Contributions to the Stable Model Semantics of Logic Programs with Negation

Stefania Costantini
Dipartimento di Scienze dell'Informazione
Universita' degli Studi di Milano
via Comelico 39/41, I-20135 Milano (Italy)
costanti@imiucca.csi.unimi.it

Abstract

Understanding the stable model semantics is an important topic in Logic Programming and Non-Monotonic Reasoning. In fact, stable models are closely related to extensions in default logic, and to other non-monotonic formalisms like TMS and abduction. In this paper we propose two new explicit characterizations of the stable model semantics. First, we specify a simple requirement for checking stability of a minimal model. Second, we characterize stable models in terms of their "difference" with respect to the set of true atoms of the well-founded model of the program. This provides a method for: efficiently computing stable models whenever the Herbrand base is finite; in many cases when the Herbrand base is infinite, computing the basic sets of assumptions on which the stable models are based. The method may help ensure correctness of any procedural semantics based on stable models, like for instance abduction.

1 Introduction

The stable model semantics for logic programs with negation has received great interest since its introduction in [7], [8] because it has an elegant and simple definition, and is closely related to the autoepistemic approach to non-monotonic reasoning [7]. It has also been discussed, for the following reasons.

First, there are programs without stable models, and programs with several stable models. At the beginning in fact, the attention was mainly on "well-behaved" programs, i.e. programs with a unique stable model, which in many cases coincides with the well-founded model [24]. More recently however, programs with several stable models have been discovered to have important connections to TMS [3] (stable models of a logic program correspond to grounded extensions of a suitably related TMS [14] [5]) and to abduction [10] [11].

Second, because the definition of stable models is implicit and non-constructive. It is however possible to calculate stable models (at least for programs with a finite Herbrand base). In fact, several procedures have been proposed, based on: TMS or ATMS [6], [18], [9]; extensions/modifications of the well-founded model construction [19], [12], [21]; linear programming techniques [1]; program transformation and case analysis [20].

In this paper we propose two original characterizations of the stable model semantics, different from [8]. The main aim is that of contributing to a better understanding of the stable model semantics, in the direction of identifying the syntactic properties of programs which determine the existence and the number of stable models. The first new characterization specifies a simple requirement for checking stability of minimal models. The second one explicitly characterizes stable models in terms of their "difference" with respect to the set of true atoms of the well-founded model of the program. This difference can be identified by an analysis of the syntactic form of the program, and, in many cases, when the set of undefined atoms in the well-founded model is finite, it can actually be calculated. Thus, this new characterization gives a method for: (i) computing the stable models if the Herbrand base of the program is finite; (ii) computing the basic sets of assumptions on which the stable models are based, and checking for the existence of stable models, in many cases when the Herbrand base is infinite, but the set of undefined atoms is finite. This may be useful for any procedural semantics based on stable models. We will outline in fact how the method can help ensure correctness of the abductive procedure introduced in [4] also on non-locally-stratified programs.

After Section 2, which reports some useful definitions, in Section 3 we introduce the new criterion for checking stability of a minimal model, called Σ-test.

In Section 4 we show that any method for calculating the well-founded model can be used for calculating a specific set of minimal models, candidate to be checked for stability (possibly by the Σ-test). These results are preliminary to those of the following Sections, where the particular characterization adopted for minimal models which are possibly stable is usefully exploited. Thus, we do not try a comparison with other methods to compute stable models using computation of minimal models as a first step, like those in [1], which are aimed at practical efficient implementation.

Sections 5 and 6 present the main results. In Section 5, we prove that the problem of finding the stable models of a program P is equivalent to that of finding the stable models of a suitable subprogram. This is because they constitute the "difference" between the stable models of the original program, and the set of true atoms of its well-founded model. This property was (independently) discovered and exploited in [21], where the stable models of this subprogram are computed by means of an intelligent "branch-and-bound" algorithm. We may notice that there is a basic difference between the point of view adopted here and in [21], with respect to all the other approaches appeared in the literature, which compute the stable models starting from the whole program. In Section 6, we show how the subprogram can be further reduced, and these "difference sets" (called *s-generators*) explicitly characterized, thus explicitly characterizing the stable models of the program.

In Section 7 we state a condition for actually computing s-generators of stable models. This is, to the best of our knowledge, the weakest condition ever stated in the literature for computing or characterizing the stable model semantics. Finally, we discuss how the characterization of stable models via s-generators is related to the sets of abductive hypotheses and the abductive procedure introduced in [4].

2 Preliminary Definitions and Observations

In the rest of this paper we will consider general programs, i.e. Horn-clause programs with negation. As base reference for semantics of general logic programs we take [15], which is entirely devoted to a systematic exposition and comparison of the various proposed semantics. As a base reference for semantics of positive logic programs we take [13].

Clauses in a general program have an atom as conclusion, and a conjunction of literals as conditions, where a literal is either an atom (positive literal) or the negation of an atom (negative literal). When saying "a program" we mean (unless explicitly specified otherwise) a general program. In the following, let P be a program. We say that an atom A *occurs positively* or vice versa *occurs negatively* in P if it appears as a positive literal or vice versa in a negative literal in the conditions of a clause in P. Negation is indicated with *not*. By "interpretations" and "models" we mean Herbrand interpretations and Herbrand models respectively. Since an Herbrand interpretation is a model of a program P if and only if it is a model of its ground instantiation [15] (Cor. 4.1), we will assume, without further mention (and as it is customary in the literature) that every program P has already been instantiated. That is, by P we mean the (possibly infinite) propositional theory consisting of all ground instances of clauses from P. By H_P and B_P we indicate respectively the Herbrand Universe and the Herbrand Base of P.

In this paper we will extensively refer to the Stable Model Semantics [8] and to the Well-Founded Model semantics [24] of general programs.

The Well-Founded model (WFM for short) of P is unique, and is in general 3-valued. Following [17], [15] we indicate the WFM of P (or WFM_P for short) with $< T(P); F(P) >$, or with $< T; F >$ if there is no ambiguity about P. $T \subseteq B_P$ is the set of atoms which are true w.r.t. the WFM, $F \subseteq B_P$ the set of atoms which are false. All atoms belonging to $B_P - (T \cup F)$, have truth value undefined. We conventionally indicate the set of undefined atoms with $U(P)$, or simply U.

The Stable Model semantics is not based on a unique model. In fact, a program admits in general more than one stable model. Stable models are minimal models of P [15] (Prop. 6.4, Gelfond and Lifschitz).

A program with a two-valued WFM is called saturated [15] (Def. 6.22). We call a two-valued WFM a $2-WFM$. For a saturated program, the $2-WFM$ coincides [15] (Th. 6.21, Van Gelder, Ross and Schlipf) with the unique stable model, and is therefore a minimal model of P.

Various equivalent definitions of Well-Founded Model appeared in the literature [24], [17], [15], [2]. In the following, when mentioning "the definition of WFM" we will refer to the definition below, which rephrases [15] (Def. 6.19-6.20) or also [17] (Def. 3.2-3.4).

