Implementing Logic Programs with Ordered Disjunction Using *asprin*

Joohyung Lee and Zhun Yang
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, USA
{joolee, zyang90}@asu.edu

Abstract

Logic Programs with Ordered Disjunction (LPOD) is an extension of standard answer set programs to handle preference using the high-level construct of ordered disjunction whereas *asprin* is a recently proposed, general, flexible, and extensible framework that provides low-level constructs for representing preference in answer set programming. We present an encoding of LPOD in the language of *asprin* and the implementation LPOD2ASPRIN based on the encoding. Unlike the known method that applies only to a fragment of LPOD via the translation to Answer Set Optimization (ASO), our translation is general, direct, and simpler. It also leads to more efficient computation of LPOD using *asprin*.[1]

1 Introduction

Logic Programs with Ordered Disjunction (LPOD) (Brewka 2002) is an extension of standard answer set programs to handle preference using the high-level construct of ordered disjunction. *asprin* (Brewka et al. 2015b) is a recently proposed, general, flexible, and extensible framework for expressing and computing preferences in answer set programming, and, as such, the preference specification in the language of *asprin* is in a lower level than LPOD. Representing high-level preference constructs in the language of *asprin* could be verbose, and end-users may find it complicated to use. To alleviate the problem, *asprin* provides a library that implements several preference types, such as `subset`, `less(weight)`, and `ASO`. However, LPOD preference types are not one of them.

In (Brewka, Niemelä, and Syrjänen 2004), LPOD is implemented using SMODELS by interleaving the execution of two ASP programs—a *generator* which produces candidate answer sets and a *tester* which checks whether a given candidate answer set is most preferred or produces a more preferred answer set otherwise. In principle, the encodings in (Brewka, Niemelä, and Syrjänen 2004) can be used with *asprin* to implement LPOD. However, this method introduces a large number of translation rules and auxiliary atoms since it does not utilize the main component of *asprin*, preference statements.

In fact, it is known that using preference statements, some fragment of LPOD can be succinctly represented in the language of *asprin* via the translation into Answer Set Optimization (ASO). Brewka, Niemelä, and Truszczynski (2003) show how to turn LPOD under Pareto-preference into ASO programs, and Brewka et al. (2015a) show that ASO programs can be represented in *asprin*. By combining the two results, the fragment of LPOD can be represented in *asprin*. It is also mentioned that LPOD under inclusion-preference can be turned into "ranked" ASO (Brewka et al. 2015a) but the representation appears quite complicated. Furthermore, it is not known how the results apply to the other LPOD preference criteria.

This paper presents a more direct and simpler translation from LPOD into the language of *asprin*, handling all four preference criteria from (Brewka 2005) in a uniform way. Based on the translation, we implemented the system LPOD2ASPRIN, which translates LPOD programs into the input language of *asprin* and internally invokes the *asprin* system. Our experiments show that the system is more scalable than the other methods of computing LPOD.

The paper is organized as follows. Section 2 reviews LPOD and *asprin*. Section 3 presents a translation that turns LPOD into the language of *asprin*. Section 4 presents the LPOD2ASPRIN system and Section 5 compares its performance with other methods of computing LPOD. Section 6 discusses the related work. Selected proofs are given in the appendix.

2 Review of LPOD and *asprin*

2.1 Review: LPOD

We review the definition of LPOD by (Brewka 2002). As in that paper, for simplicity, we assume the underlying signature is propositional.

Syntax: A (propositional) LPOD Π is $\Pi_{reg} \cup \Pi_{od}$, where its *regular part* Π_{reg} consists of usual ASP rules

$$\mathit{Head} \leftarrow \mathit{Body}$$

and its *ordered disjunction part* Π_{od} consists of *LPOD rules* of the form

$$C^1 \times \cdots \times C^n \leftarrow \mathit{Body} \tag{1}$$

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] A short version of this paper is to appear in the *16th International Conference on Principles of Knowledge Representation and Reasoning KR 2018*. This paper contains more details about the system LPOD2ASPRIN and includes experimental results.

in which C^i are atoms, n is at least 2, and *Body* is a conjunction of atoms possibly preceded by *not*.[2] Rule (1) says "when *Body* is true, if possible then C^1; if C^1 is not possible then C^2; ...; if all of $C^1, \ldots, C^{n-1}$ are not possible then C^n." It is not the case that none of $C^1, \ldots, C^n$ is true when *Body* is true.

Semantics: For an LPOD rule (1), its *i-th option* ($i = 1, \ldots, n$) is defined as

$$C^i \leftarrow Body, not\ C^1, \ldots, not\ C^{i-1}.$$

A *split program* of an LPOD Π is obtained from Π by replacing each LPOD rule in Π_{od} by one of its options. A set S of atoms is a *candidate answer set* of Π if it is an answer set of a split program of Π.

Example 1 *(From (Brewka 2002)) The following LPOD* Π_1,

$$\begin{array}{lcl} a \times b & \leftarrow & not\ c \\ b \times c & \leftarrow & not\ d, \end{array}$$

has four split programs:

$$\begin{array}{ll} a \leftarrow not\ c & a \leftarrow not\ c \\ b \leftarrow not\ d & c \leftarrow not\ d, not\ b \\ \\ b \leftarrow not\ c, not\ a & b \leftarrow not\ c, not\ a \\ b \leftarrow not\ d & c \leftarrow not\ d, not\ b. \end{array}$$

Each of them has the following answer sets respectively, which are the candidate answer sets of Π_1.

$$\begin{array}{ll} \{a, b\} & \{c\} \\ \{b\} & \{b\}, \{c\}. \end{array}$$

A candidate answer set S of Π is said to *satisfy* rule (1)

- to degree 1 if S does not satisfy *Body*, and
- to degree j ($1 \leq j \leq n$) if S satisfies *Body* and $j = min\{k \mid C^k \in S\}$.

The notion of satisfaction degrees are the basis of defining a preference relation on the candidate answer sets of Π. For a candidate answer set S, let $S^i(\Pi)$ denote the set of rules in Π_{od} satisfied by S to degree i. For candidate answer sets S_1 and S_2 of Π, (Brewka 2005) introduces the following four preference criteria.

1. **Cardinality-Preferred:** S_1 is *cardinality-preferred* to S_2 ($S_1 >^c S_2$) if there is a positive integer i such that $|S_1^i(\Pi)| > |S_2^i(\Pi)|$, and $|S_1^j(\Pi)| = |S_2^j(\Pi)|$ for all $j < i$.
2. **Inclusion-Preferred:** S_1 is *inclusion-preferred* to S_2 ($S_1 >^i S_2$) if there is a positive integer i such that $S_2^i(\Pi) \subset S_1^i(\Pi)$, and $S_1^j(\Pi) = S_2^j(\Pi)$ for all $j < i$.
3. **Pareto-Preferred:** S_1 is *Pareto-preferred* to S_2 ($S_1 >^p S_2$) if there is a rule that is satisfied to a lower degree in S_1 than in S_2, and there is no rule that is satisfied to a lower degree in S_2 than in S_1.
4. **Penalty-Sum-Preferred:** S_1 is *penalty-sum-preferred* to S_2 ($S_1 >^{ps} S_2$) if the sum of the satisfaction degrees of all rules is smaller in S_1 than in S_2.

