Preferred well-founded semantics for logic programming by alternating fixpoints

Torsten Schaub[*] **and Kewen Wang**
Institut für Informatik, Universität Potsdam
Postfach 900327, D–14439 Potsdam, Germany
`{torsten,kewen}@cs.uni-potsdam.de`

Abstract

We analyze the problem of defining well-founded semantics for ordered logic programs within a general framework based on alternating fixpoint theory. We start by showing that generalizations of existing answer set approaches to preference are too weak in the setting of well-founded semantics. We then specify some informal yet intuitive criteria and propose a semantical framework for preference handling that is more suitable for defining well-founded semantics for ordered logic programs. The suitability of the new approach is convinced by the fact that many attractive properties are satisfied by our semantics. In particular, our semantics is still correct with respect to various existing answer sets semantics while it successfully overcomes the weakness of their generalization to well-founded semantics. Finally, we indicate how an existing preferred well-founded semantics can be captured within our semantical framework.

Keywords: well-founded semantics, preference, alternating fixpoints, extended logic programs.

1 Introduction

Preferences constitute a very natural and effective way of resolving indeterminate situations. For example, in scheduling not all deadlines may be simultaneously satisfiable, and in configuration various goals may not be simultaneously met. Preferences among deadlines and goals may allow for an acceptable, non-optimal solution. In legal reasoning, laws may apply in different situations, but laws may also conflict with each other. Conflicts are resolved by appeal to higher-level principles such as authority or recency. So federal laws will have a higher priority than state laws, and newer laws will take priority over old. Further preferences, such as authority holding sway over recency, may also be required. In fact, while logical preference handling constitutes already an indispensable means in legal reasoning systems (cf. [16, 22]), it is also advancing in other application areas such as intelligent agents and e-commerce [18], information-site selection [14], and the resolution of grammatical ambiguities [11].

The increasing practical interest in preferences is also reflected by the large number of proposals for preference handling in logic programming, including [23, 6, 15, 31, 17, 8, 13, 28], and related areas, such as default logic [3, 5, 12]. A common approach in such work has been to employ meta-formalisms for characterizing "preferred answer sets". This has led to a diversity of approaches that are hardly comparable due to considerably different methods of formal characterization. As a consequence, there is no homogeneous account of preference.

In [24], we started addressing this shortcoming by proposing a uniform semantical framework for extended logic programming with preferences. To be precise, we develop an (alternating) fixpoint theory for so-called *ordered logic programs*, building on the basic ideas in [27]. An ordered logic program is an extended logic program whose rules are subject to a strict partial order. In analogy to standard logic programming, such a program is then interpreted by means of an associated fixpoint operator. Different semantics are obtained by distinguishing different subsets of the respective set of alternating fixpoints. As a result, several different approaches to defining preferred answer sets, including [9, 10, 13], can all be captured within our framework and each of these preference strategies is based on an operator, which plays the same role as the consequence operator in the setting of normal logic programs.

In this paper, we show that the preference strategies for

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

defining answer sets turn out to be too weak in the setting of well-founded semantics. For this reason, we propose a new approach to preference handling for logic programs that seems to be more appropriate for well-founded semantics. In fact, we show that for a resulting instance of this approach some attractive properties. We also discuss the relation of our preferred well-founded semantics to other approaches [21, 6, 30].

2 Definitions and notation

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; it is called *normal* if it contains no classical negation symbol $\neg$.

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 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 $Cn(\Pi)$. With these formalities at hand, we can define *answer set semantics* for extended logic programs: A set X of literals is an *answer set* of a program Π iff $Cn(\Pi^X) = X$.

For capturing even more semantics within a similar framework, van Gelder defines in [27] the operator $C_\Pi(X)$ as $Cn(\Pi^X)$. It is important to 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. For instance, given a program Π, the least alternating fixed point of A_Π is known to amount to its *well-founded semantics*. Answer sets of Π are simply alternating fixed points of A_Π that are also fixed points of C_Π.

Alternative inductive characterizations for the operators Cn, C_Π, and A_Π can be obtained by appeal to *immediate consequence operators* [26, 19]. Let Π be a basic program and X a set of literals. The *immediate consequence operator* T_Π is defined as follows:

$$T_\Pi X = \{\mathit{head}(r) \mid r \in \Pi \text{ and } \mathit{body}(r) \subseteq X\}$$

if X is consistent, and $T_\Pi X = Lit$ otherwise. Iterated applications of T_Π are written as T^j_Π for $j \geq 0$, where $T^0_\Pi X = X$ and $T^i_\Pi X = T_\Pi T^{i-1}_\Pi X$ for $i \geq 1$. It is well-known that $Cn(\Pi) = \bigcup_{i \geq 0} T^i_\Pi \emptyset$, for any basic program Π. Also, for any answer set X of program Π, it holds that $X = \bigcup_{i \geq 0} T^i_{\Pi^X} \emptyset$.

