Notions of Strong Equivalence for Logic Programs with Ordered Disjunction*

Wolfgang Faber
Department of Mathematics
University of Calabria
Via P. Bucci, cubo 30b
87036 Rende (CS), Italy
wf@wfaber.com

Hans Tompits
Institut für Informationssysteme 184/3
Technische Universität Wien
Favoritenstrasse 9-11
A-1040 Vienna, Austria
tompits@kr.tuwien.ac.at

Stefan Woltran
Institut für Informationssysteme 184/2
Technische Universität Wien
Favoritenstrasse 9-11
A-1040 Vienna, Austria
woltran@dbai.tuwien.ac.at

Abstract

Ordered disjunctions have been introduced as a simple, yet expressive approach for representing preferential knowledge by means of logic programs. The semantics for the resulting language is based on the answer-set semantics, but comes in different variants, depending on the particular interpretation of preference aggregation associated to the ordered disjunction connective. While in standard answer-set programming the question of when a program is to be considered equivalent to another received increasing attention in recent years, this problem has not been addressed for programs with ordered disjunctions so far. In this paper, we discuss the concept of strong equivalence in this setting. We introduce different versions of strong equivalence for programs with ordered disjunctions and provide model-theoretic characterisations, extending well-known ones for strong equivalence between ordinary logic programs. Furthermore, we discuss the relationships between the proposed notions and study their computational complexity.

Introduction

During the last decade, *answer-set programming* (ASP) has become an increasingly acknowledged tool for declarative knowledge representation and reasoning (Gelfond & Lifschitz 1988; Marek & Truszczyński 1999; Niemelä 1999; Baral 2002). A main advantage of ASP is that it is based on solid theoretical foundations while being able to model commonsense reasoning in an arguably satisfactory way. The availability of efficient solvers has furthermore stimulated its use in practical applications in recent years. This development had quite some implications on ASP research. For example, increasingly large applications require features for modular programming. Another issue is the fact that in applications, ASP code is often generated automatically by so-called *frontends*, calling for optimisation methods which remove redundancies, as also found in database query optimisers. For these purposes, the fairly recently suggested notion of *strong equivalence* for ASP (Lifschitz, Pearce, & Valverde 2001; Turner 2003) can be used. Intuitively, two programs P and Q are strongly equivalent iff, for any program R, $P \cup R$ and $Q \cup R$ have the same answer sets. To put it another way, two ASP programs are strongly equivalent if they can be used interchangeably in any context (accordingly, the program R above is also referred to as the *context program*). This gives a handle on showing the equivalence of ASP modules. Moreover, if a program is strongly equivalent to a subprogram of itself, then one can always use the subprogram instead of the original program, yielding potential for optimisation.

Among the different lines of ASP research, many extensions of the basic formalism have been proposed—an important one is the modelling of preferences in ASP (Delgrande *et al.* 2004). Strongly rooted in the research of nonmonotonic formalisms, the ability to specify preferences is acknowledged to be particularly beneficial to ASP, since they constitute a very natural and effective way of resolving indeterminate solutions.

A recent means of representing preferences is ASP with *ordered disjunctions* (Brewka 2002; Brewka, Niemelä, & Syrjänen 2004). The basic idea is to augment the syntax by a designated operator "$\times$" to form ordered disjunctions. Programs of this form (called *logic programs with ordered disjunctions*, or LPODs) can be evaluated in a standard way or with respect to different preferential semantics which take the occurrences of this new operator into account. The system *psmodels* (Brewka, Niemelä, & Syrjänen 2004) serves as a computational engine for these semantics. Applications of logic programs with ordered disjunctions include policy languages (Bertino, Mileo, & Provetti 2005), planning (Zepeda *et al.* 2005), and game theory (Foo, Meyer, & Brewka 2004). Also, extended ASP formalisms, like CR-Prolog (Balduccini & Mellarkod 2003), have been enhanced with ordered disjunctions.

In this paper, we examine how the inclusion of preferences in the form of ordered disjunctions affects equivalence, and in particular strong equivalence of ASP. To this end, we introduce different notions of strong equivalence for LPODs. The distinguishing aspects of these notions are (i) whether the context programs are arbitrary LPODs or just ordinary programs, i.e., whether the context may change preferential information, and (ii) whether the semantics is taken in terms of standard answer sets or preferred an-

*This work was partially supported by the Austrian Science Fund (FWF) under grant P18019 and by M.I.U.R. within projects "Potenziamento e Applicazioni della Programmazione Logica Disgiuntiva" and "Sistemi basati sulla logica per la rappresentazione di conoscenza: estensioni e tecniche di ottimizzazione".
Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

swer sets. Following Brewka, Niemelä, & Syrjänen (2004), we study three different preference strategies, viz. Pareto-, inclusion-, and cardinality-based relations (identified using the letters p, i, and c, respectively). More formally, we introduce the following relations: for all LPODs P and Q,

- $P \equiv_s Q$ holds iff the standard answer sets of P and Q coincide under any extension by ordinary programs;
- $P \equiv_{s,\times} Q$ holds iff the standard answer sets of P and Q coincide under any extension by LPODs;
- $P \equiv_s^\sigma Q$ holds iff the σ-preferred answer sets of P and Q coincide under any extension by ordinary programs (for $\sigma \in \{p, i, c\}$); and
- $P \equiv_{s,\times}^\sigma Q$ holds iff the σ-preferred answer sets of P and Q coincide under any extension by LPODs (for $\sigma \in \{p, i, c\}$).

For all of these notions we provide novel model-theoretic characterisations and we discuss relations among them. Furthermore, the notions coincide for ordinary programs, so they properly generalise the usual concept of strong equivalence. Interestingly, the two relations on standard answer sets, $\equiv_s$ and $\equiv_{s,\times}$, coincide and can be characterised in a similar fashion as strong equivalence for ordinary programs. That is to say, this characterisation uses a generalisation of the concept of an *SE-model* (Turner 2003), based on a novel notion of a reduct, extending the usual reduct as introduced by Gelfond & Lifschitz (1988). For the two relations on preferred answer sets, $\equiv_s^\sigma$ and $\equiv_{s,\times}^\sigma$, it turns out that, for each $\sigma \in \{p, i, c\}$, $\equiv_{s,\times}^\sigma$ is a proper subrelation of $\equiv_s^\sigma$. Hence, for all LPODs P, Q, $P \equiv_{s,\times}^\sigma Q$ implies $P \equiv_s^\sigma Q$, but the converse direction does not hold in general. The model-theoretic characterisations for these notions include the ones for $\equiv_s$ but in addition require specific conditions (with respect to the chosen preference strategy) on answer-sets candidates.

Our analysis about the relationships between the different introduced equivalence notions actually provides a complete picture. Moreover, we also discuss the computational complexity of equivalence checking, showing that equivalence checking for LPODs under the considered notions has the same worst-case complexity as checking strong equivalence for ordinary programs, viz. this task is co-NP-complete for each equivalence notion.

Preliminaries

A *logic program with ordered disjunction* (LPOD) (Brewka, Niemelä, & Syrjänen 2004) is a finite set of rules of the form

$$p_1 \times \cdots \times p_k \leftarrow p_{k+1}, \ldots, p_m, \mathit{not}\ p_{m+1}, \ldots, \mathit{not}\ p_n, \quad (1)$$

where $1 \leq k \leq m \leq n$, and each p_i $(1 \leq i \leq n)$ is an *atom*[1] from a universe U;[2] $\mathcal{P}_U$ denotes the set of all LPODs over universe U. Let r be a rule of form (1). We call k the *arity* of r, and denote it by $\alpha(r)$. Furthermore, r is *normal* if $k = 1$ and *Horn* if $k = 1$ and $m = n$. We use $\mathit{head}(r) = \{p_1, \ldots, p_k\}$ to denote the *head* of r and $\mathit{head}_j(r)$ to denote the j-th element, p_j, from the head of r (for $j > k$, let $\mathit{head}_j(r) = p_k$). Moreover, we define $\mathit{body}(r) = \{p_{k+1}, \ldots, p_m, \mathit{not}\ p_{m+1}, \ldots, \mathit{not}\ p_n\}$, called the *body* of r, $\mathit{body}^+(r) = \{p_{k+1}, \ldots, p_m\}$, and $\mathit{body}^-(r) = \{p_{m+1}, \ldots, p_n\}$. The *$j$-th option* $(1 \leq j \leq k)$ of r is defined as

$$r[j] = p_j \leftarrow p_{k+1}, \ldots, p_m, \mathit{not}\ p_{m+1}, \ldots, \mathit{not}\ p_n, \mathit{not}\ p_1, \ldots, \mathit{not}\ p_{j-1}.$$

We will also write a rule r of form (1) as $p_1 \times \cdots \times p_k \leftarrow \mathit{body}(r)$ whenever convenient, and $r[j]$ as $\mathit{head}_j(r) \leftarrow \mathit{body}(r), \mathit{not}\ \mathit{head}_1(r), \ldots, \mathit{not}\ \mathit{head}_{j-1}(r)$.

For a program P, $\mathit{atoms}(P)$ is given by $\bigcup_{r \in P} (\mathit{head}(r) \cup \mathit{body}^+(r) \cup \mathit{body}^-(r))$. Furthermore, we say that P is *over* V if $\mathit{atoms}(P) \subseteq V$. A program is *normal*, or a logic program (LP) *simpliciter*, if each rule in it is normal. A program is *Horn* if each rule in it is Horn. A *split program* of an LPOD is obtained by replacing each rule by one of its options. Clearly, any split program is normal, and a normal program is the unique split program of itself. The set of all split programs of an LPOD P is denoted as $SP(P)$.

An interpretation I, i.e., a set of atoms, *satisfies* a rule r, symbolically $I \models r$, iff $I \cap \mathit{head}(r) \neq \emptyset$ whenever $\mathit{body}^+(r) \subseteq I$ and $I \cap \mathit{body}^-(r) = \emptyset$ jointly hold. An interpretation I satisfies an LPOD P, symbolically $I \models P$, iff $I \models r$, for each $r \in P$. I is then also called a (*classical*) *model* of P.