A candidate answer set S of Π is a *k-preferred* ($k \in \{c, i, p, ps\}$) *answer set* if there is no candidate answer set S' of Π such that $S' >^k S$.

When Π_{od} contains m LPOD rules, the *satisfaction degree list* of a candidate answer set S of Π is $(d_1, \ldots, d_m)$ where d_i is the degree to which S satisfies rule i in Π_{od}.

Example 1 (Continued) Recall that Π_1 has three candidate answer sets: $\{a, b\}$, $\{b\}$, and $\{c\}$. Their satisfaction degree lists are (1,1), (2,1), and (1,2), respectively. One can check that $\{a, b\}$ is the only preferred answer set according to any of the four preference criteria.

The following example shows differences in preferred answer sets depending on the different preference criteria.

Example 2 *To illustrate the difference among the four preference criteria, consider the following LPOD* Π_2 *about picking a hotel near the Grand Canyon.* $hotel(1)$ *is a 2 star hotel but is close to the Grand Canyon,* $hotel(2)$ *is a 3 star hotel and the distance is medium, and* $hotel(3)$ *is a 4 star hotel but is too far.*

$$close \times med \times far \times tooFar$$
$$star4 \times star3 \times star2$$

$$1\{hotel(X) : X = 1..3\}1$$
$$\bot \leftarrow hotel(1),\ not\ close$$
$$\bot \leftarrow hotel(1),\ not\ star2$$
$$\bot \leftarrow hotel(2),\ not\ med$$
$$\bot \leftarrow hotel(2),\ not\ star3$$
$$\bot \leftarrow hotel(3),\ not\ tooFar$$
$$\bot \leftarrow hotel(3),\ not\ star4$$

Π_2 *has* 4×3 *split programs but only the following three programs are consistent (The regular part of* Π_2 *is not listed).*

$$close$$
$$star2 \leftarrow not\ star4, not\ star3$$

$$med \leftarrow not\ close$$
$$star3 \leftarrow not\ star4$$

$$tooFar \leftarrow not\ close, not\ med, not\ far$$
$$star4$$

The candidate answer sets of Π_2 *and their satisfaction degree lists are*

$$\begin{array}{ll} S_1 = \{hotel(1), close, star2, \ldots\}, & (1, 3) \\ S_2 = \{hotel(2), med, star3, \ldots\}, & (2, 2) \\ S_3 = \{hotel(3), tooFar, star4, \ldots\}, & (4, 1) \end{array}$$

By definition, the cardinality-preferred answer set of Π_2 *is* S_1*, the inclusion-preferred answer sets are* S_1 *and* S_3*, the Pareto-preferred answer sets are* S_1*,* S_2 *and* S_3*, while the penalty-sum-preferred answer sets are* S_1 *and* S_2*.*

2.2 Review: *asprin*

asprin computes the most preferred answer sets of an ASP program P according to a preference specification $\hat{F}_s$ by repeated calls to CLINGO as in Figure 1. First, an arbitrary

[2] In (Brewka 2002), a usual ASP rule is viewed as a special case of a rule with ordered disjunction when $n = 1$ but in this paper, we distinguish them. This simplifies the presentation of the translation and also allows us to consider LPOD programs that are more general than the original definition by allowing modern ASP constructs such as aggregates.

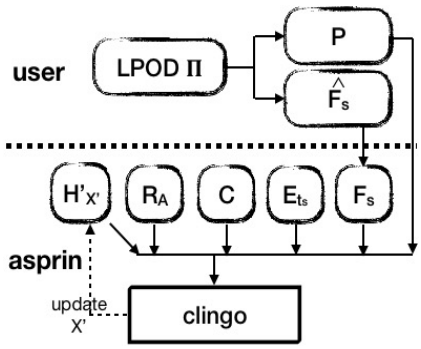

Figure 1: *asprin* Framework

answer set of P is generated as X'. Second, *asprin* tries to find an answer set X of P that is better than (i.e., preferred to) X' by running CLINGO on $P \cup F_s \cup E_{t_s} \cup H'_{X'} \cup R_A \cup C$, each of which is defined below. If CLINGO finds an answer set, which encodes the answer set X of P that is "better" than X', *asprin* replaces X' by X, and repeats the second step until CLINGO finds no answer sets, at which point X' is determined to be a most preferred answer set.

1. P is the *base program*, which consists of usual ASP rules. The answer sets of P are the "candidate answer sets" to apply a preference criterion.
2. $\hat{F}_s$ is the *preference specification* consisting of a single *optimization directive* of the form
$$\#optimize(s) \tag{2}$$
and a single [3] *preference statement* of the form
$$\#preference(s,t)\{e_1;\ldots;e_n\} \tag{3}$$
where $n \geq 0$; and s is the name of the preference statement; and t is its type (i.e., preference criterion). Each e_i is a *preference element* of the form
$$\phi_1 >> \cdots >> \phi_m$$
where $m \geq 1$ and each ϕ_i is a literal (an atom possibly preceded by *not*). [4] Intuitively, each index $1,\ldots,m$ gives the rank of the corresponding literal. The preference statement (3) declares a preference relation named s: each preference element in $\{e_1,\ldots,e_n\}$ gives a ranking of a set of literals while preference type t determines in which case one candidate answer set is better than another given the rankings. The optimization directive (2) tells *asprin* to restrict its reasoning mode to the preference relation declared by the preference statement whose name is s.
3. F_s is obtained from the preference specification $\hat{F}_s$ by turning the optimization directive (2) into an ASP fact
$$optimize(s)$$
and turning the preference statement (3) into an ASP fact
$$preference(s,t)$$
along with
$$preference(s,i,j,for(t_{\phi_j}),())$$
for each j-th literal ϕ_j in the i-th preference element e_i in (3). The term t_{ϕ_j} is defined as a if the literal ϕ_j is an atom a, and is $neg(a)$ if the literal ϕ_j is "*not* a". [5]
4. E_{t_s} is the *preference encoding* for t_s, where t_s is the type of the preference statement named s. It defines a reserved predicate $better(s)$, which is true iff there exists a candidate answer set X that is preferred to X' according to preference type t_s and the facts in F_s. In Section 3.3, we show four preference encodings $E_{lpod(c)}$, $E_{lpod(i)}$, $E_{lpod(p)}$, and $E_{lpod(ps)}$ for each of the four preference types (i.e., criteria) for LPOD.
5. $H'_{X'}$ is the set of ASP facts
$$\{holds'(a) \mid a \in X'\}$$
which reifies the atoms in X' in the form of $holds'(\cdot)$.[6]
6. R_A is the set of ASP rules
$$\{holds(a) \leftarrow a \mid a \text{ is an atom in } P\}$$
which reifies the atoms in any candidate answer set X in the form of $holds(\cdot)$.
7. C is a set of (domain-independent) ASP rules as follows.[7]
$$\bot \leftarrow \textit{not } better(S), optimize(S). \tag{4}$$
$$holds(neg(A)) \leftarrow \textit{not } holds(A), \quad preference(\_,\_,\_,for(neg(A)),\_). \tag{5}$$
$$holds'(neg(A)) \leftarrow \textit{not } holds'(A), \quad preference(\_,\_,\_,for(neg(A)),\_). \tag{6}$$
Rule (4) instructs the *asprin* system to find an answer set X that is better than X' according to the preference statement S. Rule (5) is about X, which is reified in the form of $holds(\cdot)$: for the literal of the form "*not* A" in the preference statement (3), it says $holds(neg(A))$ is true if $holds(A)$ is false in the reified X (i.e., $X \not\models A$). Similarly, rule (6) is about X', which is reified in the form of $holds'(\cdot)$.

