a], or constraint languages tailored for timetabling [YKNW94]. A special "timetable specification language" has been used for complexity analyses of the problem [CK96]. While some of these languages are capable of expressing the basic assignment problem (i.e., the satisfaction of the strict constraints) in an elegant and declarative way, those languages fail to capture preferences equally well.

In this paper, we show that a suitable extension of disjunctive datalog (function-free disjunctive logic programs) allows us to represent school timetabling with various preferences in a natural and highly declarative way.

In Section 2 we give basic syntax and semantics definitions of the language and its extension, which will then be used to represent school timetabling problems in Section 3. In 3.1 we will first describe the basic assignment problem. In 3.3 we will then augment this representation by various preferences using the same declarative framework.

2 Language Definition

We assume that the reader is familiar with the basic concepts of deductive databases [Ull89, CGT90] and logic programming [Llo84]. The language we will use in this paper is an extension of disjunctive datalog by (strong and weak) constraints. Similar extensions (except for the notion of weights and with slightly different syntax) have been defined in [BLR97b, BLR97a].

2.1 Syntax

A *term* is either a constant or a variable[1]. An *atom* is of the form $\mathtt{a}(\mathtt{t}_1, ..., \mathtt{t}_\mathtt{n})$, where $\mathtt{a}$ is a *predicate* of arity n and $\mathtt{t}_1, ..., \mathtt{t}_\mathtt{n}$ are terms. A *literal* is either a *positive* literal p or a *negative* literal $\mathtt{not}\ p$, where p is an atom.

A *(disjunctive) rule* r is a clause of the form

$$\mathtt{a}_1 \vee \cdots \vee \mathtt{a}_\mathtt{n} \leftarrow \mathtt{b}_1, \cdots, \mathtt{b}_\mathtt{k}, \mathtt{not}\ \mathtt{b}_{\mathtt{k}+1}, \cdots, \mathtt{not}\ \mathtt{b}_\mathtt{m}. \qquad \mathtt{n} \geq 1,\ \mathtt{m} \geq 0$$

where $\mathtt{a}_1, \cdots, \mathtt{a}_\mathtt{n}, \mathtt{b}_1, \cdots, \mathtt{b}_\mathtt{m}$ are atoms.

A *strong constraint* (integrity constraint) is a syntactic of the form $\leftarrow \mathtt{L}_1, \cdots, \mathtt{L}_\mathtt{m}.$, where each $\mathtt{L}_\mathtt{i}$, $1 \leq i \leq m$, is a literal (i.e., it is a rule with empty head).

A *weak constraint* is a syntactic of the form $\Leftarrow \mathtt{L}_1, \cdots, \mathtt{L}_\mathtt{m}.[\mathtt{l} : \mathtt{w}]$, where each L_i, $1 \leq i \leq m$, is a literal and l and w are integer numbers, $l \geq 1, w \geq 0$. l denotes the *priority layer* and w the *weight* among the layer. Both l and w may be omitted, defaulting to 1 and 0, respectively.

[1]Note that function symbols are not part of datalog.

All possible rules, strong and weak constraints comprise the language $\text{DATALOG}^{not,\vee,w}$. A $\text{DATALOG}^{not,\vee,w}$ program $\mathcal{P}$ is a finite subset of the language $\text{DATALOG}^{not,\vee,w}$.

Herbrand models and interpretations are defined in the usual way.

2.2 Semantics

For the constraint-free part of $\text{DATALOG}^{not,\vee,w}$ programs many semantics have been proposed, all which are defined as a subset of the Herbrand Models of the program in question. We refer to [Dix95] for a comprehensive survey of these semantics. In our examples we assume that the models of the chosen semantics are at least minimal w.r.t. set inclusion, which applies to all relevant semantics.

We will refer to the models of a chosen semantics for the constraint-free part of a $\text{DATALOG}^{not,\vee,w}$ program as *candidate models*.

The meaning of the constraints is then defined as follows: A constraint (strong or weak) is *satisfied*, iff at least one of its literals is falsified by the candidate model[2]. Any model of a $\text{DATALOG}^{not,\vee,w}$ program $\mathcal{P}$ must satisfy all strong constraints.