Constructive Definition of WFM *The $WFM < T; F >$ of a program P is the least fixpoint of the sequence of partial models M_α, where*

$$
\begin{array}{l}
M_0 =< \emptyset; \emptyset > \\
M_{\alpha+1} =< T_{\alpha+1}; F_{\alpha+1} >
\end{array}
$$

$$
\begin{array}{l}
T_{\alpha+1} = T_\alpha \cup T_{M_\alpha} \ \textit{where} \\
\qquad T_{M_\alpha} = \{A : A \in \bigcup_{n<\omega} T_{M_\alpha} \uparrow n \textit{ and } A \notin T_\alpha\} \\
\qquad\qquad \textit{given } T_{M_\alpha} \uparrow 0 = \emptyset \textit{ and} \\
\qquad\qquad A \in T_{M_\alpha} \uparrow n \textit{ iff} \qquad\qquad \textit{(1)} \\
\qquad\qquad \textit{there exists a clause } A : -L_1, \ldots, L_m \textit{ in } P \\
\qquad\qquad \textit{such that } \forall i \leq m \textit{ either } L_i \textit{ is true in } M_\alpha \textit{ or} \\
\qquad\qquad L_i \textit{ is an atom and } L_i \in T_{M_\alpha} \uparrow (n-1)
\end{array}
$$

$$
\begin{array}{l}
F_{\alpha+1} = F_\alpha \cup F_{M_\alpha} \ \textit{where} \\
\qquad F_{M_\alpha} = \{A : A \in \bigcap_{n<\omega} F_{M_\alpha} \downarrow n \textit{ and } A \notin F_\alpha\} \\
\qquad\qquad \textit{given } F_{M_\alpha} \downarrow 0 = H_P \textit{ and} \\
\qquad\qquad A \in F_{M_\alpha} \downarrow n \textit{ iff} \qquad\qquad \textit{(2)} \\
\qquad\qquad \textit{one of the following conditions holds:} \\
\qquad\qquad\qquad \textit{1. There is no clause in } P \textit{ with conclusion } A. \\
\qquad\qquad\qquad \textit{2. For every clause } A : -L_1, \ldots, L_m \textit{ in } P \\
\qquad\qquad\qquad\quad \exists i \leq m \textit{ such that either} \\
\qquad\qquad\qquad\qquad \textit{(i) } L_i \textit{ is false in } M_a \textit{ or} \\
\qquad\qquad\qquad\qquad \textit{(ii) } L_i \textit{ is an atom and } L_i \in F_{M_\alpha} \downarrow (n-1) \quad \Box
\end{array}
$$

According to the above definition, subsets of T and F are computed at each step of the construction. The size of the subsets (i.e., the degree of approximation of T and F) increases step by step until no more new true or false atoms can be computed, and therefore the fixpoint is reached. In fact, in a given step, T_{M_α} and F_{M_α} are the *new* true and false atoms that can be computed starting from the approximation reached in the previous step (given by the sets T_α and F_α). Thus, in the fixpoint, we have $M_\alpha =< T; F >$, and $T_{M_\alpha} = T_{<T;F>} = \emptyset$ and $F_{M_\alpha} = F_{<T;F>} = \emptyset$. Notice also that, in the construction of T_{M_α} and F_{M_α}, the entire current approximation is computed, by means of a fixpoint computation. For T_{M_α}, the computation starts from $T_{M_\alpha} \uparrow 0 = \emptyset$ and consists in the union of the partial results. For F_{M_α}, the computation starts from $F_{M_\alpha} \downarrow 0 = H_P$ and consists in the intersection of the partial results. For notational convenience however, only the new true and false atoms are finally selected. In the fixpoint of the whole sequence, i. e. when $T_{M_\alpha} = T_{<T;F>} = \emptyset$ and $F_{M_\alpha} = F_{<T;F>} = \emptyset$, we have (by applying the definition) $\bigcup_{n<\omega} T_{<T;F>} \uparrow n = T$ and $\bigcap_{n<\omega} F_{<T;F>} \downarrow n = F$.

The following Definitions and Proposition present some basic useful properties of the Well-Founded model. These properties are already known (at least (i)-(iii)), but it is useful for what follows to state them explicitly. Given two sets $S1$ and $S2$, by $S1 - S2$ we mean the set composed of all the elements of $S1$ which are not elements of $S2$.

Definition 2.1 *Let $A \in B_P$. A support set S for A is a finite set of clauses of P such that:*

a) exactly one clause in S has conclusion A;

b) for every B occurring positively in some clause in S, there exists a clause in S with conclusion B. □

Definition 2.2 *Let $A \in B_P$ and S be a support set for A. Let C be the clause in S with conclusion A, and D be any atom different from A which occurs in some clause in S. S is acyclic if*

a) there is no clause in S where A occurs positively

b) $S - \{C\}$ is an acyclic support set for D. □

An acyclic support set for A is simply a set of clauses which allow the derivation of A, whenever the negative conditions are true.

Proposition 2.1 *Let $< T; F >$ be the WFM of P, and U the set of undefined atoms. Let $A \in B_P$.*

(i) $A \in T$ iff there exists a clause in P
$A :- B_1, \ldots, B_m, notC_1, \ldots, notC_n,\ m \geq 0, n \geq 0$, *such that* $\{B_1, \ldots, B_m\} \subseteq T$ *and* $\{C1, ..., Cn\} \subseteq F$.

(ii) $A \in F$ iff either there is no clause in P with conclusion A, or for every clause in P
$A :- B_1, \ldots, B_m, notC_1, \ldots, notC_n, m \geq 0, n \geq 0$
either $\{B_1, \ldots, B_m\} \cap F \neq \emptyset$ *or* $\{C1, ..., Cn\} \cap T \neq \emptyset$.

(iii) $A \in U$ iff there exists a clause in P with conclusion A, and for every such clause, of the form
$A :- B_1, \ldots, B_m, notC_1, \ldots, notC_n, m \geq 0, n \geq 0$,

either there exists $I1 \subseteq \{B_1, \ldots, B_m\}, I1 \subseteq U$,
or there exists $I2 \subseteq \{C1, ..., Cn\}, I2 \subseteq U$,
where:

(a) $I1 \cup I2 \neq \emptyset$
(b) $\{B_1, \ldots, B_m\} - I1 \subseteq T$ and $\{C1, ..., Cn\} - I2 \subseteq F$

(iv) For every $A \in T$ there exists an acyclic support set S for A.

(v) For every $A \in B_P$ such that every support set for A is cyclic, $A \in F$.

Proof: By the Definition of WFM, $< T; F >$ is the least fixpoint of the sequence M_α. As observed in the comments to the Definition, in the fixpoint we have $\bigcup_{n<\omega} T_{<T;F>} \uparrow n = T$ and $\bigcap_{n<\omega} F_{<T;F>} \downarrow n = F$. Then, cases (1) and (2) in the Constructive Definition of WFM exactly correspond to (i) and (ii).

The $only - if$ part of (iii) also follows immediately. Assume the if part of (iii) does not hold. If there is no clause with conclusion A, then by (ii) $A \in F$. If (a) does not hold, i.e. $I1 \cup I2 = \emptyset$, then either (i) or (ii) must be the case. If (b) does not hold, then by (ii) $A \in F$.

