Weak Order Equivalence for Logic Programs with Preferences

Kathrin Konczak

Institut für Informatik, Universität Potsdam, Postfach 90 03 27, D–14439 Potsdam
konczak@cs.uni-potsdam.de

Abstract. Recently, notions of equivalence for Answer Set Programming have been studied intensively and were shown to be beneficial for modular programming and automated optimization. In [9], the novel notion of strong equivalence for logic programs with rule preferences (so-called ordered logic programs) has been defined and necessary and sufficient conditions for programs being strongly equivalent have been presented. In this paper, we extend this work and analyze a weaker notion of equivalence for ordered logic programs. Whereas strong equivalence makes great demands on ordered logic programs, the weakened notion enables program transformations for simplifying preference relations.

1 Introduction

During the last decade, Answer Set Programming (ASP) [10] has become an increasingly acknowledged tool for knowledge representation and reasoning. A main advantage of ASP is that it is based on solid theoretical foundations, while being able to model common-sense reasoning in an arguably satisfactory way. The availability of efficient solvers has furthermore stimulated its use in practical applications in recent years. This development has quite some implications on ASP research. For example, increasingly large applications require features for modular programming. Another requirement is the fact that, in applications, ASP code is often generated automatically by so-called front-ends, calling for optimization methods which remove redundancies, as also found in database query optimizers. For these purposes the recently suggested notion of strong equivalence for ASP [12, 16] can be used. Indeed, if two ASP programs are strongly equivalent, they can be used interchangeably in any context. This gives a handle on showing the equivalence of ASP modules. If a program is strongly equivalent to a subprogram of itself, then one can always use the subprogram instead of the original program, a technique which serves as an effective optimization method.

On a different line of ASP research, many extensions of the basic formalism have been proposed. One of the most intensively studied one is the modeling of preferences [4]. Strongly rooted in the research of non-monotonic formalisms, the ability to specify preferences is acknowledged to be particularly beneficial for ASP, since preferences constitute a very natural way of resolving indeterminate solutions. E.g., preferences have been successfully used for timetabling, auctioning, and configuration. A sophisticated application for information site selection is presented in [6]. The emergence of such applications thus also calls for optimization methods, as for standard ASP.

In [9], we have generalized the notion of strong equivalence to ASP with preferences. Since a plethora of formalisms and semantics has been introduced for extending ASP by preferences, we have limited ourselves to ordered programs, where preferences are defined among rules, which semantically act as filters over answer sets. In [9], we have found out that strong order equivalent programs must be strongly equivalent in the standard sense, have to coincide on their preference relations, and have to coincide on their set of rules contributing to answer sets. Since these conditions are very strict, we examine in this work a weaker notion of strong order equivalence. Whereas for strongly order equivalent programs we require that they have the same preferred answer sets no matter which *ordered* program we add, the weakened notion only stipulates that they have the same preferred answer sets no matter which *normal* logic program we add.

Section 2 briefly sketches strong order equivalence. In Section 3 we define and characterize weak order equivalence and give some relations to strong order equivalence. In Section 4 we present new program transformations under weak order equivalence, which simplify preference relations and the underlying logic program. In Section 5, we compare strong and weak order equivalence and point out further research issues.

2 Background

A *logic program* is a finite set of rules $p_0 \leftarrow p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n$, where $n \geq m \geq 0$, and each p_i $(0 \leq i \leq n)$ is an *atom*. For such a rule r, we let $head(r)$ denote the *head*, p_0, of r and $body(r)$ the *body*, $\{p_1, \ldots, p_m,\ not\ p_{m+1}, \ldots, not\ p_n\}$, of r. Let $body^+(r) = \{p_1, \ldots, p_m\}$ be the positive and $body^-(r) = \{p_{m+1}, \ldots, p_n\}$ be the negative part of the body of rule r. For a set of rules Π, we write $head(\Pi) = \{head(r) \mid r \in \Pi\}$. A program is *basic* if $body^-(r) = \emptyset$ for all its rules. The *reduct*, Π^X, of a program Π *relative to* a set X of atoms is defined by $\Pi^X = \{head(r) \leftarrow body^+(r) \mid r \in \Pi, body^-(r) \cap X = \emptyset\}$. A set of atoms X is *closed under* a basic program Π if for any $r \in \Pi$, $head(r) \in X$ if $body^+(r) \subseteq X$. The smallest set of atoms being closed under a basic program Π is denoted by $Cn(\Pi)$. Then, a set X of atoms is an *answer set* of a program Π if $Cn(\Pi^X) = X$. We use $AS(\Pi)$ for denoting the set of all answer sets of Π. Given a set X of atoms, we define $R_\Pi(X) = \{r \in \Pi \mid body^+(r) \subseteq X, body^-(r) \cap X = \emptyset\}$ as *generating rules* of X.

An *ordered (logic) program* is a pair $(\Pi, <)$, where Π is a logic program and $< \subseteq \Pi \times \Pi$ is a strict partial order. Given, $r_1, r_2 \in \Pi$, the relation $r_1 < r_2$ expresses that r_2 has *higher priority* than r_1. This informal interpretation can be made precise in different ways. In what follows, we consider three such interpretations: D– [3], B– [2], and W– preference [18]. Given $(\Pi, <)$, all of them use $<$ for selecting preferred answer sets among the standard answer sets of Π. We write $(\Pi, \emptyset)$ whenever no preferences are specified. Due to space restrictions, we give below a formal definition of D–preference, and explain more informally B– and W– preferences [2, 18].

