Generating maximal models using the stable model semantics

Juan Carlos Nieves[1] and Mauricio Osorio[2]

[1] Universitat Politècnica de Catalunya
Software Department (LSI)
c/Jordi Girona 1-3, E08034, Barcelona, Spain
jcnieves@lsi.upc.edu

[2] Universidad de las Américas, CENTIA,
Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@gmail.com

Abstract. Given a propositional formula X, we present a mapping that constructs a general program P, such that the maximal models of X correspond to the stable models of P, after intersecting each stable model with the relevant atoms. Maximal models are of interest for different applications.

1 Introduction

Generating maximal models of a propositional formula is of interest in different applications. For instance, it has been shown that in argumentation theory the preferred semantics of an argumentation framework can be expressed in terms of maximal models of a propositional formula [1]. Also, to compute maximal models in model-preference default inference [5], it is an essential tool.

Given a propositional formula X, we present a mapping that constructs a general program P, such that the maximal models of X correspond to the stable models of P, after intersecting each stable model with the relevant atoms. We present this mapping as a sequence of basic steps in a way that they represent a simple methodology that can be applied to any problem of this nature. In fact, we can infer the *maximal models* of a formula by using *disjunctive answer set solvers e.g.,* DLV [2]. Nowadays, there are fast answer set solvers *e.g.,* DLV [2], SMODELS [6], which have contributed to extend the applications of Answer Set Programming (ASP).

The rest of the paper is divided as follows: In §2, some basic concepts of logic programs and stable model semantics are presented. In §3, our characterization of the maximal models in terms of stable models is presented. Finally in the last section, we present our conclusions.

2 Background

In this section, we present the syntax of a valid logic program in ASP and the definition of a stable model.

2.1 Logic Programs: Syntax

The language of a propositional logic has an alphabet consisting of

(i) proposition symbols: $p_0, p_1, ...$
(ii) connectives : $\vee, \wedge, \leftarrow, \neg, \bot, \top$
(iii) auxiliary symbols : (,).

where $\vee, \wedge, \leftarrow$ are binary connectives, $\neg$ is an unary connective and $\bot, \top$ are zero-ary connectives. The proposition symbols and $\bot$ stand for the indecomposable propositions, which we call *atoms*, or *atomic propositions*. A literal is an atom, a, or the negation of an atom $\neg a$. Given a set of atoms $\{a_1, ..., a_n\}$, we write $\neg\{a_1, ..., a_n\}$ to denote the set of literals $\{\neg a_1, ..., \neg a_n\}$. Formulas are constructed as usual in logic. A theory T is a finite set of formulas. By $\mathcal{L}_T$, we denote the signature of T , namely the set of atoms that occurs in T.

A general clause, C, is denoted by $a_1 \vee \ldots \vee a_m \leftarrow l_1, \ldots, l_n$,[3] where $m \geq 0$, $n \geq 0$, each a_i is an atom, and each l_i is a literal. When every literal l_i is an atom, the clause is considered positive. When $n = 0$ and $m > 0$, the clause is an abbreviation of $a_1 \vee \ldots \vee a_m \leftarrow \top$, where $\top$ is $\neg\bot$. When $m = 0$ the clause is an abbreviation of $\bot \leftarrow l_1, \ldots, l_n$. Clauses of this form are called constraints (the rest, non-constraint clauses). A general program, P, is a finite set of general clauses. A positive general program P is a finite set of positive general clauses. A given program $\{\alpha_1, ..., \alpha_n\}$ is also represented as $\{\alpha_1; ...; \alpha_n\}$ to avoid ambiguities with the use of the comma in the body of the clauses.

We point out that whenever we consider logic programs our negation $\neg$ corresponds to the default negation *not* used in Logic Programming. Also, it is convenient to remark that in this paper we are not using at all the so called strong negation used in ASP.

2.2 Stable models semantics

First, to define the stable model semantics, let us define some relevant concepts.