Assume finally that there exists $A \in T$ for which (iv) does not hold. Since $A \in T$, clearly A has a support set. Then, every possible support set must be cyclic. This means, it consists of a set of clauses $A : - \ldots, E_1, \ldots,$ $E_1 : - \ldots, E_2, \ldots,$ $\cdots$, $E_r : - \ldots, A, \ldots$. By the Definition of T, for A to belong to T it must be $\{E_1, \ldots, E_r\} \subseteq T$ for at least one such set, say S. Notice however that clauses in S are still satisfied in $< T; F >$ if $\{A, E_1, \ldots, Er\} \subseteq F$. This is a contradiction, since $< T; F >$ is a minimal 3-valued model of P [15]. Therefore it follows that, if every support set for A is cyclic, then A must be false, which is point (v). □

Below is the specification of the Stable Model semantics. The following definitions were introduced in [8], and are reported in [15] (Def. 6.17-6.18).

Definition (The Gelfond-Lifschitz Operator). *Let I be a 2-valued interpretation of P. A GL-transformation of P modulo I is a new program P/I obtained from P by performing the following two reductions:*

1. *removing from P all clauses which contain a negative premise $notA$ such that $notA$ is false in I (equivalently, $A \in I$);*

2. *removing from the remaining clauses those negative premises $notA$ such that $notA$ is true in I (equivalently, $A \notin I$).*

P/I is clearly a positive Horn-clause program, with Least Herbrand Model J. Let $\Gamma(I) = J$. □

Fixed points of the operator G are minimal models of P. The stable model semantics is defined as follows.

Definition *A 2-valued interpretation I of P is called a Stable Model of P if $\Gamma(I) = I$.* □

It may be useful to recall the constructive definition of the Least Herbrand Model LM_P of a positive program [13] (Van Emden and Kowalski).
$LM_P = T_P \uparrow \omega = \bigcup_{n<\omega} T_P \uparrow n, where$
$T_P \uparrow 0 = \emptyset$,
$T_P \uparrow (n+1) = T_P(T_P \uparrow n)$, and
$T_P(I) = \{A : A : -B_1, \ldots, B_m$ is a clause in P, and $\{B_1, \ldots, B_m\} \subseteq I\}$

3 Observations on the Stable Model Semantics

It is useful to state a simple property of Γ that, to the best of our knowledge, has never appeared in the literature.

Theorem 3.1 *Let M be a model of P. Then $\Gamma(M) \subseteq M$.*

Proof: Since $\Gamma(M)$ is the least Herbrand model of the positive program P/M, $\Gamma(M) = T_{P/M} \uparrow \omega$. The proof is by induction on n.
Base. $n = 1$. $\forall A \in T_{P/M} = 1, A \in M$

For A to belong to $T_{P/M} \uparrow 1$, there exists a clause K in P
$A : -notC1, \ldots, notCk, k \geq 0, \{C1, \ldots, Ck\} \cap M = \emptyset$ Then, for K to be satisfied in M it must be $A \in M$.
Ind. step Assume that $\forall A \in T_{P/M} \uparrow n, A \in M$.

For A to belong to $T_{P/M} \uparrow (n+1)$, there exists a clause K in P
$A : -B_1, \ldots, B_m, notC1, \ldots, notCk, k \geq 0,$
$\{B_1, \ldots, B_m\} \subseteq T_{P/M} \uparrow n, \{C1, \ldots, Ck\} \cap M = \emptyset$ Since, by the induction hypothesis, $\{B_1, \ldots, B_m\} \subseteq M$, for K to be satisfied in M it must be $A \in M$.
□

Stable models are based [7], [8] on autoepistemic logic, which is *grounded in premises*. I.e., no conclusion can be drawn that is not soundly derivable from a set of premises (facts of the program). Then, assume that the only way of deriving A is by a clause $A : -\ldots notC \ldots$, which is logically equivalent to $A \vee C : -\ldots$ In a stable model, if C is true then A cannot be true. In fact, since A depends on the negation of a true atom, the conclusion A is not grounded in premises. This means in practice that C implies $notA$. In other words, in a stable model the disjunction $A \vee C$ is interpreted as an exclusive or (*ex-or*). All this clearly extends to indirect dependencies. We may notice that, if any atom A in a program depends (directly or indirectly) on $notA$, and there is no other possible derivation for A, then the program has no stable models, since $A\, ex - or\, A$ cannot possibly hold in any model.

Example 3.1 *Given the program:*
p:-not p.
the clause is interpreted as
p ex-or p.
Therefore, there are no stable models. □

Example 3.2 *Given the program:*
p:-not p.
p:-not a.
$\{p\}$ *is a stable model. In fact, since there is the possibility of deriving p from not a, then the first clause simply means* $p \vee p$*, which is true in* $\{p\}$. $\square$

Starting from these observations, we are able to introduce a characterization of the stable model semantics in an alternative way with respect to [8]. Some preliminary definitions are in order.

Definition 3.1 *Let M be a minimal model of P. A clause K of P*
$A : -B_1, \ldots, B_m, notC_1, \ldots, notC_n \; m \geq 0, n \geq 0$
is coherent in M (or in short M-coherent) iff
$A \notin \{B_1, \ldots, B_m, C_1, \ldots, Cn\}$, $\{B_1, \ldots, B_m\} \subseteq M$,
and $\{C_1, \ldots, C_n\} \cap M = \emptyset$.
K is called an M-coherent clause for A. $\square$

Definition 3.2 *Let M be a minimal model of P.* $A \in B_P$ *is called coherent in M (or M-coherent) iff there exists an M-coherent clause for A in P.* $\square$

Definition 3.3 *Let M be a minimal model of P.* $A \in M$ *is coherently supported in M iff there exists an acyclic support set S for A composed of M-coherent clauses. We call S an M-coherent support set.* $\square$

Whenever there is no ambiguity about M, we will often say "coherent" instead of "M-coherent" and "coherently supported" instead of "coherently supported in M".

Example 3.3 *Consider the following program.*
$a : -b.$ $\quad K1$
$b : -a.$ $\quad K2$
$a : -not\, b.$ $\quad K3$
Given the minimal model {a,b}, it is easy to see that atom a is not coherently supported. In fact, since clause K3 is not coherent, any possible coherent support set S for a must contain K1. {K1} is not a support set, since it violates condition b of Definition 2.1. {K1,K2} is a support set but it is cyclic (see Definition 2.2). Similarly it is possible to conclude that b is not coherently supported. $\square$

Example 3.4 *Given the program:*
$p : -not\, p.$ $\quad K1$
$p : -not\, a.$ $\quad K2$
the atom p is coherently supported with respect to the model $\{p\}$*, since {K2} is a coherent support set.* $\square$

Example 3.5 *Consider the following program.*

$a :- b.$ $K1$
$b :- c.$ $K2$
$c :- a.$ $K3$
$b :- not\, d.$ $K4$

With the minimal model {a,b,c}, the atom b is coherently supported by {K4}, the atom a by {K1,K4}, and c by {K3,K1,K4}. If K4 were instead b:-not a, no atom would have been coherently supported w.r.t. {a,b,c}. □

We are now able to prove the following.

Theorem 3.2 *A minimal model M of P is a stable model iff (stability condition) $\forall A \in M$ the following conditions hold.*

(a) A is coherent in M.