The *reduct* (Gelfond & Lifschitz 1988) of an LP P *relative to* an interpretation I is defined by $P^I = \{\mathit{head}(r) \leftarrow \mathit{body}^+(r) \mid r \in P, \mathit{body}^-(r) \cap I = \emptyset\}$. The smallest interpretation satisfying a Horn program P is denoted by $Cn(P)$. An interpretation I is an *answer set* of a normal program P if $Cn(P^I) = I$. The answer sets of an LPOD P are defined as the collection of all answer sets of its split programs.[3] We use $AS(P)$ for denoting the set of all answer sets of P. Hence, $AS(P) = \{I \mid \exists P' \in SP(P) : I \in AS(P')\}$. In order to distinguish the answer sets as introduced here to the preferred answer sets defined below, we call elements from $AS(P)$ also the *standard* answer sets of P.

Let r be a rule of arity k. An interpretation I *satisfies* r *to degree* j, in symbols $I \models_j r$, for $1 \leq j \leq k$, if $I \models r[j]$ and for all $1 \leq i < j$, $I \not\models r[i]$. Note that $I \models_1 r$ also holds if the body of r is not satisfied by I. Intuitively, satisfying a rule to degree 1 means that there is "no better way" to satisfy it. We also use $d_I(r)$ to denote the degree to which r is satisfied under I. This concept gives rise to program partitions, formed by rules which are satisfied to the same degree. Given a program P, an interpretation I, and a degree j, we denote these partitions by $P_I[j] = \{r \in P \mid I \models_j r\}$.

In contrast to Brewka, Niemelä, & Syrjänen (2004), we define the concept of preferences in a more general way:

[1] In contrast to previous work (Brewka 2002; Brewka, Niemelä, & Syrjänen 2004), we do not consider strong negation for reasons of simplicity.

[2] In general, we assume that this universe is fixed and suitably large. Usually, we assume that the universe amounts to the set of atoms occurring in the considered programs.

[3] Brewka, Niemelä, & Syrjänen (2004) provide an alternative definition of answer sets for LPODs based on a reduct. We introduce a different notion of a reduct for our purposes later, however.

While in (Brewka, Niemelä, & Syrjänen 2004) preferences (of different kinds) are always defined with respect to a program, we would like to abstract from programs and refer to a "type" of preference without fixing the program. A *preference schema* (or *preference*, for short) is a mapping $\sigma : \mathcal{P}_U \rightarrow 2^{2^U \times 2^U}$ assigning each LPOD P over universe U a *preference relation* σ between interpretations over U, denoted by $>_P^\sigma$. We call $>_P^\sigma$ also an *instance* of preference σ.

We now define four preference schemata, ϵ, p, i, and, c, respectively referring to the *empty*, a *Pareto-based*, an *inclusion-based*, and a *cardinality-based* preference. Their instances, $>_P^\epsilon$, $>_P^p$, $>_P^i$, and $>_P^c$, are as follows: Let I, J be (classical) models of a program P. Then,

- $I >_P^\epsilon J$ never holds,
- $I >_P^p J$ iff there is a rule $r \in P$ such that $d_I(r) < d_J(r)$, and for no $r \in P$, $d_I(r) > d_J(r)$,
- $I >_P^i J$ iff there is a k such that $P_I[k] \supset P_J[k]$, and for all $j < k$, $P_I[j] = P_J[j]$,
- $I >_P^c J$ iff there is a k such that $|P_I[k]| > |P_J[k]|$, and for all $j < k$, $|P_I[j]| = |P_J[j]|$.

Given a preference σ, an interpretation I is a *σ-preferred answer set* of an LPOD P $I \in AS(P)$ and there is no $J \in AS(P)$ such that $J >_P^\sigma I$. Intuitively, the σ-preferred answer set of an LPOD P are the standard answers sets of P which are maximal with respect to $>_P^\sigma$. The set of all σ-preferred answer sets of an LPOD P is denoted by $AS^\sigma(P)$. We observe that $AS^\epsilon(P) = AS(P)$.

Note that, for the preferences defined above, $I >_P^\epsilon J$ trivially implies $I >_P^p J$, $I >_P^p J$ implies $I >_P^i J$, and $I >_P^i J$ implies $I >_P^c J$, for any program P. Hence, $AS^c(P) \subseteq AS^i(P) \subseteq AS^p(P) \subseteq AS(P)$. There are programs for which all subset-relations are proper, as the following example demonstrates.

Example 1 *Consider the following program P (we label rules with ordered disjunctions for easier reference):*

$$\begin{array}{ll} a \leftarrow not\ b, not\ c; & b \leftarrow not\ a, not\ c; \\ c \leftarrow not\ a, not\ b; & z \leftarrow b; \quad z \leftarrow c; \\ r_a : b \times d \leftarrow a; & r_b : c \times e \times f \leftarrow a; \\ r_c : a \times b \times c \leftarrow z; & r_d : c \times b \leftarrow z. \end{array}$$

The standard answer sets of P are $A_1 = \{a, d, e\}$, $A_2 = \{a, d, f\}$, $A_3 = \{b, z\}$, and $A_4 = \{c, z\}$, with the following rule-satisfaction degrees:

	1	2	3
A_1	$P \setminus \{r_a, r_b\}$	$\{r_a, r_b\}$	$\emptyset$
A_2	$P \setminus \{r_a, r_b\}$	$\{r_a\}$	$\{r_b\}$
A_3	$P \setminus \{r_c, r_d\}$	$\{r_c, r_d\}$	$\emptyset$
A_4	$P \setminus \{r_c\}$	$\emptyset$	$\{r_c\}$

We have that $A_4 >_P^c A_2$, $A_4 >_P^c A_3$, and $A_4 >_P^c A_1$, and therefore $AS^c(P) = \{A_4\}$. We also have $A_1 >_P^i A_2$, $A_4 >_P^i A_3$, but $A_4 \not>_P^i A_1$ and $A_3 \not>_P^i A_1$, hence $AS^i(P) = \{A_1, A_4\}$. But then we have $A_4 \not>_P^p A_3$ ($d_{A_4}(r_d) < d_{A_3}(r_d)$, but $d_{A_4}(r_c) > d_{A_3}(r_c)$) and also $A_2 \not>_P^p A_3$ and $A_1 \not>_P^p A_3$. In addition, $A_1 >_P^p A_2$ holds, and therefore $AS^p(P) = \{A_1, A_3, A_4\}$.

We conclude this section with a discussion of different equivalence notions. Two LPODs, P and Q, are said to be (*ordinarily*) *equivalent*, denoted $P \equiv Q$, iff $AS(P) = AS(Q)$, and *σ-equivalent*, denoted $P \equiv^\sigma Q$, iff $AS^\sigma(P) = AS^\sigma(Q)$, for any preference σ. Two LPs P_1 and P_2 are *strongly equivalent* (Lifschitz, Pearce, & Valverde 2001), denoted $P_1 \equiv_s P_2$, iff, for any LP P, $AS(P_1 \cup P) = AS(P_2 \cup P)$. Following Turner (2003), strong equivalence between LPs can be characterised as follows: Let P be an LP (over U) and X, Y sets of atoms such that $X \subseteq Y \subseteq U$. The pair (X, Y) is an *SE-model* (over U) of P if $Y \models P$ and $X \models P^Y$. By $SE(P)$ we denote the set of all SE-models of P. Then, for any LPs P_1 and P_2, $P_1 \equiv_s P_2$ iff $SE(P_1) = SE(P_2)$.

Defining a Reduct for LPODs

As noted above, we now provide a definition of a reduct which properly extends the usual one due to Gelfond & Lifschitz (1988), and which allows us to characterise answer sets of LPODs just in the same way as answer sets of LPs.

Definition 1 *Let P be an LPOD and I an interpretation. Then, P^I is given by*

$$\{head_j(r) \leftarrow body^+(r) \mid r \in P, I \cap body^-(r) = \emptyset,$$
$$I \models_j r\} \cup \{head_{\alpha(r)}(r) \leftarrow body^+(r) \mid r \in P, I \not\models r\}.$$

In other words, for a rule $r = p_1 \times \cdots \times p_k \leftarrow body(r)$, we take the positive part, $p_j \leftarrow body^+(r)$, of the j-th option of r to build the reduct P^I, in case $I \cap body^-(r) = \emptyset$ and r is satisfied to degree j by I; if r is not satisfied by I (note that in this case $I \cap body^-(r) = \emptyset$ holds as well), we take the positive part of the least option, i.e., $head_{\alpha(r)}(r) \leftarrow body^+(r)$. Therefore, all rules $r \in P$ with $I \cap body^-(r) \neq \emptyset$ are not taken into account in the construction of P^I, which is in accordance with the original concept of a program reduct. In particular, for a normal program P, our definition of a reduct coincides with the usual notion of a reduct, since, for any normal rule r, we have that $r = r[1]$, and thus $I \models r$ iff $I \models_1 r$, hence $r \in P^I$ iff $I \cap body^-(r) = \emptyset$. Thus, P^I as defined in the background for LPs, is properly generalised to LPODs by Definition 1. The difference to the reduct $P_\times^I$ as defined by Brewka, Niemelä, & Syrjänen (2004) is that rules r from P with $I \not\models r$ are not necessarily present in $P_\times^I$.

We furthermore note that the reduct as defined by Brewka, Niemelä, & Syrjänen (2004) differs from the original notion of a reduct as defined by Gelfond & Lifschitz (1988) on some normal programs, while the reduct of Definition 1 does not. For example, for the program $P = \{a \leftarrow b\}$, we have $P_\times^{\{b\}} = \emptyset$, while $P^{\{b\}} = \{a \leftarrow b\}$ according to Definition 1 and Gelfond & Lifschitz (1988). Also, observe that $\{b\} \models P_\times^{\{b\}}$ while $\{b\} \not\models P$, a scenario that cannot occur with the reduct of Definition 1. Indeed, this is a consequence of the general property stated next.

Lemma 1 *For each LPOD P and each interpretation I, $I \models P$ iff $I \models P^I$.*

Proof. ($\Rightarrow$) From $I \models P$, we have, for each $r \in P$, $I \models r$, and thus $I \models_j r$, for some degree j. Thus, for each $r \in P$,

we have either $I \cap body^-(r) \neq \emptyset$ or $I \models head_j(r) \leftarrow body^+(r)$, and $I \models P^I$ follows.