For the weak constraints, only those candidate models are considered which minimise the sum of weights of the violated constraints in the greatest priority layer, and among them those which minimise the sum of weights of the violated constraints in the previous layer, etc. This can be expressed by an objective function for $\mathcal{P}$ and a candidate model M:

$$
\begin{aligned}
&f_{\mathcal{P}}(1) = 1\\
&f_{\mathcal{P}}(n) = f_{\mathcal{P}}(n-1) \cdot WC^{\mathcal{P}} \cdot w_{max}^{\mathcal{P}} + 1, \quad n > 1\\
&H_M^{\mathcal{P}} = \sum_{i=1}^{l_{max}^{\mathcal{P}}} (f_{\mathcal{P}}(i) \cdot \sum_{N \in N_i^{M,\mathcal{P}}} w_N)
\end{aligned}
$$

where $WC^{\mathcal{P}}$ denotes the total number of weak constraints in $\mathcal{P}$, $w_{max}^{\mathcal{P}}$ and $l_{max}^{\mathcal{P}}$ denote the maximum weight and maximum layer of a weak constraint in $\mathcal{P}$, respectively, $N_i^{M,\mathcal{P}}$ denotes the set of weak constraints in layer i which are violated and w_N denotes the weight of the weak constraint N.

All candidate models which satisfy all strong constraints and for which $H_M^{\mathcal{P}}$ is minimal among all candidate models are considered the *preferred models* w.r.t. the candidate model semantics.

[2]This applies to both total and partial semantics.

As an example, consider the following program $\mathcal{P}_s$

$$\mathsf{a} \vee \mathsf{b} \leftarrow \mathsf{c}.$$
$$\mathsf{c}.$$
$$\Leftarrow \mathsf{a}, \mathsf{c}.[2:1]$$
$$\Leftarrow \mathsf{b}.[1:10]$$

The minimal models of this example are $M_1 = \{\mathsf{a}, \mathsf{c}\}$ and $M_2 = \{\mathsf{b}, \mathsf{c}\}$. In this case $f_{\mathcal{P}_s}(1) = 1$, $f_{\mathcal{P}_s}(2) = 21$, $H^{\mathcal{P}_s}_{M_1} = 21$, and $H^{\mathcal{P}}_{M_2} = 10$. So M_2 is preferred over M_1. Indeed, M_1 violates the more important weak constraint.

3 School Timetabling

In the literature several formulations of school timetabling problems exist, the most relevant of which are NP-complete [CK96, Sch95]. It has been recognised that timetabling problems should be regarded as optimisation problems rather than just assignment problems since in practice a timetable need not only be valid but should also satisfy various preferences [Sch95, CDM98].

We will give a modular formulation of such an optimisation problem by first providing a definition for the traditional assignment problem (TTP, for Timetabling Problem) in 3.1 and 3.2, consisting only of rules and strong constraints. Afterwards, in 3.3 and 3.4, we will extend TTP by a hierarchy of desired properties (following [CDM98]) expressed by weak constraints. In the course of this we will have to introduce a few auxiliary predicates in order to express certain properties of a timetable.

3.1 TTP – The Basic NP-complete Problem

The input is a set of m classes $C = \{c_1, \ldots, c_m\}$, a set of n teachers $T = \{t_1, \ldots, t_n\}$, a set of l periods $P = \{p_1, \ldots, p_l\}$, and a set of requirements $R = \{(c_x, t_y, r_{x,y}) \mid c_x \in C, t_y \in T\}$, which state that teacher t_y has to give $r_{x,y}$ lectures to class c_y. In addition, both classes and teachers may be unavailable during certain periods, which is represented as a set of periods for each class ($\forall c \in C : U_c \subseteq P$) and each teacher ($\forall t \in T : U_t \subseteq P$).

The problem now is to assign lectures to periods in such a way that no class and no teacher are involved in more than one lecture during any period, that all of the teaching requirements are met, and that all unavailabilities are honored.