(b) if A occurs negatively in P, then A is coherently supported in M.

Proof: Let M be a stable model and $A \in M$. Assume the thesis does not hold. Assume first that point (a) does not hold. If there is no clause in P with conclusion A, then there is no clause in P/M with conclusion A. By the definition of Γ, $A \notin \Gamma(M)$ and thus M cannot be stable. If every clause $A :- B_1, \ldots, B_m, notC_1, \ldots, notC_n$ in P is not M-coherent, then one of the following is the case. $\{C_1, \ldots, C_n\} \cap M \neq \emptyset$: in this case, by the definition of GL-transformation, there is no clause in P/M with conclusion A. By the definition of Γ, $A \notin \Gamma(M)$ and thus M cannot be stable. $\exists B_i \notin M$, $i \leq m$: in this case, for every clause in P/M with conclusion A, that clause cannot be used to derive A, and thus A does not belong to the least model of P/M. This means $A \notin \Gamma(M)$, and thus M cannot be stable. Assume now that point (b) does not hold. This implies that there is no way of deriving A in P/M, and thus M cannot be stable, since $A \notin \Gamma(M)$.

Assume the stability condition holds. By Theorem 3.1, $\Gamma(M) \subseteq M$. We prove that $M \subseteq \Gamma(M)$. For every $A \in M$, point (a) of the hypothesis and the definition of the GL-operator imply there is a clause K in P/M $A :- B_1, \ldots, B_m, A \notin \{B_1, \ldots, B_m\}$, $\{B_1, \ldots, B_m\} \subseteq M$. Clearly, K is the counterpart in P/M of a coherent clause of P. For simplicity, we will call also K "coherent". If $m = 0$, obviously $A \in \Gamma(M)$. Otherwise, we can iterate the same reasoning for each $B_i, i \leq m$, and one of the following two cases hold. (i) A is derivable in P/M. I.e., for every B_i, by applying a coherent clause and again iterating the reasoning, we at last reach facts of P/M. In this case, $A \in \Gamma(M)$. (ii) A is not derivable in P/M, and then $A \notin \Gamma(M)$. This means, for every possible choice of coherent clauses, A depends (directly or indirectly) on an atom C involved in a circularity: i.e, C depends on C1 which depends on C (where possibly $A = C$ or $C1 = C$). Since all the selected clauses are coherent, all of $A, C, C1 \in M$. But this cannot be the case. In fact, if neither of $A, C, C1$ occur negatively in P, then $A, C, C1 \in M$ implies M is not minimal. If some of $A, C, C1$ occurs negatively in P, then such a circularity is forbidden by condition (b) of the hypothesis. Then, we can conclude $\Gamma(M) = M$, which means that M is stable. □

Corollary 3.1 *If P does not contain positive circularities, then a minimal model M of P is a stable model iff $\forall A \in M$ A is coherent in M.* □

Example 3.6 *Let P be the following program.*
a:-b.
b:-a.
a:-not b.
The only minimal model is $\{a,b\}$, which does not satisfy the stability condition. In fact the atom b, which occurs negatively, is not coherently supported. It is easy to verify by applying the GL-operator that $\{a,b\}$ is not stable □

Theorem 3.2 provides a method for checking stability, i.e. verifying the stability condition, alternative to the GL-transformation. We call this check Σ-test. The Σ-test is very efficient for loop-free programs, and in general is quite handy for proofs. For instance, from Theorem 3.2 it follows immediately that the WFM of a saturated program is stable, since it is minimal, and satisfies the Σ-test by construction (see Proposition 2.1).

The limitation is that the Σ-test is applicable to minimal models only. In the next Section, we characterize a set of minimal models of P, among which are the stable models. As a first step, we remind the reader that WFM_P is contained in any stable model M of P. In fact, as proved in [24], (Cor.5.7), for every stable model M of P, $T(P) \subseteq M$, $F(P) \subseteq B_P - M$. A relevant well-known consequence of this property is that stable models of P can be obtained by suitably assigning truth values true/false to the atoms in U.

4 Calculating Minimal Models via Generators

In this Section we show that any method for calculating the Well-Founded Model of a program P can be used for calculating all the minimal models of P which are supersets of $T(P)$, among which are the stable models of P. These are not the main results of the paper, but are preliminary results, necessary for what follows. We call any model M of P such that $T(P) \subseteq M$ a *WFM-compatible* model. It is easy to see that every literal which is true in WFM_P is true in any WFM–compatible model M of P.

Let $P1$ be the subprogram of P consisting of the clauses composed of literals which are either true or false in WFM_P. Precisely:

Definition 4.1 *Let P1 be obtained from P by selecting all clauses*
$A : -B_1, \ldots, B_m, notC_1, \ldots, notC_n, m \geq 0, n \geq 0$
such that $\{A, B_1, \ldots, B_m, C_1, \ldots, C_n\} \subseteq T(P) \cup F(P)$ □

Lemma 4.1 *T(P) is a minimal model of P1.*

Proof: Since WFM_P is a minimal 3-valued model of P, it is also a 3-valued model of $P1$. The 3-valued valuation function on which the definition of WFM is based [15], (Def. 4.3) coincides with the classical 2-valued truth valuation, as far as undefined atoms are not involved. Therefore, since clauses of $P1$ do not contain (by definition) undefined atoms, $T(P)$ is a two-valued model of $P1$. Assume $T(P)$ is not minimal. Then, $\exists A \in T(P)$ such that $T(P) - A$ is still a model. By Proposition 2.1, $A \in T(P)$ implies there is a clause K with conclusion A in P, where the conditions of K do not include A, and are true in WFM_P. Then, K is in $P1$. Clearly the conditions of K are still true in $T(P) - A$, which implies K is not satisfied in $T(P) - A$. Therefore, $T(P) - A$ cannot be a model, i.e. $T(P)$ is a minimal model of $P1$. $\square$

Definition 4.2 *Let $N \subseteq B_P$. We indicate by $P.N$ the program obtained by adding to P the atoms of N as facts (unit clauses).* $\square$

Notice that $B_P = B_{P.N}$. It is useful for what follows to prove that, if $N \subseteq U$, then the true and false atoms of WFM_P are still true and false (respectively) in $WFM_{P.N}$. That is, previous conclusions are not affected if some undefined atoms are assumed to be true.

Lemma 4.2 *Let $N \subseteq U$. $T(P) \subseteq T(P.N)$ and $F(P) \subseteq F(P.N)$*

Proof: By Proposition 2.1 (points (i) and (ii)), every atom in T and F is supported by at least one clause not depending of atoms in U, and thus not depending of atoms in N. Therefore, the truth value of atoms in T(P) and F(P) cannot be affected by adding atoms of N as facts. $\square$

Definition 4.3 *A set $G \subseteq U$ such that $P.G$ is saturated, is called a generator.* $\square$

Theorem 4.1 *If G is a generator, then T(P.G) is a model of P.*

Proof: Since $T(P.G)$ is a minimal model of $P.G$, which contains P as a proper subprogram, then it is also a model of P (though in general not minimal). $\square$

Given a generator G, we say that the model $MD = T(P.G)$ of P is G–generated, or equivalently that G is a generator for MD. Below we show that every WFM–compatible model M of P has at least one generator, which is $G = M - T(P)$. This means that, if we assume as facts those atoms in M which are undefined in WFM_P, then the well-founded model of the resulting program is saturated, and the set of true atoms coincides with M.