($\Leftarrow$) If $I \not\models P$, then there exists some $r \in P$ such that $body^+(r) \subseteq I$ and $I \cap (head(r) \cup body^-(r)) = \emptyset$. This implies $body^+(r) \subseteq I$ and $I \cap head(r) = \emptyset$. Hence, for any $1 \leq i \leq \alpha(r)$, $I \not\models head_i(r) \leftarrow body^+(r)$. This holds, in particular, for $i = \alpha(r)$ and since $p_{\alpha(r)} \leftarrow body^+(r)$ is contained in P^I, we get $I \not\models P^I$. □

Lemma 2 *Let P be an LPOD, $S \in SP(P)$, and I an interpretation. Then, $I \models S$ implies $I \models P$.*

Proof. Suppose $I \not\models P$. Hence, there exists a rule $r \in P$ such that $body^+(r) \subseteq I$ and $I \cap (head(r) \cup body^-(r)) = \emptyset$. Since $S \in SP(P)$, the j-th option of r, $r[j]$, is contained in S, for some j. Since $body^+(r[j]) = body^+(r)$, and $(head(r[j]) \cup body^-(r[j])) \subseteq (head(r) \cup body^-(r))$, we get $body^+(r[j]) \subseteq I$ and $I \cap (head(r[j]) \cup body^-(r[j])) = \emptyset$, thus $I \not\models S$. □

Now we can use our notion of reduct to characterise the standard answers of an LPOD.

Theorem 3 *Let P be an LPOD and I an interpretation. Then, $I \in AS(P)$ iff $I = Cn(P^I)$.*

Proof. ($\Leftarrow$) Assume $I = Cn(P^I)$, and consider the program S which contains, for each rule $r \in P$ the j-th option, $r[j]$, of r if $I \models_j r$ for some $1 \leq j \leq k$, and the $\alpha(r)$-th option of r otherwise. By construction, $S \in SP(P)$ and $P^I = S^I$. Since $I = Cn(P^I)$ by hypothesis, we get $I = Cn(S^I)$, and thus $I \in AS(P)$.

($\Rightarrow$) From $I \in AS(P)$ we get that there exists a split program $S \in SP(P)$ such that $I = Cn(S^I)$. We show that $I = Cn(P^I)$. From $I \models S^I$ (and since S is an LP) we know $I \models S$. By Lemma 2, $I \models P$, and thus, by Lemma 1, we get $I \models P^I$. It remains to show that for each $J \subset I$, $J \not\models P^I$. So, fix some $J \subset I$. We know $J \not\models S^I$, i.e. $J \not\models \{r[j]\}^I$, for some j-th option of a rule $r \in P$. From $J \not\models \{r[j]\}^I$, we get $I \cap body^-(r) = \emptyset$, $I \cap \{head_1(r), \ldots, head_{j-1}(r)\} = \emptyset$, $body^+(r) \subseteq J$, and $head_j(r) \notin J$. But $head_j(r) \in I$ has to hold, otherwise $I \not\models S$. But then, we have $body^+(r) \subseteq I$, since $J \subset I$, $I \cap body^-(r) = \emptyset$, $head_j(r) \in I$ and $I \cap \{head_1(r), \ldots, head_{j-1}(r)\} = \emptyset$. Therefore, by definition, $I \models_j r$, and since $I \cap body^-(r) = \emptyset$, we get that $head_j(r) \leftarrow body^+(r)$ is contained in P^I. Consequently, $J \not\models P^I$. □

We also mention a basic property, which one expects from a reduct, and which clearly holds by the definition which defines a reduct rule by rule.

Proposition 4 *For any LPOD P, Q, and any interpretation I, $(P \cup Q)^I = P^I \cup Q^I$.*

Non-preferential Strong Equivalence

In this section, we extend the concept of strong equivalence to LPODs by comparing their standard answer sets. As we will see later, this notion is also underlying (preferential) strong equivalence which relies on comparisons of preferred answer sets. However, already for standard strong equivalence we can, in principle, distinguish between two possible scenarios for the types of programs which are considered as context of the comparison. In fact, we distinguish between a non-preferential context, which refers to any normal program, or an arbitrary context which also includes the entire class of LPODs.

Definition 2 *Let P and Q be two LPODs. Then, P and Q are* standard strongly equivalent for non-preferential context, *symbolically $P \equiv_s Q$, iff, for any LP R, $(P \cup R) \equiv (Q \cup R)$. P and Q are* standard strongly equivalent (for arbitrary context)*, symbolically $P \equiv_{s,\times} Q$, iff, for any LPOD R, $(P \cup R) \equiv (Q \cup R)$.*

In order to characterise these strong-equivalence notions between LPODs, we define the notion of an SE-model for LPODs in the same way as done for LPs, but using our new notion of a reduct.

Definition 3 *A pair (X, Y) of interpretations with $X \subseteq Y$ is an* SE-model *of an LPOD P iff $Y \models P$ and $X \models P^Y$. The set of all SE-models of an LPOD P is denoted by $SE(P)$.*

Our generalised notion for LPODs shares some important property with the traditional notion of SE-models for LPs—in particular, the following one, which easily follows from Proposition 4 and the fact the satisfaction is defined in a standard way.

Proposition 5 *For all LPODs P, Q, $SE(P) \cap SE(Q) = SE(P \cup Q)$.*

The next result shows that the extended concept of an SE-model characterises both LPOD- and LP-strong equivalence. Thus, the latter two notions coincide.

Theorem 6 *For all LPODs P, Q, the following statements are equivalent:* (1) $P \equiv_{s,\times} Q$; (2) $P \equiv_s Q$; (3) $SE(P) = SE(Q)$.

Proof. The proof proceeds basically along the lines of the corresponding proof by Turner (Turner 2003). Recall that following Theorem 3, for any LPOD P, $I \in AS(P)$ iff $I \models P$ and no $J \subset I$ satisfies $J \models P^I$.

(1) $\Rightarrow$ (2): Follows by definition.

(2) $\Rightarrow$ (3): Suppose, without loss of generality, $(X, Y) \in SE(P) \setminus SE(Q)$.

Case 1: $X = Y$. Then, $Y \models P$ and $Y \not\models Q$. Clearly, $Y \in AS(P \cup \{y \leftarrow \mid y \in Y\})$ but $Y \notin AS(Q \cup \{y \leftarrow \mid y \in Y\})$.

Case 2: $X \subset Y$ and $(Y, Y) \in SE(P) \cap SE(Q)$. Take

$$R = \{x \leftarrow \mid x \in X\} \cup \{p \leftarrow q \mid p, q \in Y \setminus X\}.$$

Then, $Y \models Q \cup R$, and, for each $Z \subset Y$ with $X \neq Z$, $Z \not\models R^Y = R$. Since $X \not\models Q^Y$, by hypothesis $(X, Y) \notin SE(Q)$, we obtain that no $U \subset Y$ satisfies $U \models (Q \cup R)^Y$. Consequently, $Y \in AS(Q \cup R)$. On the other hand, $X \models P^Y$ by hypothesis, and $X \models R^Y$ is easily checked, since $R = R^Y$. But then, $X \models P^Y \cup R^Y = (P \cup R)^Y$. Hence,

$Y \notin AS(P \cup R)$. In both cases we used an LP R to show $AS(P \cup R) \neq AS(Q \cup R)$, hence $P \not\equiv_s Q$.
(3) $\Rightarrow$ (1). Suppose there exists an LPOD R such that $AS(P \cup R) \neq AS(Q \cup R)$. Without loss of generality, assume that $Y \in AS(P \cup R) \setminus AS(Q \cup R)$. We get $Y \models P$ and $Y \models R$, and thus have two cases for $Y \notin AS(Q \cup R)$. First, $Y \not\models Q$. We immediately get $(Y, Y) \notin SE(Q)$ and are done, since $(Y, Y) \in SE(P)$ holds in view of $Y \models P$. So suppose $Y \models Q$ but some $X \subset Y$ satisfies $(Q \cup R)^Y$. Then, $X \models Q^Y$ and we obtain $(X, Y) \in SE(Q)$. On the other hand, since $X \models R^Y$, we have $X \not\models P^Y$, otherwise $X \models P^Y \cup R^Y = (P \cup R)^Y$, which contradicts the assumption $Y \in AS(P \cup R)$. From $X \not\models P^Y$, we get $(X, Y) \notin SE(P)$. $\square$

By definition, it is clear that standard strong equivalence implies standard equivalence between LPODs. The example given next shows that such an implication does, in general, not hold for σ-equivalence, where $\sigma \in \{p, i, c\}$.

Example 2 *Consider the programs*

$$\begin{aligned} P &= \{c \times a \times b;\ c \leftarrow a, b;\ d \leftarrow c, not\ d\}, \\ Q &= \{c \times b \times a;\ c \leftarrow a, b;\ d \leftarrow c, not\ d\}. \end{aligned}$$

We compute the SE-models of these two programs. To this end, let us first establish the interpretations (over $\{a, b, c, d\}$) satisfying P, and resp., Q. Recall that the notion of satisfaction does not take care of the actual order of atoms in rule heads, and thus it is quite obvious that P and Q are satisfied by the same interpretations, viz. $\{a\}$, $\{b\}$, $\{a, d\}$, $\{b, d\}$, $\{a, c, d\}$, $\{b, c, d\}$, $\{c, d\}$, $\{a, b, c, d\}$. To see that also the (non-total) SE-models of P and Q coincide we show that $P^I = Q^I$ holds for each model I from above. In fact, there is only one rule which differs in P and Q, so we have to check $\{c \times a \times b\}^I = \{c \times b \times a\}^I$, for each such I. By definition of the reduct $\{c \times a \times b\}^I \neq \{c \times b \times a\}^I$ is only possible for interpretations I containing $\{a, b\}$ but not c. However no such interpretation satisfies the two programs. Thus, we have shown $P \equiv_s Q$.

Moreover, $AS(P) = AS(Q) = \{\{a\}, \{b\}\}$. But for each preference $\sigma \in \{p, i, c\}$, the respective preference relations include $\{a\} >_P^\sigma \{b\}$ and $\{b\} >_Q^\sigma \{a\}$, yielding $AS^p(P) = AS^i(P) = AS^c(P) = \{\{a\}\}$ and $AS^p(Q) = AS^i(Q) = AS^c(Q) = \{\{b\}\}$. Hence, $P \not\equiv^\sigma Q$, for $\sigma \in \{p, i, c\}$.