Given a program P and a preference specification $\hat{F}_s$, we say an answer set X of P is a *preferred answer set* of P w.r.t. $\hat{F}_s$ if $P \cup F_s \cup E_t \cup H'_X \cup R_A \cup C$ has no answer set, where t is the type of the preference statement s declared in $\hat{F}_s$.

[3] *asprin* allows multiple preference statements in the input but for simplicity of the presentation we assume a single preference statement.

[4] In general, *asprin* allows for a more general syntax of preference specification and preference element. For the purpose of this paper, it is sufficient to consider this simple fragment.

[5] The last term is empty because we consider ϕ_j as a non-weighted formula.

[6] Note that this is based on the definition of H'_X, which is the set of ASP facts $\{holds'(a) \mid a \in X\}$.

[7] In general, C contains more rules such as the rule to define $holds(or(A,B))$. They are omitted because they are not related to our translation.

3 Representing LPOD in *asprin*

Let Π be an LPOD where Π_{od} consists of m propositional rules as follows.

$$\begin{array}{ll} 1: & C_1^1 \times \cdots \times C_1^{n_1} \leftarrow \mathit{Body}_1 \\ & \cdots \\ m: & C_m^1 \times \cdots \times C_m^{n_m} \leftarrow \mathit{Body}_m \end{array} \quad (7)$$

where $1, \ldots, m$ are rule indices; $n_i \geq 2$ for $1 \leq i \leq m$.

In the following subsections, we present the component programs of *asprin* that encode LPOD Π, namely, P, $\hat{F}_s$, E_{t_s}. The other components, F_s, H'_X, R_A and C are generated as described above.

3.1 Base Program P

For the LPOD program $\Pi = \Pi_{reg} \cup \Pi_{od}$, the base program P contains all rules in Π_{reg} and, for each LPOD rule

$$C_i^1 \times \cdots \times C_i^{n_i} \leftarrow \mathit{Body}_i$$

in Π_{od}, P contains

$$\mathit{body}_i \leftarrow \mathit{Body}_i \quad (8)$$

$$\{C_i^1\} \leftarrow \mathit{body}_i \quad (9)$$

$$\cdots$$

$$\{C_i^{n_i-1}\} \leftarrow \mathit{body}_i, \mathit{not}\ C_i^1, \ldots, \mathit{not}\ C_i^{n_i-2} \quad (10)$$

$$C_i^{n_i} \leftarrow \mathit{body}_i, \mathit{not}\ C_i^1, \ldots, \mathit{not}\ C_i^{n_i-1} \quad (11)$$

Rule (8) defines the case when the body of rule i is true. Rules (9)–(10) say that if the body of rule i is true and each C_i^j is false ($j \in \{1, \ldots, k-1\}$), then C_i^k is possibly true. Rule (11) says that if the body of rule i is true and C_i^j is false for all $j \in \{1, \ldots, n_i - 1\}$, then $C_i^{n_i}$ must be true.

The above method of generating candidate answer sets using choice rules is from (Cabalar 2011). It is not difficult to check that the answer sets of this program P are the candidate answer sets of LPOD Π (ignoring body_i atoms).

Proposition 1 *For any LPOD Π and any set X of atoms in Π, X is a candidate answer set of Π iff $X \cup \{\mathit{body}_i \mid X$ satisfies the body of rule i in $\Pi_{od}\}$ is an answer set of P.*

Example 2 (Continued) For LPOD Π_2, the P-component of the *asprin* program is as follows.

```
body_1.
{close} :- body_1.
{med} :- body_1, not close.
{far} :- body_1, not close, not med.
tooFar :- body_1, not close, not med, not far.

body_2.
{star4} :- body_2.
{star3} :- body_2, not star4.
star2 :- body_2, not star4, not star3.

1{hotel(X): X=1..3}1.
:- hotel(1), not close.      :- hotel(1), not star2.
:- hotel(2), not med.        :- hotel(2), not star3.
:- hotel(3), not tooFar.     :- hotel(3), not star4.
```

The answer sets of the P-component are

$$\begin{array}{lllll} \{\mathit{hotel}(1), & \mathit{close}, & \mathit{star}2, & \mathit{body}_1, & \mathit{body}_2\} \\ \{\mathit{hotel}(2), & \mathit{med}, & \mathit{star}3, & \mathit{body}_1, & \mathit{body}_2\} \\ \{\mathit{hotel}(3), & \mathit{tooFar}, & \mathit{star}4, & \mathit{body}_1, & \mathit{body}_2\} \end{array}$$

which are exactly the unions of the candidate answer sets of Π and $\{\mathit{body}_1, \mathit{body}_2\}$.

3.2 Preference Specification $\hat{F}_s$

$\hat{F}_s$ contains an optimization directive

$$\#\mathit{optimize}(s)$$

and a preference statement

$$\begin{array}{l} \#\mathit{preference}(s, \mathit{lpod}(s))\ \{ \\ \quad \mathit{not}\ \mathit{body}_1 >> C_1^1 >> \cdots >> C_1^{n_1}; \\ \quad \cdots \\ \quad \mathit{not}\ \mathit{body}_m >> C_m^1 >> \cdots >> C_m^{n_m} \\ \} \end{array} \quad (12)$$

where $s \in \{c, i, p, ps\}$ denotes one of the four preference criteria for LPOD, and each line of (12) is associated with each LPOD rule. Intuitively, to check the satisfaction degree of an LPOD rule i, we check the truth value of the literals in the order specified in the i-th preference element. We first check whether *not* body_i is true. If *not* body_i is true, i.e., the body of rule i is false, the satisfaction degree must be 1 and we do not have to check further; and if it is not the case, check whether C_i^1 is true, and so on.

