A Decision Support System for Disaster Situations

Mauricio Osorio[1], Claudia Zepeda[12], David Sol[1] and Gerardo Lazzeri[1]

[1] Universidad de las Américas, CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
`{josorio,sc098382,sol,glazzeri}@mail.udlap.mx`,
[2] Bât. Nautibus, Université Lyon I, LIRIS, 43 Bd du 11 novembre, 69622 Villeurbanne Cedex, France
`czepeda@bat710.univ-lyon1.fr`

Abstract. In this work, we present current results of our research which goal is to develop a decision support system using Answer Set Programming as an extension of a Geographic Information System to model vehicular evacuation plans. We also present why an Answer Set Programming approach seems to be appropriate to explore in order to create this extension. In order to achieve our objective, we first model the disaster scenario where we are taking advantage of different features of Answer Set Programming. Once we have modeled the disaster scenario, we will define an action language to model and give solution to vehicular evacuation plans, named ALEP. We present our current results about the main features of the language ALEP. Finally, we plan to incorporate a Case Based Reasoning component to deal with real time situations, and to test ALEP language defining a DSS, as an extension of a GIS, in order to give support in developing evacuation plans in volcano Popocatepetl. We think that this DSS will be advantageous to the "Plan Operativo Popocatepetl" office in Mexico.

1 Introduction

Our work aims at adding planning capabilities by using Answer Set programming (ASP) [12] to Geographical Information Systems (GIS). The goal is to give decision support for planning in case of disaster situations such as volcanic eruptions. Indeed, GIS contain data —such as maps, demographic data, traffic volumes— that are useful for supporting evacuation planning in case of disaster situations. However, planning in GIS usually supposes that data describing the environment is completely known and static [7], and more generally, GIS are not able to reason with incomplete or dynamic information, neither can they handle normative rules. Hence, we propose to extend GIS with such reasoning capabilities. Our system should represent actions and states of the world, in relation with real information contained in GIS databases, and it should be able to deal with incomplete and dynamic information and normative rules. It should be able to take the specification of a real emergency scenario, describing the hazard or set of hazards occurring in a real situation, to check plan consistency and propose alternative plans in case of exogenous actions. It should also give support for defining evacuation plans, finding the set of all possible evacuation plans, or finding the best plan with respect to a given criterion of preference (e.g. the safest route, the shortest route or the fastest route). In order to complete the decision support system we plan to incorporate a Case Based Reasoning (CBR) approach to use or adapt solutions (evacuation plans) that were used to solve old problems.

To add such planning capabilities to GIS, we propose to use ASP [12]. Indeed, ASP is a logic programming language with strong theoretical work and well suited for declarative knowledge representation [21, 22]. ASP is well suited for representing action and change, and it allows reasoning with both incomplete and dynamic knowledge. Different inference engines for ASP have been implemented [18, 10]. However, nowadays there are only a few real applications using ASP, hence, a goal is to merge ASP and GIS in order to make ASP more applicable. We plan to apply and validate our decision support system for modeling vehicular evacuation plans in volcano Popocatepetl area, in Puebla state in Mexico. The "Plan Operativo Popocatepetl" office in Mexico (*POP office*) has the responsibility of coordinating actions to evacuate safely people into towns located next to the risk zone of volcano Popocatepetl in case of an eruption. Currently, this task is not supported by a computer decision system, and in case of danger, the actual evacuation plan is designed using printed maps and printed reports. In this case, decisions can hardly be justified because of a lack of information [25].

We need to remark that evacuation time, time evacuees need to complete an evacuation process, consists of three main components [13]: Time needed to recognize a dangerous situation, time needed to decide which course of action take and time needed to move towards the safety area (egress time). The first and second times are influenced by behavioral and organizational factors hence are difficult to predict these times. Behavior estimation of the residents under panic situation: if each person in the zone knows where to go when evacuation starts, if they know where is the nearest easily accessible point outside the zone to which they should go as quickly as possible. The egress time is influenced by the availability of emergency exit signs, well planned evacuation procedures, constructional factors(effective width of walkway, slope of stairs), the place where people are located with respect to the time (night or day, week-end or working days) and human behavior during panic situation. Despite its strenghts, ASP may not always be able to generate solutions in real time. For this reason we are also exploring the use of other technologies to complement our system. One such technology is CBR, which adapts solutions to known problems in order to quickly generate reasonably good solutions to new problems. We have already applied CBR successfully in other domains, such as chess position analysis [14, 15], and image comparison in the context of the Popocatepetl volcano's fumaroles [2]. Furthermore, we have already developed some preliminary CBR work for the generation of emergency evacuation plans in the context of the Popocatepetl volcano [23]. In the future, we plan to incorporate some of the ideas developed in the works just mentioned to strengthen our system. We also plan to study other systems which have applied

CBR to similar problems, such as CHARADE [4], and CARICA [26], which work in the context of forest fire emergencies.

The paper is structured as follows. We show the objective and methodology. Next we present the work and current results for each step of the methodology. Finally, we present our conclusions and further work.

2 Theoretical Background

2.1 Geografic Information

In order to define evacuation plans is necessary to have real information about [13, 27, 17]: Hazards zones, destructive phenomena, roads, towns, refuges and means of transport and traffic control. Most of this information should be information from a GIS database. As usually in GIS databases, the original basic data is composed of two parts[3]. The first part contains spatial data (maps obtained either from field surveys or by the interpretation of Remotely Sensed data). The second component contains non spatial data, and is complementary to the spatial data, describing what is at a point, along a line or in a polygon, and containing socio-economic characteristics (demographic data, occupation data for a village, or traffic volume for roads). Currently, we have a GIS Databese with real information about Popocatepetl problem [24].

2.2 Answer Set Programming

Indeed, ASP is a logic programming language with strong theoretical work. It is well suited for declarative knowledge representation, allowing a simple user —that is not expert programmer— to describe a scenario in a declarative way, with a pseudo-natural language [21, 22]. It is worth to mention that using a declarative logic programming language all results of a program are automatically obtained when this program is an input for some ASP inference engine like DLV [18] or Smodels [10]. ASP is also well suited for representing action and change, and it allows reasoning with both incomplete and dynamic knowledge. For example if we know nothing about the availability of a segment of road we can assume, using ASP normative rules, that this segment of road can be used in an evacuation unless we have the specific information that it is blocked. The features of ASP that ALEP can exploit are the following:
—Declarative knowledge representation: This feature allows us to describe the scenario in a declarative way, using natural language. Hence, if a final user needs to describe some particular scenario then he/she does not have to be an expert programmer.
—Reasoning with both incomplete and dynamic knowledge: Normally, planning

[3] Geographic information has special formats to be stored and it follows some standards. There are two standards for geographic information: one defined by the Open GIS Consortium, Inc (OGC) (`http://www.opengis.org`) and the other by the Environmental Systems Research Institute, Inc. (ESRI) (`http://www.esri.com`)

in GIS is made with geometric operations supposing that data describing environment are completely known and static [7]. Using two kinds of negation ASP approach we can model exceptions and reasoning with incomplete knowledge about the environment. For example, if we know nothing about a segment of road, we can assume that this segment of road can be used in an evacuation unless have the specific information that this segment is blocked.