σ-Strong Equivalence for LP Context

We next consider strong equivalence for preferred answer sets where the context is restricted to normal programs (LPs). For this case, we provide a general characterisation which applies to any preference relations satisfying some basic criteria. As our three example types of preferences, p, i, and c, satisfy these criteria, we thus obtain concrete characterisations for these kinds of equivalence.

Definition 4 *Let P and Q be LPODs and σ any preference schema. Then, P and Q are* σ-strongly equivalent for LP contexts, *in symbols $P \equiv_s^\sigma Q$, iff, for each LP R, $(P \cup R) \equiv^\sigma (Q \cup R)$.*

Recall that $(P \cup R) \equiv^\sigma (Q \cup R)$ denotes ordinary equivalence between preferred answer sets, i.e., $AS^\sigma(P \cup R) = AS^\sigma(Q \cup R)$.

Before turning to our characterisations, we need a technical lemma.

Lemma 7 *Let Y, Z be models of an LPOD P over atoms V and let*

$$\begin{aligned} R^\dagger_{Y,Z} = \ & \{a \leftarrow not\ b;\quad b \leftarrow not\ a\} \cup \\ & \{y \leftarrow a \mid y \in Y\} \cup \{z \leftarrow b \mid z \in Z\} \cup \\ & \{w \leftarrow a, y', not\ w \mid y' \in V \setminus Y\} \cup \\ & \{w \leftarrow b, z', not\ w \mid z' \in V \setminus Z\}, \end{aligned}$$

where a, b, w are new atoms. Then, $AS(P \cup R^\dagger_{Y,Z}) = \{Y \cup \{a\}, Z \cup \{b\}\}$.

The proof is straightforward. Note that the constraint-like rules using w are required since the answer sets of LPODs do not necessarily satisfy the anti-chain property.

The basic property underlying our characterisation is the following:

Definition 5 *A preference σ is* LP-invariant *iff, for each LPOD P, each LP R, and each model Y, Z of $P \cup R$, it holds that $Y >_P^\sigma Z$ iff $Y >_{P \cup R}^\sigma Z$.*

We are now prepared to characterise σ-strong equivalence for LP contexts for any preference σ which is LP-invariant. Note that in the forthcoming result the pairs $\langle SE(P), >_P^\sigma \rangle$ are assigned to a single program, and thus do not depend on the context of a particular comparison.

Theorem 8 *For all LPODs P and Q, if σ is an LP invariant preference, then $P \equiv_s^\sigma Q$ iff $\langle SE(P), >_P^\sigma \rangle = \langle SE(Q), >_Q^\sigma \rangle$.*

Proof. ($\Rightarrow$) First, suppose $SE(P) \neq SE(Q)$. By Theorem 6, we know $P \not\equiv_s Q$. Hence, there exists an LP R such that, without loss of generality, $I \in AS(P \cup R) \setminus AS(Q \cup R)$. Let $U = atoms(P \cup Q \cup R)$ and consider the program

$$R' = R \cup \{w \leftarrow I \cup \{not\ a \mid a \in U \setminus I\}\} \cup \{w \leftarrow not\ w\},$$

where $w \notin U$. Then, $I \cup \{w\}$ is the only answer set of $P \cup R'$ and thus also σ-preferred, while $Q \cup R'$ possesses no answer set and thus, in particular, no σ-preferred answer-set. But then, since R' is an LP, $P \not\equiv_s^\sigma Q$.

Second, suppose $SE(P) = SE(Q)$ and $>_P^\sigma \neq >_Q^\sigma$. Since $SE(P) = SE(Q)$, the classical models of P and Q have to coincide, and thus $>_P^\sigma \neq >_Q^\sigma$ yields that there exist models Y, Z of both P and Q such that, without loss of generality, $Z >_P^\sigma Y$ and $Z \not>_Q^\sigma Y$. Moreover, let $V = atoms(P \cup Q)$. By Lemma 7,

$$\{Y \cup \{a\}, Z \cup \{b\}\} = AS(P \cup R^\dagger_{Y,Z}) = AS(Q \cup R^\dagger_{Y,Z}).$$

Now, since $Z >_P^\sigma Y$ and $a, b \notin P$, $Z \cup \{b\} >_P^\sigma Y \cup \{a\}$ is easily seen. Since σ is LP-invariant, and both $Z \cup \{b\}$ and $Y \cup \{a\}$ are models of $P \cup R^\dagger_{Y,Z}$, we obtain $Z \cup \{b\} >^\sigma_{P \cup R^\dagger_{Y,Z}} Y \cup \{a\}$. This shows $Y \cup \{a\} \notin AS^\sigma(P \cup R^\dagger_{Y,Z})$. By similar arguments, one can show that $Y \cup \{a\} \in AS^\sigma(Q \cup R^\dagger_{Y,Z})$. Since $R^\dagger_{Y,Z}$ is an LP, we have $P \not\equiv_s^\sigma Q$.

($\Leftarrow$) Assume $P \not\equiv_s^\sigma Q$ and, without loss of generality, $Y \in AS^\sigma(P \cup R) \setminus AS^\sigma(Q \cup R)$ for some LP R. Clearly, $Y \in AS(P \cup R)$ then holds, so in case $Y \notin AS(Q \cup R)$ we are done, since then, by definition, $P \not\equiv_s Q$, and consequently $SE(P) \neq SE(Q)$, in view of Theorem 6. So assume $Y \in AS(Q \cup R)$. Then, there exists a $Z >^\sigma_{Q \cup R} Y$ such that $Z \in AS(Q \cup R)$. If $Z \notin AS(P \cup R)$ we are again done since this yields $P \not\equiv_s Q$, i.e., $SE(P) \neq SE(Q)$. Hence, Y and Z are answer sets of both $P \cup R$ and $Q \cup R$. Thus, Y and Z have to satisfy both P and Q, as well as R. Hence, we have that (i) $Z \not>^\sigma_{P \cup R} Y$ (as $Y \in AS^\sigma(P \cup R)$) and (ii) $Z >^\sigma_{Q \cup R} Y$ (as $Y \notin AS^\sigma(Q \cup R)$). Since σ is LP-invariant, we obtain $Z \not>^\sigma_P Y$ while $Z >^\sigma_Q Y$. This shows $>^\sigma_P \neq >^\sigma_Q$. □

Concerning our concrete preference relations, we first show that they are indeed LP-invariant.

Lemma 9 *Preferences p, i, c, are all LP-invariant.*

Proof. We have to show that for each LPOD P, each LP R, and each model Y, Z of $P \cup R$, $Y >^\sigma_P Z$ iff $Y >^\sigma_{P \cup R} Z$, for $\sigma \in \{p, i, c\}$.

Since R is an LP, for every $r \in R$, we obtain $d_Y(r) = d_Z(r) = 1$ and $R_Y[1] = R_Z[1]$. Moreover, for any $k > 1$, it holds that $R_Y[k] = R_Z[k] = 0$, and hence $(P \cup R)_Y[k] = P_Y[k]$ and $(P \cup R)_Z[k] = P_Z[k]$. It follows that $|(P \cup R)_Y[1]| = |P_Y[1]| + |R_Y[1]|$ and $|(P \cup R)_Z[1]| = |P_Z[1]| + |R_Z[1]|$, and since $R_Y[1] = R_Z[1]$,

$$|(P \cup R)_Y[1]| > |(P \cup R)_Z[1]| \text{ iff } |P_Y[1]| > |P_Z[1]|$$

and

$$|(P \cup R)_Y[1]| = |(P \cup R)_Z[1]| \text{ iff } |P_Y[1]| = |P_Z[1]|.$$

Consequently, $Y >^c_P Z$ iff $Y >^c_{P \cup R} Z$.

Since $R_Y[1] = R_Z[1]$, also $(P \cup R)_Y[1] = P_Y[1] \cup R_Y[1] \supset (P \cup R)_Z[1] = P_Z[1] \cup R_Z[1]$ iff $P_Y[1] \supset P_Z[1]$, and $(P \cup R)_Y[1] = (P \cup R)_Z[1]$ iff $P_Y[1] = P_Z[1]$. Consequently, $Y >^i_P Z$ iff $Y >^i_{P \cup R} Z$.

Since, for any $r \in R$, $d_Y(r) = d_Z(r)$ if there is some rule $r' \in P \cup R$ such that $d_Y(r') < d_Z(r')$, we obtain $r' \in P$. Moreover, for no rule $r'' \in R$, $d_Y(r'') > d_Z(r'')$ can hold. Consequently, $Y >^p_P Z$ iff $Y >^p_{P \cup R} Z$. □

We thus obtain the following characterisation:

Theorem 10 *For all LPODs P, Q,*

- $P \equiv^p_s Q$ *iff* $\langle SE(P), >^p_P \rangle = \langle SE(Q), >^p_Q \rangle$,
- $P \equiv^i_s Q$ *iff* $\langle SE(P), >^i_P \rangle = \langle SE(Q), >^i_Q \rangle$,
- $P \equiv^c_s Q$ *iff* $\langle SE(P), >^c_P \rangle = \langle SE(Q), >^c_Q \rangle$.

Note that this allows also for even more flexible definitions of strong equivalence notions as follows: For instance, given two LP-invariant preferences π and π', we can ask whether, for each LP R, $AS^\pi(P \cup R) = AS^{\pi'}(Q \cup R)$ holds. A possible application would be if one wants to apply a program to a different preference relation without changing the meaning under any new information (which itself does not provide new preferential information).

σ-Strong Equivalence for Arbitrary Context

We next consider the case where LPODs can be possible context programs. Hereby, it is not only necessary that $>^\sigma$ agrees on all pairs of models, but also that all pairs of models that can become comparable with respect to $>^\sigma$ agree when adding an appropriate LPOD. It is worthwhile noting that this additional requirement applies to pairs of models that are incomparable with respect to $>^\sigma$. It turns out that this requirement gives rise to quite different conditions for different σ.

Definition 6 *Let P and Q be two LPODs. Then, P and Q are* σ-strongly equivalent for arbitrary contexts, *symbolically $P \equiv^\sigma_{s,\times} Q$, iff, for any LPOD R, $(P \cup R) \equiv^\sigma (Q \cup R)$.*

The following result is straightforward.

