Extended Ordered Disjunction programs to model Preferences

Mauricio Osorio[1], Claudia Zepeda[2], and José Luis Carballido[2]

[1] Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@googlemail.com
[2] Benemérita Universidad Atónoma de Puebla
Facultad de Ciencias de la Computación,
Puebla, Puebla, México
{czepedac,jlcarballido7}@gmail.com

(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract In [3] Brewka proposed the connective of ordered disjunction to express default knowledge with knowledge about preferences. Later, the authors of [10] define the logic programs with extended ordered disjunction that extends ordered disjunction programs to a wider class of logic programs. Here, we propose to specify a preference ordering among the answer sets of a program with respect to an ordered list of atoms using a particular kind of logic programs with extended ordered disjunction.

Keywords: Answer set programming, preferences, Extended Ordered Disjunction.

1 Introduction

Preferences are useful when we need to find feasible solutions that most satisfy a set of additional requirements of a given problem. In [3] is defined the connective $\times$, called *ordered disjunction*, to express default knowledge with knowledge about preferences in a simple way. While the disjunctive clause $a \vee b$ is satisfied equally by either a or b, to satisfy the ordered disjunctive clause $a \times b$, a will be preferred to b, i.e., a model containing a will have a better *satisfaction degree* than a model that contains b but does not contain a. For example, the natural language statement *"I prefer travel by bus to walk"* can be expressed as $travelBus \times walk$ and a model containing $travelBus$ will be preferred to a model that contains $walk$. Later, the authors of [10] define the *logic programs with extended ordered disjunction* (**ELPOD**) that extends ordered disjunction programs to a wider class of logic programs. There have been other extensions for the operator $\times$ [2,4]. However, while the extension introduced in [10] is in the context of Answer Sets, the extension introduced in [2] for the operator $\times$ is in a different context. Moreover, the extension defined in [4] is in the context of Answer Sets and the authors present interesting examples, however they propose a different methodology to define it w.r.t. the methodology proposed in [10].

It is worth mentioning that if we use a ordered disjunction rule, as is defined in [3], to *specify a preference ordering among the answer sets of a program with respect to an ordered set of atoms* then, the preferred answer set does not corresponds to the answer set that we would expect. For instance, if $P_1 = \{a \leftarrow, \quad b \leftarrow \neg c, \quad c \leftarrow \neg b, \quad d \leftarrow \neg a, \quad f \leftarrow c, \neg a, \quad e \leftarrow b, \neg a\}$, then the answer sets of this program are $\{a, b\}$ and $\{a, c\}$. Now, if we want to specify a preference ordering among the answer sets of the program with respect to the ordered set of atoms, such as $[f, c]$ then, we could consider to add to the program P the standard ordered disjunction rule $\{f \times c\}$ that stands for "if f is possible then f otherwise c". Thinking in a preference sense, the intuition indicates that the inclusion-preferred answer sets of $P \cup \{f \times c\}$ should be $\{a, c\}$. However, according to [3], we obtain two inclusion-preferred answer sets $\{a, b, f\}$ and $\{a, c, f\}$.

In this paper we propose to specify a preference ordering among the answer sets of a program with respect to an ordered list of atoms using a particular set of ELPOD's. We propose to use double default negation in each atom of the ordered rule that represents the mentioned list of atoms. Finally, we show how to compute the preferred answer sets for ELPOD's using `psmodels`.

The structure of our paper is as follows. In Section 2 describes the general background needed for reading the paper, including the definition of Answer Sets and ELPOD's. In Section 3, we present how to use double default negation to to obtain the preferred answer sets of a logic program. In Section 4 we propose how to compute the preferred answer sets. On Section 5 we present our conclusions.

2 Background

In this section we summarize some basic concepts and definitions used to understand this paper.

2.1 Logic programs and semantics

We want to stress the fact that in our approach, *a program is interpreted as a propositional theory* and that the only negation used is *default negation*. For readers not familiar with this approach, we recommend [11] for further reading. Hence, we will restrict our discussion to propositional programs.