—Constraints: We can model constrains about the scenario. For example, if know that a segment of road is blocked then this segment should not be part an evacuation route.

There are some applications using ASP, one of the is related to NASA [6], which is in the process of developing complex systems needed to improve NASA's ability to deploy humans on long distance exploration missions. Under normal circumstances, such a system will be autonomous. It should be able to react to changes in the environment, plan to performs certain goals, and detect the system's malfunctioning. In more difficult situations the system shall be able to alert humans about the problem, and to cooperate with them in finding a solution. However, nowadays there are only a few real applications using ASP, hence, a goal is merge ASP and GIS in order to make ASP more applicable.

2.3 Case-Based Reasoning

Unlike classical planning systems which rely on a knowledge base of rules, CBR relies on a memory of specific past experiences (e.g. cases) which are used in der to develop solutions for new problems. The CBR process of solving a new problem involves two main steps: Finding similar cases in memory, and adapting previous solutions to current problems. CBR reasoning is primarily a process remembering solutions to old problems, and then either adapting or comparing them, rather than composing rules to generate solutions from scratch. This property makes CBR very suitable for quickly obtaining solutions that are sufficiently good to solve new complex problems, as long as we have an appropriate case base. Even though some of the CBR-generated solutions may not be optimal, as long as they are feasible they will be useful in in the generation of evacuation plans in case of a Popocatepetl volcano emergency situation. A detailed description CBR techniques and applications can be found in [16]. There are two styles CBR systems: Problem Solving, where a final solution to the new problem must be produced, and Interpretive, where it is only necessary to present a justification, presenting an argument of why this solution may be appropriate and what may have to be changed for the solution to work for the new problem. Our primary target is to develop a Problem Solving CBR to complement the ASP system. However, even an Interpretive CBR, may provide enough information to develop a feasible solution with the interaction with a human expert. The remainder this subsection describes two of our previous experiences with CBR in different domains from which we plan to incorporate some ideas to our current application.

ICONCHESS [14] uses a case based approach to generate high-level advice chess middle game positions. To generate this advice, ICONCHESS relies on a case base of middle game chess positions. Each case is a particular position indexed by its relevant features, a list of recommended actions for each player, and the preconditions that were present in the position that suggested the recommended actions. The indexing scheme is the basis for a similarity metric used to compare new positions with those available in the case base. A comprehensive description of these metrics, as used in ICONCHESS, can be found in [14]. Here we focus on the concept of Influence, which is the one whose transfer to our current target domain is most interesting. Influence is a metric introduced in ICONCHESS as an attempt to determine the impact of the fast pieces (e.g. queens, rooks, bishops, and knights) on different parts of the board.

In ICONCHESS, the Influence Regions were fixed in the way they are traditionally defined in chess. However, it is also conceivable to think in terms of Dynamic Influence Regions, which are Influence Regions that can take any shape or size. This concept will allow us to identify similar influence patterns across the board, which may also be indicative of similarities between different positions. Implementation of Dynamic Influence Regions is currently under way. It turns out that the concept of Dynamic Influence Regions may be applied to determine those areas in the vicinity of the Popocatepetl that may be in danger given a set of climatological conditions. Section 8 explains how this can be done.

Another interesting application of CBR is an Image Based Reasoner which uses a similarity metric based on 5-dimensional (5D) graphics concepts in order to compare images. A description of this application may be found in [2]. Briefly, the idea is to map each pixel in an image into a 5D space using the two-dimensional position and color (RGB) information corresponding to each pixel. Then the 5D hypervolume of the generated image is computed and used as an initial similarity metric. Later, additional graphical transformations, such as intersections, unions, and filters are used to refine the similarity calculation. A thorough description of the mathematics used to develop the similarity metric and the algorithm to compute it can be found in [2]. Preliminary testing has been done with moderate success using some of the available images of the Popocatepetl volcano.

In section 8 shows how we plan to combine this graphical similarity metric with the concept of Influence explained previously in this section.

3 Objective and Methodology

The objective of our work is to investigate and evaluate the applicability of ASP to represent geographic information and disaster situations in order to give support in definition of emergency plans. This paper introduces an action language, named ALEP, which can be used to specify evacuation planning problems. An action language allows us to specify planning problems, i.e. find a sequence of actions that has as particular result a given goal state from a given initial state. ALEP take advantage of ASP features and uses a macroscopic approach [13] to capture the evacuees' movement. In a macroscopic approach is not considered any individual behavior during the emergency situation. It is worth to stress that ASP is a declarative logic programming language then this feature is inherited to ALEP. Hence, ALEP allows us to describe the disaster scenario and specify evacuation planning problems in a declarative way, using natural language. If a final user needs to describe some particular scenario then he/she does not have to be an expert programmer. While this paper concentrates on volcanic eruption disasters ALEP is applicable also to other kind of disasters (for instance, building evacuation). We suppose adding ALEP to a GIS would offer a better approach to specification and possible solutions of evacuation plans problems to experts. Hence, we propose developing a decision support system (DSS) as an extension of a GIS to model evacuation plans. We also propose to enhance our DSS by incorporating a CBR component to deal with real time situations that may not be solved quickly enough by ASP. In order to achieve our objective, we are obeying the following series of tasks:

- Modeling the disaster zone using a GIS database.
- Define an action language to model the disaster scenario and to specify evacuation planning problems, named ALEP.
- Develop a basic CBR tool to use or adapt previous evacuation plans.
- Testing the expressiveness of ALEP: defining a DSS for volcanic eruptions.

In the following sections we are going to describe each task. For each task we present the relevant results that currently we have, the contribution to the technical area, conclusions and recommendations for future work.