Theorem 11 *For any preference σ and LPODs P and Q, $P \equiv^\sigma_{s,\times} Q$ implies $P \equiv^\sigma_s Q$.*

However, for each of the preference schemata $\sigma \in \{p, i, c\}$, the converse of the above theorem does not hold. But, as follows from our results below, adding a further condition does ensure that $P \equiv^\sigma_s Q$ implies $P \equiv^\sigma_{s,\times} Q$. In the sequel, when referring to σ-strong equivalence, we mean $\equiv^\sigma_{s,\times}$.

Pareto Preferred Strong Equivalence For the comparison of Pareto-preferred answer sets, we note that the only way in which two uncomparable models of a program can be made answer sets and comparable by addition of an LPOD is when they satisfy all rules to the same degree. To this end, we define a relation identifying pairs of models that have this property.

Definition 7 *For any program P and interpretations Y, Z over some $U \supseteq atoms(P)$, $Y =^p_P Z$ iff Y and Z are models of P and for all $r \in P$, $d_Y(r) = d_Z(r)$.*

Note that $=^p_P \subseteq 2^U \times 2^U$, just like $>^p_P$. We then obtain the following characterisation:

As we show below $\equiv^p_{s,\times}$ is then fully characterised by comparing triples of the form $\langle SE(P), >^p_P, =^p_P \rangle$.

Lemma 12 *For all LPODs P and Q, if $P \equiv^p_{s,\times} Q$, then $\langle SE(P), >^p_P, =^p_P \rangle = \langle SE(Q), >^p_Q, =^p_Q \rangle$.*

Proof. First observe that due to Theorem 10, if either $SE(P) \neq SE(Q)$ or $>^p_P \neq >^p_Q$, then $P \not\equiv^p_s Q$. This means that there exists an LP R such that $AS^p(P \cup R) \neq AS^p(Q \cup R)$, and therefore also $P \not\equiv^p_{s,\times} Q$.

Now assume that $SE(P) = SE(Q)$ and $>^p_P = >^p_Q$, but that $=^p_P$ and $=^p_Q$ differ. That is, without loss of generality, for two models Y, Z, $Y =^p_P Z$ but $Y \neq^p_Q Z$. So, for each $r \in P$, $d_Y(r) = d_Z(r)$, but there exists an $r' \in Q$ such that $d_Y(r') \neq d_Z(r')$, and without loss of generality, let $d_Y(r') > d_Z(r')$. Consider $R = R^\dagger_{Y,Z} \cup \{a \times b\}$, with $R^\dagger_{Y,Z}$ as in Lemma 7. Then, $Y' = Y \cup \{a\}$ and $Z' = Z \cup \{b\}$ are the only standard answer sets of $P \cup R$ and $Q \cup R$. Since $d_{Y'}(a \times b) < d_{Z'}(a \times b)$ and $d_{Y'}(r) = d_{Z'}(r)$, for each $r \in P \cup R$, we obtain $Y' >^p_{P \cup R} Z'$ and thus $AS^p(P \cup R) = \{Y'\}$. On the other hand, since $d_Y(r') > d_Z(r')$ holds, we obtain $Y' \not>^p_{Q \cup R} Z'$

and $Z' \not>^p_{Q\cup R} Y'$. Therefore, $AS^p(Q \cup R) = \{Y', Z'\}$ and hence $P \not\equiv^p_{s,\times} Q$. □

Lemma 13 *For all LPODs P, Q, if $\langle SE(P), >^p_P, =^p_P\rangle = \langle SE(Q), >^p_Q, =^p_Q\rangle$, then $P \equiv^p_{s,\times} Q$.*

Proof. Assume $\langle SE(P), >^p_P, =^p_P\rangle = \langle SE(Q), >^p_Q, =^p_Q\rangle$, and let R be an arbitrary LPOD. Since $SE(P) = SE(Q)$, by Theorem 6 we know that $AS(P \cup R) = AS(Q \cup R)$. In the following, consider arbitrary $X, Y \in AS(P \cup R)$.

If $X >^p_{P\cup R} Y$ holds, there is an $r_0 \in P \cup R$ such that $d_X(r_0) < d_Y(r_0)$ and for all $r_1 \in P \cup R$ it holds that $d_X(r_1) \leq d_Y(r_1)$. If there is an $r_2 \in P$ such that $d_X(r_2) < d_Y(r_2)$, then $X >^p_P Y$, and since $>^p_P$ is equal to $>^p_Q$, also $X >^p_Q Y$ and $X >^p_{Q\cup R} Y$. If there is no such r_2, then $d_X(r_3) = d_Y(r_3)$, for each $r_3 \in P$, and hence $X =^p_P Y$, and moreover $r_0 \in R$. Since $=^p_P$ is equal to $=^p_Q$, we obtain $X >^p_{Q\cup R} Y$ also in this case.

Using a symmetric argument, we can show that $X >^p_{Q\cup R} Y$ implies $X >^p_{P\cup R} Y$, and in total $X >^p_{P\cup R} Y$ iff $X >^p_{Q\cup R} Y$. It follows that $AS^p(P\cup R) = AS^p(Q\cup R)$, and since R is arbitrary, $P \equiv^p_{s,\times} Q$. □

Lemmata 12 and 13 provide us with the following characterisation:

Theorem 14 *For all LPODs P and Q, $P \equiv^p_{s,\times} Q$ iff $\langle SE(P), >^p_P, =^p_P\rangle = \langle SE(Q), >^p_Q, =^p_Q\rangle$.*

Inclusion Preferred Strong Equivalence Different to the situation of Pareto preference, adding rules under inclusion preference may invalidate the relationship between two models in one program, but not in the other one. Indeed, this may happen if the inclusion relation holds at a different degree in the two programs. More specifically, there may be an incomparability on different satisfaction degrees for two models on the two programs or an incomparability on some satisfaction degree for two models on one program, while there is no incomparability for the other program. A unified criterion for these two situations is comparing the degree on which two models are incomparable, and setting this degree to a special value if no such degree exists. In the following definition, we choose the value 0 as this special value, as satisfaction degrees are always greater than 0.

Definition 8 *For each LPOD P, let $\delta_P : 2^U \times 2^U \to \mathbb{N}$ be the partial function defined on the models of P such that for models Y and Z*

$$\delta_P(Y,Z) = \begin{cases} k & \text{if } P_Y[k] \not\subseteq P_Z[k] \text{ and} \\ & \forall j < k : P_Y[j] = P_Z[j]; \\ 0 & \text{otherwise.} \end{cases}$$

We first observe that if these partial functions coincide for two programs, then also the respective preference relations are equal.

Theorem 15 *For all LPODs P and Q having the same models, if $\delta_P = \delta_Q$ then $>^i_P$ is equal to $>^i_Q$.*

Proof. Assume without loss of generality that there are two models Y, Z of P and Q such that $Y >^i_P Z$ but $Y \not>^i_Q Z$. Then, there is some k such that $P_Y[k] \supset P_Z[k]$ and for all $j < k$, $P_Y[j] = P_Z[j]$, while either (a) for all $i > 0$, $Q_Y[i] = Q_Z[i]$ or (b) there is some h such that $Q_Y[h] \subset Q_Z[h]$ and $Q_Y[g] = Q_Z[g]$, for all $g < h$.

We observe that $\delta_P(Y,Z) = k > 0$ while in case (a), $\delta_Q(Y,Z) = 0$ holds, and in case (b), $\delta_Q(Y,Z) = 0$ also holds, since $0 < \delta_Q(Y,Z) < h$ cannot hold as the respective sets are equal, $\delta_Q(Y,Z) = h$ cannot hold because $Q_Y[h] \subset Q_Z[h]$, and $\delta_Q(Y,Z) > h$ cannot hold either as $Q_Y[h] \neq Q_Z[h]$. Therefore, in either case $\delta_P(Y,Z) \neq \delta_Q(Y,Z)$. □

Towards our characterisation of $\equiv^i_{s,\times}$, we note the following ancillary result:

Corollary 16 *Given LPODs P and Q, if $\langle SE(P), \delta_P\rangle = \langle SE(P), \delta_Q\rangle$, then $P \equiv^i_s Q$.*

Proof. By Theorem 15 and Theorem 10. □

As we show below $\equiv^i_{s,\times}$ is then fully characterised by comparing pairs of the form $\langle SE(P), \delta_P\rangle$.

Lemma 17 *For all LPODs P and Q, if $P \equiv^i_{s,\times} Q$, then $\langle SE(P), \delta_P\rangle = \langle SE(Q), \delta_Q\rangle$.*

Proof. Suppose $\langle SE(P), \delta_P\rangle \neq \langle SE(Q), \delta_Q\rangle$. If $SE(P) \neq SE(Q)$ by Theorem 10 $P \not\equiv^i_s Q$. This means that there exists an LP R such that $AS^i(P \cup R) \neq AS^i(Q \cup R)$, and therefore also $P \not\equiv^i_{s,\times} Q$.

So let us examine the situation if $SE(P) = SE(Q)$ and (again without loss of generality) for two models Y and Z of P and Q, $k = \delta_P(Y,Z) < \delta_Q(Y,Z) = \ell$.

If $k = 0$ we differentiate two cases: (i) $\forall j > 0 : P_Y[j] = P_Z[j]$ and (ii) $\exists i : P_Y[i] \subset P_Z[i]$ and $\forall h < i : P_Y[h] = P_Z[h]$. Since $\ell = \delta_Q(Y,Z)$ we have either (a) $Q_Y[\ell] \supset Q_Z[\ell]$ or (b) both $Q_Y[\ell] \not\subseteq Q_Z[\ell]$ and $Q_Y[\ell] \not\supseteq Q_Z[\ell]$. So if (i) and (a) then $Y \not>^i_P Z$ and $Y >^i_Q Z$, if (ii) and (a) then $Z >^i_P Y$ and $Y >^i_Q Z$, and if (ii) and (b) then $Z >^i_P Y$ and $Z \not>^i_Q Y$. In these cases we obtain $P \not\equiv^i_s Q$ by Theorem 10 and hence (as discussed above) $P \not\equiv^i_{s,\times} Q$.

