Chapter 8

Alternative evacuation plan problem

Normally, in a risk zone evacuation routes are defined. Each evacuation route starts in a set of places in risk, traverses other places in risk and arrives to a place out of risk. Some times the place out of risk corresponds to a shelter. Each shelter has enough provisions and drinkable water for people. However, some hazards that can accompany a disaster can result on the blocking of the predefined evacuation routes. Normally, these predefined evacuation routes are defined by the authorities in the risk zone. Then it is necessary to define alternative evacuation plans. The *alternative evacuation plan problem* (AEP-problem) can be stated as follows:

> *There is a set of predefined evacuation routes for people living in the hazard zone. Each predefined evacuation route may have several initial points, but one single final point. In case part of a predefined evacuation route is 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 is preferred, in decreasing order, to arrive to some point belonging to an evacuation route or to some shelter or to some place out of risk.*

In this chapter we present two solutions to the alternative evacuation plan problem, both of them using Answer Set Programming and preferences. Applying preferences result to be a very natural and effective way to obtain an appropriate solution for

some problems. For instance, in situations where we have a set of desiderata about an evacuation plan that we would like to satisfy and not all of them can be simultaneously satisfied. Another example is when given a planning problem, we may obtain a high number of solutions. Then it is necessary to express preferences to choose some of the possible evacuation plans. Then the solutions presented in this chapter are aimed at the specification of such preferences among feasible plans. We have explored two approaches in answer sets: *consistency restoring rules* (see Section 4.7) and $\mathcal{PP}$ *language* (see Section 5.3). Then, in the following sections we are going to present our results.

We point out that when we present the use of consistency restoring rules to solve the alternative evacuation plan problem, we are going to present how programs with consistency restoring rules can be properly represented using ordered disjunction. Hence, we apply both approaches to obtain the alternative evacuation plans. The idea is that given the CR-program used to obtain the alternative evacuation plans, we translate it following Definition 6.11 of Section 6.2 to obtain an standard ordered disjunction program. Finally, we can use PSMODELS[1] to compute preferred answer sets under the ordered disjunction semantics.

Additionally, we point out that when we present the use of $\mathcal{PP}$ *language* to solve the alternative evacuation plan problem, we are going to show how we can consider the characteristic of the feasible alternative evacuation routes in order to prefer one of them. For instance, we would like to express that it is preferred that in an alternative evacuation route buses travel only by roads belonging to some evacuation route and it is not important if they travel or not by its assigned evacuation route.

There are some problems that cannot 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 in [22], in this chapter we also propose an extension of $\mathcal{PP}$ language where

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

propositional connectives and temporal connectives allow us to represent compactly preferences having a particular property.

At the end of this chapter, we present a brief overview about the relationship between language $\mathcal{PP}$ and propositional Linear Temporal Logic (LTL) [21, 40]. The idea is that language PP could take advantage of the working framework of LTL to express preferences.

8.1 Alternative evacuation plan problem assumptions

In this chapter we assume three things:

1. The background knowledge corresponding to the network of roads has the same representation described in Chapter 7 where the network of roads between towns in a hazard zone is a directed graph. In this directed graph some 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 to zero belong to some evacuation route. Some nodes correspond to a shelter or a position out of risk. In order to illustrate the background knowledge, we can consider again the Example 7.1 given in Chapter 7, where the knowledge background corresponds to Figure 7.2. Since we are going to illustrate some of our results using this example, we present a copy of Example 7.1 and Figure 7.2 in order to make an easier lecture of the current Chapter.

   Example 8.1. Figure 8.1 shows a directed graph that corresponds to a short representation of three evacuation routes in a particular zone. Then we can define the following background knowledge corresponding to this directed graph as follows:

```
route(2). route(1). route(0). risk(0). risk(1). risk(2). risk(3).

% node(point,route,risk)
node(1,1,3).   node(2,0,3).  node(2,1,3).  node(4,0,2). node(5,0,1).
node(11,0,2).  node(8,1,1).  node(9,1,0). node(12,0,3).
node(12,2,3).  node(15,0,2).  node(16,0,1). node(16,0,1).
node(13,0,3).  node(13,2,3). node(17,2,2). node(19,2,0).

% segment (ini,fin,route)
segment(1,2,1).  segment(2,11,0).  segment(2,4,0). segment(4,5,0).
segment(4,9,0). segment(2,8,1). segment(8,9,1). segment(12,15,0).
segment(12,17,2). segment(15,16,0). segment(16,19,0).
segment(13,15,0). segment(13,17,2). segment(17,19,2).

% townAt(town, node)
townAt(p1,1).   townInRisk(p1). townAt(p2,12).  townInRisk(p2).
townAt(p3,13).  townInRisk(p3).

% busIniAt(bus,point).
bus(b1).   busIniAt(b1,p1). bus(b2).   busIniAt(b2,p2). bus(b3).
busIniAt(b3,p3).

shelther(9). shelter(19).
```

□

2. Based on the background knowledge, we are going to consider the following sorts:

   Evacuation Route and variables of these sorts are denoted by $R, R', \ldots$,

   Risk and variables of these sorts are denoted by $Ri, Ri', \ldots$,

   Node and variables of these sorts are denoted by $N, P, Q, N', P', Q' \ldots$,

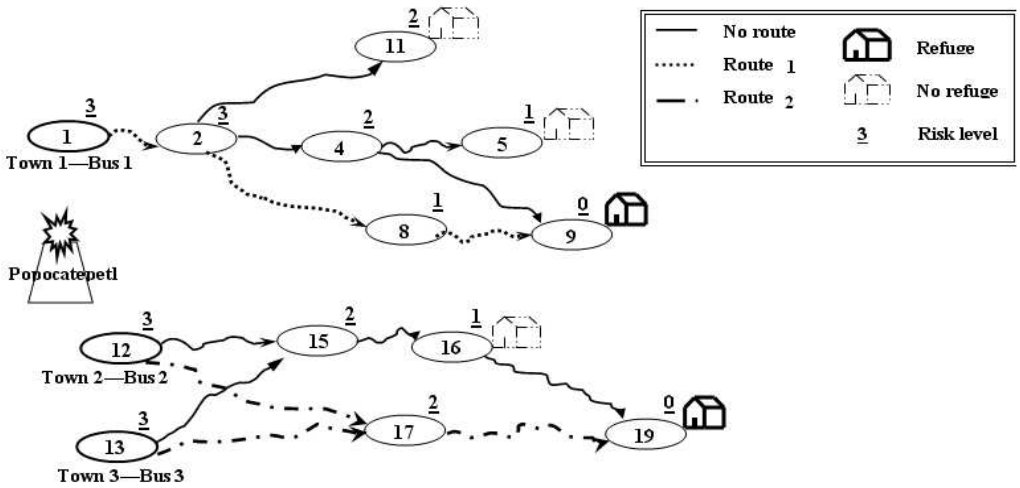

Figure 8.1: Two evacuation routes: A short example.