More formally: Find a set $A \subseteq \{(c, t, p) \mid c \in C, t \in T, p \in P\}$, such that:

$$\forall c \in C, p \in P: \quad (c, t_x, p) \in A \wedge (c, t_y, p) \in A \Longrightarrow x = y \tag{1}$$
$$\forall t \in T, p \in P: \quad (c_x, t, p) \in A \wedge (c_y, t, p) \in A \Longrightarrow x = y \tag{2}$$
$$\forall (c_x, t_y, r_{x,y}) \in R: \quad \sum_{p \in P} \chi_A((c_x, t_y, p)) = r_{x,y} \tag{3}$$
$$\forall c \in C, t \in T: \quad p \in U_c \Longrightarrow (c, t, p) \notin A \tag{4}$$
$$\forall c \in C, t \in T: \quad p \in U_t \Longrightarrow (c, t, p) \notin A \tag{5}$$

χ_A is the characteristic function for A, defined in the usual way. This TTP problem is NP-complete [Sch95].

3.2 Formulation of TTP in DATALOG$^{not,\vee,w}$

Now that we have introduced the problem in an abstract way, we can provide a concrete formalisation in our framework.

3.2.1 Representation of Input Data

We assume that all the lectures which have to be given by a teacher to some class are numbered (i.e. the first, second etc. lecture can be distinguished)[3]. So in the input database a fact `r(ci,tj,k)` is stored iff $0 < \mathtt{k} \leq r_{i,j}$ holds, which means that instead of storing the number of lectures we store each lecture seperately.

Further we assume that periods are stored as `p(q)`, if $\mathtt{q} \in P$ and that the relation $<$ is defined over them.

For representing unavailabilities two relations, `class_unavailability/2` and `teacher_unavailability/2`, are assumed to be defined for each pair of class (resp. teacher) and period, in which it (resp. (s)he) is not available.

3.2.2 Problem Formulation

Based on this representation we can now write a program that will compute all possible timetables.

Rule (6) guesses all subsets of possible assignments. Constraint (7) ensures that each lecture is not given more than once[4] Rule (8) and constraint (9) ensure that each lecture is given at least once. Constraint (10) states that lectures which differ only by their number must not be given during the same period.

There is still a problem: Due to our representation involving the number of the lecture there are several equivalent solutions. Constraint (11) enforces that lectures are planned in the order indicated by their number.

[3] If we assume that metapredicates are available, this may be simplified.

[4] We consider $\neq$ and $<$ built-in predicates, which are easy to support for any system.

$$\mathtt{a(C,T,P,K)} \vee \mathtt{na(C,T,P,K)} \leftarrow \mathtt{p(P)}, \mathtt{r(C,T,K)}., \tag{6}$$

$$\leftarrow \mathtt{a(C,T,P1,K)}, \mathtt{a(C,T,P,K)}, \mathtt{P} \neq \mathtt{P1}., \tag{7}$$

$$\mathtt{planned(C,T,K)} \leftarrow \mathtt{a(C,T,P,K)}., \tag{8}$$

$$\leftarrow \mathtt{r(C,T,K)}, \mathtt{not\ planned(C,T,K)}., \tag{9}$$

$$\leftarrow \mathtt{a(C,T,P,K)}, \mathtt{a(C,T,P,K1)}, \mathtt{K} \neq \mathtt{K1}., \tag{10}$$

$$\leftarrow \mathtt{a(C,T,P,K)}, \mathtt{a(C,T,P1,K1)}, \mathtt{P} < \mathtt{P1}, \mathtt{K1} < \mathtt{K}., \tag{11}$$

$$\leftarrow \mathtt{a(C,T1,P,K1)}, \mathtt{a(C,T2,P,K2)}, \mathtt{T1} \neq \mathtt{T2}., \tag{12}$$

$$\leftarrow \mathtt{a(C1,T,P,K1)}, \mathtt{a(C2,T,P,K2)}, \mathtt{C1} \neq \mathtt{C2}., \tag{13}$$

$$\leftarrow \mathtt{class\_unavailability(C,P)}, \mathtt{a(C,T,P,K)}., \tag{14}$$

$$\leftarrow \mathtt{teacher\_unavailability(T,P)}, \mathtt{a(C,T,P,K)}. \tag{15}$$

Figure 1: The program for the basic problem TTP

Up to now we have modelled condition 3 from above. Conditions 1 and 2 can be implemented more directly by (12) and (13), respectively. In a similar fashion conditions 4 and 5 can be formulated by (14) and (15).

