Preferred Answer Sets

Gerhard Brewka
Universität Leipzig, Institut für Informatik
Abteilung Intelligente Systeme
Augustusplatz 9/10, 04109 Leipzig, Germany
brewka@informatik.uni-leipzig.de
phone: ++49 341 97 32235, fax: ++49 341 97 32299

July 21, 1997

Abstract

The paper shows how Gelfond and Lifschitz's answer set semantics for extended logic programs can be modified to handle prioritized programs. In such programs an ordering on the program rules is used to express preferences. We show how this ordering can be used to define preferred answer sets and thus to increase the set of consequences of a program. We define a strong and a weaker notion of preferred answer sets. The first takes preferences more seriously, the second guarantees existence of preferred answer sets for finite programs possessing at least one answer set.

1 Introduction

Preferences among default rules play an important role in applications of nonmonotonic reasoning. One source of preferences that has been studied intensively is specificity [12, 16, 17]. In case of a conflict between defaults we tend to prefer the more specific one since this default provides more reliable information.

Specificity is an important source of preferences, but not the only one. In the legal domain it may, for instance, be the case that a more general rule is preferred since it represents federal law as opposed to state law [13]. In these cases preferences may be based on some basic principles regulating how conflicts among rules are to be resolved. Also in other application domains, like model based diagnosis or configuration, preferences play a fundamental role and their relevance is well-recognized.

Prioritized versions for most of the existing nonmonotonic logics have been proposed, e.g., prioritized circumscription [10], hierarchic autoepistemic logic [8], or prioritized default logic [11, 4, 1]. Somewhat surprisingly preferences have received much less attention in logic programming. In a related paper [5] the author has defined a prioritized version of well-founded semantics for extended logic programs. In this paper we investigate prioritized programs under answer set semantics [7], the second standard semantics for extended logic programs.

Let us first characterize somewhat more precisely what we want to achieve. We consider programs with an additional strict partial order on the rules. This additional information is used to solve potential conflicts, that is, we want to conclude more than in standard answer set semantics. On the other hand, unless the program with the given preference information is unsatisfiable (in a sense to be made precise) we want to conclude only literals that are contained in at least one answer set. The best way to achieve these goals is to use the preferences on rules for selecting a subset of the answer sets, the preferred answer sets. The definition of preferred answer sets is the main contribution of this paper.

As mentioned before a number of prioritized versions of default logic [14] have been defined. It is well-known that extended logic programs can be translated to default theories in a straightforward manner, and that the answer sets of a program are in 1-to-1 correspondance with the extensions of the respective default theory. One may thus conclude that the existing prioritized default logics are sufficient to specify preferred answer sets. However, it turns out that all these approaches are unsatisfactory. This will be demonstrated in Sect. 3.

Sect. 4 develops our new approach. We define a strong and a somewhat weaker notion of preferred answer sets. The strong notion takes preferences more seriously but may lead to a situation where programs possessing answer sets do not have preferred answer sets. This is avoided in the weaker notion.

Sect. 5 discusses some related literature and concludes. Sect. 2 recalls the definitions of extended logic programs and answer sets and may safely be skipped by readers familiar with these notions.

To simplify presentation we will only consider finite propositional programs in this paper.

2 Answer sets for extended logic programs: short reminder

A (propositional) extended logic program P is a collection of rules of the form

$$c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m$$

where the a_i, b_j and c are propositional literals, i.e., either propositional atoms or such atoms preceded by the classical negation sign $\neg$. The symbol *not* denotes negation by failure (weak negation), $\neg$ denotes strong negation (sometimes called explicit negation). We will call $a_1, \ldots, a_n$ the prerequisites of the rule. A rule is called prerequisite-free if $n = 0$.

We say a rule $r \in P$ of the form above is defeated by a literal l if $l = b_i$ for some $i \in \{1, \ldots, m\}$. We say r is defeated by a set of literals X if X contains a literal that defeats r.

