Applications of preferences using Answer Set Programming

Claudia Zepeda[1,3],Mauricio Osorio[1], Juan Carlos Nieves[2], Christine Solnon[3], and David Sol[1]

[1] Universidad de las Américas, CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
{josorio,sc098382,sol}@mail.udlap.mx,
[2] Universitat Politecnica de Catalunya, Departament de Lleguatges i Sistemes Informatics, c/Jordi Girona 1-3, E08034, Barcelona, Spain
jcnieves@lsi.upc.edu,
[3] LIRIS UMR 5205 CNRS, Université Lyon 1 and INSA de Lyon, 43 bd du 11 novembre, 69622 Villeurbanne cedex, France
{claudia.zepeda,christine.solnon}@liris.cnrs.fr

Abstract. Preferences are useful when the space of feasible solutions of a given problem is dense but not all these feasible solutions are equivalent w.r.t. some additional requirements. In this case, the goal is to find feasible solutions that most satisfy these additional requirements. In order to represent preferences, in this paper we use an extension of ordered disjunction programs. Ordered disjunction is an approach based on answer sets that allows us to represent alternative, ranked options for a problem. Moreover, we give a brief overview of two real applications of extended ordered programs in two different domains. The first one is in planning: evacuation planning. The second one is in argumentation: organ transplantation. In particular, we show the role of negated negative literals in extended ordered programs to obtain the preferred solution of each application.
Key words: Preferences, Answer Set Programming, Ordered Disjunction Programs, Planning.

1 Introduction

Preferences are useful when the space of feasible solutions of a given problem is dense but not all these feasible solutions are equivalent w.r.t. some additional requirements. In this case, the goal is to find feasible solutions that most satisfy these additional requirements. In [3] Brewka introduced *logic programs with ordered disjunction (LPODs)* where the connective $\times$, called *ordered disjunction*, allows a natural and simple representation of preferences. However, if we only want to specify a preference ordering among the answer sets of a program with respect to an ordered list of atoms then ordered disjunction as defined by Brewka does not work since it corresponds to a disjunction where an ordering is defined. For instance, the answer sets of the program P defined as

$\{\ a \leftarrow .\quad b \leftarrow \neg c.\quad c \leftarrow \neg b.\quad d \leftarrow \neg a.\quad f \leftarrow c, \neg a.\quad e \leftarrow b, \neg a.\ \}$ are $\{a, b\}$ and $\{a, c\}$. Then, if we consider the program P together with the ordered disjunction rule $\{f \times c\}$ that stands for "if f is possible then f otherwise c" (see [3]), we obtain two answer sets $\{a, b, f\}$ and $\{a, c, f\}$. Thinking in a preference sense, with $\{f \times c\}$ we would like to express the fact that we are more interested in answer sets containing f than answer sets containing c. Then, we would expect to obtain only $\{a, c\}$.

In order to specify a preference ordering among the answer sets of a program with respect to an ordered list of atoms, we propose to use double negation in each atom of the ordered rule that represents the mentioned list of atoms. Formally, an atom with double negation corresponds to a *negated negative literal* where the only negation used is *default negation* as we shall define in Section 2. Then, in this paper we are taking advantage of the extension of ordered disjunction programs defined in [10]. For instance, if we consider again the program P and the ordered list of atoms $\{f, c\}$, then the extended ordered disjunction program is $P \cup \{\neg\neg f \times \neg\neg c\}$ and we obtain the desired answer set $\{a, c\}$. It is worth mentioning that currently running *Psmodels* [5] we can obtain the different inclusion preferred answer sets for an ordered program as defined by Brewka, however we cannot obtain the inclusion preferred answer sets for extended ordered programs. In particular, in this paper we show how we can easily translate an extended ordered disjunction program with *negated negative literals* to a standard ordered disjunction program as defined by Brewka. Then using this translation we can run *Psmodels* to obtain the preferred answer sets of an extended ordered disjunction program.

Additionally, we can also use *negated negative literals* to obtain the maximal answer sets of a program w.r.t. a set of atoms. In [8] there is a full description of a real application using ASP to perform decision making based on an argument framework (AF) in the domain of organ transplantation. Then, we propose to use *negated negative literals* to obtain the maximal answer sets of a program characterizing an AF such that these maximal answer sets correspond to the preferred extensions of the AF.

In this paper, we also give a brief overview of an example of a real application where *negated negative literals* in extended ordered programs are useful to express preferences in planning domain: evacuation planning. The idea is to specify an ordering among the feasible plans of a planning evacuation problem using extended ordered programs.

The rest of the paper is structured as follows. In Section 2, we introduce some fundamental definitions of Answer Sets and Logic Programs with Extended Ordered Disjunction. In Section 3, we present the role of default negation in extended ordered disjunction programs. In section 4, we show how extended ordered disjunction programs may be translated to standard ordered programs so that one can use existing solvers to compute answer sets. In Sections 5 we introduce an example of a real application in planning domain where negated negative literals in extended ordered programs are useful to express preferences: evacuation planning. In Section 6 we present related work about how to use extended ordered programs to obtain the maximal answer sets of a particular program such that these maximal answer sets correspond to the preferred extensions of an argument framework. Finally in Section 7, we present conclusions and future work.

2 Background

In this section we introduce some fundamental definitions of Answer Sets and Logic Programs with Extended Ordered Disjunction.

2.1 Answer Set Programming

Using Answer Set Programming (ASP) makes it possible to describe a computational problem as a logic program whose answer sets correspond to the solutions of the given problem. Currently, there are several answer set solvers, such as: DLV[1] and SMODELS[2].