We shall use *the language of propositional logic* in the usual way, using: propositional symbols: $p, q, \ldots$; propositional connectives: $\wedge, \vee, \rightarrow, \bot$; and auxiliary symbols: (,). A *well formed propositional formula* is inductively defined as usual in logic. An *atom* is a propositional symbol. A *signature* $\mathcal{L}$ is a finite set of atoms. The *signature of a program* P, denoted as $\mathcal{L}_P$, is the set of atoms that occur in P. A *literal* is either an atom p (a positive literal) or the negation of an atom $\neg p$ (a negative literal). A *negated literal* is the negation sign $\neg$ followed by any literal, i.e. $\neg p$ or $\neg\neg p$. We remark that the only negation used in this work is *default negation* and it is represented by the symbol $\neg$. It is worth mentioning that we always can handle the other negation called *classical* or even *strong* negation, denoted by $-$, by transforming [6] the atoms with classical negation as follows: each atoms with classical negation, $-a$, that occurs in a formula is replaced by a new atom, a', and the rule $\neg(a \wedge a')$ is added. In particular, $f \rightarrow \bot$ is called *constraint* and it is also denoted as $\leftarrow f$. A *regular theory* or *logic program* is just a finite set of clauses, it can be called just *theory* or *program* where no ambiguity arises.

2.2 Answer sets

In some definitions we use Heyting's *intuitionistic* logic [12], which will be denoted by the subscript I. For a given set of atoms M and a program P, we will write $P \vdash_{\mathrm{I}} M$ to abbreviate $P \vdash_{\mathrm{I}} a$ for all $a \in M$. For a given set of atoms M and a program P, we will write $P \Vdash_{\mathrm{I}} M$ to denote that: $P \vdash_{\mathrm{I}} M$; and P is consistent w.r.t. logic I.

Now we define answer sets (or stable models) of logic programs. The stable model semantics was first defined in terms of the so called *Gelfond-Lifschitz reduction* [5] and it is usually studied in the context of syntax dependent transformations on programs. We follow an alternative approach started by Pearce [11] and also studied by Osorio et.al. [8]. This approach characterizes the answer sets of a propositional theory in terms of intuitionistic logic and it is presented in the following theorem. The notation is based on [8].

Theorem 1. *Let P be any theory and M a set of atoms. M is an answer set for P iff $P \cup \neg(\mathcal{L}_P \setminus M) \cup \neg\neg M \Vdash_{\mathrm{I}} M$.*

For instance, if we consider the program $P_1 = \{\ p \leftarrow \neg q, \quad q \leftarrow\ \}$, then we can verify that $M = \{q\}$ is an answer set of P_1 and that $P_1 \cup \{\neg p\} \cup \{\neg\neg q\} \Vdash_{\mathrm{I}} \{q\}$.

2.3 ELPOD's

In [3] the head of ordered disjunction rules is defined in terms of ground literals. In [10] the head and the body of extended ordered disjunction rules are defined in terms of well formed propositional formulas. The authors of [10] consider that a broader syntax for rules could give some benefits. For example, the use of nested expressions could simplify the task of writing logic programs and improve their readability since, it could allows us to write more concise rules and in a more natural way. Other examples are presented in [7,9].

In the following example of an ELPOD, we can see that the broader syntax results more natural, direct and intuitive: I prefer *travel by airplane* to either *travel by bus* other *travel by train*, but between *travel by bus* and *travel by train* I don't have any particular preference. Then, using an ELPOD we could just write $travelByAirplan \times (travelByBus \vee travelByTrain)$.

Definition 1. *A* logic programs with extended ordered disjunction (ELPOD) *is either a well formed propositional formula, or a formula of the form:*

$$f_1 \times \ldots \times f_n \leftarrow g \tag{1}$$

where $f_1, \ldots, f_n, g$ are well formed propositional formulas. An extended ordered disjunction program *is a finite set of extended ordered disjunction rules.*

The formulas $f_1 \ldots f_n$ are usually called the choices of a rule and their intuitive reading is as follows: if the body is true and f_1 is possible, then f_1; if f_1 is not possible, then f_2; ...; if none of $f_1, \ldots, f_{n-1}$ is possible then f_n.