Definition 1 *Let P be a logic program, X a set of literals. The X-reduct of P, denoted P^X, is the program obtained from P by*

- *deleting each rule defeated by X, and*
- *deleting all weakly negated literals from the remaining rules.*

This reduction of P is often called Gelfond-Lifschitz reduction, after its inventors.

Definition 2 *Let R be a set of rules without negation as failure. $Cn(R)$ denotes the smallest set of literals that is*

1. *closed under R, and*
2. *logically closed, i.e., either consistent or equal to the set of all literals.*

Definition 3 *Let P be a logic program, X a set of literals. Define an operator γ_P as follows:*

$$\gamma_P(X) = Cn(P^X)$$

X is an answer set of P iff $X = \gamma_P(X)$.

A literal l is a consequence of a program P under answer set semantics, denoted $l \in Ans(P)$, iff l is contained in all answer sets of P. For more on answer sets consult [7].

3 Why existing DL-approaches do not work

Different prioritzed versions of Reiter's default logic [14] have been proposed in the literature. We will show in this section that all of them suffer from weaknesses and thus cannot serve - via the standard translation from extended logic programs to default theories - as a satisfactory specification of preferred answer sets.

The first group of proposals [11, 4, 1] uses preferences to control the quasi-inductive definition of extensions [14]: in each step of the generation of extensions the defaults with highest priority whose prerequisites have already been derived are applied. Although this sounds like a very natural idea it leads to unwanted results. To show this we first have to make more precise what we count as unwanted.

People have different intuitions about the meaning of preferences among defaults. Nevertheless, it appears to us that the following principle constitutes a minimal requirement for a satisfactory treatment of preferences in default logic:

> Principle I:
> Let (D, W) be a default theory possessing an extension E_1 generated by the defaults $D' \cup \{d_1\}$ and an extension E_2 generated by the defaults $D' \cup \{d_2\}$.[1] If d_1 is preferred over d_2 then E_2 should not be maximally preferred.

It is hard to see that the use of the term "preference" could be justified when this principle is violated. Nevertheless, this is exactly what happens in the mentioned approaches. Here is an example:

1) $a : b/b$
2) $: \neg b/\neg b$
3) $: a/a$

The extensions of this default theory are $E_1 = \{a, , b\}$ and $E_2 = \{a, \neg b\}$. Although the mentioned approaches differ in technical detail they agree on the example and prefer E_2 thus violating our principle. One might say that in cases like this the preferred extensions do not take seriously what they believe: it may be the case that a less preferred default is applied although the prerequisite of a conflicting, more preferred default is believed in a preferred extension.

[1] The generating defaults for an extension E are the defaults $a : b/c$ with $a \in E$ and $\neg b \notin E$. For readers not familiar with default logic: a default rule $a{:}b/c$ corresponds to $c \leftarrow not\neg b, a$.

For people who like it less abstract we include a bird example of similar structure. Assume we are given the following default theory:

1) $bird$
2) $swims$
3) $peng : \neg flies/\neg flies$
4) $bird : flies/flies$
5) $bird \wedge swims : peng/peng$

Assume further that 3 is preferred over 4 and 4 over 5. The first preference is based on specificity, the low preference of 5 is based on the fact that the rule provides weak evidence to its conclusion only. The default theory has two Reiter extensions, namely $E_1 = Th(\{bird, swims, \neg flies, penguin\})$ and $E_2 = Th(\{bird, swims, flies, penguin\})$. The single preferred extension in all mentioned approaches is E_2 violating Principle I. The reason is that the prerequisite of 4 is derived before the prerequisite of 3 in the construction of the extension.

This unpleasant behaviour was already observed in [4]. In that paper the author tried to defend his approach arguing that there is only weak evidence for *peng*. We do not support this argument any longer, however. After all, default logic is not a logic of graded belief where degrees of evidence should play a role. Default logic models acceptance of belief based on defeasible arguments. Since *peng* is an accepted belief rule 3 should be applied and E_1 should be the preferred extension in the example.

