A Comparative Study of Logic Programs with Preference: Preliminary Report

Torsten Schaub[*] **and Kewen Wang**[†]
Institut für Informatik
Universität Potsdam
Postfach 60 15 53, D–14415 Potsdam
Germany
{torsten,kewen}@cs.uni-potsdam.de

Abstract

We are interested in semantical underpinnings for existing approaches to preference handling in extended logic programming (within the framework of answer set programming). As a starting point, we explore three different approaches that have been recently proposed in the literature. Because these approaches use rather different formal means, we furnish a series of uniform characterizations that allow us to gain insights into the relationships among these approaches. To be more precise, we provide different characterizations in terms of (i) fixpoints, (ii) order preservation, and (iii) translations into standard logic programs. While the two former provide semantics for logic programming with preference information, the latter furnishes implementation techniques for these approaches.

Introduction

Numerous approaches to logic programming with preference information have been proposed in the literature. So far, however, there is no systematic account on their structural differences, finally leading to solid semantical underpinnings. We address this shortcoming by a comparative study of a distinguished class of approaches to preference handling. This class consists of *selective* approaches remaining within the complexity class of extended logic programming (with answer sets). These approaches are selective insofar as they use preferences to distinguish certain "models" of the original program.

As a starting point, we explore three different approaches that have been recently proposed in the literature, namely the ones in (Brewka and Eiter 1999; Delgrande *et al.* 2000; Wang *et al.* 2000). Our investigation adopts characterization techniques found in the same literature in order to shed light on the relationships among these approaches. This provides us with different characterizations in terms of (i) fixpoints, (ii) order preservation, and (iii) translations into standard logic programs. While the two former provide semantics for logic programming with preference information, the latter furnishes implementation techniques for these approaches. From another perspective, one can view (iii) as an axiomatization of the underlying strategy within the object language, while (i) may be regarded as a meta-level description of the corresponding construction process. One may view (ii) as the most semantical characterization because it tells us which "models" of the original program are selected by the respective preference handling strategy.

We limit (also in view of (iii)) our investigation to approaches to preference handling that remain within NP. This excludes approach like the ones in (Rintanen 1995; Zhang and Foo 1997) that step outside the complexity class of the underlying reasoning method. This applies also to the approach in (Sakama and Inoue 1996), where preferences on literals are investigated. While the approach of (Gelfond and Son 1997) remains within NP, it advocates strategies that are non-selective. Approaches that can be addressed within this framework include (Baader and Hollunder 1993; Brewka 1994) that were originally proposed for default logic.

Proofs can be found in the full version of this paper.

Definitions and notation

We assume a basic familiarity with logic programming under *answer set semantics* (Gelfond and Lifschitz 1991). An *extended logic program* is a finite set of rules of the form

$$L_0 \leftarrow L_1, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n, \qquad (1)$$

where $n \geq m \geq 0$, and each L_i $(0 \leq i \leq n)$ is a *literal*, ie. either an atom A or the negation $\neg A$ of A. The set of all literals is denoted by Lit. Given a rule r as in (1), we let $\mathit{head}(r)$ denote the *head*, L_0, of r and $\mathit{body}(r)$ the *body*, $\{L_1, \ldots, L_m,\ \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n\}$, of r. Further, let $\mathit{body}^+(r) = \{L_1, , \ldots, L_m\}$ and $\mathit{body}^-(r) = \{L_{m+1}, \ldots, L_n\}$. A program is called *basic* if $\mathit{body}^-(r) = \emptyset$ for all its rules.

We define the reduct of a rule r as $r^+ = \mathit{head}(r) \leftarrow \mathit{body}^+(r)$. The *reduct*, Π^X, of a program Π *relative to* a set X of literals is defined by

$$\Pi^X = \{r^+ \mid r \in \Pi \text{ and } \mathit{body}^-(r) \cap X = \emptyset\}.$$

A set of literals X is *closed under* a basic program Π iff for any $r \in \Pi$, $\mathit{head}(r) \in X$ whenever $\mathit{body}^+(r) \subseteq X$. We

[*] Affiliated with the School of Computing Science at Simon Fraser University, Burnaby, Canada.

[†] On leave from Tsinghua University, Beijing, China.

say that X is *logically closed* iff it is either consistent (ie. it does not contain both a literal A and its negation $\neg A$) or equals Lit. The smallest set of literals which is both logically closed and closed under a basic program Π is denoted by $Cn(\Pi)$. Finally, a set X of literals is an *answer set* of a program Π iff $Cn(\Pi^X) = X$.

The set Γ_Π^X of all *generating rules* of an answer set X from Π is given by

$$\Gamma_\Pi^X = \{r \in \Pi \mid \mathit{body}^+(r) \subseteq X \text{ and } \mathit{body}^-(r) \cap X = \emptyset\}.$$

As van Gelder in (1993), we define $C_\Pi(X) = Cn(\Pi^X)$. Note that the operator C_Π is anti-monotonic, which implies that the operator $A_\Pi(X) = C_\Pi(C_\Pi(X))$ is monotonic. A fixpoint of A_Π is called an *alternating fixpoint* for Π. Different semantics are captured by distinguishing different groups of fixpoints of A_Π.