A reduction from extended to basic programs is avoidable with an extended consequence operator: Let Π be an extended program and X and Y be sets of literals. The *extended immediate consequence operator* $T_{\Pi,Y}$ is defined as follows:

$$\begin{aligned} T_{\Pi,Y} X \quad = \quad & \{\mathit{head}(r) \mid r \in \Pi,\ \mathit{body}^+(r) \subseteq X, \\ & \text{and } \mathit{body}^-(r) \cap Y = \emptyset\} \end{aligned} \qquad (2)$$

if X is consistent, and $T_{\Pi,Y} X = Lit$ otherwise. Iterated applications of $T_{\Pi,Y}$ are written as those of T_Π above. Clearly, we have $T_{\Pi,\emptyset} X = T_\Pi X$ for any basic program Π and $T_{\Pi,Y} X = T_{\Pi^Y} X$ for any extended program Π. Accordingly, we have for any answer set X of program Π that $X = \bigcup_{i \geq 0} T^i_{\Pi,X} \emptyset$. Finally, for dealing with the individual rules in (2), we rely on the notion of *activeness*:[1] Let $X, Y \subseteq Lit$ be two sets of literals in a program Π. A rule r in Π is *active* wrt the pair (X, Y), if $\mathit{body}^+(r) \subseteq X$ and $\mathit{body}^-(r) \cap Y = \emptyset$. Alternatively, we thus have that $T_{\Pi,Y} X = \{\mathit{head}(r) \mid r \in \Pi \text{ is active wrt } (X, Y)\}$.

Lastly, an *ordered logic program*[2] is simply 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 $r_1 < r_2$ is meant to express that r_2 has *higher priority* than r_1.[3]

3 Preferred (alternating) fixpoints

We start by describing the semantical framework given in [24], while concentrating on the formal details needed for capturing the approach introduced in [28]. The formal development of the approach in [8] and [13] is analogous and thus omitted here.

[1] Although *activeness* is implicitly present in standard logic programming (cf. definition of $T_{\Pi,Y} X$), the term as such was (to the best of our knowledge) coined in approaches dealing with preferences in default logic [3, 5]. There, however, activeness additionally stipulated that $\mathit{head}(r) \notin X$ in order to prevent multiple applications of the same rule.

[2] Also called *prioritized* logic program by some authors, as eg. in [31, 8].

[3] Some authors, eg. [8], attribute relation $<$ the inverse meaning.

The overall idea behind the obtained semantics for ordered logic program is to distinguish the "preferred" answers of a program $(\Pi, <)$ by means of fixpoint equations. That is, a set of literals X constitutes a collection of preferred answers from $(\Pi, <)$, if it satisfies the equation $\mathcal{C}_{(\Pi,<)}(X) = X$ for some operator $\mathcal{C}_{(\Pi,<)}$. In view of the classical logic programming approach described in Section 2, this makes us investigate semantics that interpret preferences as inducing selection functions on the set of standard answer sets of the underlying non-ordered program Π.

Standard answer sets are defined via a reduction of extended logic programs to basic programs. Such a reduction is inappropriate when resolving conflicts among rules by means of preferences since all such conflicts are simultaneously resolved when turning Π into Π^X. Rather conflict resolution must be addressed among the original rules in order to account for blockage between rules. In fact, once the negative body $body^-(r)$ is eliminated there is no way to detect whether $head(r') \in body^-(r)$ holds in case of $r < r'$. Our idea is therefore to characterize preferred answer sets by an inductive development that agrees with the given ordering rather than a simultaneous reduction. In terms of a standard answer set X, this means that we favor its formal characterization as $X = \bigcup_{i \geq 0} T^i_{\Pi,X}\emptyset$ over $X = Cn(\Pi^X)$. This leads us to the following definition. [4]

Definition 1 *Let $(\Pi, <)$ be an ordered logic program and let X and Y be sets of literals.*

We define the set of immediate consequences of X with respect to $(\Pi, <)$ and Y as

$$\mathcal{T}_{(\Pi,<),Y}X = \left\{ head(r) \;\middle|\; \begin{array}{ll} I. & r \text{ active wrt } (X,Y); \\ II. & \text{there is no rule } r' \\ & \text{ with } r < r' \text{such that} \\ & (a)\ r' \text{ active wrt } (Y,X) \\ & (b)\ head(r') \notin X \end{array} \right\}$$

if X is consistent, and $\mathcal{T}_{(\Pi,<),Y}X = Lit$ otherwise.

Note that $\mathcal{T}_{(\Pi,<),Y}$ is a refinement of its classical counterpart $T_{\Pi,Y}$. To see this, observe that Condition *I* embodies the standard application condition for rules given in (2)

The actual refinement takes place in Condition *II*. The idea is to apply a rule r only if the "question of applicability" has been settled for all higher-ranked rules r'. Let us illustrate this in terms of iterated applications of $\mathcal{T}_{(\Pi,<),Y}$. In these cases, X contains the set of conclusions that have been derived so far, while Y provides the putative answer set (or: $Lit \setminus Y$ provides a set of literals that can be falsified). Then, the "question of applicability" is considered to be settled for a higher ranked rule r'

- if the prerequisites of r' will never be derivable, viz. $body^+(r') \not\subseteq Y$, or
- if r' is defeated by what has been derived so far, viz. $body^-(r) \cap X \neq \emptyset$, or
- if r' or another rule with the same head have already applied, viz. $head(r') \in X$.