Theorem 4.2 *For every WFM–compatible model M of P, G=M–$T(P)$ is a generator for M and $T(P.G) = M$.*

Proof: By Lemma 4.2, atoms true and false in WFM_P are still true and false in $WFM_{P.G}$. For every $A \in G$, $A \in T(P.G)$ since A is a fact in $P.G$. It remains to prove that every other undefined atom becomes false, i.e. for every $D \in U - G$, $D \in F(P.G)$. By Proposition 2.1 (point (iii)), every clause with conclusion D has at least one undefined condition. Let such a clause, say K, be:
$D : -B_1, \ldots, B_m, notC_1, \ldots, notC_n, m \geq 0, n \geq 0,$
Since M is a model for P, K is satisfied in M. Also, M is WFM–compatible, i.e. it contains all the atoms of T(P), and no atom of F(P). Since $D \notin M$, one of the following must be the case. (1) There exists C_i, $i \leq n$, $C_i \in M$. Since M is WFM–compatible, it must be $C_i \in U$, and thus $C_i \in G$. Therefore, $notC_i$ is false in $WFM_{P.G}$, and consequently also D is false. (2) There exists B_j, $j \leq m$, $B_j \notin M$. Since M is WFM–compatible, it must be $B_j \in U$, and in particular $B_j \in U - G$. Then we can iterate for B_j the same reasoning as for D. □

Corollary 4.1 *For every WFM–compatible model M of P, $\exists G \subseteq U$ such that $T(P.G) = M$.* □

Given a WFM–compatible model M of a program P, the set $G = M - T(P)$ will be called the *maximal generator* for M. Maximal generators are very important. In fact, given WFM_P, finding maximal generators means finding WFM–compatible models of P without further calculation (just set union).

Below we show how any minimal WFM–compatible model M of P can however in general be obtained by a generator smaller than $M - T(P)$.

Example 4.1 *Consider the following program P.*
c:-d.
a:- not b.
b:- not a.
e:- a.
d.
This program has WFM $<\{c,d\};\emptyset>$ where U=$\{a,b,e\}$. P has the two (stable) models
M1 = $<\{c,d,a,e\},\{b\}>$
M2 = $<\{c,d,b\},\{a,e\}>$
It is easy to see that M2 = $T(P.\{b\})$.
M1 = $T(P.\{a,e\})$ but also M1 = $T(P.\{a\})$, since e is obtained as a consequence of a. □

Definition 4.4 *A generator S is minimal if $\forall S' \subseteq S$, S' is not a generator.* □

Theorem 4.3 *If S is a minimal generator, then $M=T(P.S)$ is a minimal model of P.*

Proof: Suppose $T(P.S)$ is not minimal. This means that there exists an A belonging to M, such that $M' = M - A$ is still a model. I.e., dropping A is of no matter for the clauses of P to be satisfied. Since $T(P)$ is a minimal model of $P1$, then A does not belong to $T(P)$. Consequently, A must either be in S or depend on some atom in S. Therefore, there exists $S' \subset S$ such that a $2-WFM$ of $P.S'$ exists, and $T(P.S') = M'$, which is a contradiction. $\square$

Among the minimal models obtained by Theorem 4.3 are the stable models of P, as well as the Weakly Perfect Model [15] of P, if they exist.

In the next Sections, we show how to explicitly characterize maximal generators of stable models. These generators can actually be calculated in many cases when the set of undefined atoms is finite, also if the whole Herbrand base is instead infinite.

5 More Observations about Stable Models

In order to reason about the stable models of a program P, it is useful to eliminate irrelevant information from P. We will prove that the relevant information about the stable models is limited to a subset of the clauses whose conclusion is undefined in WFM_P.

Given two programs PA and PB, we indicate by $PA - PB$ the program obtained by removing from PA all clauses which appear also in PB.

Definition 5.1 *Let $P2$ be the subprogram of P such that a clause of P*
$A : -B_1, \ldots, B_m, notC_1, \ldots, notC_n,\ m \geq 0, n \geq 0,$
is in $P2$ iff either $\{A, B_1, \ldots, B_m, C_1, \ldots, C_n\} \subseteq T(P) \cup F(P)$,
or $\{B_1, \ldots, B_m\} \cap F(P) \neq \emptyset$ or $\{C_1, \ldots, C_n\} \cap T(P) \neq \emptyset$. $\square$

Informally, $P2$ contains all clauses with conclusion true/false in WFM_P, and all clauses with conclusion undefined, but one of the conditions which is false in any WFM–compatible model of P. This means, given any generator G, this clause is irrelevant for the final truth value of A in the G–generated model. Clearly, $P1$ (Definition 4.1) is a subprogram of $P2$.

Lemma 5.1 *$T(P)$ is a minimal model of $P2$.*

Proof: This is a consequence of Lemma 4.1, and of the fact that every clause K in $P2 - P1$ has an undefined conclusion A (then, $A \notin T(P)$) and a false condition (then, K is satisfied in $T(P)$). $\square$

Lemma 5.2 *Let $U' = B_{P2} \cap U$.*
$< T(P); F(P) \cup U' >$ is the $2-WFM$ of $P2$.

Proof: This is an immediate consequence of the Definition of WFM and of Proposition 2.1, since every clause K in $P2$ with conclusion $A \in U'$ has a false condition. $\square$

Corollary 5.1 *T(P) is the unique stable model of P2.* $\square$

Definition 5.2 *Let* $P3$ *be the subprogram of* P *such that a clause* $A : -B_1, \ldots, B_m, notC_1, \ldots, notC_n$ *of* P *is in* $P3$ *iff* $A \in U$. $\square$

Definition 5.3 *Let P4 be P–P2 (or, equivalently, P3–P2). Let PU be the program obtained from P4 by canceling from every clause all the literals which are true in* WFM_P. $\square$

Example 5.1 *Let P be*
a:-not b,c.
b:-not a.
a:-e,f.
c:-d.
d.
WFM_P=<{c,d};{e,f}>
P2 is:
a:-e,f.
c:-d.
d.
P3 is:
a:-not b,c.
b:-not a.
a:-e,f.
P4 is:
a:-not b,c.
b:-not a.
PU is:
a:-not b.
b:-not a.
PU contains all the relevant information about undefined atoms of WFM_P*. In this case, atoms a and b are undefined because of the negative mutual recursion.* $\square$

By construction, all literals composing clauses of PU are undefined in WFM_P. Clearly, for every model MD of PU, $MD \subseteq U$.