A (*statically*) *ordered logic program*[1] is a pair $(\Pi, <)$, where Π is an extended logic program and $< \subseteq \Pi \times \Pi$ is an irreflexive and transitive relation. Given, $r_1, r_2 \in \Pi$, the relation $r_1 < r_2$ expresses that r_2 has higher priority than r_1.[2]

Preferred alternating fixpoints

The notion of answer sets (without preference) is based on a reduction of extended logic programs to basic programs (without default negation). Such a reduction is inapplicable when addressing conflicts by means of preference information since all conflicts between rules are simultaneously resolved when turning Π into Π^X. Rather conflict resolution must be characterized among the original rules in order to account for blockage between rules. That is, once the negative body $\mathit{body}^-(r)$ is eliminated there is no way to detect whether $\mathit{head}(r') \in \mathit{body}^-(r)$ holds in case of $r < r'$.

Such an approach is pursued in (Wang *et al.* 2000) for characterizing "preferred" answer sets. Following earlier approaches based on default logic (Baader and Hollunder 1993; Brewka 1994), this approach is based on the concept of *activeness*: Let $X, Y \subseteq Lit$ be two sets of literals in an ordered logic program $(\Pi, <)$. A rule r in Π is *active* wrt the pair (X, Y), if $\mathit{body}^+(r) \subseteq X$ and $\mathit{body}^-(r) \cap Y = \emptyset$.

Definition 1 (Wang *et al.*,2000) *Let $(\Pi, <)$ be an ordered logic program and let X be a set of literals. We define*

$$\begin{array}{rcl} X_0 & = & \emptyset \quad \textit{and for } i \geq 0 \\ X_{i+1} & = & X_i \cup \{\mathit{head}(r) \mid \\ & & \left.\begin{array}{ll} I. & r \in \Pi \textit{ is active wrt } (X_i, X) \textit{ and} \\ II. & \textit{there is no rule } r' \in \Pi \textit{ with } r < r' \\ & \textit{such that} \\ & (a)\ r' \textit{ is active wrt } (X, X_i) \textit{ and} \\ & (b)\ \mathit{head}(r') \notin X_i \end{array}\right\} \end{array}$$

Then, $\mathcal{C}_{(\Pi,<)}(X) = \bigcup_{i \geq 0} X_i$ if $\bigcup_{i \geq 0} X_i$ is consistent. Otherwise, $\mathcal{C}_{(\Pi,<)}(X) = Lit$.

The idea is to apply a rule r only if the question of application has been settled for all higher-ranked rules r'. That is, if either its prerequisites will never be derivable, viz. $\mathit{body}^+(r') \not\subseteq X$, or r' is defeated by what has been derived so far, viz. $\mathit{body}^-(r) \cap X_i \neq \emptyset$, or r' or another rule with the same head have already applied, viz. $\mathit{head}(r') \in X_i$.

As its original C_Π, the operator $\mathcal{C}_{(\Pi,<)}$ is anti-monotonic. Accordingly, we may define for any set $X \subseteq Lit$, the *alternating transformation* of $(\Pi, <)$ as $\mathcal{A}_{(\Pi,<)}(X) = \mathcal{C}_{(\Pi,<)}(\mathcal{C}_{(\Pi,<)}(X))$. A fixpoint of $\mathcal{A}_{(\Pi,<)}$ is called an *alternating fixpoint* of $(\Pi, <)$. Note that $\mathcal{A}_{(\Pi,<)}$ is monotonic.

Now, in analogy to Van Gelder (1993), a semantical framework for ordered logic programs in terms of sets of alternating fixpoints can be defined. Three different types of semantics are investigated in (Wang *et al.* 2000): (i) Preferred[3] answer sets, viz. alternating fixpoints being also fixpoints of $\mathcal{C}_{(\Pi,<)}$. (ii) Preferred regular extensions, viz. maximal normal[4] alternating fixpoints of $(\Pi, <)$. (iii) Preferred well-founded model, viz. the least alternating fixpoint of $(\Pi, <)$.

We put the prefix 'W-' whenever a distinction to other approaches is necessary.

For illustration, consider the following ordered logic program $(\Pi_2, <)$ due to (Baader and Hollunder 1993):

$$\begin{array}{rrcl} r_1: & \neg f & \leftarrow & p, \mathit{not}\ f \\ r_2: & w & \leftarrow & b, \mathit{not}\ \neg w \\ r_3: & f & \leftarrow & w, \mathit{not}\ \neg f \\ r_4: & b & \leftarrow & p \\ r_5: & p & \leftarrow & \end{array} \qquad r_2 < r_1 \qquad (2)$$

Observe that Π_2 admits two answer sets: $X = \{p, b, \neg f, w\}$ and $X' = \{p, b, f, w\}$. As argued in (Baader and Hollunder 1993), X is the unique W-preferred answer set. To see this, observe that

$$\begin{array}{rcl@{\qquad\qquad}rcl} X_0 & = & \emptyset & X'_0 & = & \emptyset \\ X_1 & = & \{p\} & X'_1 & = & \{p\} \\ X_2 & = & \{p, b, \neg f\} & X'_2 & = & \{p, b\} \\ X_3 & = & \{p, b, \neg f, w\} & X'_3 & = & X'_2 \neq X' \\ X_4 & = & X_3 = X & & & \end{array}$$

Note that w cannot be included into X'_3 because r_1 is active wrt (X', X'_2) and r_1 is preferred to r_2.