Definition 1. *Let $(\Pi, <)$ be an ordered logic program and X be an answer set of Π. Then, X is $<^D$–preferred, if an enumeration $\langle r_i \rangle_{i \in I}$ of Π exists s.t. for every $i, j \in I$ we have*

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

Condition 1 stipulates that the enumeration of Π is compatible with $<$. Condition 2 makes the property of supportedness explicit. Although any answer set is generated by a supported sequence of rules, in D–preferences, rules cannot be supported by lower-ranked ones. Condition 3a separates the handling of unsupported rules from preference handling. Condition 3b ensures that rules can never be blocked by lower-ranked ones. For W–preference, the previous concept of order preservation is weakened in Condition 2 and 3 for suspending both conditions, whenever the head of a preferred rule is derivable in an alternative way. Roughly speaking, B–preference additionally drops Condition 2; thus decoupling preference handling from the order induced by consecutive rule applications. For $\sigma \in \{D, W, B\}$, we define $AS^\sigma((\Pi, <))$ as the set of all $<^\sigma$-preferred answer sets of the ordered program $(\Pi, <)$. Note that these preference semantics do not always guarantee the existence of a preferred answer set. For example, the program

$$(\Pi, <) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : b \leftarrow not\ a \\ r_1 < r_2 \end{array} \right\}$$

has in all three semantics no preferred answer set, since rule r_2 is blocked by the lower ranked rule r_1.

Strong Order Equivalence has been defined and analyzed for the first time in [9]. Let $\sigma \in \{D, W, B\}$ be denoting the underlying preference semantics. Two ordered programs $(\Pi_1, <_1)$ and

$(\Pi_2, <_2)$ are $<^\sigma$-*equivalent*, denoted $(\Pi_1, <_1) \equiv^\sigma (\Pi_2, <_2)$, iff they have the same $<^\sigma$-preferred answer sets. Regarding strong equivalence for ordered programs, one has to consider *all* extensions by ordered programs. Since the union of two ordered programs is not necessarily an ordered program, we have to restrict strong equivalence to *admissible* extensions of ordered programs [9]. An ordered program $(\Pi', <')$ is an admissible extension of $(\Pi, <)$ if $(\Pi \cup \Pi', < \cup <')$ is an ordered program. [1] $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ are *strongly* $<^\sigma$-*equivalent*, denoted by $\equiv_s^\sigma$, iff for all admissible extensions $(\Pi', <')$ of $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ it holds that $(\Pi_1 \cup \Pi', <_1 \cup <')$ and $(\Pi_2 \cup \Pi', <_2 \cup <')$ are $<^\sigma$- equivalent. We say that a rule $r \in \Pi$ *contributes* to an answer set X, if there exists a program Π^* such that $X \in AS(\Pi \cup \Pi^*)$ and $r \in R_{\Pi \cup \Pi^*}(X)$. We define $Cont(\Pi)$ as the set of all rules of Π contributing to any answer set of extensions of Π. In [9] we have presented the following characterizations for strong order equivalence.

Theorem 1. *[9] Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be ordered programs.*
Then, $(\Pi_1, <_1) \equiv_s^B (\Pi_2, <_2)$ iff $\Pi_1 \equiv_s \Pi_2$, [2] *$<_1 = <_2$, and $Cont(\Pi_1) = Cont(\Pi_2)$.*

Theorem 2. *[9] Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be ordered programs and $\sigma \in \{D, W\}$.*
Then, $(\Pi_1, <_1) \equiv_s^\sigma (\Pi_2, <_2)$, iff $\Pi_1 \equiv_s \Pi_2$, $<_1 = <_2$, and $Cont(\Pi_1) \setminus \{r \in \Pi_1 \mid head(r) \in body^+(r)\} = Cont(\Pi_2) \setminus \{r \in \Pi_2 \mid head(r) \in body^+(r)\}$.

For two ordered programs, we have (i) strong $<^B$-equivalence implies strong $<^W$-equivalence and (ii) strong $<^W$-equivalence holds iff strong $<^D$-equivalence holds [9]. Moreover, in [9] program simplifications are given, where only rules not involved in preference relations and not contributing to answer sets can be removed under strong order equivalence. In what follows, we denote $PR((\Pi, <))$ as the set of all rules involved in the preference relation $<$, i.e. $PR((\Pi, <)) = \{r \in \Pi \mid \exists r' \in \Pi : r < r' \text{ or } r' < r\}$.

3 Weak order equivalence

Strong order equivalence considers all admissible extensions of ordered programs by other ordered programs. For considering a weaker notion of order equivalence, we define weak order equivalence, where we consider extensions of ordered logic programs by normal logic programs. Since the union of an ordered program and a normal logic program always represents an ordered program, we don't have to check admissibility as for strong order equivalence.

Definition 2. *(Weak order equivalence) Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be ordered logic programs and $\sigma \in \{D, B, W\}$.*

Then, $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ are weakly $<^\sigma$-equivalent, *denoted by $\equiv_w^\sigma$, iff for all normal programs Π it holds that $(\Pi_1 \cup \Pi, <_1)$ and $(\Pi_2 \cup \Pi, <_2)$ are $<^\sigma$- equivalent.*

In the following, we show some simple relationships between strong and weak order equivalence. Whenever the preference relation is empty, weak order equivalence corresponds exactly to strong equivalence of the underlying logic programs.