Example 2 (Continued) For LPOD Π_2 which contains LPOD rules

$$\begin{array}{l} \mathit{close} \times \mathit{med} \times \mathit{far} \times \mathit{tooFar} \\ \mathit{star}4 \times \mathit{star}3 \times \mathit{star}2 \end{array}$$

to find its cardinality-preferred answer sets, we set the preference criterion s to c, and let $\hat{F}_s$ be the following.

```
#optimize(c).

#preference(c, lpod(c)) {
  not body_1 >> close >> med >> far >> tooFar ;
  not body_2 >> star4 >> star3 >> star2
}.
```

asprin internally turns $\hat{F}_s$ into F_s as follows.

```
optimize(c).

preference(c, lpod(c)).

preference(c, 1, 1, for(neg(body_1)), ()).
preference(c, 1, 2, for(close), ()).
preference(c, 1, 3, for(med), ()).
preference(c, 1, 4, for(far), ()).
preference(c, 1, 5, for(tooFar), ()).

preference(c, 2, 1, for(neg(body_2)), ()).
preference(c, 2, 2, for(star4), ()).
preference(c, 2, 3, for(star3), ()).
preference(c, 2, 4, for(star2), ()).
```

The facts $optimize(c)$ and $preference(c, lpod(c))$ assert that we optimize according to the preference statement c of type $lpod(c)$ (inclusion preference). The fact $preference(c, 2, 1, for(neg(body\_2)), ())$ asserts that the first literal of the second preference element of the preference statement c is "*not* $body\_2$".

3.3 Preference Encoding E_{t_s}

The aim of E_{t_s} is to find an answer set X (reified in the form of $holds(\cdot)$) that is better than (i.e., preferred to) the current answer set X' (reified in the form of $holds'(\cdot)$) with respect to the preference type t_s.

We introduce the preference encodings E_{t_s} for each $t_s \in \{lpod(c),\ lpod(i),\ lpod(p),\ lpod(ps)\}$. Each E_{t_s} contains the common rules Deg as defined below.

Degree The aim of Deg is to find the satisfaction degree to which each LPOD rule R is satisfied by X or X'. [8]

Deg consists of the following two rules.

$$degree(R, D) \leftarrow optimize(S), preference(S, lpod(\_)), \\ preference(S, R, I, \_, \_),\ D = \#max\{1; I-1\}, \\ I = \#min\{J : holds(A), preference(S, R, J, for(A), \_)\}. \quad (13)$$

$$degree'(R, D) \leftarrow optimize(S), preference(S, lpod(\_)), \\ preference(S, R, I, \_, \_),\ D = \#max\{1; I-1\}, \\ I = \#min\{J : holds'(A), preference(S, R, J, for(A), \_)\}. \quad (14)$$

Rule (13) records the degree D to which rule R is satisfied by X (X is reified in the form of $holds(\cdot)$). It asserts that if we want to optimize according to preference statement S whose type is one of the four $lpod(\cdot)$ types, then we need to calculate the satisfaction degree D for each rule R: D is the maximum value of 1 and $I-1$ where I is the index of the first literal in the preference element R that is true in X. Rule (14) is similar to rule (13) except that it finds the satisfaction degree D of rule R for X'.

Cardinality-Preferred $E_{lpod(c)}$ contains Deg and the following two rules:

$$worse2degree(S, D) \leftarrow optimize(S), preference(S, lpod(c)), \\ degree'(\_, D), \\ \#sum\{\ \ 1, R : degree(R, D); \\ \quad -1, R : degree'(R, D)\} < 0. \quad (15)$$

$$better(S) \leftarrow optimize(S), preference(S, lpod(c)), \\ degree(\_, D), \\ \#sum\{\ \ 1, R : degree(R, D); \\ \quad -1, R : degree'(R, D)\} > 0, \\ not\ worse2degree(S, J) : J = 1..D-1. \quad (16)$$

Rule (15) defines the case when X is worse than, i.e., less preferred to, X' at degree D: X satisfies less LPOD rules to degree D than X'. In this case, there must be at least one LPOD rule that is satisfied to degree D by X', which is guaranteed by $degree'(\_, D)$. Rule (16) says that X is better than X' according to the preference type $lpod(c)$ if there exists a degree D such that X is preferred to X' at degree D (i.e., X satisfies more rules to degree D than X') and X is not worse than X' at all lower degrees. Note that "*not* $worse2degree(S, J) : J = 1..D-1$" is a *conditional literal*, and is equivalent to the conjunction of literals "*not* $worse2degree(S, J)$" for all $J \in \{1, \ldots, D-1\}$.

Inclusion-Preferred $E_{lpod(i)}$ contains Deg and two rules:

$$prf2degree(S, D) \leftarrow optimize(S), preference(S, lpod(i)), \\ degree(\_, D), \\ \#count\{J : degree(J, D), not\ degree'(J, D)\} > 0, \\ degree(J, D) : degree'(J, D). \quad (17)$$

$$better(S) \leftarrow preference(S, lpod(i)), \\ prf2degree(S, D), \\ degree(R, J) : degree'(R, J), J < D. \quad (18)$$

Rule (17) defines the case when X is preferred to X' at degree D: (i) X satisfies at least one rule to degree D; (ii) there is a rule J that is satisfied by X, but not by X', to degree D; and (iii) all rules J that are satisfied by X' to degree D are also satisfied by X to the same degree. Rule (18) says that X is better than X' according to preference type $lpod(i)$ if there exists a degree D such that X is preferred to X' at degree D, and any rule R that is satisfied by X' to a lower degree than D should also be satisfied by X to the same degree.

Pareto-Preferred $E_{lpod(p)}$ contains Deg and two rules:

$$equ(S) \leftarrow optimize(S), preference(S, lpod(p)), \\ D1 = D2 : degree(R, D1), degree'(R, D2). \quad (19)$$

$$better(S) \leftarrow optimize(S), preference(S, lpod(p)), \\ not\ equ(S), \\ D1 \leq D2 : degree(R, D1), degree'(R, D2). \quad (20)$$

Rule (19) defines that X and X' are "equivalent" if they satisfy each LPOD rule to the same degree. Rule (20) says that X is better than X' according to preference type $lpod(p)$ if X is not "equivalent" to X', and X satisfies each LPOD rule R to a degree that is the same or lower than the degree to which X' satisfies R.

Penalty-Sum-Preferred $E_{lpod(ps)}$ contains Deg and one rule:

$$better(S) \leftarrow optimize(S), preference(S, lpod(ps)), \\ \#sum\{D, R : degree(R, D); \\ \quad -D, R : degree'(R, D)\} < 0. \quad (21)$$

Rule (21) says that X is better than X' according to preference type $lpod(ps)$ if the sum of the degrees to which the LPOD rules are satisfied by X is lower than the sum of the degrees to which the LPOD rules are satisfied by X'.