Compiling order preservation

A translation of ordered logic programs $(\Pi, <)$ to standard ones Π' is developed in (Delgrande *et al.* 2000). The specific strategy used there ensures that the resulting program Π' admits only those answer sets of the original program Π that are *order preserving*:

Definition 2 *Let $(\Pi, <)$ be a statically ordered program and let X be an answer set of Π.*

Then, X is called $<$-preserving, if X is either inconsistent, or there exists an enumeration $\langle r_i \rangle_{i \in I}$ of Γ_Π^X such that for every $i, j \in I$ we have that:

0. $\mathit{body}^+(r_i) \subseteq \{\mathit{head}(r_j) \mid j < i\}$*; and*

[1] Also called *prioritized* logic program by some authors, as eg. in (Brewka and Eiter 1999).

[2] As opposed to (Brewka and Eiter 1999) that attribute relation $<$ the inverse meaning.

[3] Originally called *prioritized*.

[4] An alternating fixpoint X is normal if $X \subseteq \mathcal{C}_{(\Pi,<)}(X)$.

1. *if* $r_i < r_j$*, then* $j < i$*; and*
2. *if* $r_i < r'$ *and* $r' \in \Pi \setminus \Gamma_\Pi^X$*, then*
   (a) *body*$^+(r') \not\subseteq X$ *or*
   (b) *body*$^-(r') \cap \{head(r_j) \mid j < i\} \neq \emptyset$.

Condition 0 makes the property of *groundedness*[5] explicit. Although any standard answer set is generated by a grounded sequence of rules, we will see in the sequel that this property is weakened when preferences are at issue. Condition 1 stipulates that $\langle r_i \rangle_{i \in I}$ is *compatible* with $<$, a property invariant to all of the considered approaches. Lastly, Condition 2 is comparable with Condition II in Definition 1; it guarantees that rules can never be blocked by lower-ranked rules.

As above, $X = \{p, b, \neg f, w\}$ is the only $<$-preserving answer set of Π_2; it can be generated by the grounded sequences $\langle r_5, r_4, r_1, r_2 \rangle$ and $\langle r_5, r_1, r_4, r_2 \rangle$ both of which satisfy conditions 1 and 2. The only grounded sequence generating $X' = \{p, b, f, w\}$, namely $\langle r_5, r_4, r_2, r_3 \rangle$, violates 2b.

The corresponding translation integrates ordering information into the logic program via a special-purpose predicate symbol $\prec$. This allows also for treating ordering information in a dynamic fashion. A logic program over a propositional language $\mathcal{L}$ is said to be *dynamically* ordered iff $\mathcal{L}$ contains the following pairwise disjoint categories: (i) a set $\mathcal{N}$ of terms serving as *names* for rules; (ii) a set $\mathbf{A}$ of (propositional) atoms of a program; and (iii) a set $\mathbf{A}_\prec$ of *preference atoms* $s \prec t$, where $s, t \in \mathcal{N}$ are names. For each such program Π, we assume furthermore a bijective function $n(\cdot)$ assigning to each rule $r \in \Pi$ a name $n(r) \in \mathcal{N}$. To simplify notation, we usually write n_r instead of $n(r)$ (and we sometimes abbreviate n_{r_i} by n_i).

An atom $n_r \prec n_{r'} \in \mathbf{A}_\prec$ amounts to asserting that $r < r'$ holds. A statically ordered program $(\Pi, <)$ can thus be captured by programs containing preference atoms only among their facts; it is then expressed by the program $\Pi \cup \{(n_r \prec n_{r'}) \leftarrow \ \mid r < r'\}$.

Given $r < r'$, one wants to ensure that r' is considered before r, in the sense that, for a given answer set X, rule r' is known to be applied or defeated *ahead of* r (cf. Condition II or 2 above, respectively). This is done by translating rules so that the order of rule application can be explicitly controlled. For this purpose, one needs to be able to detect when a rule has been applied or when a rule is defeated. For a rule r, there are two cases for it not to be applied: it may be that some literal in *body*$^+(r)$ does not appear in the answer set, or it may be that a literal in *body*$^-(r)$ is in the answer set. For detecting non-applicability (i.e., blockage), for each rule r in the given program Π, a new, special-purpose atom $\mathsf{bl}(n_r)$ is introduced. Similarly, a special-purpose atom $\mathsf{ap}(n_r)$ is introduced to detect the case where a rule has been applied. For controlling application of rule r the atom $\mathsf{ok}(n_r)$ is introduced. Informally, one concludes that it is ok to apply a rule just if it is ok with respect to every $<$-greater rule; for such a $<$-greater rule r', this will be the case just when r' is known to be blocked or applied.

