On Programs with Linearly Ordered Multiple Preferences

Davy Van Nieuwenborgh*, Stijn Heymans, and Dirk Vermeir**

Dept. of Computer Science
Vrije Universiteit Brussel, VUB
Pleinlaan 2, B1050 Brussels, Belgium
{dvnieuwe,sheymans,dvermeir}@vub.ac.be

Abstract. The extended answer set semantics for logic programs allows for the defeat of rules to resolve contradictions. We propose a refinement of these semantics based on a preference relation on extended literals. This relation, a strict partial order, induces a partial order on extended answer sets. The preferred answer sets, i.e. those that are minimal w.r.t. the induced order, represent the solutions that best comply with the stated preference on extended literals. In a further extension, we propose linearly ordered programs that are equipped with a linear hierarchy of preference relations. The resulting formalism is rather expressive and essentially covers the polynomial hierarchy. E.g. the membership problem for a program with a hierarchy of height n is Σ^P_{n+1}-complete. We illustrate an application of the approach by showing how it can easily express hierarchically structured weak constraints, i.e. a layering of "desirable" constraints, such that one tries to minimize the set of violated constraints on lower levels, regardless of the violation of constraints on higher levels.

1 Introduction

In *answer set programming* (see e.g. [3, 22]) one uses a logic program to modularly describe the requirements that must be fulfilled by the solutions to a problem. The solutions then correspond to the models (answer sets) of the program, which are usually defined through (a variant of) the stable model semantics[19]. The technique has been successfully applied in problem areas such as planning[22, 13, 14], configuration and verification [27, 28], diagnosis[12, 31], game theory[11], updates[15] and database repairs[2, 29].

The traditional answer set semantics is not universal, i.e. programs may not have any answer sets at all. While natural, this poses a problem in cases where, although there is no exact solution, one would appreciate to obtain an approximate one, even if it violates some of the rules. E.g. in an over-constrained timetabling problem, an approximate solution that ignores some demands of some users may be preferable to having no schedule at all.

* Supported by the FWO

** This work was partially funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-37004 WASP project

B. Demoen and V. Lifschitz (Eds.): ICLP 2004, LNCS 3132, pp. 180-194, 2004.

The extended answer set semantics from [29, 30] achieves this by allowing for the *defeat* of problematic rules. Consider for example the rules $a \leftarrow$, $b \leftarrow$ and $\neg a \leftarrow b$. Clearly, these rules are inconsistent and have no classical answer set, while both $\{a, b\}$ and $\{\neg a, b\}$ will be recognized as extended answer sets. In $\{a, b\}$, $\neg a \leftarrow b$ is defeated by $a \leftarrow$ while in $\{\neg a, b\}$, $a \leftarrow$ is defeated by $\neg a \leftarrow b$.

In this paper, we extend the above semantics by equipping programs with a preference relation over extended literals (outcomes). Such a preference relation can be used to induce a partial order on the extended answer sets, the minimal elements of which will be preferred. In this way, the proposed extension allows one to select the more appropriate approximate solutions of an over-constrained problem.

Consider for example a news redaction that has four different news items available that are described using the following extended answer sets:

$$\begin{aligned} N_1 &= \{\mathit{local}, \mathit{politics}\} \\ N_2 &= \{\mathit{local}, \mathit{sports}\} \\ N_3 &= \{\mathit{national}, \mathit{economy}\} \\ N_4 &= \{\mathit{international}, \mathit{economy}\} \ . \end{aligned}$$

The redaction wishes to order them according to their preferences. Assuming that, regardless of the actual subject, local news is preferred over national or international items, the preference could be encoded as[1]

$$\mathit{local} < \mathit{national} < \mathit{international} < \{\mathit{economy}, \mathit{politics}, \mathit{sports}\} \ .$$

Intuitively, using the above preference relation, N_1 and N_2 should be preferred upon N_3, which should again be preferred upon N_4, i.e. $N_1, N_2 \sqsubseteq N_3 \sqsubseteq N_4$.

In the above example, only one preference relation is used, corresponding to one point of decision in the news redaction. In practice, different journalists may have conflicting preferences, and different authorities. E.g., the editor-in-chief will have the final word on which item comes first, but she will restrict herself to the selection made by the journalists. Suppose the editor-in-chief has the following preference

$$\mathit{economy} < \mathit{politics} < \mathit{sports} < \{\mathit{local}, \mathit{national}, \mathit{international}\} \ .$$

Applying this preference to the preferred items N_1 and N_2 presented by the journalists yields the most preferred item N_1.

Such hierarchies of preference relations are supported by *linearly ordered programs*, where a program is equipped with an ordered list $\langle <_i \rangle_{i=1,\ldots n}$ of preference relations on extended literals, representing the hierarchy of user preferences ($<_i$ has a higher priority than $<_{i+1}$). Semantically, preferred extended answer sets for such programs will result from first optimizing w.r.t. $<_1$, then selecting from the result the optimal sets w.r.t. $<_2$ etc. Obviously, the order in which the preference relations are applied is important, e.g. exchanging the priorities of the preference relations in the example would yield N_3 as the preferred news item.

[1] We use $a < X$, with X a set, as an abbreviation for $\{a < x \mid x \in X\}$.

It turns out that such hierarchically layered preference relations are very expressive. More specifically, we show that such programs can solve arbitrary complete problems of the polynomial hierarchy.

In [8], weak constraints are introduced as a type of constraint[2] that is "desirable" but may be violated if there are no other options, i.e. violations of weak constraints should be minimized. The framework also supports hierarchically structured weak constraints, where constraints on the lower levels are more important than constraints on higher levels. Mirroring the semantics for linearly ordered programs, solutions minimizing the violation of constraints on the lowest level are first selected, and, among those, the solutions that minimize the constraints on the second level are retained, continuing up to the highest level. Weak constraints are useful in areas such as planning, abduction and optimizations from graph theory[16, 10]. It will be shown that hierarchically structured weak constraints can be easily captured by linearly ordered programs.

The remainder of the paper is organized as follows. In Section 2, we present the extended answer set semantics together with a preference relation on extended literals and illustrate how it can be used to elegantly express common problems. Section 3 introduces linearly ordered programs, the complexity of the proposed semantics is discussed in Section 4. Before concluding and giving directions for further research in Section 6, we show in Section 5 how weak constraints can be implemented using linearly ordered programs.