Theorem 1 *For any LPOD* Π*, X is an s-preferred answer set ($s \in \{c, i, p, ps\}$) of* Π *in the sense of LPOD iff* $X \cup \{body_i \mid X$ *satisfies the body of rule* i *in* $\Pi_{od}\}$ *is a preferred answer set of* P *w.r.t.* $\hat{F}_s$ *in the sense of asprin, where* P *and* $\hat{F}_s$ *are obtained from* Π *as above.*

[8] Note that each preference element denotes an LPOD rule. We use symbol R to denote the index of the preference element in predicate "$preference(S, R, I, \_, \_)$" because R is also the index of the denoted LPOD rule.

4 LPOD2ASPRIN System

We implement system LPOD2ASPRIN as in Figure 2. The system first translates an LPOD program Π into a base program P and a preference specification $\hat{F}_s$ in the language of *asprin* as described in Sections 3.1 and 3.2, which are fed into the *asprin* system along with other component programs. We put the encodings $E_{lpod(c)}$, $E_{lpod(i)}$, $E_{lpod(p)}$, and $E_{lpod(ps)}$ in the *asprin* library. The encodings are exactly the same as those in Section 3.3 except that we eliminate the use of $\#min$ and $\#max$ by replacing rule (13) (and rule (14) accordingly) with

```
degree(R,1) :- preference(S, lpod(_)),
   preference(S,R,1,for(A),_), holds(A).
degree(R,D-1) :- preference(S, lpod(_)),
   preference(S,R,D,for(A),_), holds(A), D>1,
   not holds(B): preference(S,R,J,for(B),_), 0<J, J<D.
```

The reason for this change is because our experiments show significant speed-up with the alternative encoding.

Finally, an s-preferred answer set of Π is obtained from the output of *asprin* by removing the auxiliary atoms $body_i$.

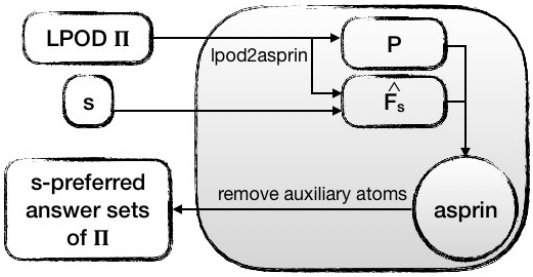

Figure 2: LPOD2ASPRIN System Overview

The LPOD2ASPRIN system homepage is

`http://reasoning.eas.asu.edu/lpod2asprin/`

which contains the source code, the tutorial, examples and some experimental results.

To find an s-preferred ($s \in \{c, i, p, ps\}$) answer set of an LPOD Π, one may execute the command

```
python lpod2asprin.py -i input.txt -type s
```

where

- `input.txt` stores the LPOD Π written in the input format of CLINGO except that the symbol `>>` is used to denote the ordered disjunction symbol; and
- `s` is one of the preference criteria in {`c`, `i`, `p`, `ps`}.

Example 2 (Continued) In the language of LPOD2ASPRIN, Π_2 is written as

```
dom(1..3).

1{hotel(X): dom(X)}1.
:- hotel(1), not close.
:- hotel(1), not star2.
:- hotel(2), not med.
:- hotel(2), not star3.
:- hotel(3), not tooFar.
:- hotel(3), not star4.

close >> med >> far >> tooFar.

star4 >> star3 >> star2.
```

If we save this program in file hotel.txt and want to find the i-preferred answer set of Π_2, we can execute

```
python lpod2asprin.py -i hotel.txt -type i
```

which outputs

```
Input LPOD program: hotel.txt
Type of LPOD preference criterion: i

asprin version 3.0.2
Reading from /asprin-3.0.2/asprin/lpod.lp ...
Solving...
Answer: 1
dom(1) dom(2) dom(3) hotel(3) tooFar star4
OPTIMUM FOUND
Answer: 2
dom(1) dom(2) dom(3) hotel(1) close star2
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
  Optimal    : 2
```

The output says that *asprin* finds two i-preferred answer sets of Π_2: $\{hotel(1), close, star2, \dots\}$ and $\{hotel(3), tooFar, star4, \dots\}$, which is as expected.

5 Experiments

Since there is no benchmarks available in the existing literature for LPOD, we designed two benchmarks and compare the run-time of LPOD2ASPRIN with two other methods: PSMODELS from (Brewka, Niemelä, and Syrjänen 2004) and using *asprin* via reduction to ASO (Brewka et al. 2015a). The latter method does not have a dedicated solver for LPOD, so we manually translate the LPOD programs into ASO programs and then into the input language of *asprin*. We only compare w.r.t. Pareto preference because it is not known how the *asprin* via ASO method can be used to represent cardinality preference and penalty-sum preference, and the representation for inclusion preference appears to be complicated.

5.1 Benchmark: abc

This benchmark is used to test how the system LPOD2ASPRIN performs with an increasing number of LPOD rules.

The **abc** example, as shown below, contains n LPOD rules, each of which contains two atoms in its head. The program generates one or two $a(i)$ and one or two $c(i)$ ($i \in \{1, \dots, n\}$), and restricts that $a(i)$ is true iff $b(i)$ is false for any $i \in \{1, \dots, n\}$. There is also a preference of $a(i)$ over $b(i)$ if $c(i)$ is true.

n	PSMODELS (2004)	*asprin* via ASO (2015a)	LPOD2ASPRIN
10	1.244s	3.977s	3.432s
15	10.088s	14.282s	11.374s
20	47.689s	45.860s	34.677s
25	2m47.842s	2m6.501s	1m32.202s
30	8m12.839s	5m1.220s	3m40.439s
35	18m26.952s	10m39.620s	7m42.605s
40	42m6.830s	20m25.556s	14m41.012s

Table 1: Experiment on the **abc** Example

```
dom(1..n).
1{a(X): dom(X)}2.                    1{c(X): dom(X)}2.
b(X) :- dom(X), not a(X).            :- a(X), b(X).

a(X) >> b(X) :- c(X).
```

Table 1 shows the run-time comparison of finding Pareto-preferred answer sets of this program with different values of n. We compare LPOD2ASPRIN with the implementation from (Brewka, Niemelä, and Syrjänen 2004) and the method via ASO reduction from (Brewka et al. 2015a) as we mentioned earlier.

In comparison, *asprin* using the reduction via ASO is more scalable than PSMODELS. However, since the semantics of ASO programs is analogous to the semantics of LPOD under Pareto preference, the reduction of LPOD into ASO programs is only straightforward under this preference. On the other hand, the LPOD2ASPRIN system works for any of the four preference criteria and scales better than the other methods.