Definition 1. *Let T be a theory. An interpretation I is a mapping from $\mathcal{L}_T$ to $\{0, 1\}$ meeting the conditions:*

1. $I(a \wedge b) = min\{I(a), I(b)\}$,
2. $I(a \vee b) = max\{I(a), I(b)\}$,
3. $I(a \leftarrow b) = 0$ *if and only if* $I(b) = 1$ *and* $I(a) = 0$,
4. $I(\neg a) = 1 - I(a)$,
5. $I(\bot) = 0$.
6. $I(\top) = 1$.

It is standard to provide interpretations only in terms of a mapping from $\mathcal{L}_T$ to $\{0, 1\}$. But then it is easy to prove that it is unique by virtue of the definition by recursion.

[3] $l_1, \ldots, l_n$ represents the formula $l_1 \wedge \cdots \wedge l_n$.

An interpretation I is called a model of P if and only if for each clause $c \in P$, $I(c) = 1$. Given a theory T and a formula α, we say that α is a logical consequence of T, denoted by $T \models \alpha$, if for every model I of T we have $I(\alpha) = 1$. A theory (or a formula) is consistent if it admits a model, otherwise it is said to be inconsistent. It is well known that $T \models \alpha$ iff $T \cup \{\neg\alpha\}$ is inconsistent.

It is possible to identify an interpretation with a subset of a given signature. For any interpretation, the corresponding subset of the signature is the set of all atoms that are true with respect to the interpretation. Conversely, given an arbitrary subset of the signature, there is a corresponding interpretation defined by specifying that the mapping assigned to an atom in the subset is equal to 1 and otherwise to 0. We use this view of interpretations freely in the rest of the paper.

We say that a model I of a theory T is a minimal model if it does not exist a model I' of T different of I such that $I' \subset I$. Maximal models are defined in the analogous way.

By using answer set programming, it is possible to describe a computational problem as a logic program whose answer sets correspond to the solutions of the given problem. The stable model semantics was first defined in terms of the so called *Gelfond-Lifschitz reduction* [3] and it is usually studied in the context of syntax dependent transformations on programs. The following definition of an answer set for general programs generalizes the definition presented in [3] and it was presented in [4].

Let P be any general program. For any set $S \subseteq \mathcal{L}_P$, let P^S be the general program obtained from P by deleting

(i) each rule that has a formula $\neg l$ in its body with $l \in S$, and then
(ii) all formulæ of the form $\neg l$ in the bodies of the remaining rules.

Clearly P^S is positive, then S is a stable model of P if and only if S is a minimal model of P^S. However, note that P^S may contain constraints.

3 Maximal models via Stable models

In this section, we provide a method for obtaining maximal models of a propositional theory in terms of stable models of a general program.

Definition 2. *Let $\mathcal{L}$ be a signature. Let $\mathcal{L}'$ be a new signature (namely $\mathcal{L} \cap \mathcal{L}' = \emptyset$) of the same cardinality as $\mathcal{L}$ that of course admits a bijective function f from $\mathcal{L}$ to $\mathcal{L}'$. We say in this case that $\mathcal{L}'$ is a copy-signature of $\mathcal{L}$. In addition, we write $f(a)$ (or also $a^\bullet$) to denote the image of a under f. For a given set of atoms N, we write $f(N)$ to denote the set $\{f(a) : a \in N\}$.*

Note that if $\mathcal{L}'$ is a copy-signature of $\mathcal{L}$ then also $\mathcal{L}$ is a copy-signature of $\mathcal{L}'$.

To explain the methodology of this section, we use a simple formula as a running example. Let our theory be $T_1 := \{a \vee b, \neg b\}$. Note that $\{a\}$ is the unique maximal model of our theory.

One can establish an important relationship between maximal and minimal models. The proof is straightforward.