In this paper, logic programs are understood as propositional theories. 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: $(,)$. An *atom* is a propositional symbol. A *literal* is either an atom a (a positive literal) or the negation of an atom $\neg a$ (a negative literal) where $\neg$ denotes *default negation* and it is the only type of negation considered in this paper. A *negated literal* is the negation sign $\neg$ followed by any literal, i.e. $\neg a$ or $\neg\neg a$. We assume that for any well formed propositional formula f, $\neg f$ is just an abbreviation of $f \rightarrow \bot$ and $\top$ is an abbreviation of $\bot \rightarrow \bot$. In particular, $f \rightarrow \bot$ is called *constraint* and it is also denoted as $\leftarrow f$. Given a set of formulas F, we define $\neg F = \{\neg f \mid f \in F\}$. Sometimes we may use *not* instead of $\neg$ and a, b instead of $a \wedge b$, following the traditional notation of logic programming. We shall define as a *clause* any well formed formula 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. We want to stress the fact that in our approach, a program is interpreted as a propositional theory. For readers not familiar with this approach, we recommend [12, 9] for further reading. We will restrict our discussion to propositional programs. As usual in answer set programming, we take for granted that programs with predicate symbols are only an abbreviation of the ground program. The signature of a program P, denoted as $\mathcal{L}_P$, is the set of atoms that occur in P. In some definitions we use Heyting's *intuitionistic* logic, 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 $a \in M$ and $P \Vdash_{\mathrm{I}} M$ to denote the fact that $P \vdash_{\mathrm{I}} M$ and P is consistent w.r.t. logic I (i.e. there is no formula A such that $P \vdash_{\mathrm{I}} A$ and $P \vdash_{\mathrm{I}} \neg A$).

We shall define answer sets (or stable models) of logic programs. The stable model semantics was first defined in terms of the so called *Gelfond-Lifschitz*

[1] http://www.dbai.tuwien.ac.at/proj/dlv/
[2] http://www.tcs.hut.fi/Software/smodels/

reduction [6] and it is usually studied in the context of syntax dependent transformations on programs. We follow an alternative approach started by Pearce [12] and also studied by Osorio et.al. [9]. This approach characterizes the answer sets for a propositional theory in terms of intuitionistic logic and it is presented in the following theorem. The notation is based on [9].

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

2.2 Logic Programs with Extended Ordered Disjunction

In [3] Brewka introduced the connective $\times$, called *ordered disjunction*, to allow an easy and natural representation of preferences and desires. 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 coffee to tea"* can be expressed as $coffee \times tea$. The definition presented here is that of [10], where ordered disjunctions is extended to wider classes of logic programs [3].

Definition 1 (Ordered Logic Programs). *An* extended ordered disjunction rule *is either a clause as defined in section 2.1, or a formula of the form: $f_1 \times \ldots \times f_n \leftarrow g$ 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; $\ldots$; 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 LPODs as presented by Brewka in [3], and we shall call them *standard ordered disjunction programs*[4]. If additionally $n = 0$ the clause is a constraint (equiv. $\bot \leftarrow g$). If $n = 1$ it is an extended clause and if $g = \top$ the clause is a fact and can be written as $f_1 \times \ldots \times f_n$. An *extended ordered disjunction program* and a *standard ordered disjunction program* as defined by Brewka can be called just *extended ordered program* and *standard ordered program* respectively where no ambiguity arises.

Now, we present the semantics of programs with extended ordered disjunction. Most of the definitions presented here are taken from [3, 5]. The only 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 Definition 1 (see [10]).

[3] Moreover, while the extension introduced in [10] is in the context of Answer Sets, the extension introduced in [4] for the operator $\times$ is in a different context.

[4] Brewka's LPODs use the strong negation connective. Here we will consider only one type of negation but this does not affect the results given in [3].

Definition 2. *[3] Let $r := f_1 \times \ldots \times f_n \leftarrow g$ be an ordered rule. For $k \leq n$ the* k-th option of r *is defined as follows: $r^k := f_k \leftarrow g, not\ f_1, \ldots not\ f_{k-1}$. Let P be an extended ordered program. 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 $r^1, \ldots, r^n$. Let M be a set of atoms. M is an answer set of P iff it is an answer set*[5] *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, *denoted by $deg_M(r)$, as follows:*

- *if $M \cup \neg(\mathcal{L}_P \setminus M) \not\vdash_{\mathrm{I}} g$, then $deg_M(r) = 1$.*
- *if $M \cup \neg(\mathcal{L}_P \setminus M) \vdash_{\mathrm{I}} g$ then $deg_M(r) = min\,\{i \mid M \cup \neg(\mathcal{L}_P \setminus M) \vdash_{\mathrm{I}} f_i\}$.*

For instance, the answer sets of the standard ordered program $P_1 = \{a \times b\}$ are $\{a\}$ and $\{b\}$ while the extended ordered program $P_2 = \{\neg\neg a \times \neg\neg b\}$ has no answer set.

