Extending and applying PP language: An answer set planning problem language

Claudia Zepeda[1,2],Mauricio Osorio[1], Christine Solnon[2], and David Sol[1]

[1] Universidad de las Américas, CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
{josorio,sc098382,sol}@mail.udlap.mx,
[2] 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. A useful approach for expressing preferences, temporal preferences and multidimensional preferences over plans is language $\mathcal{PP}$. In this paper we give an overview of a real application of this language: Evacuation planning. Moreover, we remark on some limitations of $\mathcal{PP}$ when the size of the preference is not known or is long. Then we extend $\mathcal{PP}$ by parametric preferences and we show how this extension can overcome these limitations. Also, we give a review about how $\mathcal{PP}$ can inherit all the working framework of Linear Temporal Logic to express preferences.

Key words: Preferences, Answer Set Planning, Linear Temporal Logic.

1 Introduction

Using Answer Set Programming (ASP) [4] 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]. The objective of our work is to investigate and evaluate the applicability of ASP to represent disaster situations in order to give support in definition of evacuation plans. The goal of which is to find actions to perform to put out of risk the population living in the disaster zone.

Given a planning problem expressed in an action language, it is possible to define an answer set encoding of it [2]. Then, it is possible to obtain the solution of the planning problem (the plans) from the answer sets of its answer set encoding [2]. However, given a planning problem we may obtain a large number of solutions. In this case, we need to specify preferences to select the "best" of those plans. To specify such preferences among feasible plans, [9] introduced a new language named $\mathcal{PP}$. We consider language $\mathcal{PP}$ because it allows us to express temporal preferences over plans and at different levels: the preferences in $\mathcal{PP}$ are based on the occurrence of an action in a plan, on the fluents that define

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

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.* We think that in evacuation planning it is very useful to express preferences in terms of time. In particular they are useful, when it is not possible that evacuees follow the pre-defined evacuation plan since part of the route becomes blocked. Then, they should follow an alternative evacuation plan.

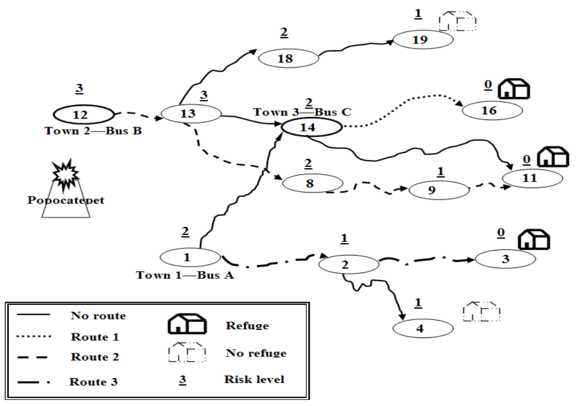

Fig. 1. Three evacuation routes: A short example.

For instance, let the directed graph in Figure 1 be a short representation of three evacuation routes in a particular zone. We define a preference about bus B, denoted as φ, to express: it is preferred that bus B travels by evacuation route 2 (path: 12, 13, 8, 9 and 11) *until* it arrives to its assigned refuge (node 11). Let us notice that preference φ is specifying a disjunction consisting of the different options that bus B has to travel by arcs belonging to evacuation route 2:

$$\mathbf{until}(\mathbf{occ}(travel(busB,12,13,1)) \vee \mathbf{occ}(travel(busB,13,8,1)) \vee \mathbf{occ}(travel(busB,8,9,1)) \vee \mathbf{occ}(travel(busB,9,11,1))\ ,\ position(busB,11)).$$

However, let us suppose evacuation route 2 has a large number of arcs. Then, in order to express φ in a similar way we would have to specify a large disjunction consisting of all arcs in the new evacuation route 2. Moreover, if this evacuation route change then we have to change also the preference. Despite in [9] it is indicated that in $\mathcal{PP}$ fluents and actions with variables are shorthand representing the set of their ground instantiations, the idea of using fluents and actions with variables is not enough to represent this type of problems.