[5] This term is borrowed from the literature on default logic.

More formally, given a dynamically ordered program Π over $\mathcal{L}$, let $\mathcal{L}^+$ be the language obtained from $\mathcal{L}$ by adding, for each $r, r' \in \Pi$, new pairwise distinct propositional atoms $\mathsf{ap}(n_r)$, $\mathsf{bl}(n_r)$, $\mathsf{ok}(n_r)$, and $\mathsf{ok}'(n_r, n_{r'})$. Then, the translation $\mathcal{T}$ maps an ordered program Π over $\mathcal{L}$ into a standard program $\mathcal{T}(\Pi)$ over $\mathcal{L}^+$ in the following way.

Definition 3 (Delgrande *et al.*,2000) *Let* $\Pi = \{r_1, \ldots, r_k\}$ *be a dynamically ordered logic program over* $\mathcal{L}$.

Then, the logic program $\mathcal{T}(\Pi)$ *over* $\mathcal{L}^+$ *is defined as* $\mathcal{T}(\Pi) = \bigcup_{r \in \Pi} \tau(r)$ *, where* $\tau(r)$ *consists of the following rules, for* $L^+ \in$ *body*$^+(r)$*,* $L^- \in$ *body*$^-(r)$*, and* $r', r'' \in \Pi$ *:*

$$
\begin{array}{rrcl}
a_1(r): & head(r) & \leftarrow & \mathsf{ap}(n_r) \\
a_2(r): & \mathsf{ap}(n_r) & \leftarrow & \mathsf{ok}(n_r), body(r) \\
b_1(r, L^+): & \mathsf{bl}(n_r) & \leftarrow & \mathsf{ok}(n_r), not\ L^+ \\
b_2(r, L^-): & \mathsf{bl}(n_r) & \leftarrow & \mathsf{ok}(n_r), L^- \\
 & & & \\
c_1(r): & \mathsf{ok}(n_r) & \leftarrow & \mathsf{ok}'(n_r, n_{r_1}), \ldots, \mathsf{ok}'(n_r, n_{r_k}) \\
c_2(r, r'): & \mathsf{ok}'(n_r, n_{r'}) & \leftarrow & not\ (n_r \prec n_{r'}) \\
c_3(r, r'): & \mathsf{ok}'(n_r, n_{r'}) & \leftarrow & (n_r \prec n_{r'}), \mathsf{ap}(n_{r'}) \\
c_4(r, r'): & \mathsf{ok}'(n_r, n_{r'}) & \leftarrow & (n_r \prec n_{r'}), \mathsf{bl}(n_{r'}) \\
 & & & \\
t(r, r', r''): & n_r \prec n_{r''} & \leftarrow & n_r \prec n_{r'}, n_{r'} \prec n_{r''} \\
as(r, r'): & \neg(n_{r'} \prec n_r) & \leftarrow & n_r \prec n_{r'}
\end{array}
$$

We write $\mathcal{T}(\Pi, <)$ rather than $\mathcal{T}(\Pi')$, whenever Π' is the dynamically ordered program capturing $(\Pi, <)$.

The first four rules of $\tau(r)$ express applicability and blocking conditions of the original rules. The second group of rules encodes the strategy for handling preferences. The first of these rules, $c_1(r)$, "quantifies" over the rules in Π. This is necessary when dealing with dynamic preferences since preferences may vary depending on the corresponding answer set. The three rules $c_2(r, r')$, $c_3(r, r')$, and $c_4(r, r')$ specify the pairwise dependency of rules in view of the given preference ordering: For any pair of rules r, r' with $n_r \prec n_{r'}$, we derive $\mathsf{ok}'(n_r, n_{r'})$ whenever $n_r \prec n_{r'}$ fails to hold, or whenever either $\mathsf{ap}(n_{r'})$ or $\mathsf{bl}(n_{r'})$ is true. This allows us to derive $\mathsf{ok}(n_r)$, indicating that r may potentially be applied whenever we have for all r' with $n_r \prec n_{r'}$ that r' has been applied or cannot be applied. It is important to note that this is only one of many strategies for dealing with preferences: different strategies are obtainable by changing the specification of $\mathsf{ok}(\cdot)$ and $\mathsf{ok}'(\cdot, \cdot)$, as we will see below.

As shown in (Delgrande *et al.* 2000), a set of literals X is a $<$-preserving answer set of Π iff $X = Y \cap \mathcal{L}$ for some answer set Y of $\mathcal{T}(\Pi, <)$. In the sequel, we refer to such answer sets as being D-*preferred*.

Synthesis

The last two sections have exposed three rather different ways of characterizing preferred answer sets. Despite their different characterizations, however, it turns out that the two approaches prefer similar answer sets.