5.2 Benchmark: n-Hotel

The **n-Hotel** example is about choosing a hotel from n candidate hotels based on the preferences over the prices, distances, and ratings of services. The input program contains three LPOD rules, each of which has n atoms in the head. The program is automatically generated with a parameter n, denoting the number of candidate hotels. Below is a program generated when $n = 4$ with random orders of price, distance, and service.

```
1{hotel(X): dom(X)}1.

:- hotel(X), price(Y), X!=Y, dom(X), dom(Y).
:- hotel(X), distance(Y), X!=Y, dom(X), dom(Y).
:- hotel(X), service(Y), X!=Y, dom(X), dom(Y).

dom(1..4).

price(3) >> price(4) >> price(1) >> price(2).

distance(4) >> distance(3) >> distance(1) >>
distance(2).

service(2) >> service(1) >> service(3) >> service(4).
```

The run-time of finding the Pareto-preferred answer sets of this program for different values of n is shown in Table 2.

n	PSMODELS (2004)	LPOD2ASPRIN
10	0.060s	0.438s
50	2.485s	1.961s
60	4.825s	1.788s
70	Terminate with No Result	3.147s
100	Terminate with No Result	6.730s
200	Terminate with No Result	47.868s
300	Terminate with No Result	1m45.006s

Table 2: Experiment on the **n-Hotel** Example

[9]

As we see, the LPOD2ASPRIN system is much more scalable than the original LPOD implementation. Besides, the original LPOD implementation cannot find any answer set if the number (n) of atoms in the head of an LPOD rule exceeds 70 whereas our system can find the preferred answer set even when $n = 800$ (800 is not an upper bound for our system, but is the biggest number we have tested).

In summary, the experimental results on the benchmarks show that the LPOD2ASPRIN system is more scalable than PSMODELS (Brewka, Niemelä, and Syrjänen 2004). In comparison with the method via ASO reduction, the method is more general to cover all preference types, and the encoding is more compact leading to more scalable computation.

6 Related Work and Conclusion

We already mentioned the method of (Brewka et al. 2015a) works under the Pareto preference. However, the reduction under inclusion preference requires a translation from LPOD to "ranked" ASO programs, which further requires a more complex reduction to *asprin*. Besides, the reductions from LPOD to ASO programs under cardinality and penalty-sum preferences were not shown. In comparison, our method reduces LPOD directly to *asprin*, which yields a simpler and uniform method that applies to all preference criteria for LPOD.

Asuncion *et al.* (2014) present a first-order semantics of logic programs with ordered disjunction by a translation into second-order logic.

Lee and Yang (2018) show a reduction from LPOD to answer set programs, where the semantics of each preference type is also represented by standard ASP rules. Their reduction is one-pass: the preferred answer sets are computed by calling an answer set solver one time by generating all candidate answer sets to which preference criteria are applied. The computation is not as scalable as LPOD2ASPRIN which makes iterative calls to CLINGO.

asprin has a library of built-in preference types, but LPOD preference is not one of them. Our preference encodings may be included in the *asprin* library to benefit the end-users.

Acknowledgments: We are grateful to the anonymous referees for their useful comments. This work was partially

[9]We did not test the *asprin* via ASO reduction because we did not implement a compiler for this method while a manual translation takes too much efforts.

supported by the National Science Foundation under Grants IIS-1526301 and IIS-1815337.

References

Asuncion, V.; Zhang, Y.; and Zhang, H. 2014. Logic programs with ordered disjunction: first-order semantics and expressiveness. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2–11. AAAI Press.

Brewka, G.; Delgrande, J.; Romero, J.; and Schaub, T. 2015a. Implementing preferences with asprin. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, 158–172. Springer.

Brewka, G.; Delgrande, J. P.; Romero, J.; and Schaub, T. 2015b. asprin: Customizing answer set preferences without a headache. In *AAAI*, 1467–1474.

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

Brewka, G.; Niemelä, I.; and Truszczynski, M. 2003. Answer set optimization. In *IJCAI*, volume 3, 867–872.

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

Brewka, G. 2005. Preferences in answer set programming. In *CAEPIA*, volume 4177, 1–10. Springer.

Cabalar, P. 2011. A logical characterisation of ordered disjunction. *AI Communications* 24(2):165–175.

Lee, J., and Yang, Z. 2018. Translating LPOD and CR-Prolog2 into standard answer set programs. *Journal of Theory and Practice of Logic Programming (TPLP)*, 18(3–4): 589–606.

Appendix: Proof of Theorem 1

Let X be a set of atoms and let σ be a signature. By $X|_\sigma$, we denote the projection of X onto σ.

Lemma 1 *Let Π be an answer set program, let X be an answer set of Π, and let $constraint$ be a rule of the form "$\bot \leftarrow$ Body". If $X \models constraint$, X is an answer set of $\Pi \cup \{constraint\}$.*

Lemma 2 *Let Π be an answer set program of signature σ. Let X be a set of atoms in σ, let Body be a conjunction of atoms (possibly preceded by not) in σ, and let a be an atom not in σ. X is an answer set of Π iff $X \cup \{a \mid X \models \text{Body}\}$ is an answer set of $\Pi' \cup \{a \leftarrow \text{Body}\}$, where Π' is obtained from Π by replacing the occurrence of Body in the body of some (i.e., from zero to all) rules in Π with a.*

Theorem 1 *For any LPOD Π and any set X of atoms in Π, X is an s-preferred answer set ($s \in \{c, i, p, ps\}$) of Π according to LPOD iff $X \cup \{body_i \mid X$ satisfies the body of rule i in $\Pi_{od}\}$ is an s-preferred answer set of P according to $asprin$, where P is the base program obtained from Π. In other words, (let $\phi(X)$ be $X \cup \{body_i \mid X$ satisfies the body of rule i in $\Pi_{od}\}$)*

(a) *X is a candidate answer set of Π and*

(b) *there is no candidate answer set X' of Π that is s-preferred to X according to LPOD*

iff

(c) *$\phi(X)$ is an answer set of P and*

(d) *$P \cup F_s \cup E_{t_s} \cup H'_{\phi(X)} \cup R_A \cup C$ has no answer set.*

Proof. **($\rightarrow$)** Let Π be an LPOD of signature σ with m LPOD rules. Let X be a candidate answer set of Π such that there is no candidate answer set X' of Π that is s-preferred to X according to LPOD. By Proposition 1, $\phi(X)$ is an answer set of P.

Assume for the sake of contradiction that $P \cup F_s \cup E_{t_s} \cup H'_{\phi(X)} \cup R_A \cup C$ has an answer set S. Let σ' be $\sigma \cup \{body_i \mid i \in \{1, \ldots, m\}\}$. Note that σ' is the signature of P. By the splitting theorem, $S|_{\sigma'}$ is an answer set of P. By Proposition 1, $S|_\sigma$ is a candidate answer set of Π. We will prove that $S|_\sigma$ is s-preferred to X, which contradicts with bullet **(b)**.

Let a be an atom in σ. Here we list some facts that will be used in the proof.