2 Ordered Programs

We use the following basic definitions and notation. A *literal* is an *atom* a or a negated atom $\neg a$. For a set or literals X, $\neg X$ denotes $\{\neg a \mid a \in X\}$ where $\neg\neg a = a$. X is *consistent* if $X \cap \neg X = \emptyset$. An interpretation I is a consistent set of (ordinary) literals.

An *extended literal* is a literal or a *naf-literal* of the form *not* l where l is a literal. The latter form denotes negation as failure. For a set of extended literals X, we use X^- to denote the set of ordinary literals underlying the naf-literals in X, i.e. $X^- = \{l \mid \textit{not}\, l \in X\}$. An extended literal l is true w.r.t. an interpretation I, denoted $I \models l$ if $l \in I$ in case l is ordinary, or $a \notin I$ if $l = \textit{not}\ a$ for some ordinary literal a. As usual, $I \models X$ for some set of (extended) literals l iff $\forall l \in X \cdot I \models l$.

An *extended rule* is a rule of the form $\alpha \leftarrow \beta$ where $\alpha \cup \beta$ is a finite set of extended literals[3] and $|\alpha| \leq 1$. An extended rule $r = \alpha \leftarrow \beta$ is *satisfied* by I, denoted $I \models r$, if $I \models \alpha$ and $\alpha \neq \emptyset$, whenever $I \models \beta$, i.e. if r is *applicable* ($I \models \beta$), then it must be *applied* ($I \models \alpha \cup \beta \wedge \alpha \neq \emptyset$). Note that this implies that a *constraint*, i.e. a rule with empty head ($\alpha = \emptyset$), can only be satisfied if it is not applicable ($I \not\models \beta$).

A countable set of extended rules is called an *extended logic program* (ELP). The *Herbrand base* $\mathcal{B}_P$ of an ELP P contains all atoms appearing in P. Further, we use $\mathcal{L}_P$ and $\mathcal{L}^*_P$ to denote the set of literals (resp. extended literals) that can be constructed from $\mathcal{B}_P$, i.e. $\mathcal{L}_P = \mathcal{B}_P \cup \neg\mathcal{B}_P$ and $\mathcal{L}^*_P = \mathcal{L}_P \cup \{\textit{not}\ l \mid l \in \mathcal{L}_P\}$. For an ELP P and an interpretation I we use $P_I \subseteq P$ to denote the *reduct* of P w.r.t. I, i.e. $P_I = \{r \in P \mid$

[2] A constraint is a rule of the form $\leftarrow \alpha$, i.e. with an empty head. Any answer set should therefore not contain α.

[3] As usual, we assume that programs have already been grounded.

$I \models r\}$, the set of rules satisfied by I. We call an interpretation I a model of a program P if $P_I = P$, i.e. I satisfies all rules in P. It is a minimal model of P if there is no model J of P such that $J \subset I$.

A *simple program* is a program without negation as failure. For simple programs P, we define an answer set of P as the minimal model of P. On the other hand, for a program P containing negation as failure, we define the *GL-reduct*[19] for P w.r.t. I, denoted P^I, as the program consisting of those rules $(\alpha \setminus \mathit{not}\, \alpha^-) \leftarrow (\beta \setminus \mathit{not}\, \beta^-)$ where $\alpha \leftarrow \beta$ is in P, $I \models \mathit{not}\ \beta^-$ and $I \models \alpha^-$.

Note that all rules in P^I are free from negation as failure, i.e. P^I is a simple program. An interpretation I is then an *answer set* of P iff I is a minimal model of the GL-reduct P^I. An extended rule $r = \alpha \leftarrow \beta$ is *defeated* w.r.t. P and I iff there exists an applied *competing rule* $r' = \alpha' \leftarrow \beta'$ such that $\{\alpha, \alpha'\}$ is inconsistent. An *extended answer set* for P is any interpretation I such that I is an answer set of P_I and each unsatisfied rule in $P \setminus P_I$ is defeated.

Example 1. Consider the ELP P shown below. The program describes a choice between speeding or not. Sticking to the indicated limit guarantees not getting a fine while speeding, when it is known that the police are carrying out checks, will definitely result in a fine. Finally, if nothing is known about checks, there is still a chance for a fine.

$$\begin{array}{rlrl}
\mathit{speeding} \leftarrow & & \mathit{fine} \leftarrow & \mathit{speeding}, \mathit{check} \\
\neg\mathit{speeding} \leftarrow & & \mathit{maybe_fine} \leftarrow & \mathit{speeding}, \mathit{not}\ \mathit{check} \\
\mathit{check} \leftarrow & & \neg\mathit{fine} \leftarrow & \neg\mathit{speeding} \\
\mathit{not}\ \mathit{check} \leftarrow & & \mathit{fine} \leftarrow & \mathit{maybe_fine} \\
\mathit{not}\ \mathit{fine} \leftarrow & \mathit{maybe_fine} & &
\end{array}$$

The above program has five possible extended answer sets, which are $M_1 = \{\mathit{speeding}, \mathit{check}, \mathit{fine}\}$, $M_2 = \{\mathit{speeding}, \mathit{maybe\_fine}\}$, $M_3 = \{\neg\mathit{speeding}, \mathit{check}, \neg\mathit{fine}\}$, $M_4 = \{\neg\mathit{speeding}, \neg\mathit{fine}\}$ and $M_5 = \{\mathit{speeding}, \mathit{maybe\_fine}, \mathit{fine}\}$.

Unlike traditional answer sets, extended answer sets are, in general, not subset minimal, i.e. an ELP P can have extended answer sets M and N with $M \subset N$, as witnessed by M_3 and M_4 in Example 1. Moreover, a program can have both answer sets and extended answer sets (that are not answer sets). E.g. the ELP $\{a \leftarrow\ ;\ \mathit{not}\ a \leftarrow \mathit{not}\ a\}$, has two extended answer sets $I = \{a\}$, which is also a traditional answer set, and $J = \emptyset$, which is not.

Often, certain extended answer sets are preferable over others. E.g., in Example 1, one would obviously prefer not to get fined, which can be represented as a strict partial order $<_1$ on literals: $<_1 = \neg\mathit{fine} < \mathcal{L}_P^* \setminus \{\neg\mathit{fine}\}$. However, in an emergency, one may prefer to ignore the speed limit, resulting in an alternative preference relation $<_2 = \{\mathit{speeding} < \neg\mathit{speeding} < C;\ \mathit{maybe\_fine} < \mathit{fine} < C\}$, where $C = \mathcal{L}_P^* \setminus \{\mathit{speeding}, \neg\mathit{speeding}, \mathit{maybe\_fine}, \mathit{fine}\}$.