Lemma 1. *Let T be a theory with signature $\mathcal{L}$ and $\mathcal{L}'$ be a copy-signature of $\mathcal{L}$. By $g(T)$ we denote the theory obtained from T by replacing every occurrence of an atom x in T by $\neg f(x)$. Then M is a maximal model of T iff $f(\mathcal{L} \setminus \mathcal{M})$ is a minimal model of $g(T)$.*

Let us come back to our example. Note that $g(T_1) := \{\neg a^\bullet \vee \neg b^\bullet, \neg\neg b^\bullet\}$. The complement of $\{a\}$ (maximal model of T_1) *w.r.t.* to $\{a, b\}$ is $\{b\}$. Now, $f(\{b\})$ is $\{b^\bullet\}$, that is precisely the minimal model of $g(T_1)$.

The second step is conceptually very simple. Transform the new theory $R := g(T)$ into a logically equivalent positive general program, denoted by $general(R)$ or just P. This is of course always possible using well known basic logical transformations. In our case $g(T_1)$ is transformed to $P_1 := \{\perp \leftarrow a^\bullet, b^\bullet; b^\bullet\}$. Clearly, $g(T)$ and P have the same set of minimal models. Moreover, the set of stable models of P are the same as the set of minimal models of $g(T)$.

For the third and final step we need to state the following lemma.

Lemma 2. *Let S be a general program of the form $S_1 \cup S_2$, where S_1 is a positive general program with signature $\mathcal{L}$ and $S_2 := \{x^\bullet \leftarrow \neg x : x \in \mathcal{L}\}$ such that $\mathcal{L}'$ is a copy-signature of $\mathcal{L}$. Then M is a stable model of S iff there exists a minimal model N of S_1 such that $M = N \cup f(\mathcal{L} \setminus N)$.*

Let us come back again to our example. Recall that $P_1 := \{\perp \leftarrow a^\bullet, b^\bullet; b^\bullet\}$. The signature of P_1 is $\{a^\bullet, b^\bullet\}$ and the copy-signature is $\{a, b\}$ Now consider the general program $P_2 := P_1 \cup \{a \leftarrow \neg a^\bullet; b \leftarrow \neg b^\bullet\}$. The unique stable model of P_2 is $\{b^\bullet, a\}$. We could obtain this model as follows: obtain a minimal model of P_1, getting $\{b^\bullet\}$. Now, the complement of this model is $\{a^\bullet\}$, and $f(\{a^\bullet\})$ is $\{a\}$. Finally we obtain $\{b^\bullet\} \cup \{a\}$ as desired. Hence if this model is intersected with the original signature, we obtain our desired maximal model of the original theory.

Let us summarize our approach. We are given $T_1 := \{a \vee b, \neg b\}$. We want to obtain a maximal model of T_1. We first construct the program $g(T_1) := \{\neg a^\bullet \vee \neg b^\bullet, \neg\neg b^\bullet\}$. Then we compute the associate positive general program $P_1 := \{\perp \leftarrow a^\bullet, b^\bullet; b^\bullet\}$. Now we add $P_2 := P_1 \cup \{a \leftarrow \neg a^\bullet; b \leftarrow \neg b^\bullet\}$. Compute a stable model of P_2 using DLV. We obtain $\{b^\bullet, a\}$. Intersect it with the original signature and we get $\{a\}$. This is a maximal model of T_1, as desired.

Acknowledgements

We are grateful to anonymous referees for their useful comments. Juan Carlos Nieves wants to thank CONACyT for his PhD Grant.

4 Conclusions

We have presented a methodology to transform a problem of generating maximal models into the problem of obtaining stable models of a general program. As a consequence we can generate the *maximal models* of a formula by using *disjunctive answer set solvers e.g.,* DLV.

References

1. P. Besnard and S. Doutre. Checking the acceptability of a set of arguments. In *Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004),*, pages 59–64, June 2004.
2. S. DLV. Vienna University of Technology. http://www.dbai.tuwien.ac.at/proj/dlv/, 1996.
3. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
4. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
5. B. Selman and H. A. Kautz. Model-preference default theories. *Artificial Intelligence*, 45(3):287–322, 1990.
6. S. SMODELS. Helsinki University of Technology. http://www.tcs.hut.fi/Software/smodels/, 1995.