The particular case where all f_i are literals and g is a conjunction of literals corresponds to the original ordered disjunction programs as presented by Brewka in [3], and as we indicated before we call them *standard ordered disjunction programs*. We recall that Brewka's ordered disjunction programs use the strong negation connective. Here we will consider only one type of negation (default negation) but this does not affect the results given in [3]. If additionally $n = 0$ the rule is a constraint, i.e, $\bot \leftarrow g$. If $n = 1$ it is an extended rule and if $g = \top$ the rule is a fact and can be written as $f_1 \times \ldots \times f_n$.

Example 1. A person should travel from his/her home to school in winter. This person prefers to travel by bus and to drink tea inside the bus rather than travel by bicycle. Additionally, he/she prefers to travel by bicycle rather than walk. This person also should consider that part of the path from his/her home to school can become blocked by snow in winter. Then, we can model this situation considering the following ELPOD P:
$winter.$
$(travelBus \wedge drinkTea) \times travelBicycle \times walk \leftarrow winter, \neg pathBlocked.$

Now, we present the semantics of ELPOD's. Most of the definitions presented here are taken from [3]. The relevant difference is the satisfaction degree. The reader may see that the satisfaction degree as defined here is just a straightforward generalization of Brewka's definition, according to our notation and Theorem 1. Hence, standard ordered programs are special cases of ELPOD's, thus all results hold for this restricted class as well.

Definition 2. *Let $r := f_1 \times \ldots \times f_n \leftarrow g$ be an extended ordered disjuntion rule. For $1 \leq k \leq n$ the* k-th option of r *is defined as follows:*

$$r^k := f_k \leftarrow g, \neg f_1, \ldots \neg f_{k-1}$$

Definition 3. *Let P be an ELPOD. P' is a* split program of P *if it is obtained by replacing each rule $r := f_1 \times \ldots \times f_n \leftarrow g$ in P by one of its options $r^1, \ldots, r^k$. M is an answer set of P iff it is an answer set*[1] *of a split program P' of P. Let M be an answer set of P and let $r := f_1 \times \ldots \times f_n \leftarrow g$ be a rule of P. We define the* satisfaction degree of r with respect to M, *denoted by $deg_M(r)$, as follows:*

– *if $M \cup \neg(\mathcal{L}_P \setminus M) \not\vdash_I g$, then $deg_M(r) = 1$.*
– *if $M \cup \neg(\mathcal{L}_P \setminus M) \vdash_I g$ then $deg_M(r) = min_{1 \leq i \leq n} \{i \mid M \cup \neg(\mathcal{L}_P \setminus M) \vdash_I f_i\}$.*

[1] Note that since we are not considering strong negation, there is no possibility of having inconsistent answer sets.

Now we present the relationship between the answer sets of an ELPOD and the satisfaction degree.

Theorem 2. *[1] Let P be an ELPOD. If M is an answer set of P then M satisfies all the rules in P to some degree.*

Definition 4. *[3] Let P be an ELPOD and M a set of literals. We define:* $S^i_M(P) = \{r \in P \mid deg_M(r) = i\}$.

Example 2. Let us consider the ELPOD P from Example 1. The split programs and answer sets of P are:

P' :
$winter.$
$(travelBus \wedge drinkTea) \leftarrow winter, \neg pathBlocked.$

P'' :
$winter.$
$travelBicycle \leftarrow$
$winter, \neg pathBlocked, \neg(travelBus \wedge drinkTea).$

P''' :
$winter.$
$walk \leftarrow winter, \neg pathBlocked,$
$\neg(travelBus \wedge drinkTea), \neg travelBicycle.$