Hence, there are some problems that can not be expressed in $\mathcal{PP}$ in a simple and natural manner. In order to have a natural representation of these kind of preferences and inspired by [7], in this paper we define $\mathcal{PP}^{par}$ language an extension of $\mathcal{PP}$ language where propositional connectives and temporal connectives allow us to represent compactly preferences having a particular property. For instance, a natural and compact representation of preference φ using a *parametric or* would be:

$$\mathbf{until}(\bigvee\{occ(travel(busB, I, F, 1)) : road(I, F, 1)\}, position(busB, 11)).$$

As we have mentioned, we think that in evacuation planning it is very useful to express preferences in terms of time. Then, in order to illustrate the usefulness of $\mathcal{PP}^{par}$ language in evacuation planning, we shall consider the real problem of finding alternative evacuation routes in the risk zone of volcano Popocatepetl in Mexico.

Finally, since $\mathcal{PP}$ is useful to express preferences over plans where the satisfaction of these preferences depends on time, in this paper we present a review of the relationship between language $\mathcal{PP}$ and propositional Linear Temporal Logic (LTL) [6, 8]. We think that language PP could take advantage of the working framework of LTL to express preferences.

The rest of the paper is structured as follows. In Section 2, we introduce some fundamental definitions about planning problems and preferences. In Section 3, we present $\mathcal{PP}^{par}$ language. In section 4 we give an overview of a real application using $\mathcal{PP}^{par}$ language: evacuation routes in the risk zone of volcano Popocatepetl in Mexico. In Section 5 we review the relationship between language $\mathcal{PP}$ and LTL. Finally in Section 6, we present our conclusions.

2 Background: Planning problems and preferences

An action signature is a couple $\langle F, A\rangle$ where F is a set of fluents and A is a set of actions. A planning problem, over a signature $\langle F, A\rangle$, is a tuple $\langle T, I, G\rangle$ where $T \subseteq \mathcal{P}(\mathbf{F}) \times \mathbf{A} \times \mathcal{P}(\mathbf{F})$ is a transition function 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, $I \subseteq F$ is the initial state and $G \subseteq F$ is the goal. The solution of a planning problem $\langle T, I, G\rangle$ over a signature $\langle F, A\rangle$ is a plan or a sequence of actions $a_1, \ldots, a_n$ to achieve its goal G starting from the initial state I, i.e. there exists a sequence of states $\sigma_0, \sigma_1 \ldots, \sigma_n$ such that $\sigma_0 = I$, $\sigma_n = G$ and $\forall i \in [1; n]$, $(\sigma_{i-1}, a_i, \sigma_i) \in T$. 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 action languages can be found in [5]. Given a planning problem expressed in an action language, it is possible to define an answer set encoding of it [2], denoted as $\Pi(T, I, G)$. Then, it is possible to obtain the solution of the planning problem (the plans) from the answer sets of $\Pi(T, I, G)$ [2].

Given a planning problem, we may obtain a high number of solutions. In this case, we need to specify preferences to select the "best" of those plans. To specify such preferences among feasible plans, [9] 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 $\psi = \varphi_1 \triangleleft \varphi_2 \triangleleft \ldots \triangleleft \varphi_k$, 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.

The set of basic desires of language $\mathcal{PP}$ can be defined inductively by the following context-free grammar $G_{PP} := (N, \Sigma, P, S)$, such that $N := \{S\}$ is the finite set of non terminals; $\Sigma := \mathbf{A} \cup \mathcal{F}_F$ is the finite set of terminals ($N \cap \Sigma = \emptyset$) where $\mathbf{A}$ and $\mathcal{F}_F$ represent the set of actions of the planning problem and the set of all fluent formulas (propositional formulas based on fluent literals) respectively; $S \in N$ is the initial symbol of the grammar; and $P := \{S \longrightarrow p|\mathbf{goal}(p)|occ(a)|S \wedge S|S \vee S|\neg S|\mathbf{next}(S)|\mathbf{until}(S,S)|\mathbf{always}(S)|\mathbf{eventually}(S)\}$ is the finite set of productions or rules where $p \in \mathcal{F}_F$ and $a \in \mathbf{A}$.