Theorem 2. *[3] Let P be an extended ordered program. If M is an answer set of P then M satisfies all the rules in P to some degree.*

Definition 3 (Preferred Answer Set). *[5] Let P be an extended ordered program and L a set of atoms. We define $S_L^i(P) = \{r \in P \mid deg_L(r) = i\}$. Let M and N be answer sets of an extended ordered program P. M is* inclusion preferred *to N, denoted as $M >_i N$, iff there is an i such that $S_N^i(P) \subset S_M^i(P)$ and for all $j < i$, $S_M^j(P) = S_N^j(P)$. M is* cardinality preferred *to N, denoted as $M >_c N$, iff there is an i such that $\left|S_M^i(P)\right| > \left|S_N^i(P)\right|$ and for all $j < i$, $\left|S_M^j(P)\right| = \left|S_N^j(P)\right|$. S is a k-*preferred answer set *(where $k \in \{inclusion, cardinality\}$) of P if S is an answer set of P and there is no S' answer set of P, $S \neq S'$, such that $S' >_k S$.*

For instance, the only inclusion preferred answer set of the standard ordered program $P_3 = \{a \times b. \quad b \leftarrow \neg a.\}$ is $\{a\}$ while the only inclusion preferred answer set of the extended ordered program $P_4 = \{\neg\neg a \times \neg\neg b. \quad b \leftarrow \neg a.\}$ is $\{b\}$. As we will see in Section 4, when a program has extended ordered rules using negated negative literals we can easily translate it to a standard ordered program and then use *Psmodels* to obtain the preferred answer sets. Then, the translation of program P_4 will be $r^\bullet \cup A \cup \{b \leftarrow \neg a\}$ where $r^\bullet = \{a^\bullet \times b^\bullet\}$ and $A = \{\leftarrow \neg a, a^\bullet. \quad a^\bullet \leftarrow \neg a^\circ. \quad a^\circ \leftarrow \neg a. \quad \leftarrow a, a^\circ. \quad \leftarrow \neg b, b^\bullet. \quad b^\bullet \leftarrow \neg b^\circ . b^\circ \leftarrow \neg b. \leftarrow b, b^\circ. \ \}$ such that $a^\bullet, b^\bullet, a^\circ, b^\circ$ are atoms that do not occur in P_4. Then, by running *Psmodels* we obtain the inclusion-preferred answer set of the standard ordered program $r^\bullet \cup A \cup \{b \leftarrow \neg a\}$: $\{b, b^\circ\}$. Finally, we can see that the intersection of the inclusion-preferred answer set with $\mathcal{L}_{P_4}$ corresponds to the inclusion-preferred answer sets of the original extended ordered program P_4: $\{b\}$.

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

3 The role of default negation in extended ordered disjunction programs

In this section, we remark on the role of negated negative literals (for instance $\neg\neg a$) in an extended ordered program with respect to the definition of Brewka, that can be found in [3].

3.1 Specifying a preference ordering among the answer sets of a program with respect to an ordered list of atoms

Since $\neg\neg a$ is equivalent to the restriction $\leftarrow \neg a$, the intuition behind $\neg\neg a$ is to indicate that a must be in the answer set of a program. Moreover, the intuitive reading of the extended ordered rule $\neg\neg a \times \neg\neg b$ is as follows: if there is an answer set containing a then this answer set is preferred; if there is no answer sets containing a, then it is preferred an answer set containing b; if there is no answer sets containing a or b then none of the answer sets are preferred. Then, while the preferred answer set of the standard ordered program $\{a \times b\}$ is $\{a\}$, the extended ordered program $\{\neg\neg a \times \neg\neg b\}$ has no answer set. Hence, 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 list of atoms. An example of this is in Section 1 where the program P and the ordered list of atoms $\{f, c\}$ are considered.

However, thinking in a preference sense and in case that the answer sets of the program do not contain any of the atoms in the given ordered list 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 an atom at the end of the extended ordered rule, this atom must be an atom that does not occur in the original program. For example, let us consider again program P of Section 1 and let us suppose that we are more interested in answer sets containing e than answer sets containing f, but in case no answer set contains either e or f, we are interested in all answer sets of P. This may be expressed by adding the following extended ordered rule to program P: $\neg\neg e \times \neg\neg f \times all\_pref$ where all_pref is an atom that does not occur in P. Therefore, we obtain two answer sets $\{a, b, all\_pref\}$ and $\{a, c, all\_pref\}$ since answer sets of P do not contain neither e nor f. These answer sets correspond to answer sets of P but including the atom all_pref. Note that the answer sets of P together with the standard ordered rule $e \times f \times all\_pref$ are $\{a, b, e\}$ and $\{a, c, e\}$.

Definition 4 (Translation of a program w.r.t. an ordered list of atoms). *Let P be a program and $C = \{c_1, c_2, \ldots, c_n\}$ be an ordered list of atoms such that $C \subseteq \mathcal{L}_P$. We define a translation of P w.r.t. C, denoted as $ord_{rule}(P, C)$, into an extended ordered program as follows: $ord_{rule}(P, C) := P \cup r_C$ such that $r_C := \neg\neg a_1 \times \neg\neg a_2 \times \ldots \times \neg\neg a_n \times all\_pref$ is an extended ordered rule defined from C where all_pref is an atom that does not occur in P.*

The following Lemma formalizes the previous discussion about the specification of an ordering among the answer sets of an extended ordered program with respect to an ordered list of atoms.

Lemma 1. *Let P be a 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 $ord_{rule}(P, C)$ iff there does not exist an inclusion preferred answer set N of $ord_{rule}(P, C)$ such that $deg_N(r_C) < deg_M(r_C)$.*

3.2 Obtaining the maximal answer sets of a program with respect to a set of atoms

We can also use negated negative literals in an extended ordered program to obtain the maximal answer sets of a program w.r.t. a set of atoms A. For instance, if the answer sets of a program P are $\{b, c, e\}$, $\{b, c, d\}$ $\{f, e\}$ and $\{e, a, c\}$ then $\{b, c, d\}$ and $\{f, e\}$ are the maximal answer sets with respect to the set of atoms $A = \{b, d, f\}$. The formal definition of a maximal answer set with respect to a set of atoms is based on the definition of maximal set with respect to a set.