A different approach was proposed in [15]. Rintanen uses a total order on (normal) defaults to induce a lexicographic order on extensions. Let $gen(E)$ denote the generating defaults for an extension E. E is then preferred over extension E' iff there is a default $r \in gen(E) \setminus gen(E')$ satisfying the condition: if r' is preferred over r and $r' \in gen(E')$ then $r' \in gen(E)$.

Unfortunately also this leads to counterintuitive results. We first formulate a second principle. The intuition behind this principle is the following: extensions should not be blamed for not applying a rule if they do not contain the prerequisite of this rule, or, in other words, conflicts should be resolved based on the preferences of those rules which are in conflict, not on the grounds that a certain decision makes other defaults applicable.

Principle II:
Let E be a preferred extension of (D, W), $d = a : b/c$ a default such that $a \notin E$. Then E is a preferred extension of $(D \cup \{d\}, W)$.

We show that Rintanen's approach violates Principle II. Consider the following default theory:[2]

1) q
2) r
3) $p : s/s$
4) $r : \neg p/\neg p$
5) $q : p/p$

Assume 3 is preferred over 4, 4 over 5. This default theory has two Reiter extensions, namely $E_1 = Th(\{q, r, \neg p\})$ and $E_2 = Th(\{q, r, p, s\})$. Intuitively, since the decision whether to believe p or not depends on 4 and 5 only, and since 4 is preferred over 5, we would expect to conclude $\neg p$, in other words, to prefer E_1.

Rintanen, however, prefers E_2 violating Principle II. The reason is that in E_2 default 3 is applied. Belief in p is thus accepted on the grounds that this allows us to apply a default of high priority. This is far from being plausible and amounts to wishful thinking.

Since all these approaches suffer from drawbacks they cannot serve as a basis for a satisfactory definition of preferred answer sets. Our new proposal will be developed in the following section.

4 Preferred answer sets

A prioritized logic program is a pair $(P, <)$ where P is an (extended) logic program and $<$ a strict partial order on P. $<$ is used to express preference information: $r_1 < r_2$ stands for "r_1 has higher priority than r_2". Our goal is to use this information to define the notion of a preferred answer set.

The definition of preferred answer sets for partially ordered programs can be reduced to the totally ordered case in a canonical way: A is a preferred answer set of $(P, <)$ iff there is a total order $<_t$ such that $<\subseteq<_t$ and A is a preferred answer set of $(P, <_t)$. We thus can restrict our discussion to totally ordered programs in the rest of the paper. The following result is immediate:

Proposition 1 *Let $<_1$ and $<_2$ be partial orders on a program P such that $<_1\subseteq<_2$. Every preferred answer set of $(P, <_2)$ is a preferred answer set of $(P, <_1)$.*

Recall that in the definition of answer sets the general case is reduced to a particularly easy special case, namely logic programs without weak negation.

[2]People who prefer interpreting symbols may read q as "quaker", p as "pacifist", r as "republican" and s as "does not like the army".

To check whether A is an answer set one has to reduce the program P with A. Then one checks whether the reduced program has consequences A.

For a given answer set A there also is a special case where it is particularly easy to check whether the preference ordering was taken into account adequately: programs without prerequisites, that is programs where the bodies of all rules contain weakly negated literals only. In this case we simply have to check whether each rule which was not applied to generate A is defeated by the consequences of applied rules of higher priority.

To do this we introduce, for prerequisite-free programs, an operator $C_<$ as follows (we assume that r_i denotes the i-th element of the total order $<$): $C_<(P)$ is the smallest logically closed superset of $\bigcup_{i=0}^{\infty} S_i$ where

$$\begin{array}{rcl} S_0 & = & \emptyset \\ S_i & = & \begin{cases} S_{i-1} & \text{if } r_i \text{ is defeated by } S_{i-1}, \\ S_{i-1} \cup \{head(r_i)\} & \text{otherwise.} \end{cases} \end{array}$$

Here $head(r)$ stands for the head of rule r. Note that this operator is not meant to derive consequences of P. It may well be the case that a rule is applied to produce $C_<(P)$ although the rule is later defeated by less preferred rules. However, if the resulting set *is* an answer set then we can be sure that all preferences were taken into account adequately.