Due to lack of space, we do not explain how to obtain the most preferred plan of a planning problem with respect to the different classes of preferences in $\mathcal{PP}$, however in [9] there are different alternatives to obtain them. Moreover, in [10] we show how we can obtain the most preferred plan with respect to an atomic preference using a simpler and easier encoding than the encoding proposed in [9].

3 $\mathcal{PP}^{par}$ language

Inspired in [7] we define $\mathcal{PP}^{par}$ language, an extension of $\mathcal{PP}$ language, where propositional connectives and temporal connectives allow us to represent compactly basic desires having a particular property.

3.1 Parametric basic desires

In $\mathcal{PP}$ fluents and actions with variables are shorthand representing the set of their ground instantiations. However, we need to specify when an action or fluent have variables. Let $\mathbf{F}^{vc}$ be the set of fluents with variables and/or constants. Let $\mathbf{A}^{vc}$ be the set of actions with variables and/or constants. A *desire set* B is $\{D : t_1, \ldots, t_n\}$ where D is (1) a fluent $f \in \mathbf{F}^{vc}$ or (2) a formula of the form $occ(\mathbf{a})$ where $\mathbf{a} \in \mathbf{A}^{vc}$ or (3) a formula of the form $goal(\mathbf{p})$ where $\mathbf{p} \in \mathbf{F}^{vc}$ and $t_1, \ldots, t_n$ is a conjunction of literals [3]. Let $\mathbf{B}$ be the set of desire sets. A *parametric and* is $\bigwedge B$ where B is a desire set. A *parametric or* is $\bigvee B$ where B is a desire set.

[3] A variable or a constant is a *term.* An *atom* is $p(t_1, \ldots, t_n)$ where p is a predicate of arity n and $t_1, \ldots, t_n$ are terms. A *literal* is either an atom a or the negation of an atom *not* a.

$G^{par} := (N^{par}, \Sigma^{par}, P^{par}, S)$ is the context-free grammar that defines the set of *parametric basic desires* where $N^{par} := \{S\}$; $\Sigma^{par} := \mathbf{A}^{vc} \cup \mathbf{F}^{vc} \cup \mathbf{B}$; $S \in N^{par}$ is the initial symbol of the grammar; and the finite set of productions or rules is P^{par} such that $P^{par} := \{S \longrightarrow p \mid \mathbf{goal}(p) \mid occ(a) \mid \bigwedge B \mid \bigvee B \mid S \wedge S \mid S \vee S \mid \neg S \mid \mathbf{next}(S) \mid \mathbf{until}(S,S) \mid \mathbf{always}(S) \mid \mathbf{eventually}(S)\}$ where $p \in \mathcal{F}_F$, $a \in \mathbf{A}$ as in language $\mathcal{PP}$ and $B \in \mathbf{B}$. For example, a parametric basic desire is:

$$\mathbf{until}(\bigvee\{occ(travel(busB, I, F, 1)) : road(I, F, 1)\}, position(busB, 11)),$$

where the desire set is: $\{occ(travel(busB, I, F, 1)) : road(I, F, 1)\}$.

3.2 Basic desire instantiation and satisfaction

Given a parametric basic desire φ of a planning problem $P = \langle T, I, G\rangle$ over a signature $\langle F, A\rangle$, let $C = F \cup A \cup BK$ denote the set of constants appearing in P where BK is the background knowledge of the problem (for instance the set of nodes and segments defining the directed graph could be part of the background knowledge). A *substitution* s is a mapping from a set of variables to the set C. Given a desire set $B = \{D : t_1, \ldots, t_n\}$, the *instantiation of set* B is the following ground set $B' = \{\{s(D) : s(t_1) \ldots s(t_n)\} \mid s \text{ is a substitution}\}$; B' is called *ground desire set.* A *ground instance* of a parametric basic desire φ is obtained if every desire set B in φ is replaced by its instantiation B'.