1. By R_A, $S \models holds(a)$ iff $S|_{\sigma'} \models a$. Since a is an atom in σ, $S \models holds(a)$ iff $S|_\sigma \models a$.
2. By $H'_{\phi(X)}$, $S \models holds'(a)$ iff $\phi(X) \models a$. Since a is an atom in σ, $S \models holds'(a)$ iff $X \models a$.
3. By F_s, S satisfies $optimize(s)$ and S satisfies $preference(s, lpod(s))$.
4. By F_s, S satisfies $preference(s, r, j, for(a), ())$ iff the $(j-1)$-th atom ($j \geq 2$) in the head of LPOD rule r is a; S satisfies $preference(s, r, 1, for(neg(body_r)), ())$ iff Π contains an LPOD rule r.
5. By rules (5) and (6) in C, for $i \in \{1, \ldots, m\}$, S satisfies $holds(neg(body_i))$ iff S does not satisfy $holds(body_i)$; S satisfies $holds'(neg(body_i))$ iff S does not satisfy $holds'(body_i)$.
6. By rule (13) in E_{t_s}, S satisfies $degree(R, D)$ iff $S|_\sigma$ satisfies LPOD rule R to degree D. This is because for any LPOD rule r, in case $S \models degree(r, 1)$,

   - $S \models degree(r, 1)$

   iff (by rule (13), bullet 3, and bullet 4)

   - in the case when $I = 1$:
     - $S \models preference(s, r, 1, for(neg(body_r)), \_)$, and
     - $S \models holds(neg(body_r))$
   - or in the case when $I = 2$: there exists an atom a such that
     - $S \models preference(s, r, 2, for(a)), \_)$, and
     - $S \models holds(a)$, and
     - $S \not\models holds(neg(body_r))$

   iff (by bullets 4 and 5)

   - in the case when $I = 1$:
     - Π contains an LPOD rule r, and
     - S does not satisfy $holds(body_r)$
   - or in the case when $I = 2$: there exists an atom a such that
     - the 1-st atom in the head of LPOD rule r is a, and

– $S \models holds(a)$, and
– $S \models holds(body_r)$

iff (by bullet 1 and rule (8))

- in the case when $I = 1$:
– $S|_\sigma$ does not satisfy the body of LPOD rule r
- or in the case when $I = 2$:
– $S|_\sigma$ satisfies the first atom in the head of LPOD rule r, and
– $S|_\sigma$ satisfies the body of LPOD rule r

iff (by definition)

- $S|_\sigma$ satisfies LPOD rule r to degree 1;

and in case $S \models degree(r, d)$ where d is greater than 1,

- $S \models degree(r, d)$, $d \geq 2$

iff (by rule (13), bullet 3, and bullet 4)

- there exists an atom a such that
– $S \models preference(s, r, d + 1, for(a)), \_)$, and
– $S \models holds(a)$, and
- $S \not\models holds(b)$ for any b and d' such that $S \models preference(s, r, d' + 1, for(b)), \_)$ and $d' < d$

iff (by bullet 4)

- there exists an atom a such that
– the d-th atom in the head of LPOD rule r is a, and
– $S \models holds(a)$, and
- $S \not\models holds(b)$ for any d'-th atom b in the head of LPOD rule r where $d' < d$

iff (by bullet 1)

- $S|_\sigma$ satisfies the d-th atom in the head of LPOD rule r, and
- $S|_\sigma$ does not satisfy the d'-th atom in the head of LPOD rule r for any $d' < d$

iff (by definition)

- $S|_\sigma$ satisfies LPOD rule r to degree d.

7. Similarly, by rule (14) in E_{t_s}, S satisfies $degree'(R, D)$ iff X satisfies LPOD rule R to degree D. The proof is analogous to that in bullet 6.

Now, we will prove that $S|_\sigma$ is s-preferred to X for $s \in \{i, p, ps\}$. The proof for the case when s is c (cardinally preference) is not shown due to the limit of space.

- **Inclusion-Preferred** By bullet 3, S satisfies $optimize(i)$ and $preference(i, lpod(i))$. Since S satisfies rule (4), S must satisfy $better(i)$. Besides,
– $S \models better(i)$

iff (by rule (18))

– there exists an integer d such that
∗ $S \models prf2degree(i, d)$, and
∗ for any LPOD rule $r \in \{1, \ldots, m\}$ and any degree j such that $j < d$, if $S \models degree'(r, j)$, then $S \models degree(r, j)$

iff (by rule (17))

– there exists an integer d such that
∗ there exists at least one LPOD rule j such that $S \models degree(j, d)$ and $S \not\models degree'(j, d)$, and for any LPOD rule j, if $S \models degree'(j, d)$, then $S \models degree(j, d)$, and
∗ for any LPOD rule $r \in \{1, \ldots, m\}$ and any degree j such that $j < d$, if $S \models degree'(r, j)$, then $S \models degree(r, j)$

iff (by bullet 6 and bullet 7)

– there exists an integer d such that
∗ the set of LPOD rules that are satisfied to degree d by X is a proper subset of the set of LPOD rules that are satisfied to degree d by $S|_\sigma$, and
∗ for any degree j such that $j < d$, the set of LPOD rules that are satisfied to degree j by X is a subset of the set of LPOD rules that are satisfied to degree j by $S|_\sigma$

iff (by definition)

– $S|_\sigma$ is i-preferred to X according to LPOD.

So $S|_\sigma$ is i-preferred to X according to LPOD. Contradiction.

- **Pareto-Preferred** Since S satisfies rule (4), and satisfies $optimize(p)$ (in F_s), S must satisfy $better(p)$. Besides,
– $S \models better(p)$

iff (by rule (20))

– $S \not\models equ(p)$, and
– for any LPOD rule $r \in \{1, \ldots, m\}$, if $S \models degree(r, d_1)$ and $S \models degree'(r, d_2)$ for any d_1 and d_2, then $d_1 \leq d_2$

iff (by rule (19))

– it is not the case that for all LPOD rule $r \in \{1, \ldots, m\}$, $S \models degree(r, d)$ iff $S \models degree'(r, d)$ for any d, and
– for any LPOD rule $r \in \{1, \ldots, m\}$, if $S \models degree(r, d_1)$ and $S \models degree'(r, d_2)$ for any d_1 and d_2, then $d_1 \leq d_2$

iff (by bullet 6)

– there is an LPOD rule that is satisfied to a lower degree in $S|_\sigma$ than in X, and there is no rule that is satisfied to a lower degree in $\phi(X)$ than in $S|_\sigma$

iff (by definition)

– $S|_\sigma$ is p-preferred to X according to LPOD.

So $S|_\sigma$ is p-preferred to X according to LPOD. Contradiction.

- **Penalty-Sum-Preferred** Since S satisfies rule (4), and satisfies $optimize(ps)$ (in F_s), S must satisfy $better(ps)$. Besides,
– $S \models better(ps)$

iff (by rule (21))

– $\sum_{R:S \models degree(R,D)} D < \sum_{R:S \models degree'(R,D)} D$

iff (by bullet 6)

– the sum of the satisfaction degrees of all rules is smaller in $S|_\sigma$ than in X

iff (by definition)

– $S|_\sigma$ is ps-preferred to X according to LPOD.

So $S|_\sigma$ is ps-preferred to X according to LPOD. Contradiction.