Lemma 1. *Let Π_1 and Π_2 be logic programs and $\sigma \in \{D, W, B\}$.*

Then, $(\Pi_1, \emptyset)$ and $(\Pi_2, \emptyset)$ are weakly $<^\sigma$-equivalent iff Π_1 and Π_2 are strongly equivalent.

Also, strong order equivalence implies weak order equivalence.

Lemma 2. *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be ordered logic programs and $\sigma \in \{D, W, B\}$.*

If $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ are strongly $<^\sigma$-equivalent, then $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ are weakly $<^\sigma$-equivalent.

As for strong order equivalence, weak order equivalence requires strong equivalence of the underlying logic programs [9].

[1] The union of two ordered programs is formally defined in [9].
[2] $\Pi_1 \equiv_s \Pi_2$ denotes strong equivalence of Π_1 and Π_2.

Theorem 3. *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be ordered logic programs.*
If $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ are weakly $<^\sigma$-equivalent, for $\sigma \in \{D, W, B\}$, then $\Pi_1 \equiv_s \Pi_2$.

In [9] we have shown that strong order equivalence requires that the programs must coincide on their preference relations (cf. Theorem 1 and 2). Interestingly, this is not required for weak order equivalence. That is, weak order equivalent programs can differ on their preference relations. To see this, consider the following example:

$$(\Pi, <) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : b \leftarrow a \\ r_2 < r_1 \end{array} \right\}$$

We obtain $(\Pi, <) \equiv^\sigma_w (\Pi, \emptyset)$ for $\sigma \in \{D, W, B\}$, since the preference $r_2 < r_1$ never selects answer sets as non-preferred, no matter which extension by normal programs is considered. Regarding strong order equivalence, we take

$$(\Pi', <') = \left\{ \begin{array}{ll} r_3 : y \leftarrow \mathit{not}\ x & \\ r_1 : a \leftarrow & r_1 <' r_3 \\ r_2 : b \leftarrow a & r_4 <' r_2 \\ r_4 : x \leftarrow & \\ r_5 : a \leftarrow z & \\ r_6 : z \leftarrow & \end{array} \right\}$$

and obtain $AS^\sigma((\Pi \cup \Pi', <')) = \{\{a, b, x, z\}\}$ but $AS^\sigma((\Pi \cup \Pi', < \cup <')) = \emptyset$ since the additional preference $r_2 < r_1$ discards the answer set $\{a, b, x, z\}$ as non-preferred. Hence, under weak order equivalence preference relations can be removed, which is not allowed under strong order equivalence. In Section 4 we will have a closer look on program transformations simplifying preference relations.

In contrast to strong order equivalence, weak order equivalent programs can also differ on their sets of generating rules and hence, on their sets of rules contributing to answer sets. For example, consider the following ordered programs:

$$(\Pi_1, <_1) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : b \leftarrow \\ r_2 <_1 r_1 \end{array} \right\} \text{ and } (\Pi_2, <_2) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : b \leftarrow \\ r_3 : a \leftarrow b \\ r_2 <_2 r_1 \end{array} \right\}$$

We obtain $(\Pi_1, <_1) \equiv^\sigma_w (\Pi_2, <_2)$ for $\sigma \in \{D, W, B\}$. By taking

$$(\Pi', <') = \left\{ \begin{array}{l} r_4 : y \leftarrow \mathit{not}\ a \\ r_1 : a \leftarrow \\ r_5 : b \leftarrow x \\ r_6 : x \leftarrow \\ r_1 < r_4 \end{array} \right\}$$

we obtain $AS^\sigma((\Pi_1 \cup \Pi', <_1 \cup <')) = \emptyset$ but $AS^\sigma((\Pi_2 \cup \Pi', <_2 \cup <')) = \{\{a, b, x\}\}$ since r_3 is used to block rule r_4 in an order preserving way. Hence, $(\Pi_1, <_1) \not\equiv^\sigma_s (\Pi_2, <_2)$ for all $\sigma \in \{D, W, B\}$. In Section 4, we will reconsider program simplifications known from (standard) strong equivalence and analyze them under weak order equivalence.

Whenever two ordered programs are weakly order equivalent, we have to impose further conditions for achieving strong order equivalence.

Lemma 3. *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be ordered programs such that $(\Pi_1, <_1) \equiv^B_w (\Pi_2, <_2)$.*
If $<_1 = <_2$ and $Cont(\Pi_1) = Cont(\Pi_2)$, then $(\Pi_1, <_1) \equiv^B_s (\Pi_2, <_2)$.

Lemma 4. *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be ordered programs such that $(\Pi_1, <_1) \equiv^\sigma_w (\Pi_2, <_2)$ and $\sigma \in \{D, W\}$.*
If $<_1 = <_2$ and $Cont(\Pi_1) \setminus \{r \in \Pi_1 \mid head(r) \in body^+(r)\} = Cont(\Pi_2) \setminus \{r \in \Pi_2 \mid head(r) \in body^+(r)\}$, then $(\Pi_1, <_1) \equiv^\sigma_s (\Pi_2, <_2)$.