For instance, let φ be the parametric basic desire described in Section 1:

$$\mathbf{until}(\bigvee\{occ(travel(busB, I, F, 1)) : road(I, F, 1)\}, position(busB, 11)).$$

If we consider the directed graph in Figure 1 then, the ground instance of φ is the following:

$$\begin{aligned}&\mathbf{until}(\mathbf{occ}(travel(busB, 12, 13, 1)) \vee \mathbf{occ}(travel(busB, 13, 8, 1)) \vee \\ &\qquad \mathbf{occ}(travel(busB, 8, 9, 1)) \vee \mathbf{occ}(travel(busB, 9, 11, 1)) \ , position(busB, 11)).\end{aligned}$$

Given a history $\alpha = s_0 a_1 s_1 a_2 s_2 \ldots a_n s_n$ of a planning problem (see section 2) and φ a basic desire formula, in [9] is defined when α satisfies φ (written as $\alpha \models \varphi$). Since a parametric basic desire works as an abbreviation of a basic desire, given a parametric basic desire φ' and an history α it is enough to apply over the ground instance of φ' the definition of satisfaction for basic desires given in [9] to check whether α satisfies φ'.

4 Finding alternative routes in the risk zone of volcano Popocatepelt

Let us consider the real problem of finding alternative evacuation routes in the risk zone of volcano Popocatepetl in Mexico. 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 of 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.*

Previously we have worked in this problem [4]. We have a detailed description of the problem. Also we have presented a partial solution to it using CR-Prolog [1], an extension of ASP with *consistency restoring rules*. Another partial solution to this problem shows how CR-Prolog programs can be translated into standard ordered disjunction logic programs as defined by Brewka [3] .

In this paper is given an overview of a more complete solution of the problem about finding alternative evacuation routes using language $\mathcal{PP}^{par}$.

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,R)` where `P` and `Q` are nodes and `R` is the route number. Segments with route number different of zero belong to some evacuation route. An exogenous action which causes `road(P,Q,R)` to become blocked results in a fact of the form `blocked(P,Q,R)`. The action `travel(P,Q,R)` 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. In particular, if we consider the directed graph in Figure 1 then we are considering three evacuation routes where the buses A, B and C travel from an initial location to the assigned refuge. Also we can define $\Pi(D, I, G)$ as follows:

```
% initial and final conditions
initially(position(busA, 1, 3)).
initially(position(busB, 12, 2)).
initially(position(busC, 14, 1)).
finally(position(B,N,R)) :- bus(B), node(N,R), refuge(N).
% fluents
 fluent(position(B,Q,R)) :- bus(B), node(Q,R).
 fluent(blocked (P,Q,R)) :- road(P,Q,R).
% actions travel by a segment of road
```

[4] We do not write the references of the work since it is not allowed write self-references in the paper.

```
 action(travel(B,P,Q,R)) :- bus(B),road(P,Q,R).
% Dynamic causal laws
 caused(position(B,Q,R),travel(B,P,Q,R)) :- bus(B),road(P,Q,R).
 caused(neg(position(B,P,R)),travel(B,P,Q,R)) :- bus(B),road(P,Q,R).
% Executability Conditions
 noaction_if(travel(B,P,Q,R),neg(position(B,P,R))):- bus(B),road(P,Q,R).
 noaction_if(travel(B,P,Q,R),blocked(P,Q,R)) :-  bus(B),road(P,Q,R).
```

Using $\mathcal{PP}^{par}$ we define the following parametric basic desires in order to define the associated atomic preference of this planning problem.