In general, we introduce, for an ELP P, a strict partial order[4] $<$ on extended literals, such that, for two extended literals l_1 and l_2, $l_1 < l_2$ expresses that l_1 is more preferred

[4] A strict partial order $<$ on a set X is a binary relation on X that is antisymmetric, anti-reflexive and transitive. The relation $<$ is well-founded if every nonempty subset of X has a $<$-minimal element.

than l_2. This preference relation induces a partial order[5] $\sqsubseteq$ on the extended answer sets of P.

Definition 1. *An **ordered program** is a pair $\langle P, <\rangle$ where P is an ELP and $<$ is a strict well-founded[6] partial order on $\mathcal{L}_P^*$. For subsets $M, N \subseteq \mathcal{L}_P$, we define $M \sqsubseteq N$ iff $\forall n \in \{l \in \mathcal{L}_P^* \mid N \models l \wedge M \not\models l\} \cdot \exists m \in \{l \in \mathcal{L}_P^* \mid M \models l \wedge N \not\models l\} \cdot m < n$. A **preferred answer set** of $\langle P, <\rangle$ is an extended answer set of P that is minimal w.r.t. $\sqsubseteq$ among the set of extended answer sets of P.*

Intuitively, M is preferable over N, i.e. $M \sqsubseteq N$, if every literal n from $N \setminus M$ is "countered" by a "better" literal $m < n$ from $M \setminus N$.

Applying the above definition on Example 1 with $<_1$ as described above, yields both M_3 and M_4 as preferred answer sets, while M_2 is the only preferred answer set w.r.t. $<_2$, which fits our intuition in both cases.

In the sequel, we will specify a strict partial order over a set X using an expression of the form

$$L = \{x_1 < y_1, \ldots, x_n < y_n, z_1, \ldots, z_m\}$$

which stands, unless explicitly stated otherwise, for the strict partial order defined by

$$\begin{array}{rl} & \{x_i < y_i \mid 1 \leq i \leq n\} \\ \cup & \{x_i < u \mid 1 \leq i \leq n \ \wedge \ u \in U\} \\ \cup & \{y_i < u \mid 1 \leq i \leq n \ \wedge \ u \in U\} \\ \cup & \{z_j < u \mid 1 \leq j \leq m \ \wedge \ u \in U\} \end{array}$$

with U the set of elements not occurring in L, i.e. $U = X \setminus K$ with $K = \{x_i \mid 1 \leq i \leq n\} \cup \{y_i \mid 1 \leq i \leq n\} \cup \{z_j \mid 1 \leq j \leq m\}$.

This notation implies that extended literals outside of L are least preferred and thus cannot influence the preference among extended answer sets. E.g. if $L = \{a < b\}$ then both $\{a, c\} \sqsubset \{b\}$ and $\{a\} \sqsubset \{b, c\}$ (where $M \sqsubset N$ iff $M \sqsubseteq N$ and $M \neq N$).

Example 2. The program below offers a choice between a large and a small drink to go with spicy or mild food.

$$\begin{array}{rl} large\_drink & \leftarrow not\ small\_drink \\ small\_drink & \leftarrow not\ large\_drink \\ spicy & \leftarrow not\ mild \\ mild & \leftarrow not\ spicy \\ large\_drink & \leftarrow spicy \end{array}$$

There are three extended answer sets: $M_1 = \{large\_drink, spicy\}$, $M_2 = \{large\_drink, mild\}$ and $M_3 = \{small\_drink, mild\}$.

A smaller drink is preferred and there is no particular preference between $mild$ and $spicy$, yielding $\{small\_drink < large\_drink,\ mild,\ spicy\}$ to describe the preferences. The preference for a small drink causes $M_3 \sqsubseteq M_2$ while M_1 is incomparable with both M_2 and M_3. Thus both M_1 and M_3 are preferred.

[5] That $\sqsubseteq$ is a partial order follows from Theorem 6 in [29].

[6] It is easy to verify that, if $<$ is empty, then $M \sqsubseteq N$ iff $\{l \mid N \models l\} \subseteq \{l \mid M \models l\}$ which reduces to $N = M$, i.e. all extended answer sets are preferred.

There is no "one true way" to induce a preference relation $\sqsubseteq$ over extended answer sets from a particular ordering $<$ over extended literals. E.g. [26], which deals with traditional answer sets of extended disjunctive programs, proposes a different method which is shown below, adapted to the current framework.

Definition 2. *For an **ordered program** $\langle P, < \rangle$ and subsets $M, N \subseteq \mathcal{L}_P$, we define $M \sqsubseteq_s N$ iff*

- $M = N$ *(reflexive), or*
- $\exists L \cdot M \sqsubseteq_s L \ \wedge \ L \sqsubseteq_s N$ *(transitive), or*
- $\exists e_1 \in \{l \in \mathcal{L}_P^* \mid M \models l \wedge N \not\models l\}$ *such that*

$$\begin{aligned} &\exists e_2 \in \{l \in \mathcal{L}_P^* \mid N \models l \wedge M \not\models l\} \cdot e_1 < e_2 \\ \wedge\ &\neg \exists e_3 \in \{l \in \mathcal{L}_P^* \mid N \models l \wedge M \not\models l\} \cdot e_3 < e_1 \ . \end{aligned}$$

Obviously, $\sqsubseteq_s$ is also a partial order.

Theorem 1. *Let $\langle P, < \rangle$ be an ordered program and let M and N be two extended answer sets of P. Then $M \sqsubseteq N$ implies that $M \sqsubseteq_s N$.*

The other direction is in general not true, as appears from the following example.

Example 3. Consider the program P depicted below.

$$\begin{array}{rlrl} \mathit{shares} \leftarrow & \mathit{not\ cash} & \mathit{cash} \leftarrow & \mathit{not\ shares} \\ \$\mathit{100} \leftarrow & \mathit{shares} & \$\mathit{1000} \leftarrow & \mathit{cash} \\ \mathit{stock\_options} \leftarrow & \mathit{shares} & & \end{array}$$

This program has two extended answer sets, i.e. $M_1 = \{cash, \$\mathit{1000}\}$ and $M_2 = \{shares, \$\mathit{100}, stock\_options\}$.