4 Modeling the Disaster Zone Using a GIS Database

In order to achieve a correct representation of the disaster zone, we represent the network of roads as a directed graph, where the directed graph edges and the non spatial data about roads are in one-to-one correspondence. In this graph, towns are connected with other towns by roads and each road is made up by segments. Some vertices of the set of directed graph edges have a relationship with identifiers of towns. An evacuation route is a path on this directed graph. For instance, the network of roads in figure 1. Hence, we use non-spatial data about segments roads and towns in order to define the background knowledge for the ASP approach that we are developing. The construction of this representation can be done in semi-automatic way. Using a GIS tool, it is possible to save non-spatial data a text file, each line of this file corresponding among other information to the identifiers of initial and final nodes of road segments used to define the directed graph. However, the GIS data is not always consistent about non spatial data (for instance, duplication of identifiers of nodes) and it is necessary to correct this information in order to use it.

5 Overview of Language ALEP

In an evacuation is necessary to remove all residents from a danger zone to safety quick as possible and with utmost reliability. Normally, danger zones are connected to other zones by roads and roads are formed by a set of segments. Evacuation experts should select some of these roads in order to define evacuation routes. Each evacuation route can start at different locations, pass by several zones in risk or pass by locations where another evacuation route passes (some segments belong to more than one evacuation route). In order to model the disaster scenario and to obtain the evacuation plans we are defining an action language, named ALEP. ALEP is an extension of language $\mathcal{K}$ [11]. In the following subsections we present briefly language $\mathcal{K}$ and the features of language ALEP as an extension af language $\mathcal{K}$.

5.1 Language $\mathcal{K}$

$\mathcal{K}$ is a logic-based planning language, well suited for planning under incomplete knowledge where transitions between states of knowledge can be described. The states of knowledge are described as a set of fluents. $\mathcal{K}$ is close in spirit to ASP semantics and exploits the power of ASP to deal with incomplete knowledge. Currently there is a planning system supporting $\mathcal{K}$, named DLV-K [3], that is implemented on top of DLV [10]. The main features of language $\mathcal{K}$ are the following: Definition of *actions and fluents*; *causations rules* (`caused f if B after A`) to specify the effect of actions and allow transitions between states of knowledge; *default negation* (`not`) that allows for natural modeling of inertial properties, default properties and dealing with incomplete knowledge; *executability of actions* (`executable action if fluent 1,..., fluent n`); definition of *initial state constraints* to specify the initial state (it is preceded by the keyword `initially`); *parallel execution* of actions (this can be prohibited by the statement `noConcurrency`); handling of *complete and incomplete knowledge* because the language allows one to represent transitions between possible states of the world (`total fluent`); definition of *goals* and *plans* where the plan for a goal is a sequence of actions whose execution leads from an initial state to a state where the goal is true. For a through description of the language $\mathcal{K}$ we refer to [11].

5.2 Features of Language ALEP

The main features of language ALEP are briefly summarized as follows.

Road set specify the list of directed edges used to represent the network of roads as a directed graph. Some of these segments belong to an evacuation route. In case of evacuation routes have some repeated vertices, these vertices belong to the first evacuation route where they are defined. The final vertex of each evacuation route corresponds to a refuge where people are out of risk. The statements used in definition of the network of roads are:

`route r formed_by` $V[n]$ `connected_to` V_j.

Intuitively, the above statement says: evacuation route `r` is the path on the directed graph defined by the sequence of `n` vertices $V[n]$, where the final vertex of this path is V_j that is part of a defined evacuation route; connected_to part is allowed to be empty which means that the last vertex of the sequence of vertices $V[n]$ is not connected to any other evacuation route.

`no_route` $V[n]$ `connected_to` V_j.

Intuitively, the above statement says: the path on the directed graph defined by the sequence of n vertices $V[n]$ where the final vertex of this path is V_j that is part or not of a defined evacuation route: connected_to part is allowed to be empty which means that the last vertex of the sequence of vertices $V[n]$ is not connected to any route.

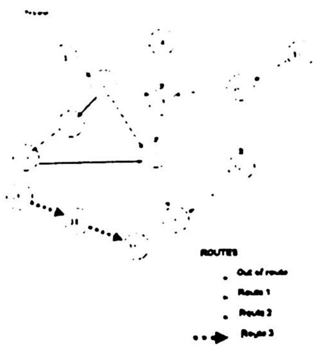

Fig. 1. A short example of a network of roads as a directed graph

For instance, the network of roads in figure 1 is represented as:

```
route 1 formed_by 1,2,3,5,6.
route 1 formed_by 4 connected to 3.
route 2 formed_by 11,10 connected to 3.
route 2 formed_by 10,12 connected to 6.
route 3 formed_by 13,14,15.
no_route 2,8,9 connected_to 5.
no_route 2 connected_to 5.
```

Location rules Location rules specify the relationship between danger locations and the directed graph vertices. The statements used in defining this relationship is: `t at` V_j.

For instance, if Calpan town is located in vertex `1` in figure 1 this can be represented as `calpan at 1`.

Unblocked segments of an evacuation route Evacuation is defined as to take all residents away from a given location that has been considered as a danger zone to safety as quick as possible and with utmost reliability [13]. In order to get reliability it is expected that evacuees follow the evacuation routes defined by experts in case of an eruption. However, a disaster can have a effect on evacuation route segments: blockage. Hence, is useful to know the segments of road that are free in order to define an alternative route. The queries used to obtain the set of unblocked segments is the following: `partial_route(S,T)` For instance, if in figure 1 the vertex 3 of evacuation route 1 is blocked then the result of `partial_route(1,6)` is $\{1,2,5,6\}$ thaf correspond to the set of unblocked vertices or route 1.

Alternative evacuation routes When part of an evacuation route is blocked, it is necessary to define alternative routes where evacuees travel toward their assigned refuge. These alternative routes should use alternative segments. An alternative segment is a segment of road that does not belong to an evacuation route. Moreover, these alternative routes should arrive to some point belonging to an evacuation route, to some refuge or to some place out of risk. The more an alternative route uses evacuation route segments, the better an alternative route become. We need to model the alternative evacuation route problem in a disaster situation. This model should consider the use of alternative segments as main parameter, where the set of segments of road (alternative and evacuation route segments) is part of the input and the minimal number of alternative segments used in the overall evacuation is the output. The queries used to obtain the alternative evacuation routes are the following:

`all_alternatives_routes(S,T)`

informally states that it is necessary to find all alternative evacuation routes from vertex S to vertex T such that each of them has the minimal number of alternative segments and the sets of alternative segments for each alternative evacuation route are disjoint.

`minimal_alternative_route(S,T)`

informally states that it is necessary to find the alternative evacuation route with the minimal number of alternative segments from vertex S to vertex T.