Town and variables of these sorts are denoted by $T, T', \ldots$,

Bus and variables of these sorts are denoted by $B, B', \ldots$.

3. We consider the alternative evacuation planning problem (D, I, G) specified in Language $\mathcal{A}$, denoted as aEv_{plan}, where the domain has only one action, *travel(B,P,Q,R)*, and three fluents: *position(B,Q,R)*, *blocked(P,Q,R)*, and *end(N)*.

   The fluent *position(B,Q,R)* indicates that bus B is at position Q of evacuation route R. The fluent *blocked(P,Q,R)* indicates that the segment of road from P to Q of route R is blocked. The fluent *end(B)* indicates that bus B is in a position that corresponds to a refuge.

   The action *travel(B,P,Q,R)* allows bus B to travel from P to Q if the three following executability conditions hold:

   (a) There is an unblocked segment of road from P to Q such that this segment belong to an evacuation route, i.e., there exist a $segment(P, Q, R)$ with R different to zero.

(b) Bus B is at position P, i.e., $position(B, Q, R)$ is true.

(c) Nodes P and Q belong to the same evacuation route R, i.e., there exist $node(P, R)$ and $node(Q, R)$ with R different to zero.

Then effects of action *travel(B,P,Q,R)* can be expressed as the three following rules:

$travel(B, P, Q, R)$ **causes** $position(B, Q, R)$ **if**

$$position(B, P, R), \sim blocked(P, Q, R), node(P, R), node(Q, R), R <> 0.$$

$travel(B, P, Q, R)$ **causes** $\sim position(B, P, R)$.

$bus(B), refuge(N), node(N, R), position(B, N, R)$ **causes** $end(B)$.

We can see that the first two rules are dynamic causal rules and the third one is a static causal rule.

Example 8.2. Let us consider the specification of the alternative evacuation planning problem aEv_{plan}; the background knowledge corresponding to the directed graph of Example 8.1 and Figure 8.1; the set of observations about the initial state $O = \{$**initially** $position(b1, 9, 1)$; **initially** $position(b2, 12, 2)$;
initially $position(b3, 13, 2)\}$; and the goal $G = \{end(b1).\ end(b2).\ end(b3).\}$. Then a possible encoding $\pi(D, O, G, l)$ in Answer Set Programming for aEv_{plan} is the following:

```
route(2). route(1). route(0). risk(0). risk(1). risk(2). risk(3).

% node(point,route,risk)
node(1,1,3).   node(2,0,3).  node(2,1,3).  node(4,0,2). node(5,0,1).
node(11,0,2).  node(8,1,1).  node(9,1,0). node(12,0,3).
node(12,2,3). node(15,0,2).  node(16,0,1). node(16,0,1).
node(13,0,3).  node(13,2,3). node(17,2,2). node(19,2,0).
```

```
% segment (ini,fin,route)
segment(1,2,1).  segment(2,11,0).  segment(2,4,0). segment(4,5,0).
segment(4,9,0). segment(2,8,1). segment(8,9,1). segment(12,15,0).
segment(12,17,2). segment(15,16,0). segment(16,19,0).
segment(13,15,0). segment(13,17,2). segment(17,19,2).

% townAt(town, node)
townAt(p1,1).   townInRisk(p1). townAt(p2,12).  townInRisk(p2).
townAt(p3,13).  townInRisk(p3).

% busIniAt(bus,point).
bus(b1).   busIniAt(b1,p1). bus(b2).   busIniAt(b2,p2). bus(b3).
busIniAt(b3,p3).

shelther(9). shelter(19).

% initially bus B is at node N of  route R.
initially(position(B, N, R)).

% goal: finally  bus B at an end-node.
finally(end(B)).

% fluents:
% position of bus B is at node Q of  route R.
fluent(position(B,Q,R)).

% road from node P to node Q of route R is blocked.
fluent(blocked (P,Q,R)).

% bus B at an end-node.
fluent(end(B)):- shelter(B).
```

```
% action travel:
% bus B  travels by the road from node P to node Q of route R,
% where R<>0, i.e, buses only travel by an evacuation route.
action(travel(B,P,Q,R)).

% Dynamic causal rules:
% if bus B  travels by the road from node P to node Q
% of route R with R<>0, then B is at position Q of route R.
caused(position(B,Q,R),travel(B,P,Q,R)) .

% if bus B  travels by the road from node P to node Q of route R,
% then B is not at position P of route R.
caused(neg(position(B,P,R)),travel(B,P,Q,R)).

% Static causal rule:
% if bus B  is at position P of route R such that
% in node P there is a shelter, then B is at an end-position.
caused(end(B), position(B,P,R)).

% Executability Conditions:
% bus B cannot  travel by the road from node P to node Q of route R,
% if B is not at position P of route R.
noaction_if(travel(B,P,Q,R),neg(position(B,P,R))).

% bus B cannot  travel by the road from node P to node Q of route R,
% if the road from node P to node Q of route R is blocked.
noaction_if(travel(B,P,Q,R),blocked(P,Q,R)).
```

Then, the plans that correspond to the answer sets of $\pi(D, O, G, l)$ with $l = 3$ are

the following:

time 0	time 1	time 2
travel(b1,1,2,1)	travel(b1,2,8,1)	travel(b1,8,9,1)
travel(b2,12,17,2)	travel(b2,17,19,2)	—
travel(b3,13,17,2)	travel(b3,17,19,2)	—

The encoding $\pi(D, O, G, l)$ obtains all possible paths of buses from their initial positions to their final positions. In this example, all buses should follow their pre-defined evacuation route, exactly as it is expected when there are predefined evacuation routes. In the plan we assumed that each action take one unit of time.