Characterizing D-preference

We start by providing a fixpoint definition for D-*preference*. For this purpose, we assume a bijective mapping $rule(\cdot)$

from rule heads to rules, that is, $rule(head(r)) = r$; accordingly, $rule(\{head(r) \mid r \in R\}) = R$. Such mappings can be defined in a bijective way by distinguishing different occurrences of literals.

Definition 4 *Let $(\Pi, <)$ be a statically ordered logic program and let X be a set of literals. We define*

$$
\begin{array}{rcl}
X_0 & = & \emptyset \qquad \textit{and for } i \geq 0 \\
X_{i+1} & = & X_i \cup \{head(r) \mid \\
 & & \left.\begin{array}{rl}
I. & r \in \Pi \textit{ is active wrt } (X_i, X) \textit{ and} \\
II. & \textit{there is no rule } r' \in \Pi \textit{ with } r < r' \\
 & \textit{such that} \\
 & (a)\ r' \textit{ is active wrt } (X, X_i) \textit{ and} \\
 & (b)\ r' \notin rule(X_i)
\end{array}\right\}
\end{array}
$$

Then, $\mathcal{C}^{\mathrm{D}}_{(\Pi,<)}(X) = \bigcup_{i \geq 0} X_i$ if $\bigcup_{i \geq 0} X_i$ is consistent. Otherwise, $\mathcal{C}^{\mathrm{D}}_{(\Pi,<)}(X) = Lit$.

The difference between this definition and Definition 1 manifests itself in IIb. While D-preference requires that a higher-ranked rule has effectively applied, W-preference contents itself with the presence of the head of the rule, no matter whether this was supplied by the rule itself.

This difference is nicely illustrated by program $(\Pi_3, <)$:

$$
\begin{array}{rclcl}
r_1: & a & \leftarrow & not\ b & \qquad r_2 < r_1 \\
r_2: & b & \leftarrow & & \\
r_3: & a & \leftarrow & &
\end{array} \tag{3}
$$

While the only answer set $\{a, b\}$ is W-preferred set, there is no D-preferred answer set. This is the same with program $(\Pi_3', <)$ obtained by replacing r_1 with $r_1' : a \leftarrow b$.

We have the following result providing three alternative characterizations of D-preferred answer sets.

Theorem 1 *Let $(\Pi, <)$ be a statically ordered logic program over $\mathcal{L}$ and let X be a set of literals. Then, the following propositions are equivalent.*

1. *$\mathcal{C}^{\mathrm{D}}_{(\Pi,<)}(X) = X$;*
2. *$X = Y \cap \mathcal{L}$ for some answer set Y of $\mathcal{T}(\Pi, <)$;*
3. *X is a $<$-preserving answer set of Π.*

While the last result dealt with effective answer sets, the next one shows that applying operator $\mathcal{C}^{\mathrm{D}}_{(\Pi,<)}$ is equivalent to the application of van Gelder's operator $C_{\Pi'}$ to the translated program $\mathcal{T}(\Pi, <)$.

Theorem 2 *Let $(\Pi, <)$ be a statically ordered logic program over $\mathcal{L}$ and let X be a set of literals over $\mathcal{L}$.*

Then, we have that $\mathcal{C}^{\mathrm{D}}_{(\Pi,<)}(X) = C_{\mathcal{T}(\Pi,<)}(Y) \cap \mathcal{L}$ for some set of literals Y over $\mathcal{L}^+$ such that $X = Y \cap \mathcal{L}$.

This result is important because it allows us to use the translation $\mathcal{T}(\Pi, <)$ for implementing further semantics by appeal to the alternating fixpoint idea.

Characterizing W-preference

We start by showing how W-preference can be characterized in terms of order preservation.

Definition 5 *Let $(\Pi, <)$ be a statically ordered program and let X be an answer set of Π.*

Then, X is called $<^{\mathrm{W}}$-preserving, if X is either inconsistent, or there exists an enumeration $\langle r_i \rangle_{i \in I}$ of Γ_Π^X such that for every $i, j \in I$ we have that:

0(a) $body^+(r_i) \subseteq \{head(r_j) \mid j < i\}$ or
(b) $head(r_i) \in \{head(r_j) \mid j < i\}$; and

1. if $r_i < r_j$, then $j < i$; and

2. if $r_i < r'$ and $r' \in \Pi \setminus \Gamma_\Pi^X$, then
(a) $body^+(r') \not\subseteq X$ or
(b) $body^-(r') \cap \{head(r_j) \mid j < i\} \neq \emptyset$ or
(c) $head(r') \in \{head(r_j) \mid j < i\}$.

The primary difference of this concept of order preservation to the original one is clearly the weaker notion of groundedness. This involves the rules in Γ_Π^X (via Condition 0b) as well as those in $\Pi \setminus \Gamma_\Pi^X$ (via Condition 2c). The rest of the definition is the same as in Definition 2. For instance, answer set $\{a, b\}$ of Π_3 is generated by the $<^{\mathrm{W}}$-preserving rule sequence $\langle r_3, r_2 \rangle$. Note that r_1 satisfies 2c but neither 2a nor 2b. For a complement, in $(\Pi_3', <)$, r_1' is dealt with via Condition 0b.