3.3 TTOP – Optimisation Problem Extending the Basic Problem

While unavailabilities are constraints which have to be strictly met, there usually are a number of additional, more or less desirable constraints.

[Sch95] states several approaches to express such desiderata. One of the most recent, practical and flexible formulations is described in [CDM98]. In this article several groups of desired properties are identified:

- Feasibility Conditions σ
- Didactic Requirements Δ
- Organisational Requirements Ω
- Teacher Preferences Π

σ is basically what we have defined in the previous section, though [CDM98] do not only consider periods but also associate those periods with days.

Δ describes several desirable criteria, which a timetable should have from a pedagogical point of view. Examples are "no more than x teaching hours a day for each teacher", "A class should not have the same teacher every day during its last hour.", "uniform distribution of the hours with the same teacher over the week", "Lecture X should be given in pairs of periods.".

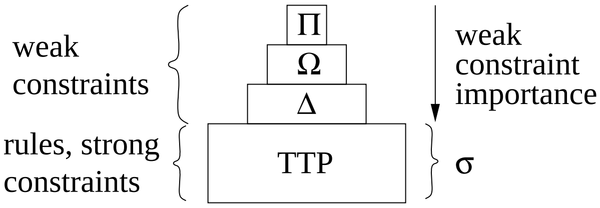

Figure 2: Design of the optimisation problem

Ω are requirements which are useful from the organisational point of view. Examples are "at least 2 teaching hours per day for each teacher", "as few holes as possible in the teachers' schedules", "Concentrate holes on one day for parent-teacher meeting hours.", "Do not assign all available teachers in one period."

Π are each teacher's individual preferences, e.g. "Teacher X does not like teaching in the morning.", "Teacher X does not want to teach in the afternoon.", or "X prefers having a particular day off.".

[CDM98] formulate this problem as an objective function which should be minimised. In our opinion, this approach goes into the right direction, albeit the representation is not optimal. The main point of criticism from the representational point of view is that compulsory constraints are represented in the same way as the desired properties.

3.4 Formulation of TTOP in DATALOG$^{not,\vee,w}$

We will extend the basic problem using the weak constraint framework. Instead of defining an objective function explicitly, we use priority layers to establish the hierarchy among the different classes of desiderata and weights to specify preferences within these classes. Figure 2 shows how we extend the basic problem by weak constraints and how they relate to the description in [CDM98].

Many preferences are related to the notion of days, which is not yet included in our representation. We assume that `period_day/2` relates periods to days.

Δ, Ω, and Π contain desirable criteria of different importance — among these classes and also within them. [CDM98] suggests to view Δ and Ω as on about the same level of importance and Π to be less important. However, this choice is arbitrary and we believe that didactic requirements should in general be more important than organisational ones. So we will create a hierarchy of three priority layers of non-strict constraints instead of just two.

We will present one example of each group of preferences. The other constraints which have been vaguely described above are also representable in our framework.

Concerning Δ, the representation of "Uniform distribution of the hours with the same teacher over the week for each class" can be accomplished as follows:

$$\begin{aligned} \Leftarrow\ & \mathtt{a(C,T,P1,K1),a(C,T,P2,K2),} \\ & \mathtt{period\_day(P1,D),period\_day(P2,D),P1 \neq P2.} \qquad [3:\Delta_1] \end{aligned} \tag{16}$$

The meaning is that two lectures should possibly not be scheduled on the same day, thereby the distribution over several days is maximised. Δ_1 defines the weight of this weak constraint relative to the other weak constraints in the same priority layer.