6 Language ALEP

As we mentioned above ALEP is an extension of language $\mathcal{K}$ [11]. Therefore, in this subsection we only introduce syntax and semantics of ALEP which are based on syntax and semantics of $\mathcal{K}$.

6.1 Syntax

Road set declaration The network of roads corresponds to a directed graph $G = (V, A)$. Evacuation routes has to be declared using **route declaration** of the form

`route r formed_by` $V[n]$ `connected_to` V_m

where r is a constant as defined in language $\mathcal{K}$, $V[n]$ corresponds to a sequence of $n \geq 1$ vertices $V_1, \ldots, V_n \in V$ and $V_j \in V$ such that there is a simple path from vertex V_1 to vertex V_j. If $j = n$, the `connected_to` part can be skipped.

A road out af an evacuation route has to be declared using **no_route declaration** of the form

`no_route` $V[n]$ `connected_to` V_j

where $V[n]$ corresponds to a sequence of $n \geq 1$ vertices $V_1, \ldots, V_n \in V$ and $V_j \in V$ such that there is a simple path from vertex V_1 to vertex V_j. If $j = n$, the

`connected_`
`to` part can be skipped.

Location rules Location rules are used to define locations and they are of the form:

`t at` V_j where `t` is a town identifier and $V_j \in V$.

Partial route queries Partial route queries are used to obtain the set of unblocked segments of an evacuation route and they are of the form

`partial_route(S,T)` where $S, T \in V$.

Alternative route queries When part of an evacuation route is blocked, it is necessary to define alternative routes. The alternative route queries are of the form:

```
all_alternatives_routes(S,T)
minimal_alternative_route(S,T)
```

where $S, T \in V$.

6.2 Semantics

Because of lack of space, in this paper *partial route queries* and *alternative route queries* are introduced as library functions. However, it is straightforward to define a formal semantic for them. In this section we present a mathematical structure useful to express from it in a uniform way different semantics: Semantic Contents. Among the possible semantics that can be defined with *Semantic Contents* are the semantics for *alternative route queries* and *partial route queries*. In order to model alternative evacuation routes is possible to use the semantics of *CR-Programs* [5] as in [28, 30] is presented. The semantics of CR-Programs is defined in terms of *minimal generalized answer sets*. In [5] the authors give a translation of CR-programs into abductive logic programs, which they call *hard reduct*, and the semantics are defined as the minimal generalized answer sets of the hard reduct of the program. In order to model partial evacuation routes is possible to use *k-minimal stable models* from [1].

Once we have constructed the Semantic Contents of a program we show how to find from it –in a uniform way – the variants of answer sets such as *minimal generalized answer sets* (the semantics for partial route queries), the standard definition of answer sets, W_s stable models, and a notion which is very similar to *k-minimal stable models* (the semantics for partial route queries). It is important to remark that k-minimal stable models are defined only for disjunctive programs, but the similar notion introduced in this paper is defined generically for any theory.

In [29] we defined the Semantic Contents of a program as a set of pairs obtained from the union of the program and a set of formulas, all of them satisfying certain properties. In this paper we introduce *compositionality* in answer sets via its semantic contents. Compositionality is inspired on the the interest [8, 9] of having a general principle on which both, the language and the meta-language for combining software components, have a formal mathematical semantics, thus providing firm foundations for reasoning about programs and program compositions.

It is important to emphasize that we can obtain the Semantic Contents of a program for every logic that satisfies few basic properties. Hence, our approach can be applied in other nonmonotonic languages such as Partial Order Programming. Semantic Contents is based in *intuitionistic* logic, for readers not familiar with this logic, we recommend [22, 21, 20] for further reading.

Now, we are going to present the definition of Semantic Contents as in [29] and the compositionality in answer sets via its semantic contents defined in this paper.

Definition 1. *Let P be a program (with a finite set of formulas), we define the semantic contents, denoted by $SC(P)$, as a set of pairs $< S, T >$ satisfying the following properties:*

1. *T is a deductively closed consistent extension of P (abbreviated as dcc extension of P) w.r.t. $\mathcal{L}$,*
2. *S is a set of formulas that $S \cup P \vdash T$ and*
3. *$\forall S' \subset S$, $S' \cup P \nvdash T$.*

The set S is called an abductive and the set T is called a scenario. If SC is a semantic contents, then $SC_S := \{X :< X, Y >\in SC\}$ and $SC_T := \{Y :< X, Y >\in SC\}$. Note that if P is inconsistent then SC is the empty contents. We write $SC_{\mathcal{L}}$ to denote the set of atoms that occur in SC. We have the following trivial lemmas.

Lemma 1. *Let P be a program over a signature $\mathcal{L}$. Then $th(P) = K(SC_T(P))$.*

Lemma 2. *Let P_1 and P_2 two programs over a signature $\mathcal{L}$. Then $th(P_1 \cup P_2) := K(SC_{1_T} \cap SC_{2_T})$.*

From the lemma 2 mentioned above and by the abuse of notation, we write $K(SC_1, SC_2) := K(SC_{1_T} \cap SC_{2_T})$, where SC_1 and SC_2 are two semantic contents. We also write $SC_T(SC_1, SC_2)$ to denote $SC_{1_T} \cap SC_{2_T}$.

Now we define an operator $+$ between semantic contents.

Definition 2. *Let SC_1 and SC_2 two semantic contents. Then $SC_1 + SC_2$ is a set of pairs of the form $< A \setminus K(SC_1, SC_2), T >$ such that $T \in SC_T(SC_1, SC_2)$ and $< A, T >\in SC_1$.*

It is easy to prove that if we choose $< A, T >\in SC_2$ in definition 2 then the defined $SC_1 + SC_2$ does not change. The following theorem affirms that we can have *compositionality* in answer sets via its semantic contents. It is important to remark that this theorem holds for every logic that satisfies few basic properties. The proof of Theorem 1 is presented in the appendix.

Theorem 1. *For every pair of programs P_1 and P_2, $SC(P_1 \cup P_2) = SC(P_1) + SC(P_2)$.*

The following lemma can be used to justify reductions of programs.

Lemma 3. *Let P and P' be two programs such that P is equivalent to P'. Then P and P' have the same semantic contents.*

Finding variants of answer sets from semantic contents We show how to obtain different semantics based on Answer sets such as: minimal generalized answer sets, the standard definition of answer sets, W_s stable models and a notion similar to k-minimal stable models. All this is done by using only the semantic contents of a program.

We start by defining a model and the semantics of a program P over a signature $\mathcal{L}$ in terms of $ASC_R(P)$, where $ASC_R(P)$ is a selected set of pairs from the semantic contents of P w.r.t. R a subset of $\mathcal{L}$.