Definition 5 (Maximal set w.r.t. a set A). *[8] Let $\{S_i : i \in I\}$ be a collection of subsets of U such that $\bigcup_{i \in I} S_i = U$ and $A \subseteq U$. We say that S_i is a maximal set w.r.t. A among the collection $\{S_i : i \in I\}$ iff there is no S_j with $j \neq i$ such that $(S_i \cap A) \subset (S_j \cap A)$.*

Definition 6 (Maximal answer set w.r.t. a set A). *[8] Let P be a consistent program and $\{M_i : i \in I\}$ be the collection of answer sets of P. Let $A \subseteq \mathcal{L}_P$. We say that M_i is a maximal answer set w.r.t. A iff M_i is an answer set of P such that M_i is a maximal set w.r.t. A among the collection of answer sets of P.*

In order to obtain the maximal answer sets with respect to a set of atoms, the original program P is extended with a set of extended ordered rules using negated negative literals. Each extended ordered rule is defined from an atom in the given set of atoms A. For instance, in the previous example where $A = \{b, d, f\}$ the set of extended ordered rules is the following: $\{\neg\neg b \times b^\bullet.\ \neg\neg d \times d^\bullet.\ \neg\neg f \times f^\bullet.\}$ where $b^\bullet$, $d^\bullet$ and $f^\bullet$ are atoms that do not occur in the original program. Then the extended ordered program is the following: $P \cup \{\neg\neg b \times b^\bullet.\ \neg\neg d \times d^\bullet.\ \neg\neg f \times f^\bullet.\}$

The following Lemma formalizes our previous discussion about the use of negated negative literals in an extended ordered program to obtain the maximal answer sets of a program w.r.t. a set of atoms.

Definition 7. *Let P be a program and $S \subseteq \mathcal{L}_P$. We define a translation of P w.r.t. S into an ordered program, denoted by $ord_{set}(P, S)$: First, we define a set of orderd clauses w.r.t. S as follows: $C_S = \{\neg\neg a \times a^\bullet \mid a \in S \text{ and } a^\bullet \notin \mathcal{L}_P\}$. Then, $ord_{set}(P, S) = P \cup C_S$.*

Lemma 2. *Let P be a program and M be an answer set of P. Let $S \subseteq \mathcal{L}_P$. Then M is an inclusion-preferred answer set of $ord_{set}(P, S)$ iff $M \cap \mathcal{L}_P$ is a maximal answer set of P w.r.t. S.*

4 Computing preferred answer sets for extended ordered programs

It is worth mentioning that neither running *Psmodels* [5] nor following the definition given by Brewka [3] for ordered disjunction we can obtain the inclusion preferred answer sets for extended ordered programs. 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 easily translate it to a standard ordered program and then use *Psmodels* to obtain the preferred answer sets. In the following definition and lemma the atoms $a^\bullet$, a°, are atoms that do not occur in the original program P.

Definition 8. *Let $\neg\neg a$ be a negated negative literal. We define the associated set of rules of $\neg\neg a$ as follows:*
$R(\neg\neg a) := \{ \quad \leftarrow \neg a, a^\bullet. \qquad a^\bullet \leftarrow \neg a^\circ. \qquad a^\circ \leftarrow \neg a. \qquad \leftarrow a, a^\circ. \quad \}.$

Lemma 3. *Let P be a program and let $C = \{c_1, c_2, \ldots, c_n\}$ be a set of atoms such that $C \subseteq \mathcal{L}_P$. Let $r_C := \neg\neg c_1 \times \neg\neg c_2 \times \ldots \times \neg\neg c_n \times all\_pref$ be an extended ordered rule defined from C where all_pref is an atom that does not occur in P. Let $A = \{R(\neg\neg c_i) | \neg\neg c_i \in r_C$ and $1 \leq i \leq n\}$ and $r^\bullet_C = \{c^\bullet_1 \times c^\bullet_2 \times \ldots \times c^\bullet_n \times all\_pref\}$ where $c^\bullet_i$, $1 \leq i \leq n$ are atoms that occur in A. Then M is an inclusion-preferred answer set of $P \cup r^\bullet_C \cup A$ iff $M \cap \mathcal{L}_P$ is an inclusion-preferred answer set of $P \cup r_C$.*

For instance, if we consider the program P of Section 1 and the set of atoms $C = \{f, c\}$ then $r_C = \neg\neg f \times \neg\neg c \times all\_pref$,
$A = \{\leftarrow \neg f, f^\bullet.\ f^\bullet \leftarrow \neg f^\circ.\ f^\circ \leftarrow \neg f.\ \leftarrow f, f^\circ.$
$\quad \leftarrow \neg c, c^\bullet.\ c^\bullet \leftarrow \neg c^\circ.\ c^\circ \leftarrow \neg c.\ \leftarrow c, c^\circ.\ \}$ and
$r^\bullet_C = \{f^\bullet \times c^\bullet \times all\_pref\}$.

Then, by running *Psmodels* we obtain the following inclusion-preferred answer set of the standard ordered program $P \cup r^\bullet_C \cup A$: $\{a, c, c^\bullet, f^\circ\}$. 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 it was described in Section **??**, i.e., $\{a, c\}$.