The preference $<= \{\$\mathit{1000} < \$\mathit{100}\}$, where we take $<$ as is, i.e. without applying the expansion from the previous page, expresses no preference between cash, shares and stock options, except that, obviously, a larger amount of cash is preferred over a smaller amount. Using the $\sqsubseteq_s$ preference relation, we get $M_1 \sqsubseteq_s M_2$ and $M_2 \not\sqsubseteq_s M_1$, while for the $\sqsubseteq$ relation both $M_1 \not\sqsubseteq M_2$ and $M_2 \not\sqsubseteq M_1$ holds. This makes M_1 preferred w.r.t. $\sqsubseteq_s$, while both M_1 and M_2 are preferred w.r.t. $\sqsubseteq$. Note that, e.g. $M_1 \not\sqsubseteq M_2$ because M_1 cannot counter the $stock\_options$ of M_2 by something more preferred.

In general, $\sqsubseteq$ makes no decision between extended answer sets containing unrelated literals in their differences, while $\sqsubseteq_s$ is more credulous, preferring e.g. \$1000 over \$100 and some "unknown" $stock\_options$.

In the next section, the skeptical approach of $\sqsubseteq$ will turn out to be useful since it allows for new information to refine an earlier result. E.g., if, in the above example, it is later learned that the stock options provide great value, $stock\_options < \$\mathit{1000}$, might be added, possibly in a different preference relation, and used to prefer M_2 among the earlier "best choices" M_1 and M_2.

From Theorem 1, the following is immediate.

Corollary 1. *Let $R = \langle P, < \rangle$ be an ordered program. Then, the preferred answer sets of R w.r.t. $\sqsubseteq_s$ are also preferred w.r.t. $\sqsubseteq$.*

3 Linear n-Ordered Programs

A strict partial order on literals, as defined in the previous section, is a powerful and flexible tool to express a wide range of preferences. However, in practice, it is sometimes useful to have different layers of preferences, each applied in turn. As an example, consider the staff selection procedure of a company. Job applicants are divided into certain profiles, e.g. either female or male, old or young, experienced or not. Further, it is believed that inexperienced applicants tend to be ambitious, which is captured by the following program.

$$\begin{array}{rlrl} \mathit{female} & \leftarrow \mathit{not}\ \mathit{male} & \mathit{male} & \leftarrow \mathit{not}\ \mathit{female} \\ \mathit{old} & \leftarrow \mathit{not}\ \mathit{young} & \mathit{young} & \leftarrow \mathit{not}\ \mathit{old} \\ \mathit{experienced} & \leftarrow \mathit{not}\ \mathit{inexperienced} & \mathit{inexperienced} & \leftarrow \mathit{not}\ \mathit{experienced} \\ \mathit{ambitious} & \leftarrow \mathit{inexperienced} & & \end{array}$$

The decision to hire a new staff member goes through a chain of decision makers. On the lowest, and most preferred, level, company policy is implemented. It stipulates that experienced persons are to be preferred over inexperienced and ambitious persons, i.e. $<_1 = \{\mathit{experienced} < \{\mathit{inexperienced}, \mathit{ambitious}\}\}$. On the second level, the financial department prefers young and inexperienced employees, since they tend to cost less, i.e. $<_2 = \{\mathit{young} < \mathit{old}, \mathit{inexperienced} < \mathit{experienced}\}$. On the last, weakest, level, the manager prefers a woman to enforce her largely male team, i.e. $<_3 = \{\mathit{female} < \mathit{male}\}$.

In this example, any preferred extended answer set should be preferred w.r.t. $\sqsubseteq_1$ (induced by $<_1$) among all extended answer sets and, furthermore, among the $\sqsubseteq_1$-preferred sets, it should also be $\sqsubseteq_2$-preferred (where $\sqsubseteq_2$ is induced by $<_2$). Finally, the preferred answer sets of the complete problem are the $\sqsubseteq_2$-preferred sets which are also $\sqsubseteq_3$-preferred (where $\sqsubseteq_3$ is induced by $<_3$).

Formally, we extend ordered programs, by allowing a linearly ordered set of preference relations $<_1, \ldots, <_n$ for an ELP P, where $<_1$ is the order with the highest priority.

Definition 3. *A* ***linearly ordered program*** *(LOLP) is a pair $\langle P, \langle <_i \rangle_{i=1,\ldots,n} \rangle$ where P is an ELP and $\langle <_i \rangle_{i=1,\ldots,n}$ is a sequence of (strict partial order) preference relations $<_1, \ldots, <_n$. Each of these orders $<_i$ induces a preference relation $\sqsubseteq_i$ between extended answer sets, as in Definition 1.*

We define the preference up to a certain order of extended answer sets by induction.

Definition 4. *Let $\langle P, \langle <_i \rangle_{i=1,\ldots,n} \rangle$ be a LOLP. An extended answer M set is preferable up to $<_i$, $1 \le i \le n$, iff*

- *$i = 1$ and M is preferred w.r.t. $\sqsubseteq_i$, or*
- *$i > 1$, M is preferable up to $<_{i-1}$, and there is no N, preferable up to $<_{i-1}$, such that $N \sqsubset_i M$.*

An extended answer set M of P is ***preferred*** *if it is preferable up to $<_n$.*

Continuing the above example, we have eight extended answer sets for the program, which are all preferable up to $<_0$. After applying $<_1$, only four of them are left, i.e. $M_1 = \{experienced, old, female\}$, $M_2 = \{experienced, young, female\}$, $M_3 = \{experienced, male, young\}$ and $M_4 = \{experienced, old, male\}$, which fits the company policy to drop inexperienced ambitious people. When $<_2$ is applied on these four remaining extended answer sets, only M_2 and M_3 are kept as preferable up to $<_2$. Finally, the manager will select M_2 as the only extended answer set preferable up to $<_3$.

Note that rearranging the chain of orders gives, in general, different results. E.g., interchanging $<_1$ with $<_2$ yields $\{young, female, ambitious, inexperienced\}$ as the only extended answer set preferable up to $<_3$.

4 Complexity

We first recall briefly some relevant notions of complexity theory (see e.g. [24, 3] for a nice introduction). The class P (NP) represents the problems that are deterministically (nondeterministically) decidable in polynomial time, while $coNP$ contains the problems whose complement are in NP.

The polynomial hierarchy, denoted PH, is made up of three classes of problems, i.e. Δ_k^P, Σ_k^P and Π_k^P, $k \geq 0$, which are defined as follows:

1. $\Delta_0^P = \Sigma_0^P = \Pi_0^P = P$; and
2. $\Delta_{k+1}^P = P^{\Sigma_k^P}$, $\Sigma_{k+1}^P = NP^{\Sigma_k^P}$, $\Pi_{k+1}^P = co\Sigma_{k+1}^P$.