The preference semantics yield an increasing number of preferred answer sets, i.e. $AS^D((\Pi,<)) \subseteq AS^W((\Pi,<)) \subseteq AS^B((\Pi,<)) \subseteq AS(\Pi)$ for any ordered program $(\Pi,<)$. In [9] we have shown that strong $<^B$-equivalence of two ordered programs implies strong $<^W$-equivalence and that strong $<^W$-equivalence holds if and only if strong $<^D$-equivalence holds. Hence, the differences between the D- and W-semantics disappear under strong order equivalence. Furthermore, the differences between the B-semantics and the D- and W-semantics are strengthens under strong oder equivalence, since the B-semantics decouples preference handling from rule application. Under weak order equivalence, we observe that any relationship between these three semantics disappears.

Theorem 4. *For any $\sigma, \sigma' \in \{D, W, B\}$ with $\sigma \neq \sigma'$, there exist ordered logic programs $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ such that $(\Pi_1, <_1) \equiv_w^\sigma (\Pi_2, <_2)$ but $(\Pi_1, <_1) \not\equiv_w^{\sigma'} (\Pi_2, <_2)$.*

This can be seen by considering the following examples:

$$(\Pi_1, <) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : a \leftarrow a \\ r_1 < r_2 \end{array} \right\} \tag{1}$$

We have $(\Pi_1, <) \equiv_w^B (\Pi_1, \emptyset)$ but $(\Pi_1, <) \not\equiv_w^\sigma (\Pi_1, \emptyset)$ for $\sigma \in \{D, W\}$, since $AS^\sigma((\Pi_1, <)) = \emptyset$ and $\{a\} \in AS^\sigma((\Pi_1, \emptyset))$. Hence, $\equiv_w^B \not\Rightarrow \equiv_w^\sigma$ for $\sigma \in \{D, W\}$. For the ordered program

$$(\Pi_2, <) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : b \leftarrow a \\ r_3 : b \leftarrow \\ r_1 < r_2 \end{array} \right\} \tag{2}$$

we obtain $(\Pi_2, <) \equiv_w^W (\Pi_2, \emptyset)$ but $(\Pi_2, <) \not\equiv_w^D (\Pi_2, \emptyset)$, since $AS^D((\Pi_2, \emptyset)) = \{\{a, b\}\}$ and $(\Pi_2, <)$ has no $<^D$-preferred answer set. Hence, $\equiv_w^W \not\Rightarrow \equiv_w^D$.

For $\equiv_w^W \not\Rightarrow \equiv_w^B$, let us consider

$$(\Pi_{3a}, <) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : a \leftarrow a \\ r_3 : y \leftarrow \mathit{not}\ a \\ r_1 < r_3 \end{array} \right\} \text{ and } (\Pi_{3b}, <) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_3 : y \leftarrow \mathit{not}\ a \\ r_1 < r_3 \end{array} \right\} \tag{3}$$

We have $(\Pi_{3a}, <) \equiv_w^W (\Pi_{3b}, <)$, but $(\Pi_{3a}, <) \not\equiv_w^B (\Pi_{3b}, <)$ since $AS^B((\Pi_{3b}, <)) = \emptyset$ and $\{a\} \in AS^B((\Pi_{3a}, <))$ To see $\equiv_w^D \not\Rightarrow \equiv_w^\sigma$, for $\sigma \in \{W, B\}$, let us consider

$$(\Pi_{4a}, <) = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : b \leftarrow \\ r_3 : a \leftarrow \mathit{not}\ b \\ r_2 < r_3 \end{array} \right\} \text{ and } (\Pi_{4b}, <') = \left\{ \begin{array}{l} r_1 : a \leftarrow \\ r_2 : b \leftarrow \\ r_4 : c \leftarrow \mathit{not}\ b \\ r_2 <' r_4 \end{array} \right\} \tag{4}$$

We observe $(\Pi_{4a}, <) \equiv_w^D (\Pi_{4b}, <')$, but $AS^\sigma((\Pi_{4b}, <')) = \emptyset$ and $AS^\sigma((\Pi_{4a}, <)) = \{\{a, b\}\}$ for $\sigma \in \{W, B\}$.

4 Transformations

In this section we provide some simplifications of ordered programs under weak order equivalence. Program simplifications that can be applied under strong order equivalence are also allowed under weak order equivalence. That is, $(\Pi, <) \equiv_w^\sigma (\Pi \setminus \{r\}, <)$ holds for rules r, where (i) $r \notin PR((\Pi, <))$, $\mathit{head}(r) \in \mathit{body}^+(r)$ and $\sigma \in \{D, W\}$, or (ii) $r \notin PR((\Pi, <))$, $r \notin \mathit{Cont}(\Pi)$, and $\sigma \in \{D, W, B\}$ [9]. Note that preference relations cannot be simplified under strong order equivalence (cf. Theorems 1 and 2).

4.1 Simplifications of preference relations

In contrast to strong order equivalence, we consider now program transformations under weak order equivalence simplifying preference relations. Whenever we remove a preference relation $r < r'$, all other preference relationships are maintained, e.g. we have $r^* < r < r'$ and remove $r < r'$, we keep the relations $r^* < r$ and $r^* < r'$.

Preference relations reflecting the order of rule application in enumerations (cf. Definition 1) are redundant under weak order equivalence and can be removed.

Theorem 5. *Let $(\Pi, <)$ be an ordered logic program and $r_1, r_2 \in \Pi$ such that $body(r_1) = \emptyset$, $head(r_1) \in body(r_2)$ and $r_2 < r_1$.*

Then, $(\Pi, <) \equiv_w^\sigma (\Pi, <')$ for $<' = < \setminus \{r_2 < r_1\}$ and $\sigma \in \{D, W, B\}$.