□

8.2 Using Consistency Restoring Programs

Let us suppose that part of a pre-defined evacuation route is blocked. If the action travel allows buses to travel only by an evacuation route then it is not possible to define an evacuation plan. For instance, if we consider Example 8.2 and we suppose that part of the evacuation route 2 is blocked because $road(12, 17, 2)$ is blocked. Then the encoding $\pi(D, O, G, l)$ of Example 8.2 is inconsistent and it is not possible to obtain the evacuation plans.

Hence, we propose to extend a planning problem modeled using Answer Set Planning adding CR-Rules (see Section 4.7) [47, 50]. The idea is to obtain the alternative evacuation plans taking advantage of the definition of a CR-rule and use it only if there is no way to obtain a plan when part of the pre-defined evacuation route is blocked. Hence, the CR-rules are defined over the actions that are part of the planning problem modeled. In case that a CR-rule is applied then it restores consistency and allows to obtain the alternative evacuation plans.

In particular, the CR-rule that we propose to add is the following:

$$r_2 : action(travel(B, P, Q, R')) \stackrel{+}{\leftarrow} bus(B), road(P, Q, R').$$

The intuition of the CR-rule r_2 is that it is possible to travel from P to Q if there is a segment of road from P to Q. In contrast to the regular action *travel* defined in $\pi(D, O, G, l)$, this CR-rule does not check if the road from node P to node Q belongs to an evacuation route, i.e, it is not important whether R' is equal to zero or not. In this way, r_2 allows us to find the alternative evacuation plans. The following example shows how we can use CR-rules to obtain the alternative evacuation plans.

Example 8.3. Going back to our Example 8.2, let us suppose that part of the evacuation route 2 is blocked because $road(12, 17, 2)$ is blocked, i.e, we add **initially** $blocked(12, 17)$ to $\pi(D, O, G, l)$. Since, it is not possible that bus $b2$ follows the predefined evacuation route, then $\pi(D, O, G, l)$ is inconsistent. In order to restore consistency and obtain an alternative plan we add to the program $\pi(D, O, G, l)$ the following CR-rule:

$$r_2 : action(travel(B, P, Q, R')) \stackrel{+}{\leftarrow} bus(B), road(P, Q, R').$$

As we described in in Section 4.7, this CR-rule should be used only if there is no way to obtain a plan when part of the evacuation route is blocked.

Then, we can rewrite the program as follows:

```
% initially bus B is at node N of  route R.
initially(position(B, N, R)).


% goal: finally  bus B at an end-node.
finally(end(B)).


% fluents:
```

```
% position of bus B is at node Q of  route R.
fluent(position(B,Q,R)).

% road from node P to node Q of route R is blocked.
fluent(blocked (P,Q,R)).

% bus B at an end-node.
fluent(end(B)).

% REGULAR action travel:
% bus B  travels by the road from node P to node Q of route R,
% where R<>0, i.e, buses only travel by an evacuation route.
action(travel(B,P,Q,R)).

% CR-rule where R' can be or not zero,
r_2: action(travel(B,P,Q,R')) :- + bus(B), road(P,Q,R').


% Dynamic causal rules:
% if bus B  travels by the road from node P to node Q
% of route R with R<>0, then B is at position Q of route R.
caused(position(B,Q,R),travel(B,P,Q,R)) .

% if bus B  travels by the road from node P to node Q of route R,
% then B is not at position P of route R.
caused(neg(position(B,P,R)),travel(B,P,Q,R)).

% Static causal rule:
% if bus B  is at position P of route R such that
% in node P there is a shelter, then B is at an end-position.
caused(end(B), position(B,P,R)).
```

```
% Executability Conditions:
% bus B can not  travel by the road from node P to node Q of route R,
% if B is not at position P of route R.
noaction_if(travel(B,P,Q,R),neg(position(B,P,R))).

% bus B can not  travel by the road from node P to node Q of route R,
% if the road from node P to node Q of route R is blocked.
noaction_if(travel(B,P,Q,R),blocked(P,Q,R)).
```

Then, we obtain four alternative evacuation plans, i.e., four answer sets:

Plan1:

time 0	time 1	time 2
travel(b1,1,2,1)	travel(b1,2,4,0)	travel(b1,4,9,0)
travel(b2,12,15,2)	travel(b2,15,16,0)	travel(b2,16,19,0)
travel(b3,13,15,2)	travel(b3,15,16,0)	travel(b3,16,19,0)

Plan2:

time 0	time 1	time 2
travel(b1,1,2,1)	travel(b1,2,8,1)	travel(b1,8,9,1)
travel(b2,12,15,0)	travel(b2,15,16,0)	travel(b2,16,19,0)
travel(b3,13,15,0)	travel(b3,15,16,0)	travel(b3,16,19,0)

Plan3:

time 0	time 1	time 2
travel(b1,1,2,1)	travel(b1,2,8,1)	travel(b1,8,9,1)
travel(b2,12,15,0)	travel(b2,15,16,0)	travel(b2,16,19,0)
travel(b3,13,17,2)	travel(b3,17,19,2)	—

Plan4:

time 0	time 1	time 2
travel(b1,1,2,1)	travel(b1,2,4,0)	travel(b1,4,9,0)
travel(b2,12,15,0)	travel(b2,15,16,0)	travel(b2,16,19,0)
travel(b3,13,17,2)	travel(b3,17,19,2)	—

The encoding $\pi(D, O, G, l)$ in Example 8.3 obtains all possible paths of buses from their initial positions to their final positions. For instance, Plan 3 says that bus $b1$ and bus $b3$ should follow their pre-defined evacuation routes while bus $b2$ should travel by nodes out of an evacuation route. □