The class $P^{\Sigma_k^P}$ ($NP^{\Sigma_k^P}$) represents the problems decidable in deterministic (nondeterministic) polynomial time using an oracle for problems in Σ_k^P, where an oracle is a subroutine capable of solving Σ_k^P problems in unit time. Note that $\Delta_1^P = P$, $\Sigma_1^P = NP$ and $\Pi_1^P = coNP$. Further, it is obvious that $\Sigma_k^P \subseteq \Sigma_k^P \cup \Pi_k^P \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P$, but for $k \geq 1$ any equality is considered unlikely. Further, the class PH is defined by $PH = \bigcup_{k=0}^{\infty} \Sigma_k^P$.

A language L is called complete for a complexity class C if both L is in C and L is hard for C. Showing that L is hard is normally done by reducing a known complete decision problem into a decision problem in L. For the classes Σ_k^P and Π_k^P with $k > 0$ a known complete, under polynomial time transformation, problem is checking whether a quantified boolean formula (QBF) ϕ is valid. Note that this does not hold for the class PH for which no complete problem is known unless $P = NP$.

Quantified boolean formulas are expressions of the form $Q_1X_1Q_2X_2 \ldots Q_kX_k \cdot G$, where $k \geq 1$, G is a Boolean expression over the atoms of the pairwise nonempty sets of variables $X_1, \ldots, X_k$ and the Q_i's, for $i = 1, \ldots, k$ are alternating quantifiers from $\{\exists, \forall\}$. When $Q_1 = \exists$, the QBF is k-existential, when $Q_1 = \forall$ we say it is k-universal. We use $QBF_{k,\exists}$ ($QBF_{k,\forall}$) to denote the set of all valid k-existential (k-universal) QBFs.

Deciding, for a given k-existential (k-universal) QBF ϕ, whether $\phi \in QBF_{k,\exists}$ ($\phi \in QBF_{k,\forall}$) is a Σ_k^P-complete (Π_k^P-complete) problem.

The following results shed some light on the complexity of the preferred answer set semantics for linear n-ordered logic programs.

First of all, checking whether an interpretation I is an extended answer set of an ELP P is in P, because (a) checking if each rule in P is either satisfied or defeated w.r.t. I, (b) applying the GL-reduct on P_I w.r.t. I, i.e. computing $(P_I)^I$, and (c) checking whether the positive program $(P_I)^I$ has I as its unique minimal model, can all be done in polynomial time.

On the other hand, the complexity of checking whether an extended answer set M is not preferable up to a certain $<_n$ depends on n, as shown in the next lemma.

Lemma 1. *Let $\langle P, \langle <_i \rangle_{i=1,\dots,n} \rangle$ be a LOLP, and let M be an extended answer set of P. Checking whether M is not preferable up to $<_n$ is in Σ_n^P.*

Proof. The proof is by induction on n.

The base case, i.e. $n = 0$, holds vacuously as checking whether M is an extended answer set is in $P = \Sigma_0^P$.

For the induction step, checking that M is not preferable up to $<_n$ can be done by (a) checking that M is (or is not) preferable up to $<_{n-1}$, which is in Σ_{n-1}^P due to the induction hypothesis; and (b) guessing, if M is preferable up to $<_{n-1}$, an interpretation $N \sqsubset_n M$ and checking that it is not the case that N is not preferable up to $<_{n-1}$, which is again in Σ_{n-1}^P due to the induction hypothesis. As a result, at most two calls are made to a Σ_{n-1}^P oracle and at most one guess is made, yielding that the problem itself is in $NP^{\Sigma_{n-1}^P} = \Sigma_n^P$. □

Using the above yields the following theorem about the complexity of LOLPs.

Theorem 2. *Let $\langle P, \langle <_i \rangle_{i=1,\dots,n} \rangle$ be a LOLP and l a literal. Deciding whether there is a preferred answer set containing l is in Σ_{n+1}^P.*

Proof. The task can be performed by an NP-algorithm that guesses an interpretation $M \ni l$ and checks that it is not the case that M is not preferable up to level n. Due to Lemma 1, the latter is in Σ_n^P, so the former is in $NP^{\Sigma_n^P} = \Sigma_{n+1}^P$. □

Theorem 3. *Let $\langle P, \langle <_i \rangle_{i=1,\dots,n} \rangle$ be a LOLP and l a literal. Deciding whether every preferred answer set contains l is in Π_{n+1}^P.*

Proof. Due to Theorem 2, finding a preferred answer set M not containing l, i.e. $l \notin M$, is in Σ_{n+1}^P. Hence, the complement of the problem is in Π_{n+1}^P. □

To prove hardness, we provide a reduction of deciding validity of QBFs by means of LOLPs.

Theorem 4. *The problem of deciding, given a LOLP $\langle P, \langle <_i \rangle_{i=1,\dots,n} \rangle$ and a literal l, whether there exists a preferred answer set containing l is Σ_{n+1}^P-hard.*

Proof. (Sketch). Let $\phi = \exists X_1 \forall X_2 \dots QX_{n+1} \cdot G \in QBF_{n+1,\exists}$, where $Q = \forall$ if n is odd and $Q = \exists$ otherwise. We assume, without loss of generality, that G is in disjunctive normal form, i.e. $G = \vee_{c \in C} C$ where C is a set of sets of literals over $X_1 \cup \dots \cup X_{n+1}$ and each $c \in C$ has to be read as a conjunction. In what follows, we will write $l <_i X_{p \dots q}$ to denote the longer $\{l <_i x \; ; \; l <_i \neg x \mid x \in X_j \wedge p \leq j \leq q\}$.

The LOLP $\langle P_\phi, \langle <_i \rangle_{i=1,\ldots,n} \rangle$ corresponding to ϕ is defined by the ELP P_ϕ:

$$\begin{array}{ll} P_1 : & \{x \leftarrow ;\ \neg x \leftarrow \mid x \in X_i \wedge 1 \le i \le n+1\} \\ P_2 : & \{g \leftarrow c \mid c \in C\} \\ P_3 : & sat \leftarrow g \\ P_4 : & \neg sat \leftarrow not\ g \end{array}$$

and the sequence $\langle <_i \rangle_{i=1,\ldots,n}$ of orders defined by

$$\begin{array}{l} \{\neg sat <_n sat, g <_n X_{2\ldots n+1},\ X_1\} \\ \{sat, g <_{n-1} \neg sat <_{n-1} X_{3\ldots n+1},\ X_1,\ X_2\} \\ \ldots \\ \{w <_1 w' <_1 X_{n+1\ldots n+1},\ X_1,\ \ldots,\ X_n\} \end{array}$$

where $w = \neg sat$ and $w' = sat, g$ if n is odd; and $w = sat, g$ and $w' = \neg sat$ otherwise.