Proof 5 Assume $(\Pi, <) \not\equiv_w^\sigma (\Pi, <')$ holds for $<' = < \setminus \{r_2 < r_1\}$. Then, there exists an Π' such that $(\Pi \cup \Pi', <) \not\equiv^\sigma (\Pi \cup \Pi', <')$. Since $<' \subseteq <$, we have $AS^\sigma((\Pi \cup \Pi', <)) \subseteq AS^\sigma((\Pi \cup \Pi', <'))$ [15]. That is, there exists an $X \in AS^\sigma((\Pi \cup \Pi', <'))$ such that $X \notin AS^\sigma((\Pi \cup \Pi', <))$. Since $X \in AS^\sigma((\Pi \cup \Pi', <'))$, there exists an $<^\sigma$-preserving enumeration E of $\Pi \cup \Pi'$ wrt X. Whenever r_1 precedes r_2 in this enumeration, E is also an $<^\sigma$-preserving enumeration for $(\Pi \cup \Pi', <)$, which is a contradiction to $X \notin AS^\sigma((\Pi \cup \Pi', <))$. Hence, for all $<^\sigma$-preserving enumeration E wrt $(\Pi \cup \Pi', <')$ and X we have that r_2 precedes r_1. All rules that are higher preferred than r_1 are higher preferred than r_2, since $<$ is transitive. Hence, all rules higher preferred than r_1 must precede r_2 in an order preserving enumeration. Thus, r_1 could precede r_2 , which leads to a contradiction to the assumption. □

Since B- preferences decouple preference handling from rule application, we can delete all preferences between rules r_1 and r_2 that are applicable wrt any answer set and any extension by a logic program and where $head(r_1) \in body^+(r_2)$. For this, we define similarly to the T_Π operator [13] the following:

$$
\begin{array}{ll}
A^0(\Pi) & = \{head(r) \mid r \in \Pi, body(r) = \emptyset\} \\
A^{i+1}(\Pi) & = \{head(r) \mid r \in \Pi, body^-(r) = \emptyset, body^+(r) \subseteq A^i(\Pi)\} \\
A(\Pi) & = \bigcup_{0 \leq i} A^i(\Pi)
\end{array}
$$

The set $A(\Pi)$ covers atoms that are true in every answer set. We say that $r \in Appl(\Pi)$ whenever $body^-(r) = \emptyset$ and $body^+(r) \subseteq A(\Pi)$. That is, r is a generating rule in all program extensions Π' of Π and for all answer sets X of $\Pi \cup \Pi'$.

The following theorem states that we can remove a preference relation between rules in $Appl(\Pi)$ under the B-semantics.

Theorem 6. *Let $(\Pi, <)$ be an ordered logic program and $r_1, r_2 \in Appl(\Pi)$ such that $head(r_1) \in body^+(r_2)$. Then, $(\Pi, <) \equiv_w^B (\Pi, <')$ for $<' = < \setminus \{r_2 < r_1, r_1 < r_2\}$.*

Proof 6 Assume, $(\Pi, <) \not\equiv_w^B (\Pi, <')$ holds for $<' = < \setminus \{r_2 < r_1, r_1 < r_2\}$. Then, there exists an $X \in AS^B((\Pi \cup \Pi', <'))$ such that $X \notin AS^B((\Pi \cup \Pi', <))$.

Let be $<' = < \setminus \{r_2 < r_1\}$. Since $r_1 \in Appl(\Pi)$ and all rules higher preferred than r_1 are also higher preferred than r_2, there always exists an $<^B$- preserving enumeration where r_1 is enumerated before r_2. Hence, there exists a $<^B$- preserving enumeration of $(\Pi \cup \Pi', <)$ that is also order preserving for $(\Pi \cup \Pi', <')$ wrt X. Hence, $X \in AS^B((\Pi \cup \Pi', <))$. The case $<' = < \setminus \{r_1 < r_2\}$ is analogous. □

The B-semantics allows to block a rule r by a lower ranked one r' as long as $head(r)$ is in the answer set [2]. By stipulating $head(r) \in A(\Pi)$, we make sure that $head(r)$ is in any resulting answer set and hence, we can remove the corresponding preference relation.

Theorem 7. *Let $(\Pi, <)$ be an ordered logic program and $r_1, r_2 \in \Pi$ such that $r_1 \in Appl(\Pi)$, $head(r_1) \in body^-(r_2)$ and $head(r_2) \in A(\Pi)$.*

Then, $(\Pi, <) \equiv_w^B (\Pi, <')$ for $<' = < \setminus \{r_1 < r_2\}$.

Proof 7 Assume $(\Pi,<) \not\equiv_w^B (\Pi,<')$ holds for $<'=< \setminus\{r_1 < r_2\}$. Then, there exists an $X \in AS^B((\Pi \cup \Pi',<'))$ such that $X \notin AS^B((\Pi \cup \Pi',<))$. Let E be an $<^B$-preserving enumeration wrt $(\Pi \cup \Pi',<')$ and X. We observe that r_2 is blocked by r_1 and $head(r_2) \in A(\Pi)$. Hence, $head(r_2) \in X$ since $A(\Pi) \subseteq X$. Thus, r_2 can be enumerated directly before r_1 . For this reason $X \in AS^B((\Pi \cup \Pi',<))$. $\square$

E.g., for the program $(\Pi,<) = \{r_1 : a \leftarrow,\ r_2 : b \leftarrow not\ a,\ r_3 : b \leftarrow,\ r_1 < r_2\}$ we obtain $(\Pi,<) \equiv_w^B (\Pi,\emptyset)$. For the W-semantics, this simplifications is not directly conferrable. There, rules can be applied and blocked by lower ranked ones, whenever the head of such a rule is derived earlier in an order preserving enumeration. But this can only be guaranteed by considering possible enumerations of rules from Π.