8.2.1 Using standard ordered disjunction programs to compute the alternative evacuation plans from a CR-Program

In Section 4.7 we reviewed CR-programs. We saw that the semantics of a CR-program is defined in terms of the *minimal generalized answer sets* of a particular abductive logic program. This abductive logic program is based on the original program and a set of abducibles which corresponds to a subset of the signature of this original program. Additionally, in Section 6.2, we proposed a characterization of minimal generalized answer sets in terms of ordered disjunction programs. This theoretical result proves that both abductive programs and programs with CR-rules can be properly represented using ordered disjunction. Hence, we can apply both approaches to obtain the alternative evacuation plans. The idea is that given the CR-program used to obtain the alternative evacuation plans, we translate it following the Definition 6.11 of Section 6.2 to obtain an standard ordered disjunction program. Finally, we can use PSMODELS[2] to compute preferred answer sets under the ordered disjunction semantics. The following example illustrates this idea.

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

Example 8.4. Using the *ordered translation* presented in Section 6.2 Definition 6.11, we can rewrite the program in Example 8.3 where the CR-rules have been translated into ordered rules.

```
% initially bus B is at node N of  route R.
initially(position(B, N, R)).

% goal: finally  bus B at an end-node.
finally(end(B)).

% fluents:
% position of bus B is at node Q of  route R.
fluent(position(B,Q,R)).

% road from node P to node Q of route R is blocked.
fluent(blocked (P,Q,R)).

% bus B at an end-node.
fluent(end(B)).

% actions travel:
% bus B  travels by the road from node P to node Q of route R,
% where R<>0, i.e, buses only travel by an evacuation route.
action(travel(B,P,Q,R)).

% The CR-rule r_2: action(travel(B,P,Q,R')) :- + bus(B), road(P,Q,R').
% where R' can be or not zero,
% is rewritten as follows:
 aux_travel(B,P,Q,R') x action(travel(B,P,Q,R')).

% Dynamic causal rules:
% if bus B  travels by the road from node P to node Q
```

```
% of route R with R<>0, then B is at position Q of route R.
caused(position(B,Q,R),travel(B,P,Q,R)) .

% if bus B  travels by the road from node P to node Q of route R,
% then B is not at position P of route R.
caused(neg(position(B,P,R)),travel(B,P,Q,R)).

% Static causal rule:
% if bus B  is at position P of route R such that
% in node P there is a shelter, then B is at an end-position.
caused(end(B), position(B,P,R)).

% Executability Conditions:
% bus B can not  travel by the road from node P to node Q of route R,
% if B is not at position P of route R.
noaction_if(travel(B,P,Q,R),neg(position(B,P,R))).

% bus B can not  travel by the road from node P to node Q of route R,
% if the road from node P to node Q of route R is blocked.
noaction_if(travel(B,P,Q,R),blocked(P,Q,R)).
```

Running this program in *PSMODELS* [9], we obtain the same alternative evacuation plans presented in Example 8.3. □

8.3 Using $\mathcal{PP}$ Language

As we have seen, by using CR-rules makes it possible to obtain the alternative evacuation plans however, these alternative evacuation plans do not consider any other characteristic of the path that they follow. For instance, we would like to express that it is preferred that in an alternative evacuation route buses travel only by roads be-

longing to some evacuation route and it is not important if they travel or not by its assigned evacuation route. Hence in this section, we consider language $\mathcal{PP}$ in order to express preferences at different levels over the alternative plans that we are interested.

Then in this section 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 as we saw in Section 8.3.1. 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 form a place in risk following the defined evacuation routes. However, if *eventually* part of the evacuation route is blocked then evacuees will travel out of some evacuation route *until* they arrive to some shelter. The following example shows the use of language $\mathcal{PP}$ to express preferences among evacuation plans.

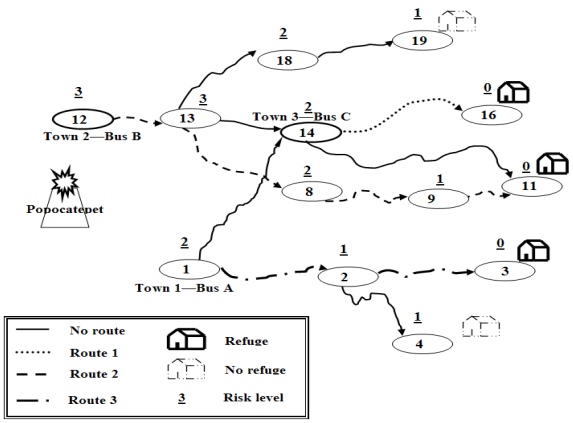

Figure 8.2: Three evacuation routes: A short example.

Example 8.5. Let us consider the specification of the alternative evacuation planning problem aEv_{plan} from Section 8 but now action travel allows buses to travel by segments that are or not part of an evacuation route , i.e., action *travel(B,P,Q,R)* allows bus B

to travel from P to Q if the three following executability conditions hold:

1. There is an unblocked segment of road from P to Q such that this segment belong or not to an evacuation route, i.e., there exist a $segment(P, Q, R)$ with R different or equal to zero.
2. Bus B is at position P, i.e., fluent $position(B, Q, R)$ is true.
3. Nodes P and Q belong or not to the same evacuation route R, i.e., there exist $node(P, R)$ and $node(Q, R)$ with R different to zero.