5 Application to a real planning problem

In this section, we give a brief overview of a real application where negated negative literals in extended ordered programs are useful: evacuation planning. We start giving a short description of planning problems and we introduce how we can express plan preferences as an extended ordered program. Then we give a brief overview of a language for planning preference specification called $\mathcal{PP}$ and we remark on the appropriateness of $\mathcal{PP}$ for expressing evacuation planning. Finally, we briefly describe the solution to the real problem of finding alternative evacuation routes in volcano Popocatepetl using extended ordered programs.

5.1 Defining planning problems with preferences

A planning problem (D, I, G) is defined by three components: the domain description D, the initial conditions I, and the goal G. A planning problem can be formally represented using action languages [7]. One of these action languages is language β. The alphabet of the language β consists of two nonempty disjoint sets of symbols F and A. F is called the set of fluents and A is called the set of actions. A fluent represents the property of an object in a world. A state of the world σ is a collection of fluents. Language β is based on the concept of a transition relation $T \subseteq \mathcal{P}(F) \times A \times \mathcal{P}(F)$ such that $(\sigma_i, a_j, \sigma_k) \in T$ means that action a_j allows one to go from state σ_i to state σ_k. The solution of a planning problem corresponds to a plan or a sequence of actions $a_1, \ldots, a_n$ to achieve its goal G, i.e., the solution is a sequence of actions $a_1, \ldots, a_n$ such that $D \models_I G\ after\ a_1, \ldots, a_n$. The sequence $\sigma_0, a_1, \sigma_1 \ldots, a_n, \sigma_n$ where $\sigma_1, \ldots, \sigma_n$ are states and $(\sigma_{i-1}, a_i, \sigma_i) \in T$, $1 \leq i \leq n$ is called a *history* of the transition system T. A full description about language β can be found in [7]. Given a planning problem expressed in language β, it is possible to define an answer set encoding of it [2], denoted as $\Pi(D, I, G)$. Then, it is possible to obtain the solution of the planning problem (the plan) from the answer sets of $\Pi(D, I, G)$ [2].

Given a planning problem, we may obtain a high number of solutions. In this case, we need to specify an ordered list of criteria of preference $(c_1, \ldots, c_n)$ to select the "best" of those plans. To specify such preferences among feasible plans, [13] introduced a new language named $\mathcal{PP}$. We consider this language $\mathcal{PP}$ because it allows us to express temporal preferences over plans: the preferences in $\mathcal{PP}$ are based on the occurrence of an action in a plan, on the fluents that define a state in the plan, on the moment when an action occurs or a fluent holds in a state or on some combination of all them. The preferences representing time are expressed using the temporal connectives *next, always, until* and *eventually.* The combination of them can be defined using three different classes of preferences:

—A *basic desire*, denoted as φ, is a $\mathcal{PP}$ formula expressing a preference about a trajectory with respect to the execution of some specific action or with respect to the states that the trajectory gets when an action is executed.

—An *atomic preference*, denoted as $\psi = \varphi_1 \lhd \varphi_2 \lhd \ldots \lhd \varphi_k$, is a formula that gives the order in which a set of basic desires formulas should be satisfied.

—A *general preference* is a formula based on atomic preferences.

5.2 Computing answer sets of planning problems with preferences

In order to compute the preferred trajectories of a planning problem $\langle D, I, G\rangle$ w.r.t. ψ a preference of any of the three classes, [13] defines the answer set encoding $\Pi(D, I, G, \psi)$ as $\Pi(D, I, G) \cup \Pi_\psi \cup \Pi_{sat}$ where $\Pi(D, I, G)$ is the answer set encoding of the planning problem as defined in [2], Π_ψ is the encoding of the preference formula ψ and Π_{sat} are the set of rules for checking of basic desire formula satisfaction. Moreover, if M is an answer set of $\Pi(D, I, G)$, then α_M denotes the trajectory achieving the goal G represented by M.

It is worth mentioning that in particular [13] shows how we can obtain the most preferred trajectory with respect to a basic desire or an atomic preference. It is assigned a weight to each component of the preference formula, then the weight of each trajectory is obtained based on the weight of each component of the preference formula satisfied by the trajectory. Finally, in order to obtain the most preferred trajectory, i.e., the answer set with maximal weight it is used the **maximize** construct in SMODELS. In [13] it is recommended to use **jsmodels** since SMODELS has some restrictions on using the **maximize** construct. Moreover, in [13] it is showed how an atomic preference of $\mathcal{PP}$ can be mapped to a collection of standard ordered rules as defined by Brewka in order to obtain the most preferred trajectory. However, the use of weights or the mapping results in a complicated encoding. We now show that extended ordered rules with negated negative literals allows a simpler and easier encoding. This encoding is based on Corollary 1 of Lemma 1.

Corollary 1. *Let $P = \Pi(D, I, G)$ be an answer set encoding of a planning problem (D, I, G). Let $C = \{c_1, c_2, \ldots, c_n\}$ be an ordered list of atoms such that $C \subseteq \mathcal{L}_P$. Let A be the set of actions such that $A \subset \mathcal{L}_P$.Then $M \cap A$ is a preferred plan w.r.t. C iff M is an inclusion-preferred answer set of $ord_{rule}(P, C)$.*

In order to obtain the most preferred trajectory using Corollary 1, given $P = \langle D, I, G\rangle$ a planning problem and $\psi = \varphi_1 \triangleleft \varphi_2 \triangleleft \ldots \triangleleft \varphi_n$ an atomic preference formula of P we do the following :

— First, we obtain C_ψ the ordered list of atoms from ψ: We define the transformation function T of the basic desire φ_i, $1 \le i \le n$ as follow: $T(\varphi_i) := c_i \leftarrow \varphi_i$ such that $c_i \notin \mathcal{L}_P$. Then, we define the associated ordered list of rules of ψ as follow: $S_\psi = \{T(\varphi_i) | \varphi_i \in \psi, 1 \le i \le n\}$. And we define C_ψ the associated ordered list of atoms w.r.t. ψ as follow: $\{c_1, \ldots, c_n | c_i \leftarrow \varphi_i \in S_\psi \textit{ and } 1 \le i \le n\}$.