The first two conditions show why activeness of r' is stipulated wrt (Y, X), as opposed to (X, Y) in Condition *I*. The last condition serves somehow two purposes: First, it detects whether the higher ranked rule r' has applied and, second, it suspends the preference $r < r'$ whenever the head of the higher ranked has already been derived by another rule. This suspension of preference constitutes a distinguishing feature of the approach at hand; this is discussed in detail in [24] in connection with other approaches to preference handling.

As with T_Π and $T_{\Pi,Y}$, iterated applications of $\mathcal{T}_{(\Pi,<),Y}$ are written as $\mathcal{T}^j_{(\Pi,<),Y}$ for $j \geq 0$, where $\mathcal{T}^0_{(\Pi,<),Y}X = X$ and $\mathcal{T}^i_{(\Pi,<),Y}X = \mathcal{T}_{(\Pi,<),Y}\mathcal{T}^{i-1}_{(\Pi,<),Y}X$ for $i \geq 1$. This allows us to define the counterpart of fixpoint operator C_Π for ordered programs:

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

We define $\mathcal{C}_{(\Pi,<)}(X) = \bigcup_{i \geq 0} \mathcal{T}^i_{(\Pi,<),X}\emptyset$.

In analogy to $\mathcal{T}_{(\Pi,<),Y}$ and $T_{\Pi,Y}$, operator $\mathcal{C}_{(\Pi,<)}$ is a refinement of its classical counterpart C_Π. The major difference of our definition from van Gelder's is that we directly obtain the consequences from Π (and Y). Unlike this, the usual approach (without preferences) first obtains a basic program Π^Y from Π and then the consequences are derived from this basic program Π^Y.

A preferred answer set is defined as a fixpoint of $\mathcal{C}_{(\Pi,<)}$.

In analogy to van Gelder [27], we may define the *alternating transformation* for an ordered logic program $(\Pi, <)$ as $\mathcal{A}_{(\Pi,<)}(X) = \mathcal{C}_{(\Pi,<)}(\mathcal{C}_{(\Pi,<)}(X))$. A fixpoint of $\mathcal{A}_{(\Pi,<)}$ is called an *alternating fixpoint* of $(\Pi, <)$. Given that $\mathcal{C}_{(\Pi,<)}$ is anti-monotonic [24], we get that $\mathcal{A}_{(\Pi,<)}(X)$ is monotonic. According to results tracing back to Tarski [25], this implies that $\mathcal{A}_{(\Pi,<)}$ possesses a least and a greatest fixpoint, denoted by *lfp*$\mathcal{A}_{(\Pi,<)}$ and *gfp*$\mathcal{A}_{(\Pi,<)}$, respectively.

Different semantics of ordered logic programs are obtained by distinguishing different subsets of the respective set of alternating fixpoints. In fact, the preferred answer set semantics constitute instances of the overall framework. To

[4] Fixpoint operators for the approaches in [8] and [13] are obtained by appropriate modifications to Condition I and II in Definition 1; cf. [24].

see this, observe that each fixpoint of $\mathcal{C}_{(\Pi,<)}$ is also a fixpoint of $\mathcal{A}_{(\Pi,<)}$.

4 Preferring least alternating fixpoints?

Let us now investigate the least alternating fixpoint of $\mathcal{A}_{(\Pi,<)}$and with it the comportment of the previous fixpoint operator in the setting of *well-founded* semantics. As opposed to answer sets semantics, this semantics relies on 3-valued models (or, partial models). Such a model consists of three parts: the set of true literals, the set of false literals, and the set of unknown literals. Given that the union of these three sets is Lit, it is sufficient to specify two of the three sets for determining a 3-valued interpretation. Accordingly, a 3-valued interpretation I is a pair (X, Y) where X and Y are sets of literals with $X \cap Y = \emptyset$. That is, $L \in X$ means that L is true in I, while $L \in Y$ means that L is false in I. Otherwise, L is considered to be unknown in I.

Well-founded semantics constitutes another major semantics for logic programs. In contrast to answers sets semantics, it aims at characterizing skeptical conclusions comprised in a single so-called *well-founded* model of the underlying program. This model can be characterized within the alternating fixpoint theory in terms of the least fixpoint of operator A_Π. That is, the well-founded model of a program Π is given by the 3-valued interpretation $(\mathit{lfp} A_\Pi, \mathit{Lit} \setminus C_\Pi \mathit{lfp} A_\Pi)$. Hence, it is sufficient to consider the least alternating fixpoint of a program, since it determines its well-founded model. We therefore refer to the least alternating fixpoint of Π as the *well-founded set* of Π. The set $\mathit{Lit} \setminus C_\Pi \mathit{lfp} A_\Pi$ is usually referred to as the *unfounded* set of Π.