We also consider the background knowledge corresponding to the directed graph of Figure 8.2, the set of observations about the initial state $O = \{$**initially** $position(busA, 1, 2)$; **initially** $position(busB, 12, 3)$; **initially** $position(busC, 14, 2)\}$; and the goal $G = \{end(busA).\ end(busB).\ end(busC).\}$. Then a possible encoding $\pi_1(D, O, G, l)$ in Answer Set Programming is the following:

```
route(3). route(2). route(1). route(0).
risk(0). risk(1). risk(2). risk(3).

% node(point,route,risk)
node(1,3,2).  node(2,3,1).  node(3,3,0).  node(4,0,1). node(5,0,1).
node(12,2,3). node(13,2,2).  node(8,2,2). node(9,0,1). node(11,2,0).
node(14,1,2). node(16,1,0). node(18,0,2). node(19,0,1).

% segment (ini,fin,route)
segment(1,2,1).  segment(2,3,3).  segment(2,4,0). segment(1,14,0).
segment(14,11,0). segment(13,14,0). segment(13,18,0).
segment(18,19,0). segment(12,13,2). segment(13,8,2). segment(8,9,2).
segment(9,11,2). segment(14,16,1).

% townAt(town, node)
townAt(p1,1).  townInRisk(p1).
```

```
townAt(p2,12). townInRisk(p2).
townAt(p3,14). townInRisk(p3).

% busIniAt(bus,point).
bus(busA).   busIniAt(busA,p1).
bus(busB).   busIniAt(busB,p2).
bus(busC).   busIniAt(busC,p3).

shelter(16). shelter(11). shelter(3).

% initially bus B is at node N of  route R.
initially(position(B, N, R)).

% goal: finally  bus B at an end-node.
finally(end(B)).

% fluents:
% position of bus B is at node Q of  route R.
fluent(position(B,Q,R)).

% road from node P to node Q of route R is blocked.
fluent(blocked (P,Q,R)).

% bus B at an end-node.
fluent(end(B)):- shelter(B).

% action travel:
% bus B  travels by the road from node P to node Q of route R,
% where R is equal to 0, 1, 2 or 3, i.e, buses may travel by
% an evacuation route or not.
action(travel(B,P,Q,R)).
```

```
% Dynamic causal rules:
% if bus B  travels by the road from node P to node Q
% of route R with R<>0, then B is at position Q of route R.
caused(position(B,Q,R),travel(B,P,Q,R)) .

% if bus B  travels by the road from node P to node Q of route R,
% then B is not at position P of route R.
caused(neg(position(B,P,R)),travel(B,P,Q,R)).

% Static causal rule:
% if bus B  is at position P of route R such that
% in node P there is a shelter, then B is at an end-position.
caused(end(B), position(B,P,R)):- shelter(P).

% Executability Conditions:
% bus B cannot  travel by the road from node P to node Q of route R,
% if B is not at position P of route R.
noaction_if(travel(B,P,Q,R),neg(position(B,P,R))).

% bus B cannot  travel by the road from node P to node Q of route R,
% if the road from node P to node Q of route R is blocked.
noaction_if(travel(B,P,Q,R),blocked(P,Q,R)).
```

In particular, if we consider the directed graph in Figure 8.2 we could define have the following basic desires:

—*travelERass* to express that it is preferred that buses travel by the evacuation route assigned by the government *until* they arrive to the refuge, i.e., it is preferred that *busB* travels by segments in route 2 *until* it arrives to position 11 where it is located its assigned shelter, *busC* travels by segments in route 1 *until* it arrives to

position 16 where it is located its assigned shelter, and *busA* travels by segments in route 3 *until* it arrives to position 3 where it is located its assigned shelter.

$$
\begin{aligned}
&\mathit{travelERass} := \\
&\mathbf{until}(\mathbf{occ}(\mathit{travel}(\mathit{busB}, 12, 13, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB}, 13, 8, 2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busB}, 8, 9, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB}, 9, 11, 2)) \ , \ \mathit{position}(\mathit{busB}, 11, 2)) \ \wedge \\
&\mathbf{until}(\mathbf{occ}(\mathit{travel}(\mathit{busC}, 14, 16, 1)) \ , \mathit{position}(\mathit{busC}, 16, 1)) \wedge \\
&\mathbf{until}(\mathbf{occ}(\mathit{travel}(\mathit{busA}, 1, 2, 3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA}, 2, 3, 3)) \ , \ \mathit{position}(\mathit{busA}, 3, 3))
\end{aligned}
$$

—*travelER* to express that it is preferred that buses travel by roads belonging to some evacuation route and it is not important if they travel or not by its assigned evacuation route, i.e., it is preferred that *busA*, *busB* and *busC* could travel by segments belonging to evacuation route 1, 2 or 3 *until* they arrive to their assigned shelter:

$$
\begin{aligned}
&\mathit{travelER} := \\
&\quad \mathbf{until}(\\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busB}, 12, 13, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB}, 13, 8, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB}, 8, 9, 2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busB}, 9, 11, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB}, 14, 16, 1)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB}, 1, 2, 3)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busB}, 2, 3, 3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC}, 12, 13, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC}, 13, 8, 2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busC}, 8, 9, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC}, 9, 11, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC}, 14, 16, 1)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busC}, 1, 2, 3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC}, 2, 3, 3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA}, 12, 13, 2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busA}, 13, 8, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA}, 8, 9, 2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA}, 9, 11, 2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busA}, 14, 16, 1)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA}, 1, 2, 3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA}, 2, 3, 3)) \ , \\
&\qquad\qquad \mathit{position}(\mathit{busC}, 16, 1) \ \wedge \ \mathit{position}(\mathit{busB}, 11, 2) \ \wedge \ \mathit{position}(\mathit{busA}, 3, 3) \ )
\end{aligned}
$$

