Computing Logic Programs with Ordered Disjunction Using *asprin*

Joohyung Lee and Zhun Yang
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, AZ, 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 of LPOD called LPOD2ASPRIN based on the encoding. Unlike the known method that applies only to a fragment of LPOD, our translation is general, direct, and simpler. It also leads to more efficient computation of LPOD using *asprin*.

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 candidate 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 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.

2 Review of LPOD and *asprin*

2.1 Review: LPOD

A (propositional) LPOD Π is $\Pi_{reg} \cup \Pi_{od}$, where its *regular part* Π_{reg} consists of usual ASP rules *Head* $\leftarrow$ *Body*, and its *ordered disjunction part* Π_{od} consists of *LPOD rules* of the form

$$C^1 \times \cdots \times C^n \leftarrow \mathit{Body} \qquad (1)$$

in which C^i are atoms, n is at least 2, and *Body* is a conjunction of atoms possibly preceded by *not*. Rule (1) says "when *Body* is true, if possible then C^1; if C^1 is not possible then C^2; ...; if all of $C^1, \dots, C^{n-1}$ are not possible then C^n."

Candidate answer sets are defined via the notion of split programs in (Brewka 2002). Based on the notion of a *satisfaction degree* of a rule by a candidate answer set, Brewka (2005) defined *preferred answer sets* under four preference criteria: cardinality-preferred (c), inclusion-preferred (i), Pareto-preferred (p), penalty-sum-preferred (ps) answer sets. We refer the reader to that paper for the details.

A set X of atoms is an *s-preferred* ($s \in \{c, i, p, ps\}$) *answer set* of an LPOD Π if X is a candidate answer set of Π and there is no candidate answer set X' of Π such that X' is preferred to X w.r.t. preference criterion s.

2.2 Review: *asprin*

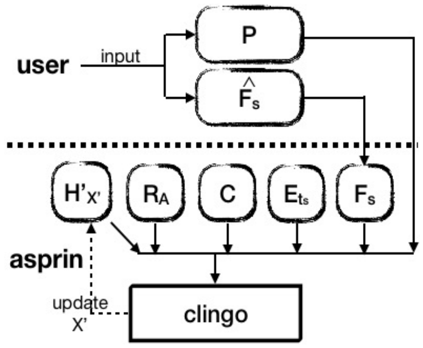

Figure 1: *asprin* Framework

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 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[1] *preference statement* of the form
$$\#preference(s,t)\{e_1; \dots ; e_n\} \tag{3}$$
where $n \geq 0$, and s is the identifier name of a preference relation, and t is the type of the preference relation. 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*).[2] Intuitively, each index $1, \dots, m$ gives the rank of the corresponding literal. The preference statement (3) declares a preference relation named s. Each preference element in $\{e_1, \dots, e_n\}$ gives a ranking of a set of literals while preference type t determines in which case one candidate answer set is better than another based on the ranking. The optimization directive (2) specifies which preference relation *asprin* uses for optimization.

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". [3]

4. E_{t_s} is the *preference encoding* for t_s, where t_s is the type of the preference relation 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 $holds'(\cdot)$.[4]

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.
$$\bot \leftarrow \textit{not } better(S), optimize(S). \tag{4}$$
$$holds(neg(A)) \leftarrow \textit{not } holds(A), \quad preference(\_,\_,\_,for(neg(A)),\_). \tag{5}$$
$$holds'(neg(A)) \leftarrow \textit{not } holds'(A), \quad preference(\_,\_,\_,for(neg(A)),\_). \tag{6}$$
Rule (4) instructs the *asprin* system to find an answer set X that is better than X' according to the preference relation 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 relation s declared in $\hat{F}_s$.

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

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

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

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

3 Representing LPOD in *asprin*

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

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

where $1, \dots, m$ 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 in Section 2.2.