In the case of arbitrary programs we perform a reduction which can be viewed as dual to the Gelfond-Lifschitz reduction: given a set of literals A we eliminate rules whose prerequisites are not in A and eliminate all prerequisites from the remaining rules. This clearly yields a prerequisite-free program to which $C_<$ can be applied. Intuitively, our construction amounts to guessing provable prerequisites and checking whether the assumption that exactly these prerequisites hold is possible under the prioritized interpretation of rules.

Definition 4 *Let P be an extended logic program, X a set of literals. Let ${}^XP$ be the program obtained from P by*

1. *deleting all rules with prerequisite L such that $L \notin X$, and*
2. *deleting all prerequisites from the remaining rules.*

Definition 5 *Let P be an extended logic program, X a set of literals. We define an operator $\lambda_<$ as follows:*

$$\lambda_<(X) = C_<({}^XP)$$

With these definitions we can introduce a first notion of preferred answer sets:

Definition 6 *Let $(P, <)$ be a prioritized logic program. A is a strongly preferred answer set of $(P, <)$ iff*

1. $\gamma_P(A) = A$, *and*
2. $\lambda_<(A) = A$.

Let us discuss some examples. In all examples we will assume that the total ordering is given implicitly by the numbering of the rules, i.e., rules with lower number are preferred to those with higher number. Here is the Tweety example:

1) p
2) b
3) $\neg f \leftarrow not\ f, p$
4) $f \leftarrow not\ \neg f, b$

We obtain two answer sets: $A_1 = \{p, b, \neg f\}$ and $A_2 = \{p, b, f\}$. We have $\lambda_<(A_2) = A_1 \neq A_2$ and thus A_2 is not strongly preferred. As intended A_1 is strongly preferred as this set is a fixed point of $\lambda_<$.

We now show that the problems with the DL-approaches discussed in Section 2 are solved in our approach. Here is the logic programming version of the swimming bird example (using obvious abbreviations):

1) b
2) s
3) $\neg f \leftarrow not\ f\ , p$
4) $f \leftarrow not \neg f, b$
5) $p \leftarrow not \neg p, b, s$

The program has two answer sets: $A_1 = \{s, b, \neg f, p\}$ and $A_2 = \{s, b, f, p\}$. As intended A_2 is not strongly preferred since $\lambda_<(A_2) = A_1$. The single strongly preferred answer set is A_1.

Also the counterexample to Rintanen is handled adequately:

1) q
2) r
3) $s \leftarrow not \neg s, p$
4) $\neg p \leftarrow not\ p\ , r$
5) $p \leftarrow not \neg p, q$

Again we obtain two answer sets, namely $A_1 = \{q, r, \neg p\}$ and $A_2 = \{q, r, p, s\}$. A_1 is strongly preferred, A_2 is not since $\lambda_<(A_2) = \{q, r, \neg p, s\}$. This is exactly what we expect.

In fact, we can show that the two principles we formulated in Sect. 3 are satisfied in our approach. The proof of the following proposition is not difficult:

Proposition 2 *The strongly preferred answer set approach satisfies Principle I and Principle II (as formulated in Sect. 3, with "default theory" replaced by "prioritized logic program", "extension" replaced by "answer set", and "default" replaced by "rule").*

Strongly preferred answer sets are not always unique, even for totally ordered programs, as the next example demonstrates:

1) $b \leftarrow not \neg b, a$
2) $c \leftarrow not\ b$
3) $a \leftarrow not\ c$

The program has two answer sets: $A_1 = \{a, b\}$ and $A_2 = \{c\}$. Both are strongly preferred.

It is not difficult to see that there are also programs with answer sets for which no strongly preferred answer set exists. Here is an example:

1) $a \leftarrow not\ b$
2) $b \leftarrow not\ a$
3) $c \leftarrow not\ c\ , a$

The single answer set of this program is $A = \{b\}$. However, $\lambda_<(A) = \{a\}$ and thus A is not strongly preferred.