$M_1 = \{winter, (travelBus \wedge drinkTea)\}$, $M_2 = \{winter, travelBicycle\}$ and $M_3 = \{winter, walk\}$ are answer sets of P. If r_1 and r_2 denote the first and second rule of program P respectively, then we have the following satisfaction degrees of each rule: $deg_{M_1}(r_1) = 1$, $deg_{M_1}(r_2) = 1$, $deg_{M_2}(r_1) = 1$, $deg_{M_2}(r_2) = 2$, $deg_{M_3}(r_1) = 1$, $deg_{M_3}(r_2) = 3$. Finally we have that, $S^1_{M_1}(P) = \{r_1, r_2\}$, $S^2_{M_1}(P) = \{\}$, $S^3_{M_1}(P) = \{\}$, $S^1_{M_2}(P) = \{r_1\}$, $S^2_{M_2}(P) = \{r_2\}$, $S^3_{M_2}(P) = \{\}$, $S^1_{M_3}(P) = \{r_1\}$, $S^2_{M_3}(P) = \{\}$, $S^3_{M_3}(P) = \{r_2\}$

Now introduce two different types of preference relations among the answer sets of an ELPOD: inclusion based preference and cardinality based preference.

Definition 5. *[3] Let M and N be answer sets of an ELPOD P. We say that M is* inclusion preferred *to N, denoted as $M >_i N$, iff there is an i such that $S^i_N(P) \subset S^i_M(P)$ and for all $j < i$, $S^j_M(P) = S^j_N(P)$. We say that M is* cardinality preferred *to N, denoted as $M >_c N$, iff there is an i such that $\left|S^i_M(P)\right| > \left|S^i_N(P)\right|$ and for all $j < i$, $\left|S^j_M(P)\right| = \left|S^j_N(P)\right|$.*

Example 3. If we consider the ELPOD P from Example 1 we can verify that $M_1 >_i M_2$ and $M_2 >_c M_3$.

Definition 6. *[3] Let M be an answer set of an ELPOD P. M is an* inclusion preferred answer set *of P if there is no answer set M' of P, $M \neq M'$, such that $M' >_i M$. M is a* cardinality preferred answer set *of P if there is no M' answer set of P, $M \neq M'$, such that $M' >_c M$.*

Example 4. If we consider the ELPOD P from Example 1, we can verify that M_1 is the inclusion preferred answer set of P and also the cardinality preferred answer set of P.

3 Expressing preferences using double negation

In order to specify a preference ordering among the answer sets of a program with respect to an ordered set of atoms, in this section we propose to use a particular set of ELPOD's. Specifically, we propose to add to the original program an extended ordered rule such that this rule is defined using the ordered set of atoms and each atom has *double default negation*.

Formally, as we defined in Background Section, an atom with double negation corresponds to a *negated negative literal* where the only negation used is *default negation*. Let us consider $\neg\neg a$ where a is an atom. Since $\neg\neg a$ is equivalent to the restriction $\leftarrow \neg a$, the intuition behind $\neg\neg a$ is to indicate that it is desirable that a holds in the model of a program.

Hence, if we consider again the previous program P_1 of the Introduction section and we want to to specify a preference ordering among the answer sets of P with respect to the ordered set of atoms $[f, c]$ we have to do the following: First we define the extended ordered disjunction rule using the ordered set of atoms $[f, c]$ such that each atom has *double default negation*, i.e, we define $\{\neg\neg f \times \neg\neg c\}$. Then, we add this extended ordered disjunction rule to the original program P_1, i.e., we define the following extended ordered disjunction program: $P_1 \cup \{\neg\neg f \times \neg\neg c\}$. It is possible to verify that we obtain the desired inclusion-preferred answer set $\{a, c\}$ from this ELPOD.