— Finally, we apply Corollary 1 to obtain α_M the most preferred trajectory w.r.t. ψ from M an inclusion-preferred answer set of $ord_{rule}(P', C_\psi)$ where $P' = \Pi(D, I, G, \psi) \cup S_\psi$.

An example where the most preferred trajectory with respect to an atomic preference is obtained using the Corollary 1 is presented in the following subsection. Obviously, the most preferred trajectory w.r.t. a basic desire is a particular case of an atomic preference. Hence, Corollary 1 works in order to obtain the most preferred trajectory w.r.t. a basic desire.

5.3 Finding alternative routes in the risk zone of the Popocatepelt

In order to illustrate the use of Lemma 1, let us consider the real problem of finding alternative evacuation routes in the risk zone of volcano Popocatepetl in Mexico. In [15, 16] we presented a detailed description of this problem and we proposed a partial solution to it using CR-Prolog [1], an extension of ASP with *consistency restoring rules*. Another partial solution to this problem was presented in [11] where we showed how CR-Prolog programs can be translated into standard ordered disjunction logic programs as defined by Brewka [3].

In this paper we give an overview of a more complete solution of the problem about finding alternative evacuation routes using language $\mathcal{PP}$. We considered to use $\mathcal{PP}$ because it allows us to express preferences over plans where the satisfaction of these preferences depends on time and on their temporal relationships. We think that in particular in evacuation planning it is very useful to express preferences in terms of time. For instance, it is *always* preferred to evacuate people from a place in risk following the defined evacuation routes. However, if at *any moment* part of the evacuation route becomes blocked then evacuees will travel by an alternative evacuation route *until* they arrive to any place out of risk. Nowadays, "Plan Operativo Popocatepetl" office in Mexico (*POP office*) is responsible of assuring safety of the people living in the risk zone of the volcano in case of an eruption. For this purpose, POP office has defined ten evacuation routes. However, some hazards that can accompany volcano eruptions (mud flows, flash floods, landslides and rockfalls, etc.) can result on the blocking of the pre-established routes. The *alternative evacuation route problem* can be stated as follows:

There is a set of predefined evacuation routes for people living in the risk area. Evacuees should travel by these routes. In case part of an evacuation route becomes inaccessible, then evacuees should search an alternative path. This alternative path can belong or not to another evacuation route. If it does not belong to an evacuation route then it should arrive to some point belonging to an evacuation route, to some refuge or to some place out of risk.

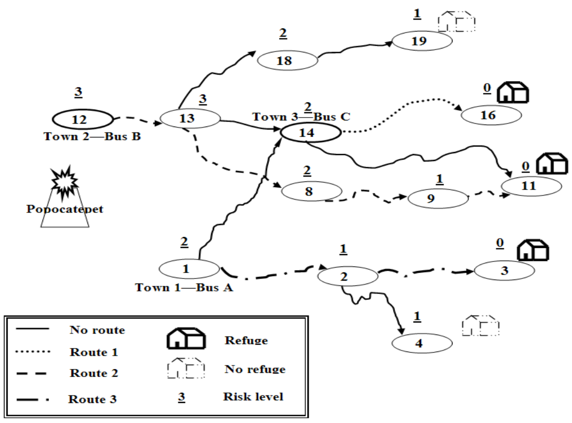

Fig. 1. Three evacuation routes: A short example.

We represent the network of roads between towns in the risk zone as a directed graph. This representation was created from an extract of our GIS database and contains real evacuation routes, towns (mostly in risk, but nearby towns not in direct risk are also included) and some additional segments that do not belong to any evacuation route, since these segments are necessary to obtain the alternative evacuation plans. We define a directed graph where nodes represent towns and evacuation routes are paths in the graph. Each segment is represented by `road(P,Q)` where `P` and `Q` are nodes. Some segments belong to evacuation routes. An exogenous action which causes `road(P,Q)` to become blocked results in a fact of the form `blocked(P,Q)`. The action `travel(P,Q)` allows to travel from `P` to `Q` if there is an unblocked segment of road from `P` to `Q`. We assumed that each action takes one unit of time.

Example 1 (Evacuation in volcano Popocatepetl).
We can define $\Pi(D, I, G)$ as follows:

```
% initial and final conditions
initially(position(busA, 1)).
initially(position(busB, 12)).
initially(position(busC, 14)).
finally(position(B,N)) :- bus(B), node(N).
% fluents
 fluent(position(B,X)) :- bus(B), node(X).
 fluent(blocked (P,Q)) :- road(P,Q).
% actions travel by a segment of road
 action(travel(B,P,Q)) :- bus(B),road(P,Q).
% Dynamic causal laws
 caused(position(B,Q),travel(B,P,Q)) :- bus(B),road(P,Q).
 caused(neg(position(B,P)),travel(B,P,Q)) :- bus(B),road(P,Q).
% Executability Conditions
 noaction_if(travel(B,P,Q),neg(position(P))):- bus(B),road(P,Q).
 noaction_if(travel(B,P,Q),blocked(P,Q)) :-  bus(B),road(P,Q).
```