—*travelERass* to express that it is preferred that *always* buses travel by the evacuation route assigned by the government *until* they arrive to the refuge:
$travelERass :=$ **until(** **always(**$\bigvee\{occ(travel(busA, I, F, 3)) : road(I, F, 3)\}$**)**, $position(busA, 3)$ **)**$\wedge$ **until(** **always(**$\bigvee\{occ(travel(busB, I, F, 2)) : road(I, F, 2)\}$**)**, $position(busB, 11)$ **)**$\wedge$ **until(** **always(**$\bigvee\{occ(travel(busC, I, F, 1)) : road(I, F, 1)\}$**)**, $position(busC, 16)$ **)**

—*travelER* to express that it is preferred that *always* buses travel by roads belonging to some evacuation route and it is not important if they travel or not by its assigned evacuation route (*neq* denotes $\neq$):
$travelER :=$ **until(** **always(** $\bigvee\{occ(travel(B, I, F, R)) : bus(B), road(I, F, R), neq(R, 0)\}$ **)**, $\bigwedge\{position(B, Fi) : bus(B), refuge(Fi)\}$ **))**

—*arriveER* to express that it is preferred that buses travel out of the evacuation route assigned *until* they travel by an evacuation route to arrive at a refuge:
$arriveER :=$ **until (** **always(** $\bigvee\{occ(travel(B, I, F, 0)) : bus(B), road(I, F, 0), \}$ **)**, **until (**$\bigvee\{occ(travel(B, I, F, R)) : bus(B), road(I, F, R), neq(R, 0)\}$ **,** $\bigwedge\{position(B, Fi) : bus(B), refuge(Fi)\}$ **))**

In a similar way we could express that *always* buses travel by a road out of an evacuation route until they arrive to any place with or without refuge (*arriveR*) or any other parametric basic desire.

A possible atomic preference ψ indicating the order in which the set of parametric basic desires formulas should be satisfied is the following:

$\psi = travelERass \triangleleft travelER \triangleleft arriveER \triangleleft arriveR$

The atomic preference ψ says that plans satisfying $travelERass$ are preferred, but otherwise plans satisfying $travelER$ are preferred, and so forth. Considering the set of segments of the directed graph in Figure 1 with no blocked segments, 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 parametric basic desire *travelERass* 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 but we add the initial condition $initially(blocked(1, 2, 3))$ to the program P (i.e. the first segment of road of evacuation route 3 is blocked). Then the most preferred trajectory w.r.t. ψ is the same for buses B and C but for bus A the route is $travel(busA, 1, 14)$ in time 1 and $travel(busA, 14, 16)$ in time 2. In this case, the most preferred trajectory satisfies the parametric basic desire *arriveER* of the atomic preference ψ, since bus A travels by a road out of any evacuation route until it arrives to node 14 of evacuation route 1 and then it get to a refuge.

5 The relationship between language $\mathcal{PP}$ and Temporal Logic

In this section, we review the relationship between language $\mathcal{PP}$ and propositional Linear Temporal Logic (LTL) [6, 8]. Since $\mathcal{PP}$ is useful to express preferences over plans where the satisfaction of these preferences depend on time, the goal of this review is to show how $\mathcal{PP}$ can inherit all the working framework of LTL. Then language PP could take advantage of the working framework of LTL to express preferences. In LTL all temporal operators are future time operators, meaning that at a given state one can only reason about the present and future states. Normally, in LTL is assumed that the set of time points is infinite, discrete an linearly ordered with a smallest element. However, in this paper we are interested in a LTL [8] where the set of time points is finite since plans of planning problems are finite that we will call $FLTL$.

5.1 Finite Temporal Logic

The language of $FLTL$ [6, 8] is given by the context-free grammar $G_T := (N, \Sigma, P, S)$ where $N := \{S\}$ is the finite set of non terminals; $\Sigma := \mathcal{V} \cup \{\bot, \rightarrow, \bigcirc, \wedge, \mathcal{U}\}$ is the finite set of terminals such that $\mathcal{V}$ is the set of atomic formulas and ($N \cap \Sigma \neq \emptyset$); $S \in N$ is the initial symbol of the grammar; and $P := \{S \longrightarrow p|S \wedge S|S \rightarrow S| \bigcirc S|S\,\mathcal{U}\,S\}$ is the finite set of productions or rules where $p \in \Sigma$.