If (i) and (b) then $Y \not>^i_P Z$, $Z \not>^i_P Y$ and $Y \not>^i_Q Z$, $Z \not>^i_Q Y$ but we construct an LPOD $R^* = R^\dagger_{Y,Z} \cup R^*_\ell$ where

$$R^*_\ell = \{v \leftarrow c_i, \mathit{not}\ v \mid 1 \leq i < \ell\} \cup \{c_1 \times \cdots \times c_{\ell-1} \times a \times b\}$$

where $c_1, \ldots, c_{\ell-1}, v$ are symbols that do not occur in P or Q. Using Lemma 7 we obtain that $AS(P \cup R^*) = AS(Q \cup R^*) = \{Y', Z'\}$, where $Y' = Y \cup \{a\}$ and $Z' = Z \cup \{b\}$.

Now for all degrees $g < \ell$, both $(P \cup R^*)_{Y'}[g] = (P \cup R^*)_{Z'}[g]$ and $(Q \cup R^*)_{Y'}[j] = (Q \cup R^*)_{Z'}[j]$ hold, whereas $(P \cup R^*)_{Y'}[\ell] \supset (P \cup R^*)_{Z'}[\ell]$ hence $Y' >^i_{P\cup R^*} Z'$, but $(Q \cup R^*)_{Y'}[\ell] \not\subseteq (Q \cup R^*)_{Z'}[\ell]$ and $(Q \cup R^*)_{Y'}[\ell] \not\supseteq (Q \cup R^*)_{Z'}[\ell]$ hence neither $Y' >^i_{Q\cup R^*} Z'$ nor $Z' >^i_{Q\cup R^*} Y'$. Therefore $AS^i(P \cup R^*) = \{Y'\} \neq AS^i(P \cup R^*) = \{Y', Z'\}$ and $P \not\equiv^i_{s,\times} Q$.

For $k > 0$, we construct an LPOD $R = R^{\dagger}_{Y,Z} \cup R^k$, similar to R^*, where

$$R^k = \{v \leftarrow c_i, not\ v \mid 1 \leq i < k\} \cup \\ \{r^+ : c_1 \times \cdots \times c_{k-1} \times b \times a \mid P_Y[k] \supset P_Z[k]\} \cup \\ \{r^- : c_1 \times \cdots \times c_{k-1} \times a \times b \mid P_Y[k] \not\supset P_Z[k]\},$$

with $c_1, \ldots, c_{k-1}, v$ being symbols that do not occur in P or Q, and $R^{\dagger}_{Y,Z}$ as in Lemma 7. Again, we first observe that $AS(P \cup R) = AS(Q \cup R) = \{Y', Z'\}$, where $Y' = Y \cup \{a\}$ and $Z' = Z \cup \{b\}$.

Suppose that $P_Y[k] \supset P_Z[k]$. Then, $d_{Y'}(r^+) = k + 1$ and $d_{Z'}(r^+) = k$, therefore neither $(P \cup R)_{Y'}[k] \subseteq (P \cup R)_{Z'}[k]$ nor $(P \cup R)_{Y'}[k] \supseteq (P \cup R)_{Z'}[k]$. Note also that $(P \cup R)_{Y'}[j] = (P \cup R)_{Z'}[j]$ for all $j < k$, so neither $Y' >^i_{P \cup R} Z'$ nor $Z' >^i_{P \cup R} Y'$ holds. If $P_Y[k] \not\supset P_Z[k]$, then since $\delta_P(Y, Z) = k$ also $P_Y[k] \not\subseteq P_Z[k]$. It follows that $(P \cup R)'_Y[k] \not\supset (P \cup R)'_Z[k]$ and $(P \cup R)'_Y[k] \not\subseteq (P \cup R)'_Z[k]$. Hence also in this case neither $Y' >^i_{P \cup R} Z'$ nor $Z' >^i_{P \cup R} Y'$ holds. In any case we therefore obtain $AS^i(P \cup R) = \{Y', Z'\}$.

Now recall that by hypothesis $\delta_Q(Y, Z) > k$, thus $Q_Y[k] = Q_Z[k]$ but now due to r^+, $(Q \cup R)_{Z'}[k] \supset (Q \cup R)_{Y'}[k]$, i.e., we have $Z' >^i_{Q \cup R} Y'$. Hence, $AS^i(Q \cup R) = \{Z'\}$, and thus $P \not\equiv^i_{s,\times} Q$. □

Lemma 18 *For all LPODs P and Q, if $\langle SE(P), \delta_P \rangle = \langle SE(Q), \delta_Q \rangle$ then $P \equiv^i_{s,\times} Q$.*

Proof. By Theorem 15 we know that $>^i_P$ is equal to $>^i_Q$ and by Corollary 16 we know $P \equiv^i_s Q$.

Let R be an arbitrary LPOD. By Theorem 6 we know $P \equiv_s Q$, and thus $AS(P \cup R) = AS(Q \cup R)$. Now consider arbitrary $X, Y \in AS(P \cup R)$. We show that $X >^i_{P \cup R} Y$ iff $X >^i_{Q \cup R} Y$, from which $P \equiv^i_{s,\times} Q$ then follows.

Assume $X >^i_{P \cup R} Y$. Hence, there exists a k such that $(P \cup R)_X[k] \supset (P \cup R)_Y[k]$ and, for each $j < k$, $(P \cup R)_X[j] = (P \cup R)_Y[j]$, i.e., $P_X[j] = P_Y[j]$ and $R_X[j] = R_Y[j]$.

First, suppose $P_X[k] \supset P_Y[k]$ and $R_X[k] \supseteq R_Y[k]$. This means that $X >^i_P Y$ and thus $X >^i_Q Y$. We also have $\delta_P(X, Y) = k$. Since $\delta_P = \delta_Q$, also $\delta_Q(X, Y) = k$ and therefore either $Q_X[k] \supset Q_Y[k]$ or $Q_X[k] \not\subseteq Q_Y[k]$ and $Q_X[k] \not\subseteq Q_Y[k]$. The latter, however, would contradict $X >^i_Q Y$, and so we know $Q_X[k] \supset Q_Y[k]$ and hence $X >^i_{Q \cup R} Y$.

Second, suppose $R_X[k] \supset R_Y[k]$ and $P_X[k] = P_Y[k]$. Hence, $\delta_P(X, Y) > k$ or $\delta_P(X, Y) = 0$. So also $\delta_Q(X, Y) > k$ or $\delta_Q(X, Y) = 0$, which implies that for each $j \leq k$, $Q_X[j] = Q_Y[j]$ (while $Q_X[j] \subset Q_Y[j]$ could hold in case of $\delta_Q(X, Y) = 0$, that means $Y >^i_Q X$ and by Theorem 15 $Y >^i_P X$ and hence also $X \not>^i_{P \cup R} Y$ which would contradict the initial assumption $X >^i_{P \cup R} Y$). It follows that $(Q \cup R)_X[k] \supset (Q \cup R)_Y[k]$, as well as $(Q \cup R)_X[j] = (Q \cup R)_Y[j]$, for each $j < k$. Hence, $X >^i_{Q \cup R} Y$.

Symmetrically, we can show that $X >^i_{Q \cup R} Y$ implies $X >^i_{P \cup R} Y$. □

Lemmata 17 and 18 provide us with the following characterisation.

Theorem 19 *For all LPODs P and Q, $P \equiv^i_{s,\times} Q$ iff $\langle SE(P), \delta_P \rangle = \langle SE(Q), \delta_Q \rangle$.*

Cardinality Preferred Strong Equivalence The remaining equivalence notion to consider is $\equiv^c_{s,\times}$. Here, the fact that the *number* of rules appearing in the context program is of relevance for making an interpretation preferred over another one, makes things more involving and cumbersome. We therefore omit proofs here. In any case, we can make use of similar concepts as before.

Definition 9 *For each LPOD P, let $\Delta_P : 2^U \times 2^U \times \mathbb{N} \to \mathbb{Z}$ be the partial function defined on the models of P such that for models Y, Z and $n \in \mathbb{N}$,*

$$\Delta_P(Y, Z, n) = |P_Y[n]| - |P_Z[n]|.$$

Different to the δ functions for inclusion preference, the condition $\Delta_P = \Delta_Q$ does not imply $>^c_P = >^c_Q$ (for LPODs P and Q). Our characterisation therefore also needs to include a comparison of these two relations.

Theorem 20 *For all LPODs P and Q, we have that $P \equiv^c_{s,\times} Q$ iff $\langle SE(P), >_P, \Delta_P \rangle = \langle SE(Q), >_Q, \Delta_Q \rangle$.*

Some Properties

We now discuss some properties of the introduced equivalence relations. First, we give a full picture on the relationship between the different concepts. Afterwards, we analyse the computational complexity of checking program equivalence.

Relationships We already know from Theorem 6 that $\equiv_s$ and $\equiv_{s,\times}$ coincide. Moreover, as a consequence of Theorems 6, 10, and 11, we obtain the following result:

Theorem 21 *For every $\sigma \in \{p, i, c\}$ and every LPOD P, Q, $P \equiv^\sigma_{s,\times} Q$ implies $P \equiv_{s,\times} Q$ (or, equivalently, $P \equiv_s Q$).*

From Theorem 10 we also know that $P \equiv^\sigma_s Q$ implies $P \equiv_s Q$, for each $\sigma \in \{p, i, c\}$. The converse, however, does not hold. In fact, Example 2 shows that LP-strong equivalence $\equiv_s$ between LPODs does not even imply (ordinary) σ-equivalence $\equiv^\sigma$. Moreover, for any preference $\sigma \in \{p, i, c\}$, we have that $P \equiv^\sigma_{s,\times} Q$ implies $P \equiv^\sigma_s Q$, but, as already noted above, in neither case the converse holds. Finally, if we compare only LPs, not surprisingly all strong-equivalence notions introduced in our work collapse to standard strong equivalence between normal logic programs, although preference information may be added in the context. This indeed shows that each notion is a generalisation of standard strong equivalence as introduced by Lifschitz, Pearce and Valverde (Lifschitz, Pearce, & Valverde 2001).

Proposition 22 *For all normal programs P, Q and every $\sigma \in \{p, i, c\}$, the following statements are equivalent:* (1) $P \equiv_s Q$, (2) $P \equiv_{s,\times} Q$, (3) $P \equiv^\sigma_s Q$, (4) $P \equiv^\sigma_{s,\times} Q$.

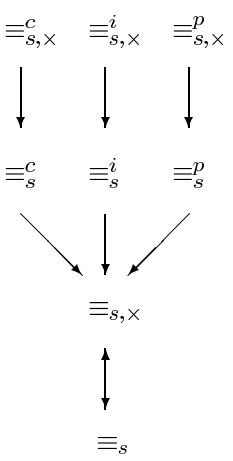

Figure 1: Relationships between equivalence notions.