Within all three semantics, the preference relation $<$ is redundant whenever all standard answer sets are also preferred ones and all rules involved in $<$ are either in $Appl(\Pi)$ or blocked wrt $A(\Pi)$.

Theorem 8. *Let $(\Pi,<)$ be an ordered logic program such that $PR(\Pi) \subseteq \{r \in Appl(\Pi)\} \cup \{r \in \Pi \mid body^-(r) \cap A(\Pi) \neq \emptyset\}$, and $AS^\sigma((\Pi,<)) = AS(\Pi)$ for some $\sigma \in \{D,W,B\}$.*
Then, $(\Pi,<) \equiv_w^\sigma (\Pi,\emptyset)$.

Proof 8 Assume $(\Pi,<) \not\equiv_w^\sigma (\Pi,\emptyset)$. Then, there exists an Π' and $X \in AS^\sigma((\Pi \cup \Pi',\emptyset))$ such that $X \notin AS^\sigma((\Pi \cup \Pi',<))$. Since all rules involved in $<$ are in $Appl(\Pi)$, there exists an order preserving enumeration of $Appl(\Pi)$. Furthermore, rules cannot be blocked by lower ranked ones, since $<$ discards no answer set as non-preferred and all non-applicable rules involved in $<$ are blocked from $A(\Pi)$. Hence, there exists an order preserving enumeration of $\Pi \cup \Pi'$ wrt X and $<$. $\square$

4.2 Transformations from standard strong equivalence

In [1, 17, 8, 14], transformations on logic programs are reported, which can be used for simplifying programs. For those modular transformations, programs are strongly equivalent to the transformed one. Such program simplifications were considered for strong order equivalence in [9]. Moreover, under strong order equivalence we can remove rules $r \notin PR((\Pi,<))$ and where either $head(r) \in body^+(r)$ or $r \notin Cont(\Pi)$ holds.

In what follows we reconsider program simplification from [1, 17, 8, 14] under the notion of weak order equivalence. Since we have presented simplifications for preference relations in the previous section, we will now assume that removable, hence redundant, rules are not involved in preference relations. That is, we separate preference simplifications from simplifications of the underlying logic programs.

The transformation **TAUT**, stating that $\Pi \equiv_s \Pi \setminus \{r\}$ for all $r \in \Pi$ where $head(r) \in body^+(r)$, is allowed under strong order equivalence for the D- and W-semantics, but not for the B-semantics. The same applies for weak order equivalence. Regarding B-semantics, let us consider

$$(\Pi,<) = \left\{ \begin{array}{l} r_1 : a \leftarrow a \\ r_2 : a \leftarrow \\ r_3 : y \leftarrow not\ a \\ r_2 < r_3 \end{array} \right\}$$

We observe $AS^B((\Pi,<)) = \{a\}$, but $AS^B((\{r_2,r_3\},<)) = \emptyset$. Hence, rules where $head(r) \in body^+(r)$ can not removed under weak $<^B$-equivalence.

The simplifications **RED**$^-$ and **CONTRA** [8] are allowed under strong order equivalence as long as the redundant rules are not involved in the preference relation. Hence, we can apply them analogously under weak order equivalence.

Let be $r_1, r_2 \in \Pi$, $head(r_1) = head(r_2)$, and $body(r_2) \subseteq body(r_1)$, then the transformation **NONMIN** states $\Pi \equiv_s \Pi \setminus \{r_1\}$. Let us consider the following example

$$(\Pi, <) = \left\{ \begin{array}{l} r_2 : a \leftarrow \\ r_1 : a \leftarrow b \\ r_3 : b \leftarrow \\ r_4 : y \leftarrow not\ a \\ r_2 < r_4 \end{array} \right\} \not\equiv_w^\sigma (\Pi \setminus \{r_1\}, <).$$

Here, we can not remove r_1 since r_2 is involved in $<$ and r_1 is used to derive $head(r_1) = head(r_2)$ in an alternative way. Hence, we suppose that this transformation can be applied under weak order equivalence whenever $r_1, r_2 \notin PR((\Pi, <))$.

Lemma 5. *Let $(\Pi, <)$ be an ordered logic program and $\sigma \in \{D, W, B\}$. Furthermore, let be $r_1, r_2 \in \Pi$ such that $head(r_1) = head(r_2)$, $body(r_2) \subseteq body(r_1)$, and $r_1, r_2 \notin PR((\Pi, <))$.*
Then, $(\Pi, <) \equiv_w^\sigma (\Pi \setminus \{r_1\}, <)$.