Interestingly, the notion of weak groundedness can be easily integrated into the translation given in the last section.

Definition 6 *Given the same prerequisites as in Definition 3.*

Then, the logic program $\mathcal{T}^{\mathrm{W}}(\Pi)$ over $\mathcal{L}^+$ is defined as $\mathcal{T}^{\mathrm{W}}(\Pi) = \bigcup_{r \in \Pi} \tau(r) \cup \{c_5(r, r') \mid r, r' \in \Pi\}$, where

$$
c_5(r, r') : \quad \mathsf{ok}'(n_r, n_{r'}) \quad \leftarrow \quad (n_r \prec n_{r'}), head(r')
$$

The purpose of $c_5(r, r')$ is to eliminate rules from the preference handling process once their head has been derived.

We have the following result, showing in particular, how W-preference is implementable via off-the-shelf logic programming systems.

Theorem 3 *Let $(\Pi, <)$ be a statically ordered logic program over $\mathcal{L}$ and let X be a set of literals. Then, the following propositions are equivalent.*

1. *$\mathcal{C}_{(\Pi,<)}(X) = X$;*
2. *$X = Y \cap \mathcal{L}$ for some answer set Y of $\mathcal{T}^{\mathrm{W}}(\Pi, <)$;*
3. *X is a $<^{\mathrm{W}}$-preserving answer set of Π.*

Another immediate consequence of this result is that W-preference does not lead to higher complexity, it remains within NP.

In analogy to what we have shown above, we have the following stronger result, opening the avenue for implementing more semantics based on W-preference:.

Theorem 4 *Let $(\Pi, <)$ be a statically ordered logic program over $\mathcal{L}$ and let X be a set of literals over $\mathcal{L}$.*

Then, we have that $\mathcal{C}_{(\Pi,<)}(X) = C_{\mathcal{T}^{\mathrm{W}}(\Pi,<)}(Y) \cap \mathcal{L}$ for some set of literals Y over $\mathcal{L}^+$ such that $X = Y \cap \mathcal{L}$.

Brewka and Eiter's concept of preference

Another approach to preference was proposed by Brewka and Eiter in (1999). For brevity, we omit technical details and simply say that an answer set is B-*preferred*; the reader is referred to (Brewka and Eiter 1999; 2000) for details.

This approach differs in two significant ways from the two approaches given above. First, the construction of answer sets is separated from verifying whether they respect the given preferences. Interestingly, this verification is done on the basis of the prerequisite-free program obtained from the original one by "evaluating" $body^+(r)$ for each rule r wrt the separately constructed (standard) answer set. Second, rules that putatively lead to counter-intuitive results are explicitly removed from the inference process. This is made explicit in (Brewka and Eiter 2000), where the following filtering transformation is defined:[6]

$$Z_X(\Pi) = \Pi \setminus \{r \in \Pi \mid head(r) \in X, body^-(r) \cap X \neq \emptyset\} \tag{4}$$

Then, by definition, an answer set of Π is B-preferred iff it is a B-preferred answer set of $Z_X(\Pi)$.

The distinguishing example of this approach is given by program $(\Pi_5, <)$:

$$\begin{array}{rrcl} r_1: & b & \leftarrow & a, not\ \neg b \\ r_2: & \neg b & \leftarrow & not\ b \\ r_3: & a & \leftarrow & not\ \neg a \end{array} \qquad r_3 < r_2 < r_1 \tag{5}$$

Program Π_5 has two standard answer sets, $\{a, b\}$ and $\{a, \neg b\}$. While the former is B-preferred, neither of them is W- or D-preferred (see below). Also, we note that both answer sets of program $(\Pi_2, <)$ are B-preferred, while only $\{p, b, \neg f, w\}$ is W- and D-preferred.

In order to shed some light on these differences, we start by providing a fixpoint characterization of B-preference:

Definition 7 *Let $(\Pi, <)$ be an ordered logic program and let X be a set of literals. We define*

$$\begin{array}{rcl} X_0 & = & \emptyset \qquad \text{and for } i \geq 0 \\ X_{i+1} & = & X_i \cup \{head(r) \mid \\ & & \left.\begin{array}{ll} I. & r \in \Pi \text{ is active wrt } (X, X) \text{ and} \\ II. & \text{there is no rule } r' \in \Pi \text{ with } r < r' \\ & \text{such that} \\ & (a)\ r' \text{ is active wrt } (X, X_i) \text{ and} \\ & (b)\ head(r') \notin X_i \end{array}\right\} \end{array}$$

Then, $\mathcal{C}^{\text{B}}_{(\Pi,<)}(X) = \bigcup_{i \geq 0} X_i$ if $\bigcup_{i \geq 0} X_i$ is consistent. Otherwise, $\mathcal{C}_{(\Pi,<)}(X) = Lit$.