(←) Let X' be a set of atoms in σ such that $\phi(X')$ is an answer set of P, and $P \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup R_A \cup C$ has no answer set. By Proposition 1, X' is a candidate answer set of Π.

Assume for the sake of contradiction that there is a candidate answer set X of Π that is s-preferred to X' according to LPOD. By Proposition 1, $\phi(X)$ is an answer set of P. We will prove that $P \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup R_A \cup C$ has some answer set S, which contradicts with bullet **(d)**. Since $\phi(X)$ is an answer set of P, it is sufficient to prove $H_{\phi(X)} \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup C$ has some answer set S, where $H_{\phi(X)}$ reifies the atoms in $\phi(X)$ into the form of $holds(\cdot)$.

First, let Π_{cur} be $H_{\phi(X)} \cup F_s \cup H'_{\phi(X')}$, and consider the answer set of Π_{cur}. Let $a(H_{\phi(X)})$ be $\{holds(a) \mid a \in \phi(X)\}$, let $a(H'_{\phi(X')})$ be $\{holds'(a) \mid a \in \phi(X')\}$, and let $a(F_s)$ denote all the atoms in F_s. It is obvious that $S_1 = a(H_{\phi(X)}) \cup a(F_s) \cup a(H'_{\phi(X')})$ is the only answer set of Π_{cur}.

Second, let's include rule (5) and rule (6) in C into Π_{cur}. By Lemma 2, $S_2 = S_1 \cup \{holds(neg(body_r)) \mid r \in \{1,\ldots,m\}, holds(body_r) \notin a(H_{\phi(X)})\} \cup \{holds'(neg(body_r)) \mid r \in \{1,\ldots,m\}, holds'(body_r) \notin a(H'_{\phi(X')})\}$ is the only answer set of Π_{cur}.

Third, let's include Deg (i.e., rule (13) and rule (14)) into Π_{cur}. By Lemma 2, $S_3 = S_2 \cup \{degree(R, D) \mid S_2$ satisfies the body of rule (13)$\} \cup \{degree'(R, D) \mid S_2$ satisfies the body of rule (14)$\}$ is the only answer set of Π_{cur}. Indeed, analogous to bullet 6 and bullet 7 in the previous direction ($\rightarrow$) of the proof, S_3 satisfies $degree(R, D)$ iff X satisfies LPOD rule R to degree D; S_3 satisfies $degree'(R, D)$ iff X' satisfies LPOD rule R to degree D. In other words, $S_3 = S_2 \cup \{degree(r, d) \mid r \in \{1,\ldots,m\}$, X satisfies rule r to degree $d\} \cup \{degree'(r, d) \mid r \in \{1,\ldots,m\}, X'$ satisfies rule r to degree $d\}$ is the only answer set of Π_{cur}.

Fourth, let's include the rules in E_{t_s} (i.e., $E_{lpod(s)}$ in Section 3.3) into Π_{cur} for each preference criterion $s \in \{i, p, ps\}$. The proof for the case when s is c is not shown due to the limit of space.

- **Inclusion-Preferred** Let's include the first rule (17) in $E_{lpod(i)}$ into Π_{cur}. Since X is i-preferred to X' according to LPOD, there exists an integer d such that
  - the set of LPOD rules that are satisfied to degree d by X' is a proper subset of the set of LPOD rules that are satisfied to degree d by X, and
  - for any degree j such that $j < d$, the set of LPOD rules that are satisfied to degree j by X' is a subset of the set of LPOD rules that are satisfied to degree j by X.

  In other words,there exists an integer d such that
  - there exists at least one LPOD rule j that is satisfied by X to degree d but is not satisfied by X' to degree d, and for any LPOD rule j', if it is satisfied by X' to degree d, it must be satisfied by X to degree d, and
  - for any degree j such that $j < d$, if an LPOD rule is satisfied by X' to degree j, it must by satisfied by X to degree j.

  Since S_3 satisfies $degree(R, D)$ iff X satisfies LPOD rule R to degree D, and S_3 satisfies $degree'(R, D)$ iff X' satisfies LPOD rule R to degree D, there exists an integer d such that

  (i) there exists at least one LPOD rule j such that $S_3 \models degree(j, d)$ and $S_3 \not\models degree'(j, d)$, and for any LPOD rule j', if $S_3 \models degree'(j', d)$, then $S_3 \models degree(j', d)$, and

  (ii) for any LPOD rule $r \in \{1,\ldots,m\}$ and any degree j such that $j < d$, if $S_3 \models degree'(r, j)$, then $S_3 \models degree(r, j)$

  Since S_3 satisfies all atoms in $a(F_s)$, according to the translation, S_3 satisfies $optimize(i)$ an $preference(i, lpod(i))$. By rule (17) and bullet **(i)** above, S_3 satisfies the body of rule (17) (where S is i and D is d). By Lemma 2, $S_3 \cup \{prf2degree(i, d)\}$ is an answer set of Π_{cur}.

  Let's include the second rule (18) in $E_{lpod(i)}$ into Π_{cur}. By Lemma 2 and bullet **(ii)** above, $S_4 = S_3 \cup \{prf2degree(i, d), better(i)\}$ is an answer set of Π_{cur}.

  Let's include rule (4) into Π_{cur}. It is easy to see that S_4 satisfies rule (4). By Lemma 1, S_4 is an answer set of $H_{\phi(X)} \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup C$. Contradiction.

- **Pareto-Preferred** Let's include the first rule (19) in $E_{lpod(p)}$ into Π_{cur}. Since X is p-preferred to X' according to LPOD, there is an LPOD rule that is satisfied to a lower degree in X than in X', and there is no rule that is satisfied to a lower degree in X' than in X. By rule (19), S_3 does not satisfy the body of rule (19) (where S is p). By Lemma 2, S_3 is an answer set of Π_{cur}.

  Let's include the second rule (20) in $E_{lpod(p)}$ into Π_{cur}. By Lemma 2, $S_4 = S_3 \cup \{better(p)\}$ is an answer set of Π_{cur}.

  Let's include rule (4) into Π_{cur}. It is easy to see that S_4 satisfies rule (4). By Lemma 1, S_4 is an answer set of $H_{\phi(X)} \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup C$. Contradiction.

- **Penalty-Sum-Preferred** Let's include $E_{lpod(ps)}$ into Π_{cur}. Since X is ps-preferred to X' according to LPOD, the sum of the satisfaction degrees of all rules is smaller in X than in X'. Thus $\sum_{R:S_3 \models degree(R,D)} D < \sum_{R:S_3 \models degree'(R,D)} D$. By Lemma 2, $S_4 = S_3 \cup \{better(ps)\}$ is an answer set of Π_{cur}.

  Let's include rule (4) into Π_{cur}. It is easy to see that S_4 satisfies rule (4). By Lemma 1, S_4 is an answer set of $H_{\phi(X)} \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup C$. Contradiction.