Definition 3. *Let $SC(P)$ be the semantic contents of a program P, R be a subset of $\mathcal{L}$ and $ASC_R(P) := \{< X, Y >\in SC : X \subseteq (R \cup \neg\mathcal{L} \cup \neg\neg\mathcal{L})\}$. We define the following:*

1. *M is a partial model of $ASC_R(P)$ if exists $< X, Y >\in ASC_R(P)$ such that $M = literals(Y)$.*
2. *M is a model of $ASC_R(P)$ if M is a partial model of $ASC_R(P)$ and $literals(M)$ is complete.*
3. *A semantics of a program P, denoted as $SEM(P)$, as a set of partial models of $ASC_R(P)$.*

Hence, we have the following corollary of lemma 3.

Corollary 1. *Let P and P' be two programs such that P is equivalent to P'. Then P and P' have the same semantics.*

The following trivial lemma is about how to obtain the answer sets of a program using its semantic contents.

Lemma 4. *Let P be a program and M a model of $ASC_\emptyset(P)$. Then $pos(M)$ is an answer set of P iff there is a pair of the form $< Z, Y >\in ASC_\emptyset(P)$ such that $M = literals(Y)$.*

Now, let $EA \subseteq \mathcal{L}$ be a set of atoms that we call *explicit abductibles*. We define an ordering among entries of semantic contents as follows:Given a semantic contents SC, $e \in SC$, $e' \in SC$, we define $e <_{EA} e'$ if one of following cases occur:

1. $e'_T \subset e_T$
2. e_T is complete, e'_T is complete, $pos(e_S) \subseteq EA$, $pos(e'_S) \subseteq EA$, $pos(e_S) \subset pos(e'_S)$.

Note that $<_{EA}$ is a strict order over SC and it is straightforward to define this order in terms of the cardinality of the sets. Now, we use the ordering among entries of the semantic contents of a program to obtain the minimal generalized answer sets of a program. The proof of this lemma is in the appendix.

Lemma 5. *Let P be a program, $EA \subseteq \mathcal{L}$ and M be a model of $ASC_{EA}(P)$. Then $pos(M)$ is a minimal generalized answer set of P w.r.t. EA iff exists X such that $< X, th(M) >$ is a minimal entry in $ASC_{EA}(P)$ w.r.t. the ordering $<_{EA}$ and M is complete.*

Now, we have the following corollary of lemma 5. It shows the relationship between our definition of minimal generalized answer set and the definition of W_s stable model given in [19]. In W_s stable model is used a set inclusion order and a cardinality order.

Corollary 2. *Let P be a program, $EA \subseteq \mathcal{L}$ and M be a model of $ASC_{EA}(P)$. Then $pos(M)$ is a minimal generalized answer set of P w.r.t. EA iff $pos(M)$ corresponds to a W_s stable model of P.*

We have shown how to obtain two of the semantics based on Answer sets: the *minimal generalized answer sets* and *W_s stable models*. This was possible thanks to the definition of an ordering among entries of the semantic contents.

Now, we use the definition 3 of a partial model in order to construct the definition of a partial answer set in terms of its semantic contents. Partial answer sets is a semantics similar to k-minimal stable models [1].

Definition 4. *Let P be a program and M a partial model of $ASC_{\emptyset}(P)$. Then M is a partial answer set of P iff is false that exists M', a partial model of $ASC_{\emptyset}(P)$, such that $|M'| > |M|$.*

The following lemma shows that our definition of partial answer sets naturally reflects its relationship with the definition of answer sets.

Lemma 6. *Let P be a program and $M \subseteq \mathcal{L}$. If P is stable consistent (P has at least one answer set) then M is an answer set of P iff M is a partial answer set of P.*

7 Testing the Expressiveness of ALEP

In this section, we present a short example of an ALEP program in order to illustrate the use of language ALEP. It is worth to mention that ALEP is a declarative logic programming language. Hence, in an implementation of ALEP language all results of an ALEP program are automatically obtained when the input for this implementation is an ALEP program. As future work we plan to develop a DSS as an extension of a GIS in order to give support in developing evacuation plans in volcano Popocatepetl. The following example uses the directed graph presented in 1 that was represented as the *road set* of subsection 5.2.

```
fluents:
position(X) requires node(X). %to indicate current position
blocked(X) requires node(X), blocked(X). %to indicate that vertex X is blocked


actions:
%action travel by the same evacuation route
travel(P,Q) requires road(P,Q,X), route(X).


always:
% final node of segment is the new position after traveling
```

```
caused  position(Q) if travel(P,Q)),node(P),node(Q).

% initial node of segment is not the new position after traveling
caused -position(P) if travel(P,Q)),node(P),node(Q).

% it is not possible travel from P to Q if currently it is not in position P
nonexecutable travel(P,Q) if not position(P).

% it is not possible travel from P to Q if Q is blocked
nonexecutable travel(P,Q) if blocked(Q).
% initial conditions: start evacuation from  vertices 1, 11 and 13
% when vertex 3 (belonging to evacuation route 1) is blocked.
initially: position(1), position(11),position(13), blocked(3).

% the goal has three subgoals:
% (1) find all alternative evacuation routes from 1 to 6
% (2) find all alternative evacuation routes from 11 to 6
% (3) find all partial evacuation routes from 13 to 15
goal:  all_alternative_route(1,6), all_alternative_route(11,6),
       partial_route(13,15)  ?   (5).
```

By default, in language $\mathcal{K}$ actions are executed concurrently. As it was described before, language ALEP is an extension of language $\mathcal{K}$ hence ALEP allows action concurrence. The result of the ALEP program are the following plans:

```
Plan 1: travel((1,2),1), travel((11,10),1), travel((13,14),1).
        travel((2,5),2), travel((10,12),2), travel((14,15),2).
        travel((5,6),3), travel((12,6),3).

Plan 2: travel((1,2),1), travel((11,10),1), travel((13,14),1).
        travel((2,8),2), travel((10,12),2), travel((14,15),2).
        travel((8,9),3), travel((12,6),1).
travel((9,5),4).
        travel((5,6),5).
```