We explained that the intuition behind an extended ordered rule using negated negative literals is to indicate that we want to specify a preference ordering among the answer sets of a program with respect to an ordered set of atoms. However, in case that the answer sets of the program do not contain any of the atoms in the given ordered set of atoms then the extended ordered rule must allow to obtain all the answer sets of the program. In order to obtain all the answer sets of the program we propose to add a positive literal to the end of the extended ordered rule that we defined as we explained above. This positive literal must be an atom that does not occur in the original program, such as the atom all_pref. For instance, let us consider again the program P_1 that has the answer sets $\{a, b\}$ and $\{a, c\}$. If we want to specify a preference ordering among the answer sets of this program with respect to the ordered set of atoms $C = [f, e]$ then we define the extended ordered rule as we explained above but we add the atom all_pref at the end of the rule, i.e., $\neg\neg f \times \neg\neg e \times all\_pref$. The new extended ordered rule indicates that we prefer the answer sets where f holds to the answer sets where e holds and if there is no answer sets of the program where f or e holds then all answer sets are preferable. It is worth to mentioning that in case that there is no answer sets of the program where f or e holds then all the answer sets will contain the atom all_pref. Hence, we have the following ELPOD, $P_2 = P_1 \cup \{\neg\neg f \times \neg\neg e \times all\_pref\}$. As we expected, we obtain the inclusion-preferred answer sets: $\{a, c, all\_pref\}$ and $\{a, b, all\_pref\}$ since there is no answer sets of the program containing f or e.

The following short examples also show the role of negated negative literals in an extended ordered program.

Example 5. Let us consider the ordered program $a \times b$ as defined in [3]. We can verify that its inclusion preferred answer set is $\{a\}$, since ordered disjunction corresponds to a disjunction where an ordering is defined. However, the ELPOD $\neg\neg a \times \neg\neg b$ has no answer sets since the extended ordered rule only indicates that we prefer the answer sets containing a to the answer sets containing b but there is no answer sets.

Example 6. Let us consider the ordered program $\{b \leftarrow \neg a, \;\; a \times b\}$. We can verify that the inclusion preferred answer set of it is $\{a\}$ since ordered disjunction corresponds to a disjunction where an ordering is defined. However, the ELPOD $\{b \leftarrow \neg a, \;\; \neg\neg a \times \neg\neg b\}$ only has $\{b\}$ as its inclusion preferred answer set, since the extended ordered rule only indicates that we prefer the answer sets containing a to the answer sets containing b and program $b \leftarrow \neg a$ has only the answer set $\{b\}$.

Now, we formalize our previous discussion about the specification of an ordering among the answer sets of a normal program with respect to an ordered set of atoms using an extended ordered program.

We start introducing a definition and a proposition that allows us to define the most preferred answer set with respect to a program and an ordered set of atoms.

Definition 7. *Let P be a normal program and let M and N be two answer sets of P. Let $C = [c_1, c_2, \ldots, c_n]$ be an ordered set of atoms. The answer set M is* preferred to the answer set N with respect to C *(denoted as $M <_C N$) if*

1. *there exists $i = min(1 \leq k \leq n)$ such that $M \cup \neg(\mathcal{L}_P \setminus M) \vdash_{\mathrm{I}} c_i$ and $N \cup \neg(\mathcal{L}_P \setminus N) \nvdash_{\mathrm{I}} c_i$, and*
2. *for all $j < i$, $M \cup \neg(\mathcal{L}_P \setminus M) \vdash_{\mathrm{I}} c_j$ and $N \cup \neg(\mathcal{L}_P \setminus N) \vdash_{\mathrm{I}} c_j$ or $M \cup \neg(\mathcal{L}_P \setminus M) \nvdash_{\mathrm{I}} c_j$ and $N \cup \neg(\mathcal{L}_P \setminus N) \nvdash_{\mathrm{I}} c_j$.*

Proposition 1. *Let C be an ordered set of atoms; then $<_C$ is a partial order.*

Given an ordered list of atoms C, an answer set M of a normal program P *is most preferred with respect to C* if there is no other answer set N of P that is preferred to M with respect to C.

Example 7. Let $P = \{a \leftarrow, \;\; c \leftarrow \neg b, \;\; b \leftarrow \neg c\}$ and $C = [b, c]$ be an ordered list of atoms. We can verify that P has two answer sets, $\{a, b\}$ and $\{a, c\}$. We also can verify that the answer set $\{a, b\}$ *is preferred to the answer set* $\{a, c\}$ *with respect to C*, i.e., $\{a, b\} <_C \{a, c\}$ and that $\{a, b\}$ is also *the most preferred answer set.*

Now, we define the extended ordered rule that we join to the original normal program in order to obtain a particular ELPOD. From this last ELPOD we obtain the most preferred answer sets of the normal program with respect to the given ordered set of atoms C.