In order to omit superfluous parentheses, the priority order of the operators is established as usually. $FLTL$ formulas are denoted as $A, A_1, \ldots, B, B_1, \ldots$. The operator $\bigcirc A$, called *nexttime* operator, reads "A holds at time point immediately after the reference point (the present time)". The operator $A\,\mathcal{U}\,B$, called *until* operator, reads "A holds until B is true". Other operators can be introduced as abbreviations, e.g., $\wedge$, $\vee$, $\leftrightarrow$, $true$, $false$ as in classical logic; $\neg A$ for $A \rightarrow \bot$; $\Diamond A$ for $true\,\mathcal{U}\,A$; $\Box A$ for $\neg\Diamond\neg A$. The operator $\Diamond A$, called *eventually* operator, reads "There is a time point after a reference point at which A holds". The operator $\Box A$, called *always* operator, reads "A holds at all time points after the reference point".

The semantics of $FLTL$ is based on a *temporal structure* $\mathbf{K}$ that consists of a finite sequence $\{\eta_0, \ldots \eta_n\}$ of mappings $\eta_i : \mathcal{V} \rightarrow \{f, t\}$, the η_i are called *states.* η_0 is the *initial state.* The finite sequence of states formalizes the informal time scale; a state is a "time point". Every state is a valuation in the classical sence. For every temporal structure $\mathbf{K}$, every $i \in \mathbb{N}_0$ and every formula F, the truth value $\mathbf{K}_i(F) \in \{f, t\}$ is inductively defined , informally meaning the "truth value of F in state η_i " [6, 8]:

1. $\mathbf{K}_i(v) = \eta_i(v)$ for $v \in \mathcal{V}$.
2. $\mathbf{K}_i(A \wedge B) = \mathbf{t}$ iff $\mathbf{K}_i(A) = \mathbf{t}$ and $\mathbf{K}_i(B) = \mathbf{t}$.
3. $\mathbf{K}_i(A \to B) = \mathbf{t}$ iff $\mathbf{K}_i(A) = \mathbf{f}$ or $\mathbf{K}_i(B) = \mathbf{t}$.
4. $\mathbf{K}_i(\bigcirc A) = \mathbf{t}$ iff $i = n$ or $\mathbf{K}_{i+1}(A) = \mathbf{t}$.
5. $\mathbf{K}_i(A\,\mathcal{U}\,B) = \mathbf{t}$ iff $\mathbf{K}_j(B) = \mathbf{t}$ for some $i < j \leq n$ and $\mathbf{K}_k(A) = \mathbf{t}$ for every k, $i \leq k < j$.

In the axiomatization of temporal logic over finite sequences it is important to recognize the final state of the sequence. Moreover, for preferences about evacuation plans it is important to recognize the state where the goal is achieved. Then, the semantics for $\bigcirc A$ that we presented above has the property that $\bigcirc false$ is true at the final state [8]. However, we can define other two alternative semantics for $\bigcirc A$:
— $\mathbf{K}_i(\bigcirc A) = \mathbf{t}$ iff $\mathbf{K}_{i+1}(A) = \mathbf{t}$. This semantics has the property that $\neg \bigcirc true$ is true only at the final state. Also, this semantics is similar to the semantics of the operator $\mathbf{next}(\psi)$ of language $\mathcal{PP}$.
— $\mathbf{K}_i(\bigcirc A) = \mathbf{t}$ iff $\mathbf{K}_{i+1}(A) = \mathbf{t}$ for $0 \leq i < n$ and $\mathbf{K}_n(\bigcirc A) = \eta_n(A)$. In this semantics the truth value for $\bigcirc A$ at the final state depends on the true value of formula A. Then, the semantics of $\bigcirc A$ has the property that the final state of the plan is infinitely repeated. Using this semantics for $\bigcirc A$ we could abbreviate the operator $\mathbf{goal}(A)$ of language $\mathcal{PP}$ that reads "A holds at the final state" as follows: $A \wedge \neg \bigcirc true$. Moreover, we think that this semantics for $\bigcirc A$ could be the most suitable for evacuation planning, because of the fact that once we are in the final state this state remains without changes. For instance, once the evacuees have achieved the shelter assigned they will remain there.