Lemma 5.3 $M \supseteq T(P)$ *is a minimal model of* P *iff* $MU = M - T(P)$ *is a minimal model of* PU*.*

Proof: Let M be a minimal model of P. Let $A : -B_1, \ldots, B_m, notC_1, \ldots, notC_n$ be a clause K in PU. By the definition of PU, K corresponds to a clause $K1$ in P, which contains some true literals in addition to the conditions of K. We show that $K1$ is satisfied in M iff K is satisfied in MU. First, $\forall D \in \{A, B_1, \ldots, B_m\}$, $D \in M$ iff $D \in MU$, since by the definition of PU, $D \notin T(P)$. Second, for every literal $L_i = notC_i$, $i \leq n$, L_i is true/false in M iff it is true/false in MU since, by the definition of PU, we have $C_i \in U$. This suffices to show that MU is a model of PU. Assume now MU is not minimal, i.e. there exists A such that $MU - A$ is still a model. By Proposition 2.1 and by the definition of PU, there is a clause K with conclusion A in PU. From the above discussion, K is satisfied in $MU - A$ iff the corresponding clause $K1$ in P is satisfied in $M - A$. This implies M is not minimal, which is a contradiction. Let MU be a minimal model of PU. By Lemma 5.1, $T(P)$ is a minimal model of $P2$. By the definition of $P2$ and PU, $T(P) \cup MU$ is a model of P. It is also minimal, since dropping any atom from either $T(P)$ or MU implies there is either a clause in $P2$, or a clause in $P - P2$ (corresponding to a clause in PU) which is not satisfied any more. $\square$

Theorem 5.1 *$M \supseteq T(P)$ is a stable model of P iff MU=M-T(P) is a stable model of PU.*

Proof: Let M be a stable model of P. Since a stable model is a minimal model [8], then by Lemma 5.3 $MU = M - T(P)$ is a minimal model of PU. We prove that MU is stable by applying the Σ-test. Assume the stability condition does not hold, i.e one of the following holds.
(i)$\exists A \in MU$ such that A is not coherent in MU. By the definition of PU, there exists a clause K in PU with conclusion A. By Definition 5.2, K has been obtained by canceling from a clause $K1$ in P literals which are true in WFM_P. Since M is WFM–compatible, these literals are still true in M. Therefore, if A is not coherent in MU, K and PU, then clearly it is not coherent in M, $K1$ and P, which is a contradiction.
(ii) $\exists A \in MU$ such that A occurs negatively in PU, but A is not coherently supported in MU. By Definitions 5.1 and 5.2, in the program $P2 = P - P4$ with unique stable model $T(P) = M - MU$ (Corollary 5.1), there is no coherent clause for either A or the atoms on which A depends in PU. Consequently, A not coherently supported in PU and MU implies A not coherently supported in P and M. Therefore, M cannot be a stable model of P in that it violates point (b) of the stability condition, which is a contradiction.

Let MU be a stable model of PU. By Lemma 5.3, $M = MU \cup T(P)$ is a minimal model for P. Assume M is not stable. Since by Corollary 5.1 $T(P)$ is a stable model of $P2$, then any clause corresponding to an atom $A \in M$ which violates the stability condition must be in $P4$. But clauses of $P4$ correspond to clauses of PU modulo the elimination of true literals. Therefore, if the stability condition does not hold for A, M and P, it does not hold for A, MU and PU, which is a contradiction. $\square$

Example 5.2 *Consider the program P in the previous Example. The stable models of PU are* $\{a\}$ *and* $\{b\}$. WFM_P *is* $<\{c,d\};\{e,f\}>$. *The stable models of P are in fact* $\{a,c,d\}$ *and* $\{b,c,d\}$. □

In [21] a procedure is outlined (developed independently of this research) for: (i) simultaneously computing WFM_P and PU; (ii) computing the stable models of PU via a branch-and-bound search algorithm.

In the next Section we show instead how the stable models of PU (called *s-generators*) can be explicitly characterized. This by further reducing PU to a program P^*, where the relevant information is made explicit. In Section 7 we discuss the conditions for actually computing PU, P^* and the s-generators. Whenever the Herbrand base of P is finite, this implies being able to compute the stable models of P. Otherwise, it is however possible in many cases to compute at least the s-generators, which represent the basic sets of assumptions on which the stable models are based.

6 Finding Generators of Stable Models

The program PU contains, as shown in the previous Section, all the relevant information about the stable models of P. Only, this information may be in implicit form. The relevant information for finding stable models concerns negative dependencies. Then, we define a variant of a program where these dependencies become explicit.

Definition 6.1 *Let Pr be a program. The unfolded version of Pr, called Pr^, is defined as follows.*

(i) For every clause K of Pr, K is a clause of Pr^.

(ii) If Pr^ contains a clause
$A : -L_1, \ldots, L_{i-1}, B, L_{i+1}, \ldots, L_n, n > 0$
and a clause
$B : -E_1, \ldots, E_m, m \geq 0$
then Pr^ contains the clause
$A : -L_1, \ldots, L_{i-1}, E_1, \ldots, E_m, L_{i+1}, \ldots, L_n, n > 0, m \geq 0$ □

The following property clearly holds.

Proposition 6.1 *Let Pr be a program and* $M \subseteq B_{P_r}$. *M is a model of Pr iff M is a model of Pr^.* □

If the Herbrand base of Pr is finite, Pr^ can be calculated by a simple unfolding procedure Ufp [22] that generates new clauses from the given ones by replacing (unfolding) any atom occurring positively with its definition (if any) in every possible way. This replacement is never performed on an atom on which it has been performed in a previous step (loop check). Ufp terminates when there are no more new atoms to unfold.

Example 6.1 *Given Pr:*
p:-q,not a.
q:-not p.
a:-not p.
Pr^ is the program:
p:-q,not a.
q:-not p.
a:-not p.
p:-not p, not a.
If Pr is the program:
a:-c.
c:-not b.
b:-d.
d:-not a.
Pr^ is:
a:-c.
c:-not b.
b:-d.
d:-not a.
a:-not b.
b:- not a. □

Proposition 6.1 allows us to find the stable models of PU by finding the stable models of $PU\hat{}$. In particular, we select a subprogram of $PU\hat{}$, suitable for identifying the stable models.

Definition 6.2 *Let P^* be the subprogram of $PU\hat{}$ composed of all clauses without positive conditions. For every clause $A : -notC_1, \ldots, notC_n$ in P^*, we rewrite it in the equivalent form:*

$$A \vee (C_1 \vee \ldots \vee C_n) \quad \square$$

Below we summarize how the program P^* can be obtained from P and WFM_P.

Reduction Procedure

1. *Cancel from P all clauses whose conclusion is not undefined in WFM_P, or such that at least one literal in the conditions is false w.r.t. WFM_P, thus obtaining $P4$.*

2. *Cancel from $P4$ all literals which are true w.r.t. WFM_P, obtaining PU.*

3. *Apply Ufp to PU, obtaining $PU\hat{}$.*

4. *Select clauses of $PU\hat{}$ without positive conditions, obtaining P^*.*

5. *For every clause of P^* $A : -notC_1, \ldots, notC_n$, rewrite it in the equivalent form:*

$$A \vee (C_1 \vee \ldots \vee C_n) \quad \Box$$

It is useful to state a simple property of P^*.

Proposition 6.2 *Let A be the conclusion of a clause in PU.*

(i) There exists a clause with conclusion A in P^* iff there exists an acyclic support set for A in PU.

(ii) There is no clause with conclusion A in P^* iff every support set for A in PU is cyclic.