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

$$\mathit{body}_i \leftarrow \mathit{Body}_i \tag{8}$$

$$\{C_i^1\} \leftarrow \mathit{body}_i \tag{9}$$

$$\cdots$$

$$\{C_i^{n_i-1}\} \leftarrow \mathit{body}_i, \mathit{not}\ C_i^1, \dots, \mathit{not}\ C_i^{n_i-2} \tag{10}$$

$$C_i^{n_i} \leftarrow \mathit{body}_i, \mathit{not}\ C_i^1, \dots, \mathit{not}\ C_i^{n_i-1} \tag{11}$$

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 ($j \in \{1, \dots, k-1\}$), 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 $j \in \{1, \dots, n_i - 1\}$, 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 1 *Consider the following LPOD Π 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.*

$$\begin{array}{ll} close \times med \times far \times tooFar. & \leftarrow hotel(2),\ \mathit{not}\ med. \\ star4 \times star3 \times star2. & \leftarrow hotel(2),\ \mathit{not}\ star3. \\ 1\{hotel(X) : X = 1..3\}1. & \leftarrow hotel(3),\ \mathit{not}\ tooFar. \\ \leftarrow hotel(1),\ \mathit{not}\ close. & \leftarrow hotel(3),\ \mathit{not}\ star4. \\ \leftarrow hotel(1),\ \mathit{not}\ star2. & \end{array}$$

The P-component of asprin obtained from this LPOD Π is the following.

```
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

$$\begin{array}{lllll} \{hotel(1), & close, & star2, & body_1, & body_2\} \\ \{hotel(2), & med, & star3, & body_1, & body_2\} \\ \{hotel(3), & tooFar, & star4, & body_1, & body_2\} \end{array}$$

which are exactly the unions of the candidate answer sets of Π and $\{body_1, body_2\}$.

3.2 Preference Specification $\hat{F}_s$

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

$$\#optimize(s)$$

and a preference statement

$$\begin{array}{l} \#preference(s, lpod(s))\ \{ \\ \quad \mathit{not}\ body_1 > C_1^1 > \cdots > C_1^{n_1}; \\ \quad\qquad \cdots \\ \quad \mathit{not}\ body_m > C_m^1 > \cdots > C_m^{n_m} \qquad \} \end{array} \tag{12}$$

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 to specify satisfaction degrees. 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 is 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 1 (Continued) For LPOD Π which contains LPOD rules

$$\begin{array}{l} close \times med \times far \times tooFar \\ star4 \times star3 \times star2 \end{array}$$

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'.

Deg consists of the following two rules.

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

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

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 relation 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:

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

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

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 $J \in \{1, \dots, D-1\}$.

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

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

$$\begin{aligned} better(S) \leftarrow\ & preference(S, lpod(i)),\\ & prf2degree(S, D),\\ & degree(R, J) : degree'(R, J), J < D. \end{aligned} \tag{18}$$

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:

$$\begin{aligned} equ(S) \leftarrow\ & optimize(S), preference(S, lpod(p)),\\ & D1 = D2 : degree(R, D1), degree'(R, D2). \end{aligned} \tag{19}$$

$$\begin{aligned} better(S) \leftarrow\ & optimize(S), preference(S, lpod(p)),\\ & not\ equ(S),\\ & D1 \le D2 : degree(R, D1), degree'(R, D2). \end{aligned} \tag{20}$$

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:

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

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.*

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 $body_i$.

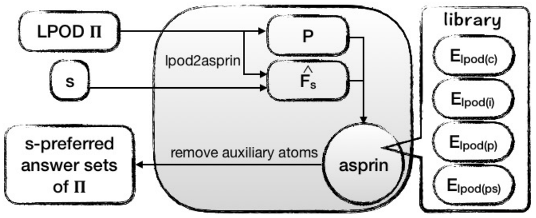

Figure 2: LPOD2ASPRIN System Overview

The LPOD2ASPRIN system homepage is

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

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

5 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 and then applying preference criteria unlike the multiple calls to CLINGO in $asprin$.

$asprin$ 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 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. *Theory and Practice of Logic Programming (TPLP)*, 18(3-4), 589-606. doi:10.1017/S1471068418000315.