Proof 5 Assume $(\Pi, <) \not\equiv_w^\sigma (\Pi \setminus \{r_1\}, <)$. Then, there exists an Π', $r_1 \notin \Pi'$, such that $(\Pi \cup \Pi', <) \not\equiv^\sigma (\Pi \cup \Pi' \setminus \{r_1\}, <)$. Abbreviatory, we write Π^* for $\Pi \cup \Pi'$. There are 2 cases. Case 1: There exists an $X \in AS^\sigma((\Pi^*, <))$ such that $X \notin AS^\sigma((\Pi^* \setminus \{r_1\}, <))$; Case 2: There exists an $X \in AS^\sigma((\Pi^* \setminus \{r_1\}, <))$ such that $X \notin AS^\sigma((\Pi^*, <))$. In Case 1, we have an $<^\sigma$- preserving enumeration of Π^* wrt X, but not for $\Pi^* \setminus \{r_1\}$. That is, r_1 is used to derive rules or to block rules in an order preserving way, which can not be done by r_2. That is, r_1 precedes r_2 in any order-preserving enumeration (otherwise r_2 can be used to derive rules or to block rules). But this is a contradiction to $body(r_2) \subseteq body(r_1)$ and $r_1, r_2 \notin PR((\Pi, <))$. In Case 2, r_1 can always be inserted at the end of the enumeration E and we get an $<^\sigma$- preserving enumeration of Π^* wrt X, which is a contradiction to the assumption. Thus, $(\Pi, <) \equiv_w^\sigma (\Pi \setminus \{r_1\}, <)$. $\Box$

Let be $r, r' \in \Pi$ such that there exists an $A \subseteq body^-(r')$ such that $head(r) \in head(r') \cup A$, $body^-(r) \subseteq body^-(r') \setminus A$ and $body^+(r) \subseteq body^+(r')$, then the transformation **S-IMP** states that $\Pi \equiv_s \Pi \setminus \{r'\}$. Since this transformation has been developed for programs with disjunction in the head of rules, we have to adapt the condition $head(r) \in head(r') \cup A$ to normal logic programs. Whenever $head(r) = head(r')$, we have exactly the transformation **NONMIN**. Whenever $head(r) \in A$ we obtain that r' never contributes to an answer set. Hence, this transformation, where $head(r) \in A$, can be applied under strong and under weak order equivalence.

The following transformation creates new rules to make other transformations applicable. Let be $r_1 \in \Pi$, where $a \in body^+(r_1)$, $G_a = \{r_2 \in \Pi \mid head(r_2) = a\}$, and $G_a \neq \emptyset$. Then, transformation **WGPPE** states that $\Pi \equiv_s \Pi \cup G'_a$ holds where $G'_a = \{head(r_1) \leftarrow (body^+(r_1) \setminus \{a\}) \cup not\ body^-(r_1) \cup body(r_2) \mid r_2 \in G_a\}$. Let us consider the following example:

$$(\Pi, <) = \left\{ \begin{array}{l} r_2 : a \leftarrow \\ r_1 : b \leftarrow a \\ r_3 : y \leftarrow not\ b \\ r_1 < r_3 \end{array} \right\} \text{ and } (\Pi \cup \{r^{G_a}\}, <) = \left\{ \begin{array}{l} r_2 : a \leftarrow \\ r_1 : b \leftarrow a \\ r^{G_a} : b \leftarrow \\ r_3 : y \leftarrow not\ b \\ r_1 < r_3 \end{array} \right\}$$

for $\sigma \in \{D, W, B\}$. We observe $(\Pi, <) \not\equiv_w^\sigma (\Pi \cup \{r^{G_a}\}, <)$. Analogously to Lemma 5, we observe that r_1 cannot be involved in $<$ if this transformation should be possible under $\equiv_w^\sigma$, since rules from G'_a are alternatives to derive rules in an order preserving way.

Lemma 6. *Let $(\Pi, <)$ be an ordered logic program and $\sigma \in \{D, W, B\}$. Furthermore, let be $r_1 \in \Pi$, $r_1 \notin PR((\Pi, <))$ $a \in body^+(r_1)$, $G_a = \{r_2 \in \Pi \mid head(r_2) = a\}$ and $G_a \neq \emptyset$. Then, $(\Pi, <) \equiv_s^\sigma (\Pi \cup G'_a, <)$ for $G'_a = \{head(r_1) \leftarrow (body^+(r_1) \setminus \{a\}) \cup not\ body^-(r_1) \cup body(r_2) \mid r_2 \in G_a\}$.*

5 Conclusions and Further work

We have presented the notion of weak order equivalence for logic programs with rule preferences, so called ordered programs. We have considered three semantics for handling preferences and have characterized weak order equivalence under these semantics. Furthermore, we have provided several program transformations for simplifying preference relations and the underlying logic programs.

The three considered preference semantics have successfully been used for information-site selection [7]. There, a prototypical environment of a movie domain has been developed, which comprises (i) basic domain knowledge, (ii) XML sources containing movie data wrapped from the Internet Movie Database [11] and other movie related data sources, and (iii) suitable site descriptions. Queries are formulated in XML-QL [5], and can be executed after site selection on the respective source. Rule-based preferences are then used to select sites such that the utility of the answer, in terms of quality of the result and other criteria is as large as possible for the user. Hence, notions of equivalence for ordered programs can be used to optimize databases.

In [9], strong order equivalence for ordered programs has been defined and characterized for the considered preference semantics. Strong order equivalence imposes rigorous conditions on ordered programs being strongly order equivalent (cf. Theorem 1 and 2), e.g. programs have to coincide on their preference relations and on rules contributing to answer sets. Hence, we have considered in this work a weaker notion of order equivalence. We have pointed out that this weaker notion imposes softer conditions on ordered programs being weakly order equivalent. More precisely, only strong equivalence of the underlying logic program is required, whereas the identicalness of preference relations and rules contributing to answer sets is not supposed.