We can use the following abbreviations of basic desires to define the associated atomic preference of this planning problem: "travel by evacuation route assigned by the government" as **travelERass**, "travel by evacuation route not assigned by the government" as **travelER**, "travel by a road out of an evacuation route until arrive to any point of an evacuation route" as **arriveER**, "travel by a road out of an evacuation route until arrive to any refuge" as **arriveRef**, "travel by a road out of an evacuation route until arrive to any place out of risk" as **arriveOR**. In particular, if we consider the directed graph in Figure 1 we have the following definition of **travelERass** basic desire.

travelERass := **always(occ**(travel(busB,12,13)) ∨ **occ**(travel(busB,13,8)) ∨
occ(travel(busB,8,9)) ∨ **occ**(travel(busB,9,11)) ∨ (position(busB,11)) **)** ∧
always(occ (travel(busC,14,16)) ∨ (position(busC,16)) **)** ∧
always (occ (travel(busA,1,2)) ∨ **occ**(travel(busA,2,3)) ∨ (position(busC,3)) **)**.

Let's notice that **travelERass** considers the three buses described in Figure 1. Due to lack of space we do not define the other basic desires, however it is not difficult to define them in a similar way. Then, the atomic preference is the following: $\psi = travelERass \triangleleft travelER \triangleleft arriveER \triangleleft arriveRef \triangleleft arriveOR$.

Then in order to obtain the most preferred trajectory of the planning problem $P = \Pi(D, I, G, \psi)$ with respect to the atomic preference ψ we follow the indications given in Subsection 5.2:

1. We obtain the associated ordered list of rules of ψ:
   $S_\psi = \{c_1 \leftarrow travelERass.\ \ c_2 \leftarrow travelER.\ \ \ c_3 \leftarrow arriveER.$
   $\quad c_4 \leftarrow arriveRef.\ \ \ c_5 \leftarrow arriveOR.\}$
2. We obtain the associated ordered list of atoms w.r.t. ψ representing the ordered list of criteria of preference: $C_\psi = \{c_1, c_2, c_3, c_4, c_5\}$.
3. Then by Definition 4 the extended ordered rule defined from C_ψ is: $r_{C_\psi} = \neg\neg c_1 \times \neg\neg c_2 \times \neg\neg c_3 \times \neg\neg c_4 \neg\neg c_5 \times no\_pref$, where no_pref is an atom that does not occur in P. Also by Definition 4 the translation of P w.r.t. C_ψ is: $ord_{rule}(P \cup S_\psi, C_\psi) = P \cup S_\psi \cup r_{C_\psi}$.
4. Finally, we apply Corollary 1 to obtain α_M a most preferred trajectory w.r.t. ψ from M an inclusion-preferred answer set of $ord_{rule}(P \cup S_\psi, C_\psi)$. At this point, it is worth describing how we can easily translate the extended ordered program $ord_{rule}(P \cup S_\psi, C_\psi)$ to a standard ordered program and then use *Psmodels* to obtain the preferred answer sets. Then, using Definition 8 to obtain the set A of associated rules for each $\neg\neg c_i$ with $1 \leq i \leq 5$ we have,

$$\begin{array}{llll} A = \{ \leftarrow \neg c_1, c_1^\bullet. & c_1^\bullet \leftarrow \neg c_1^\circ. & c_1^\circ \leftarrow \neg c_1. & \leftarrow c_1, c_1^\circ. \\ \quad\ \leftarrow \neg c_2, c_2^\bullet. & c_2^\bullet \leftarrow \neg c_2^\circ. & c_2^\circ \leftarrow \neg c_2. & \leftarrow c_2, c_2^\circ. \\ \quad\ \leftarrow \neg c_3, c_3^\bullet. & c_3^\bullet \leftarrow \neg c_3^\circ. & c_3^\circ \leftarrow \neg c_3. & \leftarrow c_3, c_3^\circ. \\ \quad\ \leftarrow \neg c_4, c_4^\bullet. & c_4^\bullet \leftarrow \neg c_4^\circ. & c_4^\circ \leftarrow \neg c_4. & \leftarrow c_4, c_4^\circ. \\ \quad\ \leftarrow \neg c_5, c_5^\bullet. & c_5^\bullet \leftarrow \neg c_5^\circ. & c_5^\circ \leftarrow \neg c_5. & \leftarrow c_5, c_5^\circ.\ \} \end{array}$$

   and the standard ordered rule is $r^\bullet_{C_\psi} = \{c_1^\bullet \times c_2^\bullet \times c_3^\bullet \times c_4^\bullet \times c_5^\bullet \times all\_pref\}$.

   Hence thanks to Lemma 3, the intersection of an inclusion-preferred answer set of $P \cup S_\psi \cup\ r^\bullet_{C_\psi} \cup A$ with $\mathcal{L}_P$ is an inclusion-preferred answer set of $P \cup S_\psi \cup r_C$, i.e., it is an inclusion-preferred answer set of the extended ordered program $ord_{rule}(P \cup S_\psi, C_\psi)$. Therefore, we can run *Psmodels* to obtain the inclusion-preferred answer sets of the standard ordered program $P \cup S_\psi \cup\ r^\bullet_{C_\psi} \cup A$.

In particular, if we consider the set of segments of the directed graph in Figure 1 with no blocked segments then the most preferred trajectory w.r.t. ψ is:

time 1: travel(busB,12,13), travel(busC,14,16), travel(busA,1,2);
time 2: travel(busB,13,8), travel(busA,2,3);
time 3: travel(busB,8,9);
time 4: travel(busB,9,11).