Definition 8. *Let P be a normal program and $C = [c_1, c_2, \ldots, c_n]$ be an ordered set of atoms such that $C \subseteq \mathcal{L}_P$. We define an* extended ordered rule defined from C*, denoted as r_C, as follows: $r_C := \neg\neg c_1 \times \neg\neg c_2 \times \ldots \times \neg\neg c_n \times all\_pref$ such that $all\_pref \notin \mathcal{L}_P$.*

Example 8. Let us consider again program P from Example 7, and also the ordered set of atoms $C = [b, c]$. Then, the *extended ordered rule defined from* C, denoted as r_C is the following: $r_C := \neg\neg b \times \neg\neg c \times all\_pref$.

Here, we present Lemma 1 that allows us to obtain the most preferred answer set of a normal program with respect to an ordered set of atoms based on a particular extended ordered program.

Lemma 1. *Let P be a normal program and let $C = [c_1, c_2, \ldots, c_n]$ be an ordered list of atoms such that $C \subseteq \mathcal{L}_P$. Let r_C be the extended ordered rule defined from C. Then M is an inclusion preferred answer set of $P \cup r_C$ iff $(M \cap \mathcal{L}_P)$ is the most preferred answer set with respect to $C \cup P$.*

Example 9. Let us consider again program P from Example 7 and also the ordered list of atoms $C = [b, c]$. Then, $P \cup r_C = \{a \leftarrow, \quad c \leftarrow \neg b, \quad b \leftarrow \neg c, \quad \neg\neg b \times \neg\neg c \times all\_pref\}$. We can verify that $\{a, b\}$ is *the most preferred answer set of P with respect to C* and it is also an inclusion preferred answer set of $P \cup r_C$.

In the following section, we show how to compute the preferred answer sets for ELPOD's using `psmodels` [3].

4 Computing preferred answer sets for extended ordered programs

`Psmodels`[2] is a software implementation useful to obtain the inclusion preferred answer sets for the ordered disjuntion programs as defined in [3]. Hence, it is not possible to obtain the inclusion preferred answer sets for extended ordered programs using `Psmodels`. The reason is that the definition given by Brewka for ordered disjunction has syntactical restrictions. However, in particular when this program has extended ordered rules using negated negative literals, we can translate easily this program to a standard ordered disjunction program (as defined by Brewka in [3]) and in this way to use `Psmodels` to obtain the preferred answer sets.

[2] http://www.tcs.hut.fi/Software/smodels/priority/

Lemma 2 allows us to translate an extended ordered program that results from joining a normal program with an extended ordered disjunction rule with negated negative literals to a standard ordered program.

In the following definition and lemma the atoms $a^\bullet$, a°, are atoms that do not occur in the original program P.

Definition 9. *Let $\neg\neg a$ be a negated negative literal. We define the associated set of $\neg\neg a$ as follows:*
$R(\neg\neg a) := \{ \ \leftarrow \neg a, a^\bullet \ , \quad a^\bullet \leftarrow \neg a^\circ \ ,$
$\quad a^\circ \leftarrow \neg a \ , \quad \leftarrow a, a^\circ \ \}.$

Lemma 2. *Let P be a normal program and let $C = [c_1, c_2, \ldots, c_n]$ be an ordered set of atoms such that $C \subseteq \mathcal{L}_P$. Let $C^\bullet = \{c_1^\bullet, c_2^\bullet, \ldots, c_n^\bullet\}$ be a set of atoms such that $C^\bullet \cap \mathcal{L}_P = \emptyset$. Let r_C be the extended ordered rule defined from C. Let $r_C^\bullet$ be the following ordered rule $c_1^\bullet \times c_2^\bullet \times \ldots \times c_n^\bullet \times all\_pref$. Let $A = \bigcup_{c_i \in C \ and \ 1 \le i \le n} R(\neg\neg c_i)$. Then M is an inclusion preferred answer set of $P \cup \{r_C^\bullet\} \cup A$ iff $M \cap \mathcal{L}_P$ is an inclusion preferred answer set of $P \cup \{r_C\}$.*