Obviously, the construction can be done in polynomial time. Intuitively, the rules in P_1 are used to guess a truth assignment for $X_1 \cup \ldots \cup X_{n+1}$. For each such truth assignment, the rules in P_2, P_3 and P_4 will decide whether the formula G is valid or not. The intuition behind the orders is to prefer those extended answer sets of P_ϕ that give a counterexample to the validity of ϕ. Only when such an example does not exist, i.e. ϕ is valid, an extended answer set containing the literal sat will be preferred.

First note that an order relation $<_k = w < w' < X_{n-k+2\ldots n+1}$ can only prefer an extended answer set M_1 upon M_2, i.e. $M_1 \sqsubset_k M_2$, if $M_1 \cap (X_1 \cup \ldots \cup X_{n-k+1}) = M_2 \cap (X_1 \cup \ldots \cup X_{n-k+1})$; otherwise we have both $M_1 \not\sqsubset_k M_2$ and $M_2 \not\sqsubset_k M_1$. Further, when $<_k = sat, g < \neg sat < X_{n-k+2\ldots n+1}$ (respectively $<_k = \neg sat < sat, g < X_{n-k+2\ldots n+1}$), then the $(n-k+2)^{th}$ quantifier, denoted Q_{n-k+2} in ϕ is $\exists$ ($\forall$ respectively).

In the sequel we use $\mathcal{M}^k$, with $0 \le k \le n$ to denote the set of extended answer sets of P_ϕ that are preferable up to $<_k$. We will show by induction that $\mathcal{M}^k$ only contains extended answer sets $M \in \mathcal{M}^k$ with $sat \in M$ iff $Q_{n-k+2} \ldots Q_{n+1} \cdot G$ is valid using $x_M^{1\ldots n-k+1}$, i.e. the truth combination over $X_1 \cup \cdots \cup X_{n-k+1}$ in M.

The base case, i.e. $k = 0$, holds vacuously, as we have, for each possible truth combination over $X_1 \cup \cdots \cup X_{n+1}$, an extended answer set $M \in \mathcal{M}^0$ containing $sat \in M$ if G is valid and $\neg sat \in M$ if G is not.

For the induction step, suppose the claim holds for $\mathcal{M}^{k-1}$ and consider $<_k$ and Q_{n-k+2}. When $Q_{n-k+2} = \exists$, $<_k$ will prefer those extended answer sets in $\mathcal{M}^{k-1}$ containing sat for a fixed truth combination X over $X_1 \cup \ldots \cup X_{n-k+1}$. By the induction hypothesis, we have that $M \in \mathcal{M}^{k-1}$ with $sat \in M$ iff $Q_{n-k+3} \ldots Q_{n+1} \cdot G$ is valid for $x_M^{1\ldots n-k+2}$. Clearly, $Q_{n-k+2} \ldots Q_{n+1} \cdot G$ is then valid for $x_M^{1\ldots n-k+1}$ iff $\mathcal{M}^k$ contains an extended answer set M with $sat \in M$.

On the other hand, when $Q_{n-k+2} = \forall$, $<_k$ will prefer those extended answer sets in $\mathcal{M}^{k-1}$ containing $\neg sat$ for a fixed truth combination X over $X_1 \cup \ldots \cup X_{n-k+1}$. By the induction hypothesis, we have that $M \in \mathcal{M}^{k-1}$ with $\neg sat \in M$ iff $Q_{n-k+3} \ldots Q_{n+1} \cdot G$ is not valid for $x_M^{1\ldots n-k+2}$. Clearly, only when $Q_{n-k+3} \ldots Q_{n+1} \cdot G$ holds for every combination of X_{n-k+2} with X, no extended answer sets with $\neg sat$ will be in $\mathcal{M}^{k-1}$ for X, and all those with sat will be passed to $\mathcal{M}^k$, yielding that $Q_{n-k+2} \ldots Q_{n+1} \cdot G$ holds for x_M iff $M \in \mathcal{M}^k$ with $sat \in M$.

Finally, by induction the above yields for $\mathcal{M}^n$, i.e. the preferred answer sets, which implies that ϕ is valid iff $\mathcal{M}^n$ contains a preferred answer set M containing sat, i.e. $\exists M \in \mathcal{M}^n \cdot sat \in M$, from which the theorem readily follows. $\square$

Theorem 5. *The problem of deciding, given a LOLP $\langle P, \langle <_i \rangle_{i=1,\dots,n} \rangle$ and a literal l, whether every preferred answer set contains l is Π^P_{n+1}-hard.*

Proof. Reconsider the LOLP in the proof of Theorem 4. Let l be a fresh atom not occurring in P_ϕ and define P'_ϕ as P_ϕ with two extra rules $l \leftarrow$ and $\neg l \leftarrow$. Clearly, showing that l does not occur in every preferred answer set is the same as showing that $\neg l$ occurs in any preferred answer set. Deciding the latter is Σ^P_{n+1}-hard by Theorem 4; thus deciding the complement of the former is Π^P_{n+1}-hard.

$\square$

The following is immediate from Theorem 2, 3, 4 and 5.

Corollary 2. *The problem of deciding, given an arbitrary LOLP $\langle P, \langle <_i \rangle_{i=1,\dots,n} \rangle$ and a literal l, whether there is a preferred answer set containing l is Σ^P_{n+1}-complete. On the other hand, deciding whether every preferred answer set contains l is Π^P_{n+1}-complete.*

5 Weak Constraints

Weak constraints were introduced in [8] as a relaxation of the concept of a constraint. Intuitively, a weak constraint is allowed to be violated, but only as a last resort, meaning that one tries to minimize the set of violated constraints. Here minimization is typically interpreted as either *subset minimality* or *cardinality minimality*. In the former, we prefer a solution that violates a set of weak constraints C_1 over one that violates a set C_2 iff $C_1 \subset C_2$, while in the latter, we would only need that C_1 contains less violated constraints than C_2, i.e. $|C_1| < |C_2|$.