As for the remaining relationships, Figure 1 provides a complete picture concerning the different equivalence notions. Implications between relations hold precisely in case there is an arc in the transitive closure of the graph in Figure 1. For illustration, we first give an example showing that $P \equiv^i_{s,\times} Q$ does not imply $P \equiv^p_s Q$, from which in turn we get that $P \equiv^i_{s,\times} Q$ does not imply $P \equiv^p_{s,\times} Q$ and $P \equiv^i_s Q$ does not imply $P \equiv^p_s Q$. Then, we also give an example that $P \equiv^c_{s,\times} Q$ does neither imply $P \equiv^p_s Q$ nor $P \equiv^i_s Q$. Counterexamples for the remaining three relations are omitted here for space reasons.

Example 3 *Consider the following two programs:*

$$\begin{aligned} P &= \{r_1 : c \times a \times b;\ a \leftarrow c;\ b \leftarrow c;\ c \leftarrow a, b\},\\ Q &= \{r_1 : c \times a \times b;\ r_2 : c \times c \times b \times a;\ a \leftarrow c;\\ &\quad\ b \leftarrow c;\ c \leftarrow a, b\}. \end{aligned}$$

Both programs have the same models, $\{a\}$, $\{b\}$, $\{a,b,c\}$, *and SE-models,* $SE(P) = SE(Q) = \{(\{a\},\{a\}), (\{b\}, \{b\}), (\{a,b,c\},\{a,b,c\})\}$. *The rule-satisfaction degrees of the models of* P *and* Q *are as follows:*

P	1	2	3
$\{a\}$	$P \setminus \{r_1\}$	$\{r_1\}$	$\emptyset$
$\{b\}$	$P \setminus \{r_1\}$	$\emptyset$	$\{r_1\}$
$\{a,b,c\}$	P	$\emptyset$	$\emptyset$

Q	1	2	3	4
$\{a\}$	$Q \setminus \{r_1, r_2\}$	$\{r_1\}$	$\emptyset$	$\{r_2\}$
$\{b\}$	$Q \setminus \{r_1, r_2\}$	$\emptyset$	$\{r_1, r_2\}$	$\emptyset$
$\{a,b,c\}$	Q	$\emptyset$	$\emptyset$	$\emptyset$

We then obtain the following inclusion relations:

$$\begin{array}{rr} \{a\} >^i_P \{b\} & \{a\} >^i_Q \{b\} \\ \{a,b,c\} >^i_P \{a\} & \{a,b,c\} >^i_Q \{a\} \\ \{a,b,c\} >^i_P \{b\} & \{a,b,c\} >^i_Q \{b\} \end{array}$$

So, $>^i_P = >^i_Q$. *Moreover, we have:*

$$\begin{array}{rr} \delta_P(\{a\},\{b\}) = 2 & \delta_Q(\{a\},\{b\}) = 2 \\ \delta_P(\{a,b,c\},\{a\}) = 1 & \delta_Q(\{a,b,c\},\{a\}) = 1 \\ \delta_P(\{a,b,c\},\{b\}) = 1 & \delta_Q(\{a,b,c\},\{b\}) = 1 \end{array}$$

From this it follows that $P \equiv^i_{s,\times} Q$ *holds. On the other hand, we obtain the following the Pareto relations.*

$$\begin{array}{rr} \{a\} >^p_P \{b\} & \{a\} \not>^p_Q \{b\} \\ \{b,d\} >^p_P \{a\} & \{b\} \not>^p_Q \{a\} \\ \{a,b,c\} >^p_P \{a\} & \{a,b,c\} >^p_Q \{a\} \\ \{a,b,c\} >^p_P \{b\} & \{a,b,c\} >^p_Q \{b\} \end{array}$$

So, $>^p_P \neq >^p_Q$, *and hence* $P \not\equiv^p_s Q$.

Example 4 *Consider the following programs:*

$$\begin{aligned} P &= \{r_1 : c \times d;\ r_2 : c \times b \times a;\ r_3 : c \times c \times d;\\ &\quad\ d \leftarrow a;\ d \leftarrow b;\ a \leftarrow c;\ b \leftarrow c;\ c \leftarrow a, b\},\\ Q &= \{r_2 : c \times b \times a;\ r_4 : c \times a \times b;\ r_5 : c \times b \times a \times d;\\ &\quad\ d \leftarrow a;\ d \leftarrow b;\ a \leftarrow c;\ b \leftarrow c;\ c \leftarrow a, b\}, \end{aligned}$$

Both programs have the same models, $\{a,d\}$, $\{b,d\}$, $\{a,b,c,d\}$, *and SE models* $SE(P) = SE(Q) = \{(\{a,d\},\{a,d\}), (\{b,d\},\{b,d\}), (\{a,b,c,d\},\{a,b,c,d\})\}$. *The rule satisfaction degrees of the models of* P *and* Q *are as follows:*

P	1	2	3
$\{a,d\}$	$P \setminus \{r_1, r_2, r_3\}$	$\{r_1\}$	$\{r_2, r_3\}$
$\{b,d\}$	$P \setminus \{r_1, r_2, r_3\}$	$\{r_1, r_2\}$	$\{r_3\}$
$\{a,b,c,d\}$	P	$\emptyset$	$\emptyset$

Q	1	2	3
$\{a,d\}$	$Q \setminus \{r_2, r_4, r_5\}$	$\{r_4\}$	$\{r_2, r_5\}$
$\{b,d\}$	$Q \setminus \{r_2, r_4, r_5\}$	$\{r_2, r_5\}$	$\{r_4\}$
$\{a,b,c,d\}$	Q	$\emptyset$	$\emptyset$

The following is easy to see.

$$\begin{array}{rr} \{b,d\} >^c_P \{a,d\} & \{b,d\} >^c_Q \{a,d\} \\ \{a,b,c,d\} >^c_P \{a,d\} & \{a,b,c,d\} >^c_Q \{a,d\} \\ \{a,b,c,d\} >^c_P \{b,d\} & \{a,b,c,d\} >^c_Q \{b,d\} \end{array}$$

So $>^c_P = >^c_Q$. *Moreover, we have:*

$$\begin{array}{r} \Delta^1_P(\{a,d\},\{b,d\}) = 0 \\ \Delta^2_P(\{a,d\},\{b,d\}) = -1 \\ \Delta^3_P(\{a,d\},\{b,d\}) = 1 \\ \Delta^1_P(\{a,b,c,d\},\{a,d\}) = 3 \\ \Delta^2_P(\{a,b,c,d\},\{a,d\}) = -1 \\ \Delta^3_P(\{a,b,c,d\},\{a,d\}) = -2 \\ \Delta^1_P(\{a,b,c,d\},\{b,d\}) = 3 \\ \Delta^2_P(\{a,b,c,d\},\{b,d\}) = -2 \\ \Delta^3_P(\{a,b,c,d\},\{b,d\}) = -1 \end{array}$$

$$\begin{array}{r} \Delta^1_Q(\{a,d\},\{b,d\}) = 0 \\ \Delta^2_Q(\{a,d\},\{b,d\}) = -1 \\ \Delta^3_Q(\{a,d\},\{b,d\}) = 1 \\ \Delta^1_Q(\{a,b,c,d\},\{a,d\}) = 3 \\ \Delta^2_Q(\{a,b,c,d\},\{a,d\}) = -1 \\ \Delta^3_Q(\{a,b,c,d\},\{a,d\}) = -2 \\ \Delta^1_Q(\{a,b,c,d\},\{b,d\}) = 3 \\ \Delta^2_Q(\{a,b,c,d\},\{b,d\}) = -2 \\ \Delta^3_Q(\{a,b,c,d\},\{b,d\}) = -1 \end{array}$$

So $P \equiv^c_{s,\times} Q$ *holds. However, we observe the following inclusion relations.*

$$\begin{array}{rl} \{b,d\} >^i_P \{a,d\} & \{a,d\} \not>^i_Q \{b,d\} \\ \{a,b,c,d\} >^i_P \{a,d\} & \{b,d\} \not>^i_Q \{a,d\} \\ \{a,b,c,d\} >^i_P \{b,d\} & \{a,b,c,d\} >^i_Q \{a,d\} \\ & \{a,b,c,d\} >^i_Q \{b,d\} \end{array}$$

Therefore, $>^i_P \neq >^i_Q$*, and as a consequence,* $P \not\equiv^i_s Q$*. Moreover, we also observe that the following Pareto relations hold.*

$$\begin{array}{rl} \{b,d\} >^p_P \{a,d\} & \{a,d\} \not>^p_Q \{b,d\} \\ \{a,b,c,d\} >^p_P \{a,d\} & \{b,d\} \not>^p_Q \{a,d\} \\ \{a,b,c,d\} >^p_P \{b,d\} & \{a,b,c,d\} >^p_Q \{a,d\} \\ & \{a,b,c,d\} >^p_Q \{b,d\} \end{array}$$

Therefore, $>^p_P \neq >^p_Q$*, and as a consequence,* $P \not\equiv^p_s Q$*.*

Computational Complexity Checking whether two LPs are strongly equivalent is well known to be co-NP-complete (Lin 2002). It turns out that this complexity bound also holds for the generalised notions studied here.

We start with some simple observations:

Lemma 23 *Given interpretations* Y*,* Z*, and an LPOD* P*, deciding any out of (i)* $Y >^\sigma_P Z$*, for* $\sigma \in \{p,i,c\}$*, (ii)* $Y =^p_P Z$*, (iii)* $\delta_P(Y,Z) = \delta_Q(Y,Z)$*, and (iv)* $\Delta_P(Y,Z,n) = \Delta_Q(Y,Z,n)$ *is feasible in polynomial time.*

Proof. For the relations $>^p_P$ and $=^p_P$ we can determine, given an interpretation Y and rule r, the degree $d_Y(r)$ in polynomial time; thus we can proceed rule by rule, and compute for each $r \in P$ its degree w.r.t. Y and Z. Simple comparisons then are sufficient to decide $Y >^p_P Z$, and likewise, $Y =^p_P Z$.