Example 10. Let us consider again the program P_1 at the beginning of this Section 3, and the set of atoms $C = [f, c]$ then $r_C = \neg\neg f \times \neg\neg c \times all\_pref$,
$A = \{\leftarrow \neg f, f^\bullet, \quad f^\bullet \leftarrow \neg f^\circ, \quad f^\circ \leftarrow \neg f, \quad \leftarrow f, f^\circ,$
$\leftarrow \neg c, c^\bullet, \quad c^\bullet \leftarrow \neg c^\circ, \quad c^\circ \leftarrow \neg c, \quad \leftarrow c, c^\circ\}$ and $r_C^\bullet = \{f^\bullet \times c^\bullet \times all\_pref\}$.

Then, by running `psmodels` we obtain the following inclusion preferred answer set of the standard ordered program $P \cup r_C^\bullet \cup A$: $\{a, c, c^\bullet, f^\circ\}$. Finally, we can see that the intersection of the answer set with $\mathcal{L}_P$ corresponds to the inclusion preferred answer sets of the original extended ordered program $P \cup r_C$ as we described before: $\{a, c\}$.

5 Conclusions

We review the syntax and semantics of ELPOD. We explain how to specify and compute the preferred answer sets of a logic program with respect to a preference order. This order is expressed as an extended ordered disjunction rule using double default negation. Finally, we show how to compute the preferred answer sets for ELPOD's using `psmodels`.

References

1. G. Brewka. Logic Programming with Ordered Disjunction. In *Proceedings of the 18th National Conference on Artificial Intelligence, AAAI-2002.* Morgan Kaufmann, 2002.
2. G. Brewka, S. Benferhat, and D. L. Berre. Qualitative choice logic. *Artif. Intell.*, 157(1-2):203–237, 2004.
3. G. Brewka, I. Niemelä, and T. Syrjänen. Implementing Ordered Disjunction Using Answer Set Solvers for Normal Programs. In *Proceedings of the 8th European Workshop Logic in Artificial Inteligence JELIA 2002.* Springer, 2002.

4. R. Confalonieri and J. C. Nieves. Nested logic programs with ordered disjunction. In *Latin-American Workshop on Non-Monotonic Reasoning 2010 (LANMR10)*, volume 677, pages 55–66, 2010.
5. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
6. M. Gelfond and V. Lifschitz. Logic Programs with Classical Negation. In D. Warren and P. Szeredi, editors, *Proceedings of the 7th Int. Conf. on Logic Programming*, pages 579–597, Jerusalem, Israel, June 1990. MIT.
7. N. Leone and S. Perri. Parametric Connectives in Disjunctive Logic Programming. In *ASP03 Answer Set Programming: Advances in Theory and Implementation*, Messina, Sicily, Sept. 2003.
8. M. Osorio, J. A. Navarro, and J. Arrazola. Applications of Intuitionistic Logic in Answer Set Programming. *Theory and Practice of Logic Programming (TPLP)*, 4:325–354, May 2004.
9. M. Osorio and M. Ortiz. Embedded Implications and Minimality in ASP. In M. H. U. G. Dietmar Seipel and O. Bartenstein, editors, *15th International Conference on Applications of Declarative Programming and Knowledge Management. INAP 2004*, Postdam, Germany, Mar. 2004.
10. M. Osorio, M. Ortiz, and C. Zepeda. Using CR-rules for evacuation planning. In G. D. I. Luna, O. F. Chaves, and M. O. Galindo, editors, *IX Ibero-american Workshops on Artificial Inteligence*, pages 56–63, 1994.
11. D. Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.
12. D. van Dalen. *Logic and Structure*. Springer, Berlin, second edition, 1980.