Subset minimality is obviously less controversial since, for cardinality minimality, it may happen that, while $|C_1| < |C_2|$, C_1 contains more important constraints than C_2[7].

In [8] a semantics for hierarchies of weak constraints is defined, where one minimizes constraints on lower levels, before minimizing, among the results of the previous levels, constraints on higher levels. Formally, weak constraints have the same syntactic form as constraints, i.e. $\leftarrow \beta$ with β a set of extended literals. We then assign the weak constraints for a certain level i to a set W_i, similar to [8], and define a *weak logic program* as consisting of a program and a hierarchy of sets of weak constraints.

Definition 5. *A **weak logic program** (WLP) is a pair $\langle P, W \rangle$ where P is a program and W is a set $\{W_1, \dots, W_n\}$, with each W_i, $1 \le i \le n$, a set of weak constraints.*

To enhance readability of the following definition, we assume an empty dummy set W_0 of weak constraints.

[7] A similar preference for subset minimality over cardinality minimality is also common in, for example, the domain of *diagnosis* [25, 31], where one tries to minimize the set of causes responsible for certain failures.

Definition 6. *Let $\langle P, W\rangle$ be a WLP. The extended answer sets of P are* ***preferable up to*** W_0*. An extended answer set M of P set is preferable up to W_i, $1 \leq i \leq n$, if*

– *M is preferable up to W_{i-1}, and*
– *there is no N, preferable up to W_{i-1}, such that $W_N^i \subset W_M^i$, where $W_N^i = \{c \mid c \in W_i \wedge N \not\models c\}$, i.e. the constraints in W_i that are violated by N.*

An extended answer set of P is a ***preferred*** *answer set of a WLP $\langle P, W\rangle$ if it is preferable up to W_n.*

LOLPs can easily implement weak constraints. Intuitively, each order $<_i$ in the hierarchy will try to minimize the violation of weak constraints in W_i.

For a WLP $\langle P, \{W_1, \ldots, W_n\}\rangle$, define the LOLP $\langle P \cup WC, \langle <_i\rangle_{i=1,\ldots,n}\rangle$ with $WC = \{c \leftarrow \beta \mid c = (\leftarrow \beta) \in W_i, 1 \leq i \leq n\}$ representing the weak constraints by rules with new atoms c, one for each constraint $\leftarrow \beta$, and each order $<_i$ in $\langle <_i\rangle_{i=1,\ldots,n}$ defined by

$$\{\mathit{not}\ c_i <_i c_i \mid c_i \in W_i\}.$$

The orders prefer extended answer sets that do not contain c since c can only be obtained by applying $c \leftarrow \beta$, corresponding to a violation of the corresponding original constraint $\leftarrow \beta$.

Theorem 6. *An extended answer set M of a WLP $\langle P, W\rangle$ is preferred iff $M \cup \{c \mid c \in W, M \not\models c\}$ is a preferred answer set of the LOLP $\langle P \cup WC, \langle <_i\rangle_{i=1,\ldots,n}\rangle$.*

The other approach to minimize the violation of weak constraints, is to take into account the cardinality of the sets of violated weak constraints, as in [8]. The following definition formalizes the notion of cardinality preferred, or c-preferred for short, answer sets.

Definition 7. *Let $\langle P, W\rangle$ be a WLP. The extended answer sets of P are* ***c-preferable up to*** W_0*. An extended answer M of P set is c-preferable up to W_i, $1 \leq i \leq n$, if*

– *M is c-preferable up to W_{i-1}, and*
– *there is no N, c-preferable up to W_{i-1}, such that $|W_N^i| < |W_M^i|$.*

An extended answer set of P is a ***c-preferred*** *answer set of an WLP $\langle P, W\rangle$ if it is c-preferable up to W_n.*

In the special case that the preferable answer sets on a level are c-preferable we have that, on the next level, the c-preferable answer sets are preferable. Denote the set of extended answer sets that are c-preferable up to W_i as $\mathcal{M}_c^i$ and the set of extended answer sets preferable up to W_i as $\mathcal{M}^i$.

Theorem 7. *Let $\langle P, W\rangle$ be a WLP and let $\mathcal{M}_c^{i-1} = \mathcal{M}^{i-1}$ for some $1 \leq i \leq n$. Then $\mathcal{M}_c^i \subseteq \mathcal{M}^i$.*

The pre-condition that every extended answer set, preferable up to W_{i-1}, has to be c-preferable is necessary, as can be seen from the following example, where we have a c-preferred answer set that is not preferred.

Example 4. Take a WLP $\langle P, \{W_1, W_2\}\rangle$ with P the program

$$\begin{aligned} \neg a &\leftarrow \\ a &\leftarrow \\ b &\leftarrow a \end{aligned}$$

and the weak constraints $W_1 = \{c_1 :\leftarrow \neg a\ ;\ c_2 :\leftarrow a\ ;\ c_3 :\leftarrow b\}$ and the second level $W_2 = \{c_4 :\leftarrow \neg a, \mathit{not}\ b\}$. The program P has two extended answer sets $M = \{\neg a\}$ and $N = \{a, b\}$. This leads to the following sets of violated constraints: $W_M^1 = \{c_1\}$, $W_N^1 = \{c_2, c_3\}$, $W_M^2 = \{c_4\}$, and $W_N^2 = \emptyset$. Then, M is c-preferable up to W_1, while N is not. Both M and N are preferable up to W_1. However, M is c-preferable up to W_2, since there are no other extended answer sets that are c-preferable up to W_1, while M is not preferable up to W_2.

If there is only one level of weak constraints we have the attractive property that c-preferred answer sets are preferred.

Corollary 3. *Let $\langle P, \{W_1\}\rangle$ be a WLP with one level of weak constraints. A c-preferred answer set of $\langle P, \{W_1\}\rangle$ is preferred.*

In this case, if the preferred answer sets are already computed, and one decides later on that c-preferred answer sets are needed, the search space can be restricted to just the preferred answer sets instead of all extended answer sets.

6 Conclusions and Directions for Further Research

Equipping logic programs with a preference relation on the rules has a relatively long history [21, 20, 18, 9, 7, 5, 32, 1, 29]. Also approaches that consider a preference relation on (extended) literals have been considered: [26] proposes explicit preferences while [4, 6] encodes dynamic preferences within the program.

In this paper, we applied such preferences on the extended answer set semantics, thus allowing the selection of preferred "approximate" answer sets for inconsistent programs. We also considered a natural extension, linearly ordered programs, where there are several preference relations. This extension increases the expressiveness of the resulting formalism to cover the polynomial hierarchy.