5.2 Inheriting the *FLTL* work framework to language $\mathcal{PP}$

In order to inherit all the $FLTL$ work framework to language $\mathcal{PP}$ we need to do the following: (1) transforming each history α of a planning problem into a finite temporal structure $\mathbf{K}_\alpha$; (2) transforming the basic desire formula φ into a temporal formula F_φ; and (3) obtain the truth value of $\mathbf{K}_i(F_\varphi)$.

Let P be a planning problem and $\alpha = s_0 a_1 s_1 a_2 s_2 \ldots a_n s_n$ a history of P. Let $\mathbf{F}$ be the set of fluents of P and $\mathbf{A}$ be the set of actions of P. Let $\mathbf{S}$ be the set of states in α, i.e., $\mathbf{S}= \{s_0, s_1, \ldots, s_n\}$. The transformation of the history α into a finite temporal structure $\mathbf{K}_\alpha$ is described as follows: First, we define a transformation function T of an action $a \in \mathbf{A}$ as follow: $T(a) := f_a$ where f_a is a fluent and $f_a \notin \mathbf{F}$. Also, we define a straightforward generalization of T over $\mathbf{A}$, the set of actions, as follows: $T(\mathbf{A}) = \{T(a) | a \in \mathbf{A}\}$. Then, the *finite temporal structure* $\mathbf{K}_\alpha$ consists of a finite sequence $\{\eta_0, \ldots \eta_n\}$ of mappings $\eta_i : \mathbf{S} \cup T(\mathbf{A}) \to \{f, t\}$. Every η_i is a valuation defined as follows: $\eta_i(f_{a_i}) = t$ iff $a_i \in \alpha$ and $\eta_i(s_i) = t$ iff $s_i \in \alpha$.

In order to transform a basic desire formula φ into a temporal formula F_φ, we only replace each occurrence of *occ*(a) in φ for the fluent obtained from $T(a)$.

Finally, for every history $\alpha = s_0 a_1 s_1 a_2 s_2 \ldots a_n s_n$ and its finite temporal structure $\mathbf{K}_\alpha$, every $i \in \mathbb{N}_0$ and every temporal formula F_φ obtained from the basic desire formula φ, the truth value $\mathbf{K}_{\alpha i}(F_\varphi) \in \{f, t\}$ is inductively defined in the same way as it is defined for a formula in $FLTL$.

6 Conclusions

In this paper we give an overview of a more complete solution of the problem about finding alternative evacuation routes using language $\mathcal{PP}^{par}$. In particular we test the usefulness of $\mathcal{PP}^{par}$ finding alternative evacuation routes in the risk zone of volcano Popocatepetl in Mexico. Due to lack of space we only show an example with a short number of evacuation routes, however we have tested with a larger number of segments representing the real zone of volcano Popocatepetl. Also, we review the relationship between $\mathcal{PP}$ and Linear Temporal Logic. Since, it is useful to express preferences over plans where the satisfaction of these preferences depend on time, the goal of this review is to show how $\mathcal{PP}$ can inherit all the working framework of LTL. Then language PP could take advantage of the working framework of LTL to express preferences.

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. 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.
5. Michael Gelfond and Vladimir Lifschitz. Action languages. *Electron. Trans. Artif. Intell.*, 2:193–210, 1998.
6. Fred Kroger. *Temporal Logic of Programs.* Springer Verlag, Berlin, 1987.
7. Nicola Leone and Simona Perri. Parametric Connectives in Disjunctive Logic Programming. In *ASP03 Answer Set Programming: Advances in Theory and Implementation*, Messina, Sicily, September 2003.
8. R. Pucella. Logic column 11: The finite and the infinite in temporal logic. *ACM SIGACT News*, 36(1):86–99, 2005.
9. Tran Cao Son and Enrico Pontelli. Planning with preferences using logic programming. In *LPNMR*, pages 247–260, 2004.
10. Claudia Zepeda, Mauricio Osorio, Juan Carlos Nieves, Christine Solnon, and David Sol. Applications of preferences using answer set programming. In *Submmited to Answer Set Programming: Advances in Theory and Implementation (ASP 2005).*