In both plans, action `travel` is executed concurrently. It is indicated that at the same time `i` (for `i` from 1 to 5) action `travel` should be executed three, two or one times. Moreover, as a result of vertex 3 is blocked and part of the goal is to find all alternative routes from 1 to 6 then there are obtained two plans. These two plans represent all alternative evacuation routes from vertex 1 to vertex 6 such that each of them has the minimal number of alternative segments and the sets of alternative segments for each alternative evacuation route are disjoint. The solution to the last subgoal, partial route(13,15), that indicates to find the set of unblocked segments from 13 to 15 in both plans is the same: travel from 13 to 14 at time 1 and travel from 14 to 15 at time 2. This solution corresponds to all segments of evcuation route 3 because there is not blocked segments in this route. We want to stress that the semantics of all_alternative_route sentence is given in terms of

minimal generalized answer sets using a set inclusion order and the semantics of partial_route is given in terms of *k-minimal stable models*.
If we replace the goal for
`goal: minimal_alternative_route(1,6) ? (5).`
then there is only one plan. This plan corresponds to the alternative evacuation route with the minimal number of alternative segments from vertex 1 to vertex 6.
`Plan 1: travel((1,2),1). travel((2,5),2). travel((5,6),3).`
The semantics of this program is also given in terms of *minimal generalized answer sets* but using an cardinality order. Since 3 is blocked, both options of finding alternative evacuation routes include roads that do not belong to the pre-established evacuation route. Now, if we replace the goal for `goal: partial_route(1,6) ? (5).` then the set of unblocked segments of the evacuation route is: `Unblocked segments: road(1,2), road(5,6).`
The semantics of this program is given in terms of *k-minimal stable models*.

8 Conclusions and Future Work

We have intruduced ALEP, an action language, which can be used to specify evacuation planning problems that can successfully generate evacuation plans for some situations in the context of a disaster sitaution like a Popocatepetl volcano eruption. However, there are still several ways to improve our work. At this point we are interested in pursuing two particular objectives: to generate a CBR component to complement the ASP planner, and to do further testing of ALEP, by defining a DSS.

As mentioned in Section 1, we plan to enhance our ASP based DSS with a CBR component. While the results obtained from the ASP planner have been satisfactory so far, the process may be very time consuming for particular situations. For this reason it may not always be possible to compute a new solution from scratch given a new real time situation that needs to be addressed almost immediately. As a result, it is necessary to combine ASP with some other technique that will allow our system to generate solutions quickly. CBR seems to be a natural choice since it is well suited to quickly generate "ball-park" solutions for new problems based on previously solved problems.

However, there are not too many real cases for our particular problem that we can use in order to create the initial case base for the CBR component. Therefore, our approach will be to initially populate our case base with artificially generated situations along with the solutions computed by the ASP planner for each of these situations. Several interesting problems arise at this point. First, it is necessary to identify all the attributes that are relevant for each situation, and then select those attributes that will compose our indices. Second, it is necessary to select the particular cases we want to solve in advance, so that they can cover the widest selection of possible cases, making it easy to adapt the solutions to solved cases to be applied to new ones. We have done some preliminary work in order to solve these problems [23]. We plan to incorporate some of the ideas developed in our previous CBR work, such as the critical regions used in the chess domain [14, 15], or the similarity metric used in [2] to our current problem. Furthermore we will analyze other systems such as those described in [26, 4] to search for other ideas that may be helpful. Eventually, we envision a CBR system, such as the one

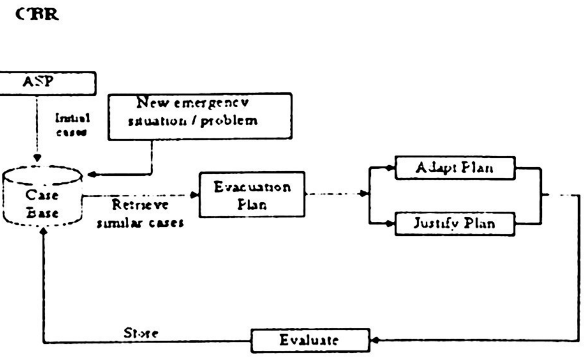

Fig. 2. CBR for planning evacuation

described in Figure 2. The initial cases will be provided by the ASP planner results. Later, when facing new situations in real time, the description of the emergency scenery will be used to retrieve evacuation plans for similar cases. The selected plan(s) will be used or adapted, and finally stored after an appropriate evaluation. We are exploring the possibility of combining the ideas presented in section 2.3 in order to generate a powerful and accurate similarity metric for disaster situations in the context of the Popocatepetl volcano. This can be accomplished following these steps: 1) Compute the degree of danger to which each region in the area comprising the volcano and its vicinity is exposed according to the current volcano and climatological conditions. This is analogous to the computation of influence in the context of chess, with the particularity that the shapes of the regions may have a much wider variation, and the computation of the Influence/Danger may be more complex than in chess. Particular attention must be placed on those areas which are densely populated or contain otherwise precious resources, which would be analogous to the computation of Influence regions around the kings in the chess domain, 2) Map these danger computations into different colors to be overlapped in the appropriate positions of the geographical area's images, 3) Use an Image Based Reasoner similar to the one presented in the previous section in order to compare different situations, 4) Incorporate this image based approach into an overall similarity metric that also includes other factors that may be relevant (e.g. availability of vehicles to carry out the evacuation).