—*travelBlSh* to express that it is preferred that buses travel by its assigned evacuation route *until eventually* part of its evacuation route is blocked and then they travel out of some evacuation route *until* they arrive at its assigned shelter, i.e., it is preferred that *busB* travels by segments in route 2 *until eventually* some of these segments are

blocked and then $busB$ travels by segments out of some evacuation route *until* it arrives to position 11 where it is located its assigned shelter, and a similar situation for $busC$ and $busA$ with routes 1 and 3 respectively:

$$
\begin{aligned}
&travelBlSh := \\
&\mathbf{until}(\mathbf{occ}(travel(busB,12,13,2)) \vee \mathbf{occ}(travel(busB,13,8,2)) \vee \\
&\quad \mathbf{occ}(travel(busB,8,9,2)) \vee \mathbf{occ}(travel(busB,9,11,2)) , \\
&\quad \mathbf{until}(\ \mathbf{eventually}(blocked(12,13,2) \vee blocked(13,8,2)) \vee blocked(8,9,2) \vee blocked(9,11,2)), \\
&\qquad travel(busB,13,18,0)) \vee travel(busB,18,19,0)) \vee travel(busB,13,14,2)) \vee \\
&\qquad travel(busB,14,11,2)) \vee travel(busB,1,14,2)) \vee position(busB,11,2)\ )\ ) \\
&\wedge \\
&\mathbf{until}(\mathbf{occ}(travel(busC,14,16,1)) , \\
&\quad \mathbf{until}(\ \mathbf{eventually}(blocked(14,16,1)), \\
&\qquad travel(busC,13,18,0)) \vee travel(busC,18,19,0)) \vee travel(busC,13,14,2)) \vee \\
&\qquad travel(busC,14,11,2)) \vee travel(busC,1,14,2)) \vee position(busC,16,1)\ )\ ) \\
&\wedge \\
&\mathbf{until}(\mathbf{occ}(travel(busA,1,2,3)) \vee \mathbf{occ}(travel(busA,2,3,3)) , \\
&\quad \mathbf{until}(\ \mathbf{eventually}(blocked(1,2,3) \vee blocked(2,3,3))), \\
&\qquad travel(busA,13,18,0)) \vee travel(busA,18,19,0)) \vee travel(busA,13,14,2)) \vee \\
&\qquad travel(busA,14,11,2)) \vee travel(busA,1,14,2)) \vee position(busA,3,3)\ )\ )
\end{aligned}
$$

In a similar way we could express any other parametric basic desire.

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

$\psi = travelERass \triangleleft travelER \triangleleft travelBlSh$

The atomic preference ψ says that plans satisfying $travelERass$ are preferred, but otherwise plans satisfying $travelER$ are preferred, but otherwise plans satisfying $travelBlSh$ are preferred. □

8.3.1 Computing answer sets of planning problems with preferences using extended ordered disjunction programs

At the end of Subsection 5.3.1 we present a brief overview from [43] about how to compute the preferred trajectories of a planning problem expressed in an action language w.r.t. ψ an atomic preference (see Definition 5.5).

First, in [43] is defined the answer set encoding of a planning program with respect to a preference formula $\Pi(D, O, G, l, \psi)$ as $\pi(D, O, G, l) \cup \Pi_\psi \cup \Pi_{sat}$ where $\pi(D, O, G, l)$ is the answer set encoding of the planning problem as defined in Subsection 5.2.2, 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. It is important to mention that if the preference formula corresponds to a basic desire formula φ then, in the encoding Π_ψ φ is associated with a unique name n_φ. Moreover, if the preference formula corresponds to an atomic preference formula $\psi = \varphi_1 \lhd \varphi_2 \lhd \ldots \lhd \varphi_n$ then, in the encoding Π_ψ each $\varphi_i, 1 \leq i \leq n$ is associated with a unique name $n_{\varphi_i}, 1 \leq i \leq n$. In [43] also is mentioned that if M is an answer set of $\pi(D, O, G, l)$, then $\alpha_M = s_0 a_0 \ldots a_{n-1} s_n$ denotes the plan achieving the goal G represented by M, where $occurs(a_i, i) \in M$ for $i \in \{0, \ldots, n-1\}$ and $s_i = \{f \mid holds(f, i) \in M\}$ for $i \in \{0, \ldots, n\}$. Then, [43] shows how to obtain the most preferred trajectory with respect to a basic desire or an atomic preference. A weight is assigned 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 sets with maximal weight is used the **maximize** construct in SMODELS. In [43] it is recommended to use **jsmodels** since SMODELS has some restrictions on using the **maximize** construct. Moreover, in [43] it is shown how an atomic preference of $\mathcal{PP}$ can be mapped to a collection of standard ordered rules as defined by Brewka

[7] in order to obtain the most preferred trajectory.

We noticed that the use of weights or the mapping results in a complicated encoding. Then in order to allow a simpler and easier encoding, we propose to use extended ordered rules with negated negative literals to compute the preferred trajectories of a planning problem expressed in an action language w.r.t. ψ an atomic preference [46]. This encoding is based on Definition 8.1 and Corollary 8.1 of Lemma 6.7 (see below).

We recall that Lemma 6.7 allows us to obtain the most preferred answer set of a program P with respect to an ordered list of atoms in $\mathcal{L}_P$ by means of a particular extended ordered disjunction program. Hence, the idea of the Corollary 8.1 is to replace in Lemma 6.7 the program P with the encoding of a planning problem with respect a preference formula $\Pi(D, O, G, l, \psi)$ and use the ordered list of unique names of in the encoding of the preference formula Π_ψ as the ordered list of atoms, and then obtaining the most preferred trajectory with respect to ψ by means of a particular extended ordered disjunction program.

Definition 8.1. Let $P = \Pi(D, O, G, l, \psi)$ be an answer set encoding of a planning program with respect to an atomic preference formula $\psi = \varphi_1 \lhd \varphi_2 \lhd \ldots \lhd \varphi_n$ as defined in [43]. Then, we define *the extended ordered rule defined from* ψ, denoted by r_ψ, as follows: $r_\psi = \neg\neg n_{\varphi_1} \times n_{\varphi_2} \ldots \times \neg\neg n_{\varphi_n} \times no\_pref$, where $n_{\varphi_i}, 1 \leq i \leq n$ is the unique name of each $\varphi_i, 1 \leq i \leq n$ in P and no_pref is an atom that does not occur in P. □

Corollary 8.1. *Let $P = \Pi(D, O, G, l, \psi)$ be an answer set encoding of a planning program with respect to an atomic preference formula $\psi = \varphi_1 \lhd \varphi_2 \lhd \ldots \lhd \varphi_n$ as defined in [43]. Let r_ψ be the extended ordered rule defined from ψ. Then M is an inclusion-preferred answer set of $P \cup r_\psi$ iff $\alpha_{M \cap \mathcal{L}_P}$ is a most preferred trajectory w.r.t. ψ.* □

Obviously, the most preferred trajectory w.r.t. a basic desire is a particular case of an atomic preference. Hence, Corollary 8.1 works in order to obtain the most preferred

trajectory w.r.t. a basic desire too.