Two views seem possible here: one might say that this is exactly what is expected in a situation like this: the priorities cannot be satisfied and thus the prioritized program should be considered inconsistent. This corresponds to a rather strict interpretation of the meaning of priorities.

However, one might also argue that the preference information should help us to select among the existing answer sets, but never render an otherwise consistent program inconsistent. In the rest of this section we show how this view can be implemented in our approach.

The basic idea is to consider in the comparison of answer sets the degree to which preferences are violated. To do so we need to define the distance between two orderings.

Definition 7 *Let $r_{1,1} <_1 \ldots <_1 r_{1,k}$ and $r_{2,1} <_2 \ldots <_2 r_{2,k}$ be total orderings on a (nonempty) finite set M with $|M| = k$. We define the distance between $<_1$ and $<_2$, denoted $D(<_1, <_2)$, inductively as follows:*

$$D(<_1, <_2) = \begin{cases} 0 & \textit{if } k = 1 \\ (j-1) + D(<'_1, <'_2) & \textit{otherwise.} \end{cases}$$

where $r_{1,1} = r_{2,j}$ (i.e., j is the position of the first element of $<_1$ in $<_2$) and $<'_1$ and $<'_2$ are the restrictions of the respective orderings to $M - \{r_{1,1}\}$.

Intuitively, the distance between two orderings is the number of position switches needed to transform the second into the first.

With this notion we can weaken the definition of strongly preferred answer sets:

Definition 8 *Let $(P, <)$ be a prioritized logic program, A an answer set of P. The preference violation degree of A, denoted $pvd(A)$, is the minimal distance between $<$ and an ordering $<'$ such that A is a strongly preferred answer set of $(P, <')$.*

It is obvious that the *pvd* is defined for each answer set since each such set is strongly preferred according to some total order (just let all rules applied in A have preference over the rules defeated by A).

Definition 9 *Let $(P, <)$ be a prioritized logic program. A is a preferred answer set of $(P, <)$ iff for no answer set A' we have $pvd(A') < pvd(A)$.*

The following results are immediate form these definitions:

Proposition 3 *The preferred and the strongly preferred answer sets of a prioritized logic program $(P, <)$ coincide whenever the latter exist.*

Strongly preferred answer sets are just those with preference violation degree zero.

Proposition 4 *Let $(P, <)$ be a prioritized logic program. If $(P, <)$ possesses at least one answer set then it possesses at least one preferred answer set.*

Note that the existence of preferred answer sets is guaranteed since we consider finite programs only (for the infinite case the possible existence of infinite descending chains needs to be taken into account). In particular, whenever there is only one single answer set (as in our example) this set will be preferred.

5 Related work and conclusion

Several approaches treating preferences in the context of logic programming have been described in the literature. We will briefly discuss how they relate to our proposal.

Kowalski and Sadri [9] proposed to consider rules with negation in the head as exceptions to more general rules and to give them higher priority. Technically, this is achieved by a redefinition of answer sets. It turns out that the original answer sets remain answer sets according to the new definition whenever they are consistent. The main achievement is that programs whose single answer set is inconsistent become consistent in the new semantics. The approach can hardly be viewed as a satisfactory treatment of preferences for several reasons:

1. preferences are implicit and highly restricted; the asymmetric treatment of positive and negative information seems unjustified,
2. it is difficult to see how, for instance, exceptions of exceptions can be represented,
3. fewer conclusions are obtained than in the original answer set semantics, contrary to what one would expect when preferences are taken into account.

It is, therefore, more reasonable to view Kowalski and Sadri's approach as a contribution to inconsistency handling rather than preference handling.