References

1. Acosta. J.C.. Arrazola. J.. Osorio. M.: Making Belief Revision with LUPS. In Azuela. S.. Humberto. J.. Arroyo Figueroa. G. eds.: XI International Conference on Computing. CIC-IPN. México. D.F. (2002).
2. Aguilera. A.. Lazzeri. S.G.. Aguila. R.P.: Image Based Reasoning Applied to the Comparison of the Popocatepetl Volcano's Fumaroles. To appear in Workshop Proceedings of Deduction and Reasoning Techniques (2004).
3. ASP_Solver: Web location of DLVk: http://www.dbai.tuwien.ac.at/proj/dlv/K/.
4. Avesani. P.. Perini. A.. Ricci. F.: Interactive Case-Based Planning for Forest Fire Management. Applied Intelligence 13(1) (2000) 41–57.
5. Balduccini, M.. Gelfond. M.: Logic Programs with Consistency-Restoring Rules. In Doherty. P.. McCarthy. J.. Williams. M.-A. eds.: International Symposium on Logical Formalization of Commonsense Reasoning. AAAI 2003 Spring Symposium Series (2003).
6. Balduccini. M., Gelfond. M.. Nogueira. M.. Watson. R.: Planning with the USA-Advisor. In Kortenkamp. D. ed.: 3rd NASA International workshop on Planning and Scheduling for Space (2002).
7. Birkin. M.. Clarke. G.. Clarke. M.. Wilson. A.: Intelligent GIS. Local decisions and strategic planning. Geoinformation International (1996).
8. Brogi, A., Contiero. S., Turini. F.: Programming by Combining General Logic Programs. Journal of Logic and Computation 9(1) (1999) 7–24.
9. Brogi, A., Contiero, S., Turini, F.: The Use of Renaming in Composing General Programs. Logic-Based Program Synthesis and Transformation - 8th International Workshop. LOPSTR'98. Selected Papers. LNCS 1559. Springer-Verlag (1999) 124-142.
10. Calimeri, F., Dell'Armi, T., Eiter. T., Faber. W., Gottlob. G., Ianni, G., Ielpa. G., Koch. C., Leone, N. Perri. S.. Pfeifer. G.. Polleres. A.: The DLV System. In Flesca. S., Ianni, G. eds.: Proceedings of the 8th European Conference on Artificial Intelligence (2002).
11. Eiter, T., Faber, W., Leone, N.. Pfeifer, G., Polleres. A.: Planning under Incomplete Knowledge. In: Proceedings of the First International Conference on Computational Logic Springer-Verlag, London. UK (2000) 807–821.
12. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In Kowalski, R., Bowen, K. eds.: 5th Conference on Logic Programming MIT Press (1988) 1070–1080.
13. Hamacher, H.W., Tjandra. S.A.: Mathematical Modelling of Evacuation Problems: A State of Art. Institut Techno- und Wirtschaftsmathematik 24 (2001).
14. Lazzeri, S.. Heller, R.: Application of Fuzzy Logic and Case Based Reasoning to the generation of high-level advice in chesse. Advances in Computer Chess, University of Maastricht, The Netherlands 8 (1996) 251–267.
15. Lazzeri, S.: A Principle Based Chess Playing Program. To appear in Workshop Proceedings of Deduction and Reasoning Techniques (2004).
16. Leake, D.B.: Case-Based Reasoning: Expriences, Lessons, and Future Directions. Menlo Park, California: AAAI Press (1996).
17. Macias. J.L., Carrasco-Nunez, G., Delgado. H., Martin. A.L.. Siebe, C.. Hoblitt. R.. Sheridan. M.F. Tilling, R.I.: Mapa de Peligros del Volcan Popocatepetl. UNAM-CENAPRED, Map with explanation booklet, 14 p (1995).

18. Niemela, I., Simons, P.: Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In: Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR), Lecture Notes in Artificial Intelligence (LNCS) 1265 (1997) 420–429.
19. Osorio, M., Nieves, J.: W_{sc} - Stable semantics for propositional theories. In Sossa, Freeman, Vizcaino eds.: Proceedings of the International Conference CIC . México, D.F (2001) 319–328.
20. Osorio, M., Corona, E.: The A-Pol System. In: Proceedings 2nd International Workshop Answer Set Programming 78 (2003). http://CEUR-WS/Vol-78/.
21. Osorio, M., Navarro, J.A., Arrazola, J.: Safe beliefs for propositional theories. Journal of Pure and Applied Logic (2004).
22. Osorio, M., Navarro, J.A., Arrazola, J.: Applications of Intuitionistic Logic in Answer Set Programming. Theory and Practice of Logic Programming (TPLP) 4 (2004) 325–354.
23. Posada, N.: Tesis profesional de Licenciatura: CBR y toma de decisiones en el contexto de un GIS: Caso del volcan Popocatepetl. Universidad de las Américas, Puebla (2001).
24. Razo, A., Sol, D.: Standard 2D and 3D geospatial data formats for a Volcano GIS. Tercer Encuentro Internacional de Ciencias de la Computacion (2001) 737–744.
25. Sol, D., Razo, A.: Natural Hazards in the Popocatpetl Volcano Zone. ESRI International Conference, San Diego CA, USA (2001).
26. Terral, S., Avesani, P., Ricci, F., Martin, E.: CARICA:Cases Acquisition and Replay in Fire Campaign Ambience Project. http://sra.itc.it/projects/carica/.
27. Volcanic Emergency Management. UNDRO and UNESCO, United Nations, New York (1985).
28. Zepeda, C., Osorio, M., Sol, D.: Towards the use of Cr-rules and Semantic Contents in ASP for planning in GIS. Technical Report 2004-010, Université Lyon I (2004).
29. Zepeda, C., Osorio, M., Sol, D.: Modeling Evacuation Planning using A-Prolog Submmited to CONIELECOMP (2005).
30. Zepeda, C., Solnon, C., Sol, D.: Planning Operation: An extension of a Geographical Information System. LA-NMR 2004 CEUR Workshop proceedings 92 (2004).

A Appendix

Theorem 1

Proof. Let $K(SC_1, SC_2) = K(SC(P_1), SC(P_2))$, $SC_1 = SC(P_1)$ and $SC_2 = SC(P_2)$. Let $< X, Y > \in SC(P_1 \cup P_2)$. Then $X \cup (P_1 \cup P_2) \vdash Y$ and $\forall X' \subset X, X' \cup (P_1 \cup P_2) \nvdash Y$. Hence $(X \cup P_2) \cup P_1 \vdash Y$ and $X \cap P_1 = \emptyset$.
Thus, $\exists P' \subseteq P_2$ such that $< X \cup P', Y > \in SC_1$ (The reader can verify that: if $P' \subseteq P_2$ then $X \cup P' \cup P_1 \vdash Y$. Now if $\exists X' \subset X$ such that $X' \cup P' \cup P_1 \vdash Y$ then $X' \cup P_2 \cup P_1 \vdash Y$ but this contradicts the hypothesis). Since $Y \in SC_T(P_1) \cap SC_T(P_2)$ then $Y \in SC_T(SC_1, SC_2)$. Hence $< (X \cup P') \setminus K(SC_1, SC_2), Y > \in SC_1 + SC_2$. But $K(SC_1, SC_2) \vdash P_2$, so $< X \setminus K(SC_1, SC_2), Y > \in SC_1 + SC_2$. But $K(SC_1, SC_2) \cap X = \emptyset$, so $< X, Y > \in SC_1 + SC_2$.
Now, suppose $< X, Y > \in SC_1 + SC_2$. Also suppose that $< X, Y > \notin SC(P_1 \cup P_2)$ (to prove by contradiction).
Hence $\exists X'$ such that $X = X' \setminus K(SC_1, SC_2)$ and $< X', Y > \in SC_1$ (and so

$X' \cup P_1 \vdash Y$). Thus by set properties $X' \subseteq X \cup K(SC_1, SC_2)$ and by monotonicity (in logic) $X \cup K(SC_1, SC_2) \cup P_1 \vdash Y$, but $P_1 \cup P_2 \vdash K(SC_1, SC_2)$. Hence $X \cup P_1 \cup P_2 \vdash Y$.

Now, by $< X, Y > \notin SC(P_1 \cup P_2)$ we have 3 cases:

(1) Y is not a *dcc* extension of $P_1 \cup P_2$. So, $Y \notin SC_T(SC_1, SC_1)$ and so $< X, Y > \notin SC_1 + SC_2$, a contradiction.

(2) Y is a *dcc* extension of $P_1 \cup P_2$, but $X \cup P_1 \cup P_2 \not\vdash Y$. However we have already shown $X \cup P_1 \cup P_2 \vdash Y$, a contradiction.

(3) Suppose $X \cup P_1 \cup P_2 \vdash Y$, Y is a *dcc* extension of $P_1 \cup P_2$ and X does not satisfies third property of definition 1. Now we have two subcases:

(a) $X \cap K(SC_1, SC_2) = \emptyset$. But by construction of $SC_1 + SC_2$, all pairs $< X', Y' > \in SC_1 + SC_2$, satisfy that $X' \cap K(SC_1, SC_2) = \emptyset$. Hence $< X, Y > \notin SC_1 + SC_2$, contradiction.

(b) $X \cap K(SC_1, SC_2) = \emptyset$. So, we assume that $\exists X' \subset X$ such that $X' \cup P_1 \cup P_2 \vdash Y$. Then $(X' \cup P_2) \cup P_1 \vdash Y$. Hence $\exists X'', X'' \subseteq X' \cup P_2$ such that $< X'', Y > \in SC_1$. So $< X'' \setminus K(SC_1, SC_2), Y > \in SC_1 + SC_2$.

We know $X'' \subseteq X \cup P_2$ and so by set properties $X'' \setminus K(SC_1, SC_2) \subseteq (X \cup P_2) \setminus K(SC_1, SC_2)$. That is, $X'' \setminus K(SC_1, SC_2) \subseteq X \setminus K(SC_1, SC_2)$. To prove that the contention is strict, take $e \in X$ and $e \notin X'$. Then by hypothesis (case b), we conclude that $e \notin X''$. We also know that $e \notin K(SC_1, SC_2)$. Hence $e \notin X'' \setminus K(SC_1, SC_2)$ but $e \in X \setminus K(SC_1, SC_2)$. So, $X'' \setminus K(SC_1, SC_2) \subset X \setminus K(SC_1, SC_2)$. Since $X \cap K(SC_1, SC_2) = \emptyset$ then $X'' \setminus K(SC_1, SC_2) \subset X$. Implying that $< X, Y > \notin SC_1 + SC_2$, contradiction.

So, in all cases we arrived to a contradiction. $\Diamond$

Lemma 5

Proof. Let M be a model of $ASC_{EA}(P)$. Suppose that $pos(M)$ is a minimal generalized answer set of P w.r.t. EA that implies that $\exists \Delta \subseteq EA$ such that $pos(M)$ is an answer set of $\Delta \cup P$ and $\exists \Delta' \subset \Delta$ such that $\Delta' \cup P$ has an answer set of P.

Hence $P \cup \Delta \cup \neg pos(M) \cup \neg\neg pos(M) \vdash_I pos(M)$ then $P \cup \Delta \cup neg(M) \cup \neg\neg pos(M) \vdash_I pos(M) \cup neg(M)$. Let N be the minimal subset of $neg(M) \cup \neg\neg pos(M)$ such that $P \cup \Delta \cup N \vdash_I pos(M) \cup neg(M)$.

Clearly $< \Delta \cup N, th(M) >$ is an entry of $ASC_{EA}(P)$ (Proof by contradiction of $\exists \Delta' \subset \Delta \cup N$ such that $\Delta' \cup P \vdash_I th(M)$: Suppose that $\exists \Delta' \subset \Delta \cup N$ such that $\Delta' \cup P \vdash_I th(M)$. Suppose $\Delta' := \Delta'_P \cup \Delta'_N$ where $\Delta'_P := \{x \in \Delta : x \in \mathcal{L}\}$ and $\Delta'_N := \{x \in \Delta : x \in \neg\mathcal{L}\}$ then we have $\Delta'_P \cup \Delta'_N \cup P \vdash_I th(M)$. Hence $P \cup \Delta'_P$ has an answer set, contradiction).

Now we prove (by contradiction) that the entry is minimal. Let $e := < \Delta \cup N, th(M) >$. Suppose that exists e' such that $e' <_{EA} e$ then we have two cases:

1) $e_T \subset e'_T$ then e'_T is inconsistent, contradiction.

2) e_T is complete, e'_T is complete, $pos(e_S) \subseteq EA$, $pos(e'_S) \subseteq EA$, $pos(e_S) \subset pos(e'_S)$. Then $P \cup pos(e'_S) \cup neg(e'_S) \vdash_I e'_T$. Hence $P \cup pos(e'_S)$ has an answer set, contradiction.

Now, suppose that exists X such that $< X, th(M) >$ is a minimal entry in $ASC(P)$ w.r.t. the ordering $<_{EA}$ such that M is complete and $pos(X) \subseteq EA$.

Then $X \subset \mathcal{L} \cup \neg\mathcal{L} \cup \neg\neg\mathcal{L}$ and $P \cup X \vdash_I th(M)$.
Hence $P \cup X \vdash_I M$.
Then $P \cup pos(X) \cup neg(X) \vdash_I pos(M) \cup neg(M)$.
Then $P \cup pos(X) \cup neg(M) \vdash_I pos(M) \cup neg(M)$.
Then $P \cup pos(X) \cup neg(M) \cup \neg\neg pos(M) \vdash_I pos(M) \cup neg(M)$.
Let $\Delta = pos(X)$. Then $P \cup \Delta \cup neg(M) \cup \neg\neg pos(X) \vdash_I pos(M)$.
Hence $pos(M)$ is an answer set of $P \cup \Delta$ such that $\Delta \subseteq EA$. We prove (by contradiction) that $pos(M)$ is a minimal generalized answer set of P w.r.t. EA. Suppose that $\exists \Delta' \subset \Delta$ such that $P \cup \Delta'$ has an answer set (then exists an M' such that $pos(M')$ is an answer set of $P \cup \Delta'$). Hence $P \cup \Delta' \cup neg(M') \cup \neg\neg pos(M') \vdash pos(M')$. Then exists $< \Delta' \cup X, th(M') > \in ASC_{EA}(P)$ where $\Delta' \subset \Delta$ and $\Delta \subset EA$ then $\Delta' \subseteq EA$. Contradiction. $\Diamond$