In the following example is illustrated how to obtain the most preferred trajectory w.r.t. an atomic preference using Corollary 8.1.

Example 8.6. Let $P = \Pi(D, O, G, l, \psi)$ be the answer set encoding of the planning program with respect to an atomic preference formula $\psi = travelERass \triangleleft travelER \triangleleft travelBlSh$ from Example 8.5.

Let r_ψ be the extended ordered rule defined from ψ, i.e., $r_\psi = \neg\neg n_{travelERass} \times n_{travelER} \times \neg\neg n_{travelBlSh} \times no\_pref$

Now, we apply Corollary 8.1 to obtain the most preferred trajectories w.r.t. ψ from the inclusion-preferred answer sets of $P \cup r_\psi$, i.e., if M is an inclusion-preferred answer set of $P \cup r_\psi$ then $\alpha_{M \cap \mathcal{L}_P}$ is a most preferred trajectory w.r.t. ψ.

At this point, it is worth recalling that we can easily translate the extended ordered program $P \cup r_\psi$ to a standard ordered program and then use *Psmodels* to obtain the preferred answer sets (see Definition 6.14 and Lemma 6.15).

In particular, if we consider the set of segments of the directed graph in Figure 8.2 with no blocked segments then the most preferred trajectory w.r.t. ψ is:

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

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 exactly as it was expected. Now, if we consider the set of segments of the directed graph in Figure 8.2 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	time 2	time 3	time 4
travel(busB,12,13)	travel(busB,13,8)	travel(busB,8,9)	travel(busB,9,11)
travel(busC,14,16)	—	—	—
travel(busA,1,14)	travel(busA,14,16)	—	—

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. □

8.4 Extending $\mathcal{PP}$ language with parametric basic desires

Let us notice that the basic desire *travelER* of Example 8.5 is specifying a disjunction consisting of the different options that buses have to travel by arcs belonging to some predefined evacuation route, i.e,

$$
\begin{aligned}
&\mathit{travelER} := \\
&\quad \mathbf{until}(\\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busB},12,13,2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB},13,8,2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB},8,9,2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busB},9,11,2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB},14,16,1)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busB},1,2,3)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busB},2,3,3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC},12,13,2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC},13,8,2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busC},8,9,2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC},9,11,2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC},14,16,1)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busC},1,2,3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busC},2,3,3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA},12,13,2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busA},13,8,2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA},8,9,2)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA},9,11,2)) \vee \\
&\qquad \mathbf{occ}(\mathit{travel}(\mathit{busA},14,16,1)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA},1,2,3)) \vee \mathbf{occ}(\mathit{travel}(\mathit{busA},2,3,3)) \quad , \\
&\qquad\qquad \mathit{position}(\mathit{busC},16,1) \ \wedge \ \mathit{position}(\mathit{busB},11,2) \ \wedge \ \mathit{position}(\mathit{busA},3,3) \quad)
\end{aligned}
$$

However, let us suppose evacuation routes have a higher number of arcs. Then, in order to express *travelER* in a similar way we would have to specify a long disjunction con-

sisting of all arcs in the new evacuation routes. Despite in [43] 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 express this type of problems. Hence, there are some preferences that cannot 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 in [22], in this subsection 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 *travelER* of Example 8.5 using a *parametric or* would be:

$$\textbf{until}(\bigvee\{occ(travel(B,I,F,R)) : bus(B), road(I,F,R), neq(R,0)\}, \bigwedge\{position(B,Fi,R) : bus(B), shelter(Fi), route(R), neq(R,0)\})$$

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 [49]:

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] .

[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.

Let $\mathbf{B}$ be the set of desire sets. A *parametric and* is $\bigwedge B$ where B is a desire set [49]. A *parametric or* is $\bigvee B$ where B is a desire set [49]. $G^{par} := (N^{par}, \Sigma^{par}, P^{par}, S)$ is the context-free grammar [49] 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 $B \in \mathbf{B}$.

Example 8.7. A parametric basic desire is the following :
until($\bigvee\{occ(travel(busB, I, F, 1)) : road(I, F, 1)\}$ **,** $position(busB, 19)$ **)** where the desire set is $\{occ(travel(busB, I, F, 1)) : road(I, F, 1)\}$. □

Given a parametric basic desire φ of a planning problem $P = \langle D, I, G\rangle$, let C denote the set of constants appearing in P; let $\mathbf{A}^c$ denote the set of ground instantiations of actions of P; and let $\mathbf{F}^c$ denote the set of ground instantiations of fluents of P. 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 $B' = \{\{s(D) : s(t_1) \ldots s(t_n)\} | s \text{ is a substitution}\}$; 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'.

Example 8.8. For instance, let us consider the parametric basic desire φ described in Example 8.7 and the directed graph in Figure 7.2. Then, the ground instance of φ is the following:
until (occ $(travel(b2, 12, 17, 1)) \vee$ **occ**$(travel(b2, 17, 19, 1))$**,** $position(b2, 19)$**).** □

Given a history $\alpha = s_0 a_1 s_1 a_2 s_2 \ldots a_n s_n$ of a planning problem and φ a basic desire formula, in [43] 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 a history α it is enough to apply over the ground instance of φ' the definition of satisfaction for basic desires given in [43] to check whether α satisfies φ' .