For $>^i_P$, one can check that for any j, computing $P_Y[j]$ (and likewise $P_Z[j]$) and thus deciding $P_Y[j] \supset P_Z[j]$ as well as $P_Y[j] = P_Z[j]$ can be done in polynomial time. Thus for deciding $Y >^i_P Z$, we can start with $j = 1$ and check whether $P_Y[j] \supset P_Z[j]$ or $P_Y[j] = P_Z[j]$ holds. In the former case we return "true", in the latter case we increment j and do the same check; otherwise, or if j reached the maximal arity of a rule in P, we return "false".

For $>^c_P$ given some k, counting the the number of rules in P satisfied to degree k under a given interpretation Y, i.e., to establish $|P_Y[j]|$ can be done in polynomial time. The same algorithm as for $>^i_P$ shows that $Y >^c_P Z$ can thus be decided in polynomial time as well.

For (iii), i.e., $\delta_P(Y,Z) = \delta_Q(Y,Z)$, it is sufficient to see that computing δ can be done in polynomial time. Once again, we can use an algorithm which starts with $j = 1$ and checks $P_Y[j] \not\subseteq P_Z[j]$ or $P_Y[j] = P_Z[j]$. In the former case we return j, in the latter case we increment j and return to the check, and otherwise we return 0. We also return 0 if j reaches the maximal arity of rules in P. We do the same for Q, and compare the values the two algorithms return. Obviously, all involved steps can be done in polynomial time.

Since, as noted earlier, $|P_Y[j]|$ can be computed in polynomial time, given an interpretation Y and an integer j, also (iv) can be determined in polynomial time. □

Theorem 24 *Given LPODs* P *and* Q*, deciding* $P \mathrel{\widetilde{\equiv}} Q$ *is* co-NP*-complete for all* $\mathrel{\widetilde{\equiv}} \in \{\equiv_s, \equiv_{s,\times}, \equiv^\sigma_s, \equiv^\sigma_{s,\times} \mid \sigma \in \{p,i,c\}\}$.

Proof. Hardness follows from Lin (2002), together with Proposition 22. Membership for $\equiv_s$ (and thus, for $\equiv_{s,\times}$) between LPODs is easily seen via Theorem 6: In fact, for the complementary problem, we can guess a pair (X,Y) of interpretations and check whether (X,Y) is SE-model of exactly one of the compared programs. SE-model-checking is feasible in polynomial time—in particular, since the construction of the reduct P^Y can be done in polynomial time and since checking $Y \models P$ and $X \models P^Y$ amounts to classical model checking. This shows NP-membership for the complementary problem. Thus, we obtain co-NP-membership for $\equiv_s$ and $\equiv_{s,\times}$. Membership for $\equiv^\sigma_s$ and $\equiv^\sigma_{s,\times}$ (for $\sigma \in \{p,i,c\}$) involves tests additional to $\equiv_s$, as we show next. Since these additional tests are independent of the check for $\equiv_s$, co-NP-membership of the entire equivalence test follows.

For $\equiv^\sigma_s$ (for $\sigma \in \{p,i,c\}$), we know from Theorem 10 that $P \equiv^\sigma_s Q$ holds iff $P \equiv_s Q$ and for each pair Y, Z of joint models of P and Q, $Y >^\sigma_P Z$ iff $Y >^\sigma_Q Z$. Hence, to decide the complementary problem of the latter test, it is sufficient to guess two interpretations Y, Z, check whether both are models of P and Q (this can be done in polynomial time), and check that either (i) $Y >^\sigma_P Z$ and $Y \not>^\sigma_Q Z$; or (ii) $Y \not>^\sigma_P Z$ and $Y >^\sigma_Q Z$. By Lemma 23 these four checks can be done in polynomial time, and thus the complement of checking whether $Y >^\sigma_P Z$ iff $Y >^\sigma_Q Z$, for each pair Y, Z of models of P and Q, is in NP. Thus $\equiv^\sigma_s$ (for $\sigma \in \{p,i,c\}$) is in co-NP.

For $\equiv^p_{s,\times}$, we know from Theorem 14, that $P \equiv^p_{s,\times} Q$ iff $\langle SE(P), >^p_P, =^p_P \rangle = \langle SE(Q), >^p_Q, =^p_Q \rangle$, i.e., $P \equiv^p_{s,\times} Q$ iff $P \equiv^p_s Q$ and $=^p_P$ is the same relation as $=^p_Q$. We have shown above that $P \equiv^p_s Q$ is in co-NP, and to see that checking whether $=^p_P$ is equal to $=^p_Q$ is also in co-NP, the same argumentation as above is sufficient, making use of the corresponding result for checking $Y =^p_P Z$ from Lemma 23.

For $\equiv^i_{s,\times}$, we know from Theorem 14, that $P \equiv^i_{s,\times} Q$ iff $(SE(P), \delta_P) = (SE(Q), \delta_Q)$, i.e., iff $P \equiv_s Q$ and $\delta_P = \delta_Q$. The first problem is already shown in co-NP. To decide $\delta_P = \delta_Q$, we once more take the complementary problem, guess models Y, Z and check that $\delta_P(Y,Z) \neq \delta_Q(Y,Z)$. By Lemma 23 this can be done in polynomial time.

Finally for $\equiv^c_{s,\times}$, we know from Theorem 20, that $P \equiv^c_{s,\times} Q$ iff $(SE(P), >_P, \Delta_P) = (SE(Q), >_Q, \Delta_Q)$, i.e., iff $P \equiv^c_s Q$ and $\Delta_P = \Delta_Q$. Again, the first problem is already shown in co-NP, and to decide $\Delta_P = \Delta_Q$, we once more can make use of Lemma 23. □

Discussion

In this paper, we discussed different notions of strong equivalence for logic programs with ordered disjunctions, extending the usual one for normal logic programs. Following Brewka (2002) and Brewka, Niemelä, & Syrjänen (2004),

we studied Pareto-, inclusion-, and cardinality-based preference relations and introduced corresponding equivalence notions based on these strategies. We provided model-theoretic characterisations and introduced to that end a novel notion of a reduct for LPODs, leading to a direct generalisation of the well-known characterisation of strong equivalence for LPs by Turner (2003).

Although $\equiv^{\sigma}_{s,\times}$, for $\sigma \in \{p, i, c\}$, is arguably the most direct generalisation of strong equivalence for normal programs, in the sense that it tests whether two LPODs have the same preferred answer sets in any context, the other strong-equivalence notions are nonetheless relevant—in fact, $\equiv^{\sigma}_{s,\times}$ can be characterised in terms of some of the other strong-equivalence notions, provided that additional conditions hold as well.

Concerning related work, to the best of our knowledge, strong equivalence with respect to programs allowing for a representation of preferences has been studied only by Faber & Konczak (2006) (called "strong order equivalence"). However, the formalism studied there differs considerably from LPODs. Already syntactically, preferences are specified among rules using a construct different from rules. In LPODs, preferences are specified among atoms using an extended rule syntax. For this reason, also the semantics of the formalisms are hardly comparable. Indeed, also the characterisations of strong order equivalence obtained by Faber and Konczak (Faber & Konczak 2006) are quite different: For instance, the preferences expressed in two strongly order equivalent programs have to be exactly equal. This also implies that the rules upon which preferences are defined must occur in both strongly order equivalent programs. Therefore, one can never substitute a rule upon which a preference is expressed by another one without losing strong order equivalence.

Interesting issues for future work include the consideration of notions for *uniform equivalence* between LPODs. We plan to apply the new equivalence notions to derive syntactic program transformations (see, e.g. Eiter *et al.*; Cabalar, Pearce, & Valverde (2004; 2007), for transformations in the context of LPs) for LPODs as a basis for LPOD optimisation. We recall that in "traditional" answer-set programming, characterisations for strong equivalence paved the way for characterising also weaker notions of equivalence, which in turn provided more potential for optimising programs. Thus, our work may serve as a starting point for further work in this direction.

References

Balduccini, M., and Mellarkod, V. 2003. CR-Prolog with ordered disjunction. In *Proc. ASP 2003*, volume 78 of *CEUR Workshop Proceedings*, 98–112. CEUR-WS.org.

Baral, C. 2002. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Bertino, E.; Mileo, A.; and Provetti, A. 2005. PDL with preferences. In *Proc. POLICY 2005*, 213–222. IEEE Computer Society.

Brewka, G.; Niemelä, I.; and Syrjänen, T. 2002. Implementing ordered disjunction using answer set solvers for normal programs. In *Proc. JELIA 2002*, 444–455. Springer.

Brewka, G.; Niemelä, I.; and Syrjänen, T. 2004. Logic programs with ordered disjunctions. *Computational Intelligence* 20(2):335–357.

Brewka, G. 2002. Logic programming with ordered disjunction. In *Proc. AAAI 2002*, 100–105. AAAI Press.

Cabalar, P.; Pearce, D.; and Valverde A. 2002. Minimal logic programs. In *Proc. ICLP 2007*, volume 4670 of LNCS, 104–118. Springer.

Delgrande, J. P.; Schaub, T.; Tompits, H.; and Wang, K. 2004. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence* 20(2):308–334.

Eiter, T.; Fink, M.; Tompits, H.; and Woltran, S. 2004. Simplifying logic programs under uniform and strong equivalence. In *Proc. LPNMR-7*, volume 2923 of LNAI, 87–99. Springer.

Faber, W., and Konczak, K. 2006. Strong order equivalence. *Annals of Mathematics and Artificial Intelligence* 47(1–2):43–78.

Foo, N. Y.; Meyer, T.; and Brewka, G. 2004. LPOD answer sets and Nash equilibria. In *Proc. ASIAN 2004*, volume 3321 of *LNCS*, 343–351. Springer.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. ICLP 1988*, 1070–1080. MIT Press.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2(4):526–541.

Lin, F. 2002. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proc. KR 2002*, 170–176. Morgan Kaufmann.

Marek, V., and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*. Springer Verlag. 375–398.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25:241–273.

Turner, H. 2003. Strong equivalence made easy: Nested expressions and weight constraints. *Theory and Practice of Logic Programming* 3(4–5):609–622.

Zepeda, C.; Osorio, M.; Nieves, J. C.; Solnon, C.; and Sol, D. 2005. Applications of preferences using answer set programming. In *Proc. ASP 2005*, volume 142 of *CEUR Workshop Proceedings*. CEUR-WS.org.