After extending these concepts to preference handling, that is, substituting the classical operators A_Π and C_Π by $\mathcal{A}_{(\Pi,<)}$ and $\mathcal{C}_{(\Pi,<)}$, respectively, one can show that (i) each ordered logic program has a unique preferred well-founded model; (ii) the preferred well-founded set is contained in any preferred answer set (while the unfounded one is not); and (iii) whenever we obtain a two-valued well-founded model, its underlying well-founded set is the unique answer set of the program.[5]

One often criticized deficiency of the standard well-founded model is that it is too skeptical. Unfortunately, this is *not* remedied by alternating the fixpoint operators of the previous sections, no matter which strategy we consider. To see this, consider the ordered logic program $(\Pi_3, <)$:

$$\begin{array}{rcccl} r_1 & = & a & \leftarrow & \mathit{not}\ b \qquad\qquad r_2 < r_1 \\ r_2 & = & b & \leftarrow & \mathit{not}\ a \end{array} \quad (3)$$

[5]No matter whether we consider the fixpoint operators for the approach in [28], [8], or [13], respectively.

The well-founded model of Π_3 is given by $(\emptyset, \emptyset)$. The same model is obtained by alternating operator $\mathcal{C}_{(\Pi_3,<)}$. Observe that $\mathcal{C}_{(\Pi_3,<)}(\emptyset) = \{a, b\}$ and $\mathcal{C}_{(\Pi_3,<)}(\{a, b\}) = \emptyset$. Consequently, $\emptyset$ is the least alternating fixpoint of $(\Pi_3, <)$.

The question is now why these operators are still too skeptical in defining well-founded semantics (although they work nicely in the setting of answer sets and regular semantics). In fact, the great advantage of a setting like that of answer sets semantics is that we deal with *direct* fixpoint equations, like $\mathcal{C}_{(\Pi,<)}(X) = X$, where the context X represents the putative answer set. This is different in the setting of well-founded semantics, where we usually start by applying an operator to a rather small context, eg. initially the empty set; this usually results in a larger set, sometimes even Lit, that constitutes then the context of the second application of the operator. Now, looking at the underlying definitions, we see that the actual preference handling condition, eg. Condition II in Definition 1 takes advantage of X for deciding applicability. The alternating character in the well-founded setting does not support this sort of analysis since it cannot provide the (putative) final result of the computation.

5 Towards a preferred well-founded semantics

In view of the failure of the above fixpoint operator(s) in the setting of well-founded semantics, the obvious question is now whether an appropriate *alternating* fixpoint operation is definable that yields a reasonable well-founded semantics for ordered logic programs. As informal guidelines, we would like that the resulting semantics (i) allows for deriving more conclusions than the standard well-founded semantics by appeal to given preferences; (ii) coincides with standard well-founded semantics in the absence of preferences; and finally (iii) approximates the previous preferred answer sets semantics.

The standard well-founded model is defined by means of the least fixpoint of the operator $A_\Pi = C_\Pi C_\Pi$. As above, we aim at integrating preferences by elaborating upon the underlying immediate consequence operator $T_{\Pi,Y}$X given in (2). As well, the basic idea is to modify this operator so that more conclusions can be derived by employing preferences. However, as discussed at the end of the previous section, the alternating iterations of C_Π face two complementary situations: those with smaller contexts and those with larger ones. Since preferences exploit these contexts, it seems reasonable to distinguish alternating applications or, at least, to concentrate on one such situation while dealing with the other one in the standard way.[6] For strengthening $A_\Pi = C_\Pi C_\Pi$, we thus have two options: either we make the outer operator derive more literals or we make the

[6]Such an approach is also pursued in [6].

inner operator derive less literals.

In what follows, we adopt the former option and elaborate upon the outer operator. The general idea is then to reduce the context considered in the second application of C_Π by appeal to preferences in order to make more rules applicable. For this purpose we remove those literals that are derived by means of less preferred, defeated rules.

Definition 3 *Let $(\Pi, <)$ be an ordered logic program and let X and Y be sets of literals.*

We define the set of immediate consequences of X with respect to $(\Pi, <)$ and Y as

$$\mathcal{T}^{\circ}_{(\Pi,<),Y} X = \{ \mathit{head}(r) \mid r \in \Pi \text{ is active wrt } (X, Y \backslash D^r_X) \}$$

where

$$D^r_X = \left\{ L \;\middle|\; \begin{array}{l} \text{for all rules } r' \in \Pi, \\ \quad \text{if } L = \mathit{head}(r') \quad \text{and} \\ \qquad \mathit{body}^+(r') \subseteq C_\Pi(\emptyset), \\ \quad \text{then } r' < r \quad \text{and} \\ \qquad\quad (\mathit{head}(r) \cup X) \cap \mathit{body}^-(r') \neq \emptyset \end{array} \right\}$$

if X is consistent, and $\mathcal{T}^{\circ}_{(\Pi,<),Y} X = \mathit{Lit}$ otherwise.