The considered preference semantics yield an increasing number of preferred answer sets, i.e. $AS^D((\Pi,<)) \subseteq AS^W((\Pi,<)) \subseteq AS^B((\Pi,<)) \subseteq AS(\Pi)$ for any ordered program $(\Pi,<)$. In [9], we have found out that under strong equivalence this relationship is changed. More precisely, if two ordered programs are strongly $<^B$- equivalent, then they are strongly $<^W$- equivalent, and they are strongly $<^W$-equivalent if and only if they are strongly $<^D$-equivalent. Interestingly, Theorem 4 shows us that under weak order equivalence no relationship between the semantics exists.

In Section 4, we have considered several program transformations on ordered programs under weak order equivalence wrt the underlying preference semantics. We have concentrated on two types of program transformations: (1) simplifications of preference relations and (2) simplifications of the underlying logic programs. This is reasonable since rules and preferences interact with each other. Theorem 5- 7 describe simplifications of preference relations. Theorem 8 shows one condition when the given preference relation is totally redundant. In further studies we want to characterize preference relations that have no influence on determing preferred answer sets.

Regarding simplifications of the underlying logic programs, we have reconsidered the transformations in [1, 17, 8, 14] wrt strong equivalence. All are considered under the aspect that removable rules are not involved in preference relations. The transformation **TAUT** is possible under weak and strong order equivalence only for the D- and W-semantics. Reductions **RED**$^-$ and **CONTRA** are possible under weak and strong order equivalence wrt all preference semantics. Transformation **NONMIN** and **WGPPE** are not possible under strong order equivalence, but they become applicable under weak order equivalence as long as one makes further restrictions to the preference relations. The simplification **S-IMP** restricted to normal logic programs falls in one case back to **NONMIN** and in the other case, the redundant rules becomes never applicable. Hence, in the last case, the transformation is possible under strong and weak order equivalence.

As with strong order equivalence, a definition of weak order equivalence in terms of SE-models [16] seems to be inappropriate since SE-models are atom-based and the considered preference semantics are rule-based.

In an extended version of this paper, we want to define other simplifications on ordered programs under weak order equivalence, where a difference between the preference semantics is appearing. In this paper, we have made a first step into this direction with Theorems 6 and 7 for the B-semantics. Also, we have studied only transformations, where one rule or preference relations are removed separately. In future work, we want to analyze program transformations, which (i) remove several rules in one step and (ii) remove rules and preference relations together. Beyond this, we want to investigate complexity issues.

Acknowledgements

The author was supported by the German Science Foundation (DFG) under grant SCHA 550/6-4, TP C and by the EC under project IST-2001-37004 WASP.

References

1. S. Brass and J. Dix. Semantics of (disjunctive) logic programs based on partial evaluation. *Journal of Logic Programming*, 40(1):1–46, 1999.
2. G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, 1999.
3. J. Delgrande, T. Schaub, and H. Tompits. A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming*, 3(2):129–187, March 2003.
4. J. Delgrande, T. Schaub, H. Tompits, and K. Wang. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*, 20(2):308–334, 2004.
5. A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language for xml. *Computer Networks*, 31(11-16):1155–1169, 1999.
6. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. A Generic Approach for Knowledge-Based Information Site Selection. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proceedings Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR-02), April 22-25, Toulouse, France*, pages 459–469. Morgan Kaufmann, 2002. Extended version Technical Report INFSYS RR-1843-02-09, TU Wien, 2002.
7. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. A generic approach for knowledge-based information-site selection. In D. Fensel, F. Giunchiglia, D. McGuiness, and M. Williams, editors, *Proceedings of the Eighth International Conference on the Principles of Knowledge Representation and Reasoning*, pages 459–469. Morgan Kaufmann Publishers, 2002.
8. T. Eiter, M. Fink, H. Tompits, and S. Woltran. Simplifying logic programs under uniform and strong equivalence. In V. Lifschitz and I. Niemelä, editors, *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04)*, volume 2923 of *Lecture Notes in Computer Science*, pages 87–99. Springer-Verlag Heidelberg, 2004.
9. W. Faber and K. Konczak. Strong equivalence for logic programs with preferences. In L. Kaelbling and A. Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 430–435, 2005.
10. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the International Conference on Logic Programming*, pages 1070–1080. The MIT Press, 1988.
11. The internet movie database. `http://imdb.com/`.
12. V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.
13. J. Lloyd. *Foundations of Logic Programming.* Symbolic Computation. Springer-Verlag, 2nd edition, 1987.
14. M. Osorio, J.A. Navarro, and J. Arrazola. Equivalence in answer set programming. In *Logic Based Program Synthesis and Transformation, 11th International Workshop (LOPSTR 2001)*, volume 2372 of *Lecture Notes in Computer Science*, pages 57–75. Springer, 2001.
15. T. Schaub and K. Wang. A semantic framework for preference handling in answer set programming. *Theory and Practice of Logic Programming*, 3(4-5):569–607, 2003.
16. H. Turner. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4-5):609–622, 2003.
17. K. Wang and L. Zhou. Comparisons and computation of well-founded semantics for disjunctive logic programs. *ACM Transactions on Computational Logic*, 6(2):295–327, 2005.
18. 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.