Proof: By the definition of PU and by Proposition 2.1, it is easy to see that every atom which occurs positively in PU has a support set. Assume at least one support set S for A is acyclic. PU has (by definition) no unit clauses. Then, if unfolding clauses in S, since S is finite and acyclic the unfolding will stop when there are no more positive atoms. I.e., the unfolding will necessarily produce a clause with conclusion A and no positive condition. Therefore, if there is no such clause every support set for A must be cyclic. $\Box$

Below is the main definition of this paper.

Definition 6.3 *A set $G \subseteq U$ is an s-generator for P iff the following conditions hold:*

(i) for every clause $A \vee (C_1 \vee \ldots \vee C_n)$ in P^* either $A \in G$, or $\exists Ci \in G, i \leq n$.

(ii) $\forall A \in G$ there exists a clause in P^* $A \vee (C_1 \vee \ldots \vee C_n)$ where $A \notin \{C_1, \ldots, C_n\}$ and $\{C_1, \ldots, C_n\} \cap G = \emptyset$.

(iii) there is no set $G' \subset G$ satisfying (i)-(ii). $\Box$

An s-generator for P will be called simply s-generator if there is no ambiguity about P. In all the examples which follow, we may notice that for every generator G, $M = T(P) \cup G$ is a stable model of P. We may also notice that there are as many s-generators as stable models. I.e., s-generators seem to be exactly the maximal generators of stable models. In the rest of this Section we prove that this is in fact the case.

Example 6.2 *Let P be the program:*
a:- not b.
b:-not a.
d:-not e, f.
e:-not d.
f:- a.

$WFM_P = <\emptyset;\emptyset>$. *After steps 1-3 of the reduction procedure we get:*

a:-not b.
b:-not a.
d:-not e, f.
d:-not e, a.
d:-not e, not b.
e:- not d.
f:-a.
f:-not b.
P^* is the following:

$a \vee b$.
$b \vee a$.
$d \vee e \vee b$.
$e \vee d$.
$f \vee b$
s-generators are: $\{a,f,d\}$, $\{a,f,e\}$, $\{b,e\}$. □

Example 6.3 *Let P be the program:*
a:-b.
b:-c.
c:-a.
a:-e.
e:-not c.
$WFM_P = <\emptyset;\emptyset>$.
It is easy to verify that P^* is:

$a \vee c$.
$b \vee c$.
$c \vee c$.
$e \vee c$.
Clearly there are no s-generators. In fact, in order to satisfy the clause $c \vee c$ (point (i) of Definition 6.3), any s-generator should contain c, but point (ii) is not satisfied for c. □

Example 6.4 *Let P be the program:*
p:-q.
a:- not b.
b:-not a.
c:-not a.
c:-not b.
$WFM_P = <\emptyset;\{p,q\}>$. *The reduction procedure drops the first clause, and P^* is:*
$a \vee b$.
$b \vee a$.

$c \vee a$.
$c \vee b$.
s-generators are: $\{a,c\}$, $\{b,c\}$. $\square$

Example 6.5 *Let P be the program:*
g:-e.
e.
h:-not f.
a:-not b.
b:-not a.
p:-not p.
p:-not a.
WFM_P=<{g,h,e};{f}>. *Steps 1-3 of the reduction procedure modify the program by dropping the first three clauses. Steps 4-5 give P*:*
$a \vee b$.
$b \vee a$.
$p \vee p$.
$p \vee a$.
It is easy to see that the only s-generator is $\{b,p\}$. *If we cancel the last clause in P (and correspondingly in P*), for the resulting program there are no s-generators.* $\square$

Example 6.6 *Consider the following program P.*
a:-not b.
b:-not c.
c:-not a.
As it is well known, P has $WFM = <\emptyset;\emptyset>$ *and no stable models. P* is:*
$a \vee b$.
$b \vee c$.
$c \vee a$.
It is easy to see that there are no s-generators. $\square$

Example 6.7 *Let P be the program:*
f:-c,q.
p:-q,not a.
q:-not p.
a:-h,not b.
b:-not a.
c.
h.
P has WFM <{c,h};$\emptyset$ >. *PU is:*
f:-q.
p:-q,not a.
q:-not p.

a:-not b.
b:-not a.
PU^ is:

f:-q.
f:-not p.
p:-q,not a.
q:-not p.
p:-not p,not a.
a:-not b.
b:-not a.
P is:*

$f \vee p$
$q \vee p$
$p \vee p \vee a$
$a \vee b$
$b \vee a$
The only s-generator is {a,f,q}. □

The aim of the rest of this Section is to prove that s-generators are the maximal generators of the stable models of P. Given results in Section 5, this requires to prove that s-generators are exactly the stable models of PU. As a first step, Theorems 6.1 and 6.2 show that an s-generator G for P is a generator for $PU\hat{}$, and that the G-generated model happens to be G itself. This implies that G is a model of $PU\hat{}$, which means G is a model of PU. Theorem 6.3 shows that G is actually a minimal model, and Theorem 6.4 finally proves it is stable.

Theorem 6.1 *Let G be an s-generator for P. G is a generator for $PU\hat{}$.*

Proof: Assume $PU\hat{}$.G is not saturated. Then, in the well-founded model of $PU\hat{}.G$ there are undefined atoms, i.e. $\exists A \in U(PU\hat{}.G)$. Let us consider the subprogram P^* of $PU\hat{}.G$. If there is no clause in P^* with conclusion A, then by Proposition 6.2 every support set for A in PU (equivalently in $PU\hat{}$) is cyclic. This is still the case in $PU\hat{}.G$: since by Proposition 6.2 and Definition 6.3 atoms involved in positive circularities cannot be in G, support sets for A in $PU\hat{}.G$ are the same as in $PU\hat{}$. Therefore, by Proposition 2.1 $A \in F(PU\hat{}.G)$, which is a contradiction. Assume now there is at least one clause in P^* with conclusion A. Any such clause has the form $A \vee (E_1 \vee \ldots \vee E_r)$, $r > 0$. Since A is undefined, certainly $A \notin G$. Then, by condition (i) of Definition 6.3, $\exists D \in \{E_1, \ldots, Er\}, D \in G$. Then, since A depends for every possible derivation on the negation of a true atom, $A \in F(PU\hat{}.G)$, which is a contradiction. □

Theorem 6.2 *Let G be an s-generator for P. $T(PU\hat{}.G) = G$.*

Proof: Assume $\exists A \in T(PU\hat{}.G) - G$. By Proposition 2.1, A has an acyclic support set S. By Proposition 6.2, there exists at least a clause with conclusion A in $P*$. Any such clause is of the form: $A \vee (E_1 \vee \ldots \vee Er)$, $r > 0$. If $A \notin G$, then by condition (i) of Definition 6.3 $\exists D \in \{E_1, \ldots, E_r\}, D \in G$. Therefore, since A depends for every possible derivation on the negation of a true atom, $A \in F(PU\hat{}.G)$, which is a contradiction. $\Box$

Corollary 6.1 *Let G be an s-generator for P. G is a model of both $PU\hat{}$ and PU.*