We say that r defeats r' wrt X if $(\mathit{head}(r) \cup X) \cap \mathit{body}^-(r') \neq \emptyset$. The set of removed literals D^r_X consists thus of those rule heads, all of whose corresponding rules are less preferred than r and defeated by r or X, viz. the literals derived so far. In fact, this condition only removes a literal such as $\mathit{head}(r')$ from Y, if all of its applicable generating rules like r' are defeated by the preferred rule r. Note that D^r_X is normally different for different rules r.

For illustration consider the rules in Π_3. For $X = \emptyset$ and $Y = \{a, b\}$, we get $D^{r_1}_\emptyset = \{b\}$ and $D^{r_2}_\emptyset = \emptyset$. In such a situation, activeness of r_1 is checked wrt $(\emptyset, \{a, b\} \setminus \{b\})$ while that of r_2 is checked wrt $(\emptyset, \{a, b\})$. When applying r_1, the removal of $D^{r_1}_\emptyset = \{b\}$ from context $\{a, b\}$ allows us to discard the conclusion of the less preferred rule r_2 that is defeated by the preferred rule r_1. This example is continued below.

Notably, the choice of D^r_X is one among many options. Unfortunately, it leads beyond the scope of this paper to investigate the overall resulting spectrum, so that we concentrate on the above definition and discuss some alternatives at the end of this section. From a general perspective, the above definition offers thus a parameterizable framework for defining well-founded semantics including preferences.

In analogy to the previous sections, we can define a consequence operator as follows.

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

We define $\mathcal{C}^{\circ}_{(\Pi,<)}(X) = \bigcup_{i \geq 0} (\mathcal{T}^{\circ}_{(\Pi,<),X})^i \emptyset$.

Of particular interest in view of an alternating fixpoint theory is that $\mathcal{C}^{\circ}_{(\Pi,<)}$ enjoys *anti-monotonicity*:

Theorem 1 *Let $(\Pi, <)$ be an ordered logic program and X_1, X_2 sets of literals.*

If $X_1 \subseteq X_2$, then $\mathcal{C}^{\circ}_{(\Pi,<)}(X_2) \subseteq \mathcal{C}^{\circ}_{(\Pi,<)}(X_1)$.

Given this, we may define a new alternating transformation of $(\Pi, <)$ as

$$\mathcal{A}^{\circ}_{(\Pi,<)} = \mathcal{C}^{\circ}_{(\Pi,<)} C_\Pi .$$

Since both $\mathcal{C}^{\circ}_{(\Pi,<)}$ and C_Π are anti-monotonic, $\mathcal{A}^{\circ}_{(\Pi,<)}$ is monotonic.

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

We define X as a preferred well-founded set of $(\Pi, <)$ iff $\mathit{lfp} \mathcal{A}^{\circ}_{(\Pi,<)} = X$.

By Tarski's Theorem [25], we get that each ordered logic program has a unique preferred well-founded set.

Theorem 2 *Let $(\Pi, <)$ be an ordered logic program.*

Then, there is a unique preferred well-founded set of $(\Pi, <)$.

Given the notion of the preferred well-founded set, we define the preferred well-founded model of an ordered program as follows.

Definition 6 *Let $(\Pi, <)$ be an ordered logic program and let X be the well-founded set of $(\Pi, <)$.*

We define the preferred well-founded model of $(\Pi, <)$ as $(X, \mathit{Lit} \setminus C_\Pi(X))$.

It is well-known that the standard well-founded semantics for extended logic programs has time complexity $O(n^2)$ [29, 4]. The complexity of the preferred well-founded semantics is still in polynomial time but it is in $O(n^3)$. The reason is that we have to additionally compute D^r_X for each $r \in \Pi$.

We first obtain the following corollary to Theorem 2.

Corollary 3 *Every ordered logic program has a unique preferred well-founded model.*

This result shows that our preferred well-founded semantics is as robust as the standard well-founded semantics.

The relationship between the standard well-founded model and the preferred well-founded model can be stated as follows.

Theorem 4 *Let (X, Y) be the preferred well-founded model of $(\Pi, <)$ and let (X', Y') be the well-founded model of Π.*

Then, we have

1. *$X' \subseteq X$ and $Y' \subseteq Y$ and*
2. *$(X, Y) = (X', Y')$, if $< = \emptyset$.*

Let us reconsider $(\Pi_3, <)$. While $(\emptyset, \emptyset)$ is the well-founded model of Π_3, its ordered counterpart $(\Pi_3, <)$ has the preferred well-founded model $(\{a\}, \{b\})$. To see this, observe that $C_{\Pi_3}\emptyset = \{a, b\}$ and $\mathcal{C}^{\circ}_{(\Pi_3,<)}(\{a, b\}) = \{a\}$. Clearly, $\{a\}$ is a fixpoint of C_{Π_3} and $\mathcal{C}^{\circ}_{(\Pi_3,<)}$. Thus, $\{a\}$ is an alternating fixpoint of $(\Pi_3, <)$. Also, we see that $\emptyset$ is not an alternating fixpoint. This implies that $\{a\}$ is the least alternating fixpoint of $(\Pi_3, <)$.