The difference between this definition[7] and its predecessors manifests itself in Condition I, where activeness is tested wrt (X, X) instead of (X_i, X) as in Definition 1 and 4. In fact, in Example (5) it is the (unprovability of the) prerequisite a of the highest-ranked rule r_1 that makes the construction of W- or D-preferred answer sets break down (cf. Definition 1 and 4). This is avoided with B-preference because once answer set $\{a, b\}$ is provided, its preference-compatibility is tested wrt the program obtained by replacing r_1 with $b \leftarrow not\ \neg b$.

B-preference can be captured by means of the following notion of order preservation:

Definition 8 *Let $(\Pi, <)$ be a statically ordered program and let X be an answer set of Π.*

Then, X is called $<^{\text{B}}$-preserving, if X is either inconsistent, or there exists an enumeration $\langle r_i \rangle_{i \in I}$ of Γ^X_Π such that, for every $i, j \in I$, we have that:

1. *if $r_i < r_j$, then $j < i$; and*
2. *if $r_i < r'$ and $r' \in \Pi \setminus \Gamma^X_\Pi$, then*
   (a) $body^+(r') \not\subseteq X$ or
   (b) $body^-(r') \cap \{head(r_j) \mid j < i\} \neq \emptyset$ or
   (c) $head(r') \in X$.

This definition differs in two ways from its predecessors. First, it drops any requirement on groundedness, expressed by Condition 0 above. This corresponds to using (X, X) instead of (X_i, X) in Definition 7. Hence, groundedness is fully disconnected from order preservation. In fact, observe that the B-preferred answer set $\{a, b\}$ of $(\Pi_5, <)$ is associated with the $<^{\text{B}}$-preserving sequence $\langle r_1, r_2 \rangle$, while the standard answer set itself is generated by the grounded sequence $\langle r_2, r_1 \rangle$.

Second, Condition 2c is more relaxed than in Definition 5. That is, any rule r' whose head is in X (as opposed to X_i) is taken as "applied". Apart from this, Condition 2c also integrates the filter-conditions from (4).[8] For illustration, consider Example (3) extended by $r_3 < r_2$:

$$\begin{array}{rrcl} r_1: & a & \leftarrow & not\ b \\ r_2: & b & \leftarrow & \\ r_3: & a & \leftarrow & \end{array} \qquad r_3 < r_2 < r_1 \tag{6}$$

While this program has no D- or W-preferred answer set, it has a B-preferred one: $\{a, b\}$ generated by $\langle r_2, r_3 \rangle$. The critical rule r_1 is handled by 2c. As a net result, Condition 2 is weaker than its counterpart in Definition 5.

We have the following results.

Theorem 5 *Let $(\Pi, <)$ be a statically ordered logic program over $\mathcal{L}$ and let X be an answer set of Π.*

Then, the following propositions are equivalent.

1. *X is B-preferred;*
2. *$\mathcal{C}^{\text{B}}_{(Z_X(\Pi),<)}(X) = X$;*
3. *$X = Y \cap \mathcal{L}$ for some answer set Y of $\mathcal{T}^{\text{B}}(\Pi, <)$ (where $\mathcal{T}^{\text{B}}$ is defined in (Delgrande* et al. *2000));*
4. *X is a $<^{\text{B}}$-preserving answer set of Π.*

Unlike theorems 1 and 3, the last result stipulates that X must be an answer set of Π. This requirement can only be dropped in case 3, while all other cases rely on this property.

[6] While this is integrated into (Brewka and Eiter 1999, Def. 4.4), it is made explicit in (Brewka and Eiter 2000, Def. 6).

[7] We have refrained from integrating (4) in order to keep the fixpoint operator comparable to its predecessors, given in the previous section. This is taken care of in the second proposition of Theorem 5.

[8] Condition $body^-(r') \cap X \neq \emptyset$ from (4) is obsolete because $r' \notin \Gamma^X_\Pi$.

Relationships

Up to now, we have tried to clarify the structural differences between the respective approaches. This has led to homogeneous characterizations that allow us to compare the examined approaches in a uniform way. As a result, we obtain insights into the relationships among these approaches.

First of all, we observe that all three approaches treat the blockage of (higher-ranked) rules in the same way. That is, a rule r' is found to be blocked if either its prerequisites in $body^+(r')$ are *never* derivable or if some member of $body^-(r')$ has been derived by higher-ranked or unrelated rules. This is reflected by the identity of conditions Ia and 2a/b in all three approaches, respectively. Although this is arguably a sensible strategy, it leads to the loss of preferred answer sets on programs like

$$\begin{array}{rcll} r_1: & a & \leftarrow & not\ b \qquad\qquad r_2 < r_1 \\ r_2: & b & \leftarrow & . \end{array}$$

Let us now discuss the differences among the approaches. The difference between D- and W-preference can be directly read off Definition 1 and 4; it manifests itself in Condition IIb and leads to the following relationship.

Theorem 6 *Every* D*-preferred answer set is* W*-preferred.*

Example (3) shows that the converse does not hold.