We can see that this most preferred trajectory satisfies the **travelERass** basic desire of the atomic preference ψ since all the buses travel by the evacuation route assigned by the government exactly as *POP office* indicates. Now, if we consider the set of segments of the directed graph in Figure 1 with segment from node 1 to node 2 blocked, i.e., if we add the initial condition $initially(blocked(1, 2))$ to the program P then the most preferred trajectory w.r.t. ψ is:

time 1: travel(busB,12,13), travel(busC,14,16), travel(busA,1,14);
time 2: travel(busB,13,8), travel(busA,14,16);
time 3: travel(busB,8,9);
time 4: travel(busB,9,11).

Now, the most preferred trajectory satisfies the **travelER** basic desire of the atomic preference ψ since $busA$ travels by a road out of the evacuation route assigned by the government until it arrives to node 14 of evacuation route 1.

6 Related work

Another possible real application of negated negative literals in extended ordered programs is in argumentation and in particular in the domain of organ transplantation. CARREL [14] is an agent-based platform to mediate organ transplants. In [8] there is a full description about CARREL-ASP, namely CARREL extended with ASP to perform decision making based on an argumentation framework in the domain of organ transplantation. The idea is to use Lemma 2 to obtain the preferred extension of an argumentation framework by getting the inclusion preferred answer sets of the extended ordered program $ord_{set}(P, A)$ as defined in Definition 7 where P corresponds to the encoding of an argumentation framework AF and A corresponds to the translation of the set of arguments of AF to the program P. It is worth mentioning that in [8] extended ordered programs are not used to obtain the preferred extensions. For details see [8] .

7 Conclusions

In this paper we have shown how we can easily translate an extended ordered program with negated negative literals to a standard ordered disjunction program as defined by Brewka thanks to the characterization of the answer sets for a propositional theory in terms of intuitionistic logic. It is worth mentioning that it is also possible to use a different approach to represent preferences instead of ordered disjunction programs like abductive logic programs, since the kind of preferences that we are using in this paper is not very complex.

We are interested in expressing more sophisticated preferences in evacuation planning. Then we will see if using general preferences of $\mathcal{PP}$ language is possible to express them. For instance, if a represents "arrive to a refuge (a place out of risk with provisions and water)", b represents "arrive to a place in risk with water" and c represents "arrive to a place in risk with food" then we would like to express a preference to indicate that we prefer the answer sets containing a to the answer sets containing b and c, but neither b is preferred to c nor c is preferred to b.

References

1. Marcello Balduccini and Michael Gelfond. Logic Programs with Consistency-Restoring Rules. In Patrick Doherty, John McCarthy, and Mary-Anne Williams, editors, *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, Mar 2003.
2. Chitta Baral. *Knowledge Representation, reasoning and declarative problem solving with Answer Sets*. Cambridge University Press, Cambridge, 2003.
3. Gerhard Brewka. Logic Programming with Ordered Disjunction. In *Proceedings of the 18th National Conference on Artificial Intelligence, AAAI-2002*. Morgan Kaufmann, 2002.
4. Gerhard Brewka, Salem Benferhat, and Daniel Le Berre. Qualitative choice logic. *Artif. Intell.*, 157(1-2):203–237, 2004.
5. Gerhard Brewka, Ilkka Niemelä, and Tommi 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.
6. Michael Gelfond and Vladimir 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.
7. Michael Gelfond and Vladimir Lifschitz. Action languages. *Electron. Trans. Artif. Intell.*, 2:193–210, 1998.
8. Juan Carlos Nieves, Mauricio Osorio, Claudia Zepeda, and Ulises Cortés. Argumentation for decision making in CARREL using Answer Set Programming. In *to appear in Proceedings of Encuentro Internacional de Ciencias de la Computación (ENC 2005)*, http://correo.udlap.mx/∼sc098382/dungsPaper/, 2005.
9. Mauricio Osorio, Juan Antonio Navarro, and José Arrazola. Applications of Intuitionistic Logic in Answer Set Programming. *Theory and Practice of Logic Programming (TPLP)*, 4:325–354, May 2004.
10. Mauricio Osorio, Magdalena Ortiz, and Matilde Hernandez. Generalized Ordered Disjunctions and its Applicatons. Unpublished. http://mail.udlap.mx/∼is103378/research/pubs/iclp/genOrdDisj.pdf, 2004.
11. Mauricio Osorio, Magdalena Ortiz, and Claudia Zepeda. Using CR-rules for evacuation planning. In Guillermo De Ita Luna, Olac Fuentes Chaves, and Mauricio Osorio Galindo, editors, *IX Ibero-american Workshops on Artificial Inteligence*, pages 56–63, 1994.
12. David Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.
13. Tran Cao Son and Enrico Pontelli. Planning with preferences using logic programming. In *LPNMR*, pages 247–260, 2004.
14. J. Vázquez-Salceda, U. Cortés, J. Padget, A. López-Navidad, and F. Caballero. Extending the CARREL system to mediate in the organ and tissue allocation processes: A first approach. *Artificial Intelligence in Medicine*, 3:233–258, 2003.
15. Claudia Zepeda, Mauricio Osorio, and David Sol. Towards the use of Cr-rules and Semantic Contents in ASP for planning in GIS. Technical Report RR-2004-010, Université Lyon I, Mars 2004.
16. Claudia Zepeda, Christine Solnon, and David Sol. Planning Operation: An extension of a Geographical Information System. In *LA-NMR 2004 CEUR Workshop proceedings*, volume 92, 2004.