This example along with the last result show that preferences allow us to strengthen the conclusions obtained by the standard well-founded semantics. That is, whenever certain conclusions are not sanctioned in the standard framework one may add appropriate preferences in order to obtain these conclusions within the overall framework of well-founded semantics.

For a complement, consider the following variation of $(\Pi_3, <)$, also discussed in [6].

$$\begin{array}{rclcl} r_1 & = & a & \leftarrow & \mathit{not}\ b \\ r_2 & = & b & \leftarrow & \mathit{not}\ c \end{array} \qquad r_2 < r_1 \qquad (4)$$

Observe that Π_4 has well-founded model $(\{b\}, \{a, c\})$. In contrast to $(\Pi_3, <)$, the preferred well-founded model of $(\Pi_4, <)$ is also $(\{b\}, \{a, c\})$. As discussed in [6] this makes sense since preferences should only enrich but not "override" an underlying well-founded model.

Another attractive property of this instance of preferred well-founded semantics is that it provides an approximation of preferred answer sets semantics.

Theorem 5 *Let (X, Y) be the preferred well-founded model of $(\Pi, <)$ and let Z be a preferred answer set of $(\Pi, <)$.*

Then, we have $X \subseteq Z$ and $Y \subseteq \mathit{Lit} \setminus Z$.

Notably, this can be shown for all aforementioned preferred answer sets semantics, no matter whether we consider the approach in [28], [8], or [13], respectively.

Finally, let us briefly discuss some alternative choices for D^r_X. In fact, whenever we express the same preferences among (negative) rules having the same head the previous definition of D^r_X is equivalent to $\{\mathit{head}(r') \mid r' < r \text{ and } (\mathit{head}(r) \cup X) \cap \mathit{body}^-(r') \neq \emptyset\}$. However, this conceptually simpler definition is inadequate when it comes to attributing different preferences to rules with the same heads as in the following example.

Consider the ordered program $(\Pi_5, <)$.

$$\begin{array}{rclcl} r_1 & = & a & \leftarrow & \\ r_2 & = & b & \leftarrow & \mathit{not}\ a \\ r_3 & = & a & \leftarrow & \mathit{not}\ b \end{array} \qquad r_3 < r_2 < r_1 \qquad (5)$$

The preferred well-founded semantics of $(\Pi_5, <)$ gives $(\{a\}, \{b\})$, while the conceptually simpler one yields $(\{a, b\}, \emptyset)$, a clearly wrong result! In the simplistic setting $D^{r_2}_{\emptyset}$ would contain the head of the third rule, discarding the fact that r_1 already defeats r_2.

Another alternative choice for D^r_X is indicated by the difference between the strategies employed in [28] and [13]. In fact, the latter implicitly distinguishes between same literals stemming from different rules. This amounts to distinguishing different occurrences of literals. For this, we may rely on the aforementioned simplistic definition of D^r_X and suppose that $\mathit{head}(r)$ provides us with occurrences of literals, like b^{r_2} instead of b. Without entering details, let us illustrate this idea by appeal to $(\Pi_5, <)$. An approach distinguishing occurrences of literals would yield $C_{\Pi_5}\emptyset = \{a^{r_1}, b^{r_2}, a^{r_3}\}$ and $\mathcal{C}^{\circ}_{(\Pi_5,<)}(\{a^{r_1}, b^{r_2}, a^{r_3}\}) = \{a^{r_1}, a^{r_3}\}$. When considering r_2, we check activeness wrt $(\emptyset, \{a^{r_1}, b^{r_2}, a^{r_3}\} \setminus \{a^{r_3}\})$, viz. $(\emptyset, \{a^{r_1}, b^{r_2}\})$. Unlike just above, a^{r_1} remains in the reduced context and r_2 is inapplicable. An elaboration of this avenue is beyond the scope of this paper, in particular, since it involves an occurrence-based development of well-founded semantics.

6 Relationships

In contrast to answer set semantics, the extension of well-founded semantics to ordered logic program has been rarely studied before. In this section we will discuss the relation of our approach to [6, 21, 30].

6.1 Relation to Brewka's Approach

Brewka defines in [6] a well-founded semantics for ordered logic programs. Notably, this approach is based on a paraconsistent extension of well-founded semantics that tolerates inconsistencies among the result of the inner operator without trivializing the overall result. Despite this deviation from standard well-founded semantics, the question remains whether Brewka's semantics can be captured within our semantical framework.

In fact, both approaches are based on quite different intuitions. While the underlying idea of Brewka's approach is to define a criterion for selecting the intended rules by employing preference, we integrate preferences into the immediate consequence operaor by individually restricting the context of application for each rule.