Interestingly, a similar relationship is obtained between W- and B-preference. In fact, Definition 8 can be interpreted as a weakening of Definition 5 by dropping Condition 0 and weakening Condition 2 (via 2c). We thus obtain the following result.

Theorem 7 *Every* W*-preferred answer set is* B*-preferred.*

Example (5) shows that the converse does not hold.

Let $\mathcal{AS}(\Pi) = \{X \mid C_\Pi(X) = X\}$ and $\mathcal{AS}_P(\Pi,<) = \{X \mid \mathcal{C}^P_{(\Pi,<)}(X) = X\}$ for $P = \text{W},\text{D},\text{B}$. Then, we obtain the following summarizing result.

Theorem 8 *Let* $(\Pi,<)$ *be a statically ordered logic program.*

Then, we have:

$$\mathcal{AS}_{\text{D}}(\Pi,<) \subseteq \mathcal{AS}_{\text{W}}(\Pi,<) \subseteq \mathcal{AS}_{\text{B}}(\Pi,<) \subseteq \mathcal{AS}(\Pi)$$

(The full paper gives further results on sufficient conditions for the coincidence of these approaches.) In principle, this hierarchy is induced by a decreasing interaction between groundedness and preference. While D-preference requires the full compatibility of both concepts, this interaction is already weakened in W-preference, before it is fully abandoned in B-preference. This is nicely reflected by the evolution of Condition 0 in definitions 2, 5, and 8.

Notably, groundedness as such is not the ultimate distinguishing factor, as demonstrated by the fact that prerequisite-free programs do not necessarily lead to the same preferred answer sets, as witnessed in (3) and (6). Rather it is the degree of interaction between groundedness and preferences that makes the difference.

Conclusion

The notion of preference seems to be pervasive in logic programming when it comes to knowledge representation. This is reflected by numerous approaches that aim at enhancing logic programming with preferences in order to improve knowledge representation capacities. Despite the large variety of approaches, however, only very little attention has been paid to their structural differences and sameness, finally leading to solid semantical underpinnings.

This work is a first step towards a systematic account to logic programming with preferences. We elaborated upon three different approaches that were originally defined in rather heterogenous ways. We obtained three alternative yet uniform ways of characterizing preferred answer sets (in terms of fixpoints, order preservation, and an axiomatic account). The underlying uniformity provided us with a deeper understanding of how and which answer sets are preferred in each approach. This has led to a clarification of their relationships and subtle differences. In particular, we revealed that the investigated approaches yield an increasing number of answer sets depending on how tight they connect preference to groundedness.

An interesting technical result of this paper is given by the equivalences between the fixpoint operators and the standard logic programming operators applied to the correspondingly transformed programs (cf. Theorem 2 and 4). This opens the avenue for further concepts of preference handling on the basis of the alternating fixpoint theory and its issuing semantics. Further research includes dynamic preferences and more efficient algorithms for different semantics in a unifying way.

Acknowledgements We would like to thank Ph. Besnard, Th. Linke and the anonymous referees for useful comments on this paper.

This work was supported by the German Science Foundation (DFG) within Project "Nichtmonotone Inferenzsysteme zur Verarbeitung konfligierender Regeln" under grant FOR 375/1-1, TP C.

References

F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. In *Proc. IJCAI'93*, p 669–674, 1993.

G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, 1999.

G. Brewka and T. Eiter. Prioritizing default logic. In St. Hölldobler, ed, *Intellectics and Computational Logic*. Kluwer, 2000. To appear.

G. Brewka. Adding priorities and specificity to default logic. In L. Pereira and D. Pearce, eds, *Proc. JELIA'94*, p 247–260. Springer, 1994.

J. Delgrande, T. Schaub, and H. Tompits. Logic programs with compiled preferences. In *Proc. ECAI 2000*, p 392–398. IOS Press, 2000.

M. Gelfond and V. Lifschitz. Classical negation in logic programs and deductive databases. *New Generation Computing*, 9:365–385, 1991.

M. Gelfond and T. Son. Reasoning with prioritized defaults. In J. Dix, L. Pereira, and T. Przymusinski, eds, *Workshop on Logic Programming and Knowledge Representation*, p 164–223. Springer, 1997.

J. Rintanen. On specificity in default logic. In *Proc. IJCAI'95*, p 1474–1479. Morgan Kaufmann, 1995.

C. Sakama and K. Inoue. Representing priorities in logic programs. In M. Maher, ed, *Proc. JCSLP'96*, p 82–96. MIT Press, 1996.

A. van Gelder. The alternating fixpoint of logic programs with negation. *J. Computer and System Science*, 47:185–120, 1993.

K. Wang, L. Zhou, and F. Lin. Alternating fixpoint theory for logic programs with priority. In *Proc. Int'l Conf. Computational Logic*, p 164-178. Springer, 2000.

Y. Zhang and N. Foo. Answer sets for prioritized logic programs. In J. Maluszynski, ed, *Proc. ISLP'97*, p 69–84. MIT Press, 1997.