Such preference hierarchies occur naturally in several application areas such as timetabling. As an application of the approach, we have shown that hierarchically structured weak constraints can be considered as a special case of linearly ordered programs.

Future work may generalize the structure of the preference relations, e.g. to arbitrary partial orders or to cyclic structures, where the latter may provide a natural model for agent communication.

A brute force prototype implementation for LOLPs is available which uses an existing answer set solver to generate all extended answer sets, and then filters out the preferred ones taking into account the given preference levels. A dedicated implementation, using existing answer set solvers, could, similarly to [6], compute preferred answer sets more directly by generating one extended answer set and then trying to generate a better one using an augmented program, which, when applied in a fixpoint computation, results in a preferred answer set.

References

1. José Júlio Alferes and Luís Moniz Pereira. Updates plus preferences. In Ojeda-Aciego et al. [23], pages 345–360.
2. Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. Specifying and querying database repairs using logic programs with exceptions. In *Proceedings of the 4th International Conference on Flexible Query Answering Systems*, pages 27–41, Warsaw, Octobre 2000. Springer-Verlag.
3. Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge Press, 2003.
4. G. Brewka. Logic programming with ordered disjunction. In *Proceedings of the 18th National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, pages 100–105, Edmonton, Canada, July 2002. AAAI Press.
5. Gerhard Brewka and Thomas Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, April 1999.
6. Gerhard Brewka, Ilkka Niemela, and Tommi Syrjanen. Implementing ordered disjunction using answer set solvers for normal programs. In Flesca et al. [17], pages 444–455.
7. Francesco Buccafurri, Wolfgang Faber, and Nicola Leone. Disjunctive logic programs with inheritance. In Danny De Schreye, editor, *Logic Programming: The 1999 International Conference*, pages 79–93, Las Cruces, New Mexico, December 1999. MIT Press.
8. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Strong and weak constraints in disjunctive datalog. In *Proceedings of the 4th International Conference on Logic Programming (LPNMR '97)*, pages 2–17, 1997.
9. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Disjunctive ordered logic: Semantics and expressiveness. In Anthony G. Cohn, Lenhard K. Schubert, and Stuart C. Shapiro, editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning*, pages 418–431, Trento, June 1998. Morgan Kaufmann.
10. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing disjunctive datalog by constraints. *Knowledge and Data Engineering*, 12(5):845–860, 2000.
11. Marina De Vos and Dirk Vermeir. Choice Logic Programs and Nash Equilibria in Strategic Games. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 266–276, Madrid, Spain, 1999. Springer Verslag.
12. Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. The diagnosis frontend of the dlv system. *AI Communications*, 12(1-2):99–111, 1999.
13. Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. Planning under incomplete knowledge. In John W. Lloyd, Verónica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luís Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *Proceedings of the First International Conference on Computational Logic (CL2000)*, volume 1861 of *Lecture Notes in Computer Science*, pages 807–821. Springer, 2000.
14. Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. The DLV^k planning system. In Flesca et al. [17], pages 541–544.
15. Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits. Considerations on updates of logic programs. In Ojeda-Aciego et al. [23], pages 2–20.
16. Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Representing school timetabling in a disjunctive logic programming language. In *Proceedings of the 13th Workshop on Logic Programming (WLP '98)*, 1998.

17. Sergio Flesca, Sergio Greco, Nicola Leone, and Giovambattista Ianni, editors. *Logic in Artificial Intelligence*, volume 2424 of *Lecture Notes in Artificial Intelligence*, Cosenza, Italy, September 2002. Springer Verlag.
18. D. Gabbay, E. Laenens, and D. Vermeir. Credulous vs. Sceptical Semantics for Ordered Logic Programs. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 208–217, Cambridge, Mass, 1991. Morgan Kaufmann.
19. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, pages 1070–1080, Seattle, Washington, August 1988. The MIT Press.
20. Robert A. Kowalski and Fariba Sadri. Logic programs with exceptions. In David H. D. Warren and Peter Szeredi, editors, *Proceedings of the 7th International Conference on Logic Programming*, pages 598–613, Jerusalem, 1990. The MIT Press.
21. Els Laenens and Dirk Vermeir. A logical basis for object oriented programming. In Jan van Eijck, editor, *European Workshop, JELIA 90*, volume 478 of *Lecture Notes in Artificial Intelligence*, pages 317–332, Amsterdam, The Netherlands, September 1990. Springer Verlag.
22. Vladimir Lifschitz. Answer set programming and plan generation. *Journal of Artificial Intelligence*, 138(1-2):39–54, 2002.
23. Manual Ojeda-Aciego, Inma P. de Guzmán, Gerhard Brewka, and Luíz Moniz Pereira, editors. *Logic in Artificial Intelligence*, volume 1919 of *Lecture Notes in Artificial Intelligence*, Malaga, Spain, September–October 2000. Springer Verlag.
24. Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.
25. Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
26. Chiaki Sakama and Katsumi Inoue. Representing priorities in logic programs. In Michael J. Maher, editor, *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming*, pages 82–96, Bonn, September 1996. MIT Press.
27. T. Soininen and I. Niemelä. Developing a declarative rule language for applications in product configuration. In *Proceedings of the First International Workshop on Practical Aspects of Declarative Languages (PADL '99)*, Lecture Notes in Computer Science, San Antonio, Texas, 1999. Springer Verslag.
28. T. Soininen, I. Niemelä, J. Tiihonen, and R. Sulonen. Representing configuration knowledge with weight constraint rules. In *Proceedings of the AAAI Spring 2001 Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge*, Stanford, USA, 2001.
29. Davy Van Nieuwenborgh and Dirk Vermeir. Preferred answer sets for ordered logic programs. In Flesca et al. [17], pages 432–443.
30. Davy Van Nieuwenborgh and Dirk Vermeir. Order and negation as failure. In Catuscia Palamidessi, editor, *Proceedings of the International Conference on Logic Programming*, volume 2916 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2003.
31. Davy Van Nieuwenborgh and Dirk Vermeir. Ordered diagnosis. In *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR2003)*, volume 2850 of *Lecture Notes in Artificial Intelligence*, pages 244–258. Springer Verlag, 2003.
32. Kewen Wang, Lizhu Zhou, and Fangzhen Lin. Alternating fixpoint theory for logic programs with priority. In *Proceedings of the International Conference on Computational Logic (CL2000)*, volume 1861 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2000.