Proof: From Theorem 4.1, Proposition 6.1 and the above Theorem. $\Box$

Theorem 6.3 *Let G be an s-generator for P. G is a minimal model of $PU\hat{}$.*

Proof: Assume G is not minimal, i.e. $\exists A \in G$ such that $G - A$ is still a model. By point (ii) of Definition 6.3, there exists a clause K in $P*$ (and then in $PU\hat{}$) $A \vee (C_1 \vee \ldots \vee C_n)$, where $A \notin \{C_1, \ldots, C_n\}$ and $\{C_1, \ldots, C_n\} \cap G = \emptyset$. Clearly, K is not satisfied in $G - A$, that thus cannot be a model. $\Box$

Corollary 6.2 *Let G be an s-generator for P. G is a minimal model of PU.*

Proof: From Proposition 6.1 and the above Theorem. $\Box$

Theorem 6.4 *G is an s-generator for P iff G is a stable model of PU.*

Proof: Let G be an s-generator for P. Since, by Corollary 6.2, G is a minimal model of PU, we can apply the Σ-check on G. We have to prove the following.
(1) for every $A \in G$, there exists a clause K in PU
$A : -B_1, \ldots, B_m, notC_1, \ldots, notC_n, m \geq 0, n \geq 0$, where
$A \notin \{B_1, \ldots, B_m, C_1, \ldots, C_n\}$ and $\{B_1, \ldots, B_m\} \subseteq G, \{C_1, \ldots, C_n\} \cap G = \emptyset$.
It is easy to see that K certainly exists, because the clause required by condition (ii) of Definition 6.3, say $K*$, can be obtained only by the unfolding of K. In fact, $K*$ has the form: $A \vee (E_1 \vee \ldots \vee E_k \vee C_1 \vee \ldots \vee C_n)$, where $k \geq 0$ and $\{E_1, \ldots, E_k, C_1, \ldots, C_n\} \cap G = \emptyset$. As an immediate consequence, $\{C_1, \ldots, C_n\} \cap G = \emptyset$. Then, since $E_1, \ldots, E_k$ result from the unfolding of $B_1, \ldots, B_m$, for each $B_i (i \leq m)$ there is a clause in $P*$
$B_i \vee (F_1 \vee \ldots \vee F_v), v \geq 0$, where $\{F_1, \ldots, F_v\} \subseteq \{E_1, \ldots, E_k\}$.
By condition (i) of Definition 6.3, $B_i \in G$, which means $\{B_1, \ldots, B_m\} \subseteq G$.
(2) For every B occurring negatively in PU, if $B \in G$ then B is coherently supported in PU and G. This is an immediate consequence of point (ii) of Definition 6.3 and of proposition 6.2.

Let G be a stable model of PU. Since the stability condition is satisfied by G, for every $A \in G$ there exists a clause K in PU of the form specified in point 1. Thus, the unfolding of K produces a clause in $P*$

$A \vee (E_1 \vee \ldots \vee E_k \vee C_1 \vee \ldots \vee C_n)$, where
$\{E_1, \ldots, E_k, C_1, \ldots, C_n\} \cap G = \emptyset$.
This satisfies condition (ii) of Definition 6.3. Assume condition (i) is not satisfied, i.e. there exists K' in P^* of the form $A \vee (C_1 \vee \ldots \vee C_n)$, and $\{A, C_1, \ldots, C_n\} \cap G = \emptyset$. Clearly, K' is not satisfied in G. This is a contradiction, because by Theorem 6.3 G is a minimal model of $PU\hat{}$, of which P^* is a subprogram. □

Below are the main results of this research.

Corollary 6.3 *M is a stable model of P iff*
$M = T(P) \cup G$, where G is an s-generator for P.

Proof: Immediate from Theorem 5.1 and Theorem 6.4. □

Corollary 6.4 *There exists a stable model of P iff*
there exists an s-generator for P.

Proof: Immediate from Corollary 6.3. □

Corollary 6.5 *Let GI be the intersection of all the s-generators of P.*
$T(P) \cup GI$ is the intersection of all the stable models of P.

Proof: Immediate from Theorem 5.1 and Theorem 6.4. □

Of course, the number of stable models of P corresponds to the number of s-generators for P.

Let us summarize the method for calculating the stable models of P, in case the Herbrand base of P is finite.

Stable Model Construction

step1. Calculate the $WFM < T; F >$ of P, and consequently the set U of undefined atoms.

step 2. From P and U, calculate P^* by applying the Reduction Procedure.

step 3. Find the s-generators $G_1, \ldots, G_r$.

step 4. Find the stable models $S_1, \ldots, S_r$ of P, where $\forall i \leq r$ $S_i = T \cup G_i$. □

7 Computability Issues, Relation to Abduction

If the Herbrand base of P is not finite, whenever P^* is computable the s-generators denote the basic sets of assumptions which characterize the stable models. Therefore it is worth stating explicitly the conditions for this to happen.

It is easy to see that computability of P^* is equivalent to computabilty of PU. As stated in Section 5, computing PU requires computing P3 and P4.

In order to compute P3, it suffices that the set U be finite. It is not necessary that either H_P or WFM_P be finite, since U can be in principle calculated independently of them (though a procedure for constructing U independently of T and F is still unknown).

Let $TF_U =< T_U; F_U >$, $T_U \subseteq T$, $F_U \subseteq F$ be the set of atoms true/false in WFM_P which occur in clauses of P3. In order to compute $P4$, it suffices that TF_U be finite, again independently of the whole WFM_P. In fact, in order to compute $P4$ from $P3$ it suffices to cancel from $P3$ every clause with a literal in the conditions which is false w.r.t. WFM_P (equivalently, false w.r.t. TF_U), and then remove from the remaining clauses all literals which are true w.r.t. WFM_P (equivalently, true w.r.t. TF_U).

We may notice that, in practical application of the Reduction Procedure, upon the assumption that TF_U be finite any proof theory for WFM_P (like for instance SLS-Resolution [16]) can be used for computing PU from $P3$.

In summary, the requirement for computing the s-generators for P is that the sets U and TF_U be finite. This is a weaker condition than requiring that the Herbrand base of P be finite, or that WFM_P be finite. To the best of our knowledge, this is in fact the weakest condition ever stated in the literature for computing or characterizing stable models.

Computing the s-generators may be useful for abductive reasoning, since there is a precise relation between s-generators and sets of abductive hypotheses. Let us in fact consider the one-to-one correspondence established in [4] between stable models and sets of abductive hypotheses. In particular, for every stable model M there exists a set of abductive hypotheses $D = (H_P - M)*$, meaning that D contains the negation of any atom in $H_P - M$. It is easy to see that, given the s-generator G for stable model M, the corresponding set of abductive hypotheses is $D = ((U - G) \cup F)*$.

For practical programs, it may be possible to suppose that s-generators are more likely to be finite than sets of abductive hypotheses. It is not unreasonable in fact to expect that in many knowledge-representation applications disjunctive information (denoted by undefined atoms) is actually expressed by a finite number of clauses.

Whenever they can be computed, s-generators may be used for ensuring correctness of the Abductive Procedure of [4] on non-locally stratified programs (on which it is presently in general not correct). Consider for instance the following program, which is a variation of an example discussed in [4]:

$h(a).$
$g(a).$
$g(f(X)) :- g(X).$
$k(X) :- not\, h(X).$
$r :- not\, r.$
$r :- q.$

$p : -not\, q.$
$q : -not\, p.$
The $WFM < T; F >$ of the above program is clearