Towards the use of Semantic Contents in ASP

Mauricio Osorio and Claudia Zepeda

Universidad de las Américas, CENTIA.
Sta. Catarina Mártir, Cholula, Puebla
72820 México
{josorio, sc098382}@mail.udlap.mx
http://mailweb.udlap.mx/∼josorio

Abstract. We introduce the notion of Semantic Contents of a program. Using only the semantic contents of a program it is possible to obtain different semantics based on Answer Set Programming such as the standard definition of answer sets, W_s stable models, minimal generalized answer sets and a new notion similar to k-minimal stable models. One of our main theorems says that we can have compositionality in answer sets via its semantic contents. The theorem removes and makes abstraction of all details specific to answer set programming. Thus, we obtain a theorem that has its application in other nonmonotonic languages such as Partial Order Programming.
Keywords: Answer sets, Compositionality, Intuitionistic logic.

1 Introduction

Answer Set Programming (ASP), also known as A-Prolog, is a logic programming language under the answer set semantics defined by Gelfond and Lifschitz in 1987 [GL88]. Currently, in order to expand the applicability of A-Prolog, have been developed several extensions of A-Prolog such as ASET-Prolog [Gel02], A-POL [NO01], etc. These extensions add to A-Prolog features like cardinality constraints, weight constraints, weak constraints, sets, aggregates, consistency-restoring rules[BG03], Partial Order clauses, etc. At the same time, the answer sets semantics has been extended itself in different directions such as the standard definition of answer sets [GL88], minimal generalized answer sets [KM90,Gel91], W_s stable models [ON01] and a notion which is very similar to k-minimal stable models from [OA02].

With the aim of having a mathematical structure that may express in a uniform way the different semantics mentioned above, we introduce the notion of Semantic Contents of a program.

We define 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. 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.

According to [SH01], a semantics is defined as compositional if, and only if, in it the different semantic attributes of each complex expression are a function of the meanings of its component expressions plus the way they are combined into the complex expressions. Hence, the authors in [SH01] also present one of the possible meanings of the principle of compositionality: "the applicability of a semantic attribute A to a complex expression E is determined by the applicability of all and sundry semantic attributes to the component expressions of E plus the structure of E in terms of these component expressions".

Moreover, in [ABT99a,ABT99b] is presented the interest 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.

Inspired on these notions about compositionality, one of our main theorems in this work says that we can have *compositionality* in answer sets via its semantic contents.

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 minimal generalized answer sets, the standard definition of answer sets, W_s stable models, and a notion which is very similar to k-minimal stable models. 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. We also present how the Semantic Contents and the minimal generalized answer sets of a program are preserved when we apply the popular transformations from [SBZ01,BDFZ01] for logic programs used to simplify the structure of programs and reduce their size.

Additionally, we know that *Planning* and *diagnosis* are tasks that are carry on by the public administration. The use of maps plays a fundamental roll in these tasks as a form of representation, visualization and analysis. Geographic Information Systems (GIS) is the technology that allows the automatic manipulation of digital maps. However, due to the large amount of data, operations of diagnostic and planning become very difficult to deal with. In our future work, we plan to work on a situation related to the Popocatepetl volcano problem: the creation of evacuation plans to put out of risk people living in the risk zones. We propose to use the Semantic Contents of a program since it represents a mathematical structure from which we can obtain in a uniform way, using only orderings among entries or filters of entries, the different answer set semantics needed to perform planing and diagnostic.

The paper is structured as follows. We start with background material needed to understand the definition of Semantic Contents. Next, we introduce the definition of Semantic Contents and give one of our main theorems about compositionality. Next, we show how we can find variants of answer sets from Semantic Contents and how we can apply some transformation to logic programs and the semantic contents of a program is preserved. Finally, we present future work and our conclusions.

2 Background

We use the language of propositional logic in order to describe rules within logic programs. Formally we consider a language built from an alphabet consisting of

atoms: $p_0, p_1, \ldots$ connectives: $\wedge, \vee, \rightarrow, \bot$
auxiliary symbols: '(', ')', '.'.

Where $\wedge, \vee, \rightarrow$ are 2-place connectives and $\bot$ is a 0-place connective. Formulas are defined as usual in logic. The formula $\neg F$ is introduced as an abbreviation of $F \rightarrow \bot$, and $F \leftrightarrow G$ as an abbreviation of $(F \rightarrow G) \wedge (G \rightarrow F)$. A theory is a finite set of propositional formulas.

A signature $\mathcal{L}$ is a finite set of atoms. We assume a fixed set in this paper. $FORM(\mathcal{L})$ is the set of formulas in that can be constructed from $\mathcal{L}$. If F is a formula then the *signature* of F, denoted as $\mathcal{L}_F$, is the set of atoms that occur in F. A *literal* is either an atom a (a positive literal) or a negated atom $\neg a$ (a negative literal).

Given a set of formulas $\mathcal{F}$, we define $\neg\mathcal{F} = \{\neg F \mid F \in \mathcal{F}\}$. We write $X \subset Y$ to denote that $X \subseteq Y$ and $X \neq Y$. Given a set of literals S, $pos(S)$ denotes the set of positive literals in S and $neg(S)$ denotes the set of negative literals in S. Given a set of formulas S, we write $literals(S)$ to denote the set of literals in S. We assume the use of intuitionistic logic in all the paper.

The set of theorems of P (namely the set $\{\alpha : P \vdash \alpha\}$, is denoted as $th(P)$). For a given set of sets of formulas R, we write $K(R)$ to denote the set of formulas that belong to every set in R. For two given sets of formulas M and P we write $P \Vdash_{\mathrm{I}} M$ if P is consistent and P proves every element in M in intuitionistic logic.

A finite set of formulas P is *consistent* [Men87] if there is no a formula A such that both A and $\neg A$ are theorems of P. A finite set of formulas P is said to be *complete* [Men87] if, for any closed formula A of P, either $P \vdash A$ or $P \vdash \neg A$. A finite set of formulas P' is said to be an *extension* of a finite set of formulas P [Men87] if every theorem of P is a theorem of P'. A finite set of formulas P is said to be *deductively closed* if for every formula A such that $P \Vdash_{\mathrm{I}} A$ then $A \in P$.

If P is a program then a generalized answer set [KM90,Gel91] is an answer set $M(\Delta)$ of $P \cup \Delta$ where $\Delta \subseteq A$ and A is a set of atoms; $M(\Delta_1) < M(\Delta_2)$ if $\Delta_1 \subset \Delta_2$. An answer set is a *minimal* if it is minimal with respect to this ordering.

We adopt the characterization of answer sets as the one given in [ONA03]. Namely, M is an *answer set* of P iff $P \cup \neg\widetilde{M} \cup \neg\neg M \Vdash_{\mathrm{I}} M$.

3 Semantic Contents

With the aim of having a mathematical structure useful to express from it in a uniform way different semantics, in these section we introduce the notion of Semantic Contents of a program.

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 second lemma 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.

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

Now, the following lemma will 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.*

4 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.*

We have the following trivial lemma 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, consider that we are given a set of atoms $EA \subseteq \mathcal{L}$ 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.

Now, we use the ordering among entries of the semantic contents of a program to obtain the minimal generalized answer sets of a program.

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 [ON01].

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 [OA02].

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.*

The following example, shows how our definition of partial answer sets agrees with the definition of answer sets when the program is stable consistent.

Example 1. Let P be the program:

$$\begin{array}{l} a \leftarrow \neg b. \\ e \leftarrow \neg d. \\ d \leftarrow \neg e. \end{array}$$

We can verify that $\{e, \neg d, \neg a\}, \{\neg e, d, \neg a\}, \{\neg a\}, \{e, \neg d, \neg b, a\}, \{\neg e, d, \neg b, a\}$, $\{\neg b, a\}, \{e, \neg d\}, \{\neg e, d\}$ are the partial models of $ASC_{\emptyset}(P)$. However, only $\{e, \neg d, \neg b, a\}, \{\neg e, d, \neg b, a\}$ are partial answer sets of P. Furthermore, we can verify that these partial answer sets are the answer sets of P, according to Lemma 6.

Now, we present a program that only has partial answer sets.

Example 2. Let P be the program:

$$a \leftarrow \neg b.$$
$$c \leftarrow \neg c.$$

$\{a\}$ and $\{\neg b, a\}$ are the partial models of $ASC_{\emptyset}(P)$ but the only partial answer set of P is $\{\neg b, a\}$. We can verify that P does not have answer sets.

In spite of k-minimal stable models and partial answer sets are similar, it is important to remark that k-minimal stable models are defined only for disjunctive programs and partial answer sets are defined generically for *any theory*.

4.1 Reductions of programs

Now we will study some popular transformations that can be applied to logic programs.

From lemma 3, we can validate that the semantics contents of a program is preserved when we apply one of the following transformations [SBZ01] for logic programs: RED$^-$ or SUB or DSuc or TAUT.

Moreover, we can show how RED$^+$ and Dloop transformations [BDFZ01] can be applied to a program P and the minimal generalized answer sets of the program P are preserved. We highlight the fact that the set of explicit abductibles, $EA \subseteq \mathcal{L}$, is included in the following definitions in a strategic way in order to take them into account when RED$^+$ or Dloop transformations are applied.

Definition 5. *Let P be a program. Let EA a subset of L_P. The RED$^+$ transformation replace a rule $\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$ by $\mathcal{A} \leftarrow \mathcal{B}^+, \neg(\mathcal{B}^- \cap HEAD(P) \cap EA)$.*

Definition 6. *Let P be a program. Let EA a subset of L_P. Let $unf(P) = \mathcal{L}_P \setminus MM(Definite(P \cup EA))$. Then we define*
Dloop$(P) = \{\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^- \in P \mid \mathcal{B}^+ \cap unf(P) = \emptyset\}$.

Lemma 7. *Let P and P' be two programs. Let EA a set of explicit abductibles. If P' is obtained of P by an application of Dloop or RED$^+$ then P and P' have the same minimal generalized answer sets w.r.t. EA.*

5 Future work: Planing and diagnostic in GIS problems using Semantic Contents

We know that A-Prolog is used for planning, diagnosing, consistency checking, and other tasks. Particularly, in [BG02] is shown how a "diagnostic module" finds possible explanations of observations when the union of a program and the corresponding observations is inconsistent (i.e. the program does not possess an answer set). However, checking consistency and finding a diagnosis as in [BG02] implies several problems. One of these problems is that we do not have way to declaratively specify preferences between possible diagnoses by performing extra observations. To avoid this problem, in [BG03] is expanded A-Prolog by consistency-restoring rules with preferences (cr-rules). Cr-rules programs are closely related to abductive logic programs [KM90,Gel91]. Moreover, the semantics of cr-rules programs is given by the notion of *minimal generalized answer set*. Cr-rules use minimal generalized answer sets in order to select the sets of cr-rules needed to restore consistency of a program. Additionally, to apply cr-rules to restore the consistency of a program is necessary to know the possible causes of inconsistency. However, it is not always possible to know which are all the possible causes of inconsistency. Hence, we propose to use *partial answer sets* to infer from a program the largest amount of knowledge in order to give support to find a possible cause of inconsistency. Finally, this new cause of inconsistency can help us to restore consistency using the cr-rules.

We propose to use the Semantic Contents of a program since it represents a mathematical structure from which we can obtain in a uniform way, using only orderings among entries or filters of entries, the different answer set semantics needed to perform planing and diagnostic. Among the different answer set semantics needed we have: answer sets, partial answer sets and minimal generalized answer sets.

6 Conclusions

In order to have a mathematical structure useful to express from it in a uniform way the different answer set semantics, in these paper we have introduced the notion of Semantic Contents of a program. It is important to emphasize that we can obtain the Semantic Contents of a program for every logic that satisfies few basic properties. Inspired on different notions about compositionality, one of our main theorems in this work says that we can have *compositionality* in answer sets via its semantic contents. The theorem removes and makes abstraction of all details specific to answer set programming. Thus, we obtain a theorem that has its application in other nonmonotonic languages such as Partial Order Programming.

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 minimal generalized answer sets, the standard definition of answer sets, W_s stable models, and partial answer sets which is a notion similar to k-minimal. It is important to remark that k-minimal stable models are defined only for disjunctive programs, but in this paper partial answer sets are defined generically for any theory. We also want to remark that our definition of partial answer sets is different from the definition of partial stable model semantics introduced in [Prz91].

We also present how the Semantics Content or the minimal generalized answer sets of a program are preserved when we apply the popular transformations from [SBZ01,BDFZ01] for logic programs used to simplify the structure of programs and reduce their size.

References

[ABT99a] S. Contiero A. Brogi and F. Turini. Programming by combining general logic programs. *Journal of Logic and Computation*, 9(1):7–24, 1999.

[ABT99b] S. Contiero A. Brogi and F. Turini. The use of renaming in composing general programs. In *Logic-Based Program Synthesis and Transformation - 8th International Workshop, LOPSTR'98, Selected Papers*, pages 124–142. LNCS 1559, Springer-Verlag, May 1999.

[BDFZ01] Stefan Brass, Jürgen Dix, Burkhard Freitag, and Ulrich Zukowski. Transformation-based bottom-up computation of the well-founded model. *Theory and Practice of Logic Programming*, 1(5):497–538, 2001.

[BG02] Marcello Balduccini and Michael Gelfond. Diagnostic reasoning with a-prolog. *Journal of Theory and Practice of Logic Programming (TPLP)*, 2002. (to appear).

[BG03] Marcello Balduccini and Michael Gelfond. Logic programs with consistency-restoring rules. In Patrick Doherty, John McCarthy, and Mary-Anne Williams, editors, *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, Mar 2003. (to appear).

[Gel91] Michael Gelfond. Epistemic approach to formalization of commonsense reasoning. Technical Report TR-91-2, University of Texas at El Paso, 1991.

[Gel02] Michael Gelfond. Representing knowledge in a-prolog. In Antonis C. Kakas and Fariba Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408, pages 413–451. Springer Verlag, Berlin, 2002.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of ICLP-88*, pages 1070–1080, 1988.

[KM90] Antonis C. Kakas and Paolo Mancarella. Generalized stable models: a semantics for abduction. In *Proceedings of ECAI-90*, pages 385–391. IOS Press, 1990.

[Men87] Ellliott Mendelson. *Introduction to Mathematical Logic*, pages 35, 68. Wadsworth & Brooks/Cole Advanced Books & and Software, 1987.

[NO01] Juan Carlos Nieves and Mauricio Osorio. Stratified partial-order logic programming. In *Conference on Artificial Intelligence (AI2001)*, volume 2056 of *Lecture Notes in Artificial Intelligence (LNCS)*, pages 225–235, 2001.

[OA02] M. Osorio and J.C. Acosta. Making belief revision with lups. *XI International Conference on Computing CIC-2002*, II:377–388, Mexico City, 2002.

[ON01] M. Osorio and J. Nieves. w_{sc} - stable semantics for propositional theories. In Vizcaino Ed: Sossa, Freeman, editor, *Proceedings of the International Conference CIC'2001*, pages 319–328. México, D.F, 2001.

[ONA03] M. Osorio, J. A. Navarro, and J. Arrazola. Applications of intuitionistic logic in answer set programming. *Journal of Theory and Practice of Logic Programming (TPLP)*, 2003. (to appear, http://www.cwi.nl/projects/alp/TPLP/).

[Prz91] T. C. Przymusinski. Stable semantics for disjunctive programs. *Special issue of the New Generation Computing Journal*, 9(3):401–424, 1991.

[Raf00] A. Raffaeta'. *Spatio-Temporal Knowledge Bases in a Constraint Logic Programming Framework with Multiple Theories*. PhD thesis, Universita' Degli Studi di Pisa Dipartimento di Informatica, 2000.

[SBZ01] B. Freitag S. Brass, J. Dix and U. Zukowski. Transformation-based bottom-up computation of the well-founded model. *Theory and Practice of Logic Programming*, 1(5):497–538, 2001.

[SH01] G. Sandu and J. Hintikka. Aspects of compositionality. *Journal of Logic, Language and Information*, 10(1):49–61, 2001.

7 Appendix: Proofs.

Proof of Lemma 5.

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 $\nexists \Delta' \subset \Delta$ such that $\Delta' \cup P$ has an answer set of P.

Hence $P \cup \Delta \cup \widetilde{\neg pos(M)} \cup \neg\neg pos(M) \Vdash_{\mathrm{I}} pos(M)$ then $P \cup \Delta \cup neg(M) \cup \neg\neg pos(M) \Vdash_{\mathrm{I}} pos(M) \cup neg(M)$. Let N be the minimal subset of $neg(M) \cup \neg\neg pos(M)$ such that $P \cup \Delta \cup N \Vdash_{\mathrm{I}} pos(M) \cup neg(M)$.

Clearly $< \Delta \cup N, th(M) >$ is an entry of $ASC_{EA}(P)$ (Proof by contradiction of $\nexists \Delta' \subset \Delta \cup N$ such that $\Delta' \cup P \Vdash_{\mathrm{I}} th(M)$: Suppose that $\exists \Delta' \subset \Delta \cup N$ such that $\Delta' \cup P \Vdash_{\mathrm{I}} th(M)$. 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 $\Delta'_P \cup \Delta'_N \cup P \Vdash_{\mathrm{I}} th(M)$. 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 $P \cup pos(e'_S) \cup neg(e'_S) \Vdash_{\mathrm{I}} e'_T$. 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 $P \cup X \Vdash_{\mathrm{I}} th(M)$.
Hence $P \cup X \Vdash_{\mathrm{I}} M$.
Then $P \cup pos(X) \cup neg(X) \Vdash_{\mathrm{I}} pos(M) \cup neg(M)$.
Then $P \cup pos(X) \cup neg(M) \Vdash_{\mathrm{I}} pos(M) \cup neg(M)$.
Then $P \cup pos(X) \cup neg(M) \cup \neg\neg pos(M) \Vdash_{\mathrm{I}} pos(M) \cup neg(M)$.
Let $\Delta = pos(X)$. Then $P \cup \Delta \cup neg(M) \cup \neg\neg pos(X) \Vdash_{\mathrm{I}} pos(M)$.
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 $P \cup \Delta' \cup neg(M') \cup \neg\neg pos(M') \Vdash_{\mathrm{I}} pos(M')$. Then exists $< \Delta' \cup X, th(M') > \in ASC_{EA}(P)$ where $\Delta' \subset \Delta$ and $\Delta \subset EA$ then $\Delta' \subseteq EA$. Contradiction. □