Regarding Ω, the organisational constraints, we have for example "At least 2 teaching hours per day for each teacher". This should rather read "Not a singleton teaching hour on any day for a teacher", since teachers may have a day off. The criterion can then be formulated in a rather straightforward manner by introducing an auxiliary predicate:

$$\begin{aligned} \mathtt{severalhours(T,D)} \leftarrow\ & \mathtt{a(C,T,P,K),a(C1,T,P1,K1),} \\ & \mathtt{period\_day(P,D),period\_day(P1,D),} \\ & \mathtt{P \neq P1.} \end{aligned} \tag{17}$$

$$\begin{aligned} \Leftarrow\ & \mathtt{a(C,T,P,K),period\_day(P,D),} \\ & \mathtt{not\ severalhours(T,D).} \qquad [2:\Omega_1] \end{aligned} \tag{18}$$

Note that the layer is lower than the one for Δ. Ω_1 describes the weight within the organisational constraints.

Now for the constraints in Π: "A particular day off." is easy. Lectures which are scheduled in periods on the particular day `d` for teacher `t`, get the weight $\Pi_2^{\mathtt{t}}$.

$$\Leftarrow \mathtt{a(I,t,P,K),period\_day(P,d).} \qquad [1:\Pi_2^{\mathtt{t}}] \tag{19}$$

4 Conclusion

We have presented how disjunctive datalog extended by strong and weak constraints can be used to represent various school timetabling problems, including various optimisation criteria, in an elegant and declarative manner.

The encoding defined in this paper also has practical relevance, since we are currently adding support for weak constraints to the `dlv` system, our prototype implementation of disjunctive datalog, which already has support for strong constraints. Up-to-date information on `dlv` and the implementation itself are available at `http://www.dbai.tuwien.ac.at/proj/dlv/`.

References

[AM98] Slim Abdennadher and Michael Marte. Constraintbasierte Stundenplanung für Universitäten. In *12. Workshop Planen und Konfigurieren (PuK-98)*, 1998. (in German) Proceedings appeared as Technical Report of the University of Paderborn, available from Benno Stein <stein@uni-paderborn.de>.

[Bar96] Victor A. Bardadym. Computer-aided school and university timetabling: The new wave. In Burke and Ross [BR96], pages 22–45.

[BLR97a] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Adding Weak Constraints to Disjunctive Datalog. In *Proceedings of the 1997 Joint Conference on Declarative Programming APPIA-GULP-PRODE'97*, Grado, Italy, June 1997.

[BLR97b] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Strong and Weak Constraints in Disjunctive Datalog. In *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR '97)*, Dagstuhl, Germany, July 1997.

[BR96] Edmund Burke and Peter Ross, editors. *Practice and Theory of Automated Timetabling, First International Conference 1995*, number 1153 in LNCS. Springer, 1996.

[CDM98] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Metaheuristics for high school timetabling. *Computational Optimization and Applications*, 9(3):275–298, 1998.

[CGT90] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.

[CK96] Tim B. Cooper and Jeffrey H. Kingston. The complexity of timetable construction problems. In Burke and Ross [BR96], pages 283–295.

[Dix95] J. Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. In *Logic, Action and Information. Proc. of the Konstanz Colloquium in Logic and Information (LogIn'92)*, pages 241–329. DeGruyter, 1995.

[HW96a] Martin Henz and Jörg Würtz. Constraint-based timetabling – a case study. *Applied Artificial Intelligence*, 10:439–453, 1996.

[HW96b] Martin Henz and Jörg Würtz. Using oz for college time tabling. In Burke and Ross [BR96], pages 283–296.

[Llo84] J.W. Lloyd. *Foundations of Logic Programming.* Springer, Berlin, 1984.

[Sch95] A. Schaerf. A survey of automated timetabling. Technical Report CS-R9567 1995, Computer Science/Department of Software Technology, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, 1995.

[Ull89] J. D. Ullman. *Principles of Database and Knowledge Base Systems.* Computer Science Press, 1989.

[YKNW94] M. Yoshikawa, K. Kaneko, Y. Nomura, and M. Watanabe. A constraint-based approach to high-school timetabling problems: A case study. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 1111–1116, 1994.