Example 8.9. If we consider the Example 8.5 then we can rewrite the basic desires using $\mathcal{PP}^{par}$ as follows:

—*travelERass* to express that it is preferred that buses travel by the evacuation route assigned by the government *until* they arrive to the refuge, i.e., it is preferred that *busB* travels by segments in route 2 *until* it arrives to position 11 where it is located its assigned shelter, *busC* travels by segments in route 1 *until* it arrives to position 16 where it is located its assigned shelter, and *busA* travels by segments in route 3 *until* it arrives to position 3 where it is located its assigned shelter.

travelERass :=

$$\textbf{until}(\bigvee\{occ(travel(busB,I,F,2)) : segment(I,F,2)\},position(busB,11,2))\wedge$$
$$\textbf{until}(\bigvee\{occ(travel(busC,I,F,1)) : segment(I,F,1)\},position(busC,16,1))\wedge$$
$$\textbf{until}(\bigvee\{occ(travel(busA,I,F,3)) : segment(I,F,3)\},position(busA,3,3))$$

——*travelER* to express that it is preferred that buses travel by roads belonging to some evacuation route and it is not important if they travel or not by its assigned evacuation route, i.e., it is preferred that *busA*, *busB* and *busC* could travel by segments belonging to evacuation route 1, 2 or 3 *until* they arrive to their assigned shelter:

travelER :=

$$\textbf{until}(\bigvee\{occ(travel(B,I,F,R)) : bus(B), segment(I,F,R), neq(R,0)\},$$
$$\bigwedge\{position(B,Fi,R) : bus(B), shelter(Fi), route(R)\})$$

——*travelBlSh* to express that it is preferred that buses travel by its assigned evacuation route *until eventually* part of its evacuation route is blocked and then they travel out of some evacuation route *until* they arrive at its assigned shelter, i.e., it is preferred that $busB$ travels by segments in route 2 *until eventually* some of these segments are blocked and then $busB$ travels by segments out of some evacuation route *until* it arrives to position 11 where it is located its assigned shelter, and a similar situation for $busC$ and $busA$ with routes 1 and 3 respectively:

travelBlSh :=

$$\begin{aligned}
&\textbf{until}(\bigvee\{occ(travel(busB,I,F,2)) : segment(I,F,1)\},\\
&\qquad\textbf{until}(\ \textbf{eventually}(\bigvee\{blocked(I,F,2) : segment(I,F,2)\}\ ) \wedge\\
&\qquad\quad\bigvee\{occ(travel(busB,I,F,0)) : segment(I,F,0)\}\ , position(busB,11,2)\ )\ )\\
\wedge\\
&\textbf{until}(\bigvee\{occ(travel(busC,I,F,1)) : segment(I,F,1)\},\\
&\qquad\textbf{until}(\ \textbf{eventually}(\bigvee\{blocked(I,F,1) : segment(I,F,1)\}\ ) \wedge\\
&\qquad\quad\bigvee\{occ(travel(busC,I,F,0)) : segment(I,F,0)\}\ , position(busC,16,1)\ )\ )\\
\wedge\\
&\textbf{until}(\bigvee\{occ(travel(busA,I,F,3)) : segment(I,F,3)\},\\
&\qquad\textbf{until}(\ \textbf{eventually}(\bigvee\{blocked(I,F,3) : segment(I,F,3)\}\ ) \wedge\\
&\qquad\quad\bigvee\{occ(travel(busA,I,F,0)) : segment(I,F,0)\}\ , position(busA,3,3)\ )\ )
\end{aligned}$$

□

8.5 Language $\mathcal{PP}$ and Temporal Logic

Since $\mathcal{PP}$ is useful to express preferences over plans where the satisfaction of these preferences depends on time, in this section we present a brief overview about the relationship between language $\mathcal{PP}$ and propositional Linear Temporal Logic (LTL) [21, 40]. Then language $\mathcal{PP}$ could take advantage of the working framework of LTL to express preferences.

8.5.1 Inheriting the LTL framework to $\mathcal{PP}$

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 and linearly ordered with a smallest element. However, we are interested in a LTL where the set of time points is finite, denoted as $FLTL$, since plans of planning problems are finite.

The language of $FLTL$ [21, 40] 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 sense. 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 " [21, 40]:

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. $\mathbf{K}_i(A \rightarrow B) = \mathbf{t}$ 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 order to inherit all the $FLTL$ framework to language $\mathcal{PP}$ we propose to do the following:

1. We transform each history α of a planning problem into a finite temporal structure $\mathbf{K}_\alpha$: 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 $T(a) := f_a$ where f_a is a fluent and $f_a \notin \mathbf{F}$. Also, we define a straightforward generalization of T over $\mathbf{A}$ as $T(\mathbf{A}) = \{T(a) | a \in \mathbf{A}\}$.

Then, the *finite temporal structure* $\mathbf{K}_\alpha$ consists of a finite sequence $\{\eta_0, \ldots \eta_n\}$ of mappings $\eta_i : \mathbf{S} \cup T(\mathbf{A}) \rightarrow \{f, t\}$. Every η_i is a valuation defined as $\eta_i(f_{a_i}) = t$ iff $a_i \in \alpha$ and $\eta_i(s_i) = t$ iff $s_i \in \alpha$.

2. Then, we transform the basic desire formula φ into a temporal formula F_φ: 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 $T(a)$.

3. Finally, we obtain the truth value of $\mathbf{K}_i(F_\varphi)$: 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 $FLTL$.

8.5.2 Taking advantage of the working framework of LTL

Once we have transformed each history α of a planning problem into a finite temporal structure $\mathbf{K}_\alpha$ and transformed the basic desire formula φ into a temporal formula F_φ then, language PP could take advantage of the working framework of LTL to express preferences. One example of this is the following.

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 in the previous Subsection has the property that $\bigcirc false$ is true at the final state since $\mathbf{K}_i(\bigcirc A) = \mathbf{t}$ iff $i = n$ [40]. However, we can define two other alternative semantics for $\bigcirc A$ [49]:

- $\mathbf{K}_i(\bigcirc A) = \mathbf{t}$ 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 **next**(ψ) 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 **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.

8.6 Conclusion

In this chapter we presented two solutions to the alternative evacuation plan problem using two approaches for preferences in Answer Set Programming. In particular we explored two approaches in answer sets: *consistency restoring rules* and $\mathcal{PP}$ *language*.

In this chapter we also proposed an extension of $\mathcal{PP}$ language where propositional connectives and temporal connectives allow us to represent compactly preferences having a particular property.

Finally, we presented a brief overview about the relationship between language $\mathcal{PP}$ and propositional Linear Temporal Logic (LTL) [21, 40].