An approach that is closer in spirit to ours is ordered logic programming [6]. An ordered logic program is a set of components forming an inheritance hierarchy. Each component consists of a set of rules. The inheritance hierarchy is used to settle conflicts among rules: rules lower in the hierarchy have preference over those higher up in the hierarchy since the former are considered more specific. A notion of a stable model for ordered logic programs can be defined (see [6] for the details). There are two main differences between ordered logic programs and our approach:

1. the preferences of ordered logic programs are predefined through the inheritance hierarchy,
2. ordered logic programs use only one kind of negation, the distinction between negation as failure and classical negation is not expressible in the language.

In [5] the author has presented a prioritized version of well-founded semantics for extended logic programs. In that paper the preference information was expressed in the logical language and could thus be derived dynamically. Here we did not want to complicate matters and used an additional ordering relation outside the language. However, the representation in the language is not much more difficult and can be done along the lines of [3]: if preferences are represented in the language then an answer set A is strongly preferred if there is a total order $<$ of the program rules such that $<$ is compatible with the preference information in A and $\lambda_<(A) = A$. Intuitively, $<$ is compatible with A if the addition of a syntactic representation of $<$ to A does not lead to inconsistency.

A final remark about complexity: Eiter (personal communication) has recently shown that the problem whether a literal L belongs to all preferred answer sets of a prioritized logic program is Θ_2^p-complete. Θ_2^p is the class of problems decidable in polynomial time with logarithmically many calls to an NP oracle, measured in the size of the input.

Acknowledgements

I would like to thank Michael Gelfond and Heinrich Herre for helpful and clarifying discussions on the topic of the paper.

References

[1] Baader, Franz, Hollunder, Bernhard, Priorities on Defaults with Prerequisite and their Application in Treating Specificity in Terminological Default Logic, Journal of Automated Reasoning, 15:41-68, 1995

[2] Baral, C., Gelfond, M., Logic Programming and Knowledge Representation, Journal of Logic Programming, 19,20:73-148, 1994

[3] Brewka, Gerhard, Reasoning About Priorities in Default Logic, Proc. AAAI-94, Seattle, 1994

[4] Brewka, Gerhard, Adding Priorities and Specificity to Default Logic, Proc. JELIA 94, Springer LNAI 838, 247-260, 1994

[5] Brewka, Gerhard, Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences, Journal of Artificial Intelligence Research, (1996), 19-36

[6] Buccafurri, F., Leone, N., Rullo, P., Stable Models and their Computation for Logic Programming with Inheritance and True Negation, Journal of Logic Programming, to appear, 1995

[7] Gelfond, M., Lifschitz, V., Logic Programs with Classical Negation, Proc. 7th Intl. Conference on Logic Programming, 1990

[8] Konolige, K., Hierarchic Autoepistemic Theories for Nonmonotonic Reasoning, Proc. AAAI, 1988

[9] Kowalski, R.A., Sadri, F., Logic Programs with Exceptions, New Generation Computing, 9 (1991), 387-400

[10] Lifschitz, V., Computing Circumscription, Proc. 9th Int. Joint Conference on Artificial Intelligence, 1985

[11] Marek, Wiktor, Truszczynski, Mirek, Nonmonotonic Logic: Context-Dependent Reasoning, Springer, 1993

[12] Poole, David, On the Comparison of Theories: Preferring the Most Specific Explanation, Proc. IJCAI-85, Los Angeles, 1985

[13] Prakken, Henry, Logical Tools for Modelling Legal Argument, dissertation, VU Amsterdam, 1993

[14] Reiter, Raymond, A Logic for Default Reasoning, Artificial Intelligence 13 (1980) 81-132.

[15] Rintanen, Jussi, On Specificity in Default Logic, Proc. International Joint Conference on Artificial Intelligence, Montreal, 1995, 1474-1479

[16] Touretzky, David S., The Mathematics of Inheritance, Pitman Research Notes in Artificial Intelligence, London, 1986

[17] Touretzky, David S., Thomason, Richmond H., Horty, John F., A Skeptic's Menagerie: Conflictors, Preemptors, Reinstaters, and Zombies in Nonmonotonic Inheritance, Proc. 12th IJCAI, Sydney, 1991