Nonetheless, it turns out that Brewka's semantics can be captured through an alternating fixpoint construction. As we show below, Brewka's modification boils down to using an alternate fixpoint operator of the form "$\mathcal{C}^{\star}_{(\Pi,<)} C^{\star}_{\Pi}$". To this end, let us first consider the difference among the underlying operators $C^{\star}_{\Pi}$ and C_{Π}. Define $Cl(\Pi)$ as the smallest set of literals which is closed under a basic program Π. Then, given a set X of literals, $C^{\star}_{\Pi}(X)$ is defined as $Cl(\Pi^X)$. Dropping the requirement of logical closure results in a paraconsistent inference operation. For example, given $\Pi = \{a \leftarrow, \neg a \leftarrow, b \leftarrow\}$, we get $Cn(\Pi) = Lit$, while $Cl(\Pi) = \{a, \neg a, b\}$. Although the corresponding adaptions are more involved, the surprising result is now that Brewka's semantics can also be captured within our overall framework, if we use the closure operator Cl instead of Cn.

Moreover, we need the following. Let $(\Pi, <)$ be an ordered logic program and X be a set of literals. We define Π^r_X as the set of rules defeated by r wrt X and $<$ as

$$\Pi^r_X = \{r' \in \Pi \mid r' < r,\ r \text{ defeats } r' \text{ wrt } X\}.$$

Notice that Π^r_X is a set of rules while D^r_X is a set of literals. Π^r_X is also different from Brewka's *Dom* (set of *dom*inated rules) in that Π^r_X is defined wrt a set X of literals rather than a set of rules.

Write $(\Pi^r_X)^+ = \{(r')^+ \mid r' \in \Pi^r_X\}$. Let $\mathcal{T}^{\star}_{(\Pi,<)}$ be the operator obtained from $\mathcal{T}^{\circ}_{(\Pi,<)}$ (in Definition 3) by replacing $Y \setminus D^r_X$ with $Cl(\Pi^Y \setminus (\Pi^r_X)^+)$. This results in a fixpoint operator $\mathcal{C}^{\star}_{(\Pi,<)}$.

As we show in the full version of this paper, Brewka's well-founded set corresponds to the least fixpoint of the alternating operator $\mathcal{C}^{\star}_{(\Pi,<)} C^{\star}_{\Pi}$. This means Brewka's well-founded semantics also enjoys an alternating fixpoint characterization.

6.2 Relation to Other Approaches

In [30], it is mentioned that a well-founded semantics with preference can be defined in terms of their operator but default negation is not allowed in their syntax. However, even for ordered logic programs without default negation, our basic semantic approach is different from the well-founded semantics in priority logic [30]. The main reason is that they interpret the priority relation $r < r'$ in a quite different way: r is blocked whenever r' is applicable. While we attribute to the program

$$\begin{array}{lllll} r_1 & = & p & \leftarrow & \qquad\qquad r_2 < r_3 \\ r_2 & = & q & \leftarrow & \end{array} \tag{6}$$

a preferred well-founded model, containing both p and q, the well-founded model of Π_6 in priority logic is $\{p\}$. That is, q cannot be inferred.

Another skeptical semantics for preference is defeasible logic, which was originally introduced by D. Nute [21] and received extensive studies in recent years [1, 2, 20]. Defeasible logic distinguishes the strict rules from defeasible rules. This already makes its semantics different from our preferred well-founded semantics.

Consider an example from [7]. The following is a theory in defeasible logic:

$$\begin{array}{llll} r'_1 & & \Rightarrow & p \qquad\qquad r'_2 < r'_3 \\ r'_2 & p & \rightarrow & q \\ r'_3 & & \Rightarrow & \neg q \end{array} \tag{7}$$

In defeasible logic, $+\delta q$ is not derivable, i. e., q cannot be defeasibly derived. As pointed out by Brewka, this means a defeasible rule having higher priority can defeat a strict rule.

The above theory can be directly translated into an ordered logic program $(\Pi, <)$ as follows:

$$\begin{array}{lllll} r_1 & = & p & \leftarrow & not\ \neg p \qquad\qquad r_2 < r_3 \\ r_2 & = & q & \leftarrow & p \\ r_3 & = & \neg q & \leftarrow & not\ q \end{array} \tag{8}$$

It can be verified that the preferred well-founded model (in our sense) is $\{p, q\}$. Therefore, q is derivable under our preferred well-founded semantics.

7 Conclusion

We have looked into the issue of how van Gelder's alternating fixpoint theory [27] for normal logic programs can be suitably extended to define the well-founded semantics for ordered logic programs (extended logic programs with preference). The key of the alternating fixpoint approach is how to specify a suitable consequence relation for ordered logic programs. We argue that the preference strategies for defining answer sets are not suitable for defining preferred well-founded semantics and then some informal criteria for preferred well-founded semantics are proposed. Based on this analysis, we have defined a well-founded semantics for ordered logic programs. This semantics allows an elegant definition and satisfies some attractive properties: (1) Each ordered logic program has a unique preferred well-founded model; (2) The preferred well-founded reasoning is no less skeptical than the standard well-founded reasoning; (3) Any conclusion under the preferred well-founded semantics is also derivable under some major preferred answer sets semantics. Our semantics is different from defeasible logic and the skeptical priority logic. An important result is the equivalence of Brewka's preferred well-founded semantics and our semantics introduced in Section 5.

Acknowledgments. This work was supported by DFG under grant FOR 375/1-1, TP C.

References

[1] G. Antoniou, D. Billington, G. Governatori, and M. Maher. A flexible framework for defeasible logics. In *Proc. AAAI/IAAI'2000*, pages 405–410. AAAI Press, 2000.

[2] G. Antoniou, D. Billington, G. Governatori, and M. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, 2001.

[3] F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. In R. Bajcsy, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 669–674. Morgan Kaufmann Publishers, 1993.

[4] C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 12:1–80, 1994.

[5] G. Brewka. Adding priorities and specificity to default logic. In L. Pereira and D. Pearce, editors, *European Workshop on Logics in Artificial Intelligence (JELIA'94)*, Lecture Notes in Artificial Intelligence, pages 247–260. Springer-Verlag, 1994.

[6] G. Brewka. Well-founded semantics for extended logic programs with dynamic preferences. *Journal of Artificial Intelligence Research*, 4:19–36, 1996.

[7] G. Brewka. On the relationship between defeasible logic and well-founded semantics. In T. Eiter, W. Faber, and M. Truszczynski, editors, *Proceedings of the Sixth International Conference on the Logic Programming and Nonmonotonic Reasoning*, pages 121–132. Springer-Verlag, 2001.

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

[9] G. Brewka and T. Eiter. Prioritizing default logic. In St. Hölldobler, editor, *Intellectics and Computational Logic — Papers in Honour of Wolfgang Bibel*, pages 27–45. Kluwer Academic Publishers, 2000.

[10] F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. In *Proceedings of the International Conference on Logic Programming*, pages 79–93. The MIT Press, 1999.

[11] B. Cui and T. Swift. Preference logic grammars: Fixed-point semantics and application to data standardization. *Artificial Intelligence*, 2001. To appear.

[12] J. Delgrande and T. Schaub. Compiling reasoning with and about preferences into default logic. In M. Pollack, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 168–174. Morgan Kaufmann Publishers, 1997.

[13] J. Delgrande, T. Schaub, and H. Tompits. Logic programs with compiled preferences. In W. Horn, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 392–398. IOS Press, 2000.

[14] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. A generic approach for knowledge-based information-site selection. In *Proceedings of the Eighth International Conference on the Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, 2002.

[15] M. Gelfond and T. Son. Reasoning with prioritized defaults. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Third International Workshop on Logic Programming and Knowledge Representation*, volume 1471 of *Lecture Notes in Computer Science*, pages 164–223. Springer-Verlag, 1997.

[16] T. Gordon. *The pleading game: An Artificial Intelligence Model of Procedural Justice*. Dissertation, Technische Hochschule Darmstadt, Alexanderstraße 10, D-64283 Darmstadt, Germany, 1993.

[17] B. Grosof. Prioritized conflict handling for logic programs. In J. Maluszynsk, editor, *Logic Programming: Proceedings of the 1997 International Symposium*, pages 197–211. The MIT Press, 1997.

[18] B. Grosof. Business rules for electronic commerce. `http://www.research.ibm.com/rules/papers.htm` 1999. IBM Research.

[19] J. Lloyd. *Foundations of Logic Programming*. Symbolic Computation. Springer-Verlag, 2nd edition, 1987.

[20] M. Maher, J. Rock, G. Antoniou, D. Billington, and D. Miller. Efficient defeasible reasoning systems. In *Proceedings of the 12th International Conference on Tools with Artificial Intelligence*, pages 384–392. IEEE Press, 2000.

[21] D. Nute. Defeasible reasoning. In *Proceedings of the 20th Hawaii International Conference on Systems Science*, pages 470–477. IEEE Press, 1987.

[22] H. Prakken. *Logical Tools for Modelling Legal Argument*. Kluwer Academic Publishers, 1997.

[23] C. Sakama and K. Inoue. Representing priorities in logic programs. In M. Maher, editor, *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming*, pages 82–96, Cambridge, 1996. The MIT Press.

[24] T. Schaub and K. Wang. A comparative study of logic programs with preference. In B. Nebel, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 597–602. Morgan Kaufmann Publishers, 2001.

[25] A. Tarski. A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[26] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

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

[28] K. Wang, L. Zhou, and F. Lin. Alternating fixpoint theory for logic programs with priority. In *Proceedings of the First International Conference on Computational Logic*, volume 1861 of *Lecture Notes in Computer Science*, pages 164–178. Springer-Verlag, 2000.

[29] C. Witteveen. Partial semantics for truth maintenance. In J. van Eijck, editor, *Logics in AI*, volume 478 of *Lecture Notes in Artificial Intelligence*, pages 544–561. Springer-Verlag, 1991.

[30] J. You, X. Wang, and L. Yuan. Nonmonotonic reasoning as prioritized argumentation. *IEEE Transactions on Knowledge and Data Engineering*, 2001. To appear.

[31] Y. Zhang and N. Foo. Answer sets for prioritized logic programs. In J. Maluszynski, editor, *Proceedings of the International Symposium on Logic Programming (ILPS-97)*, pages 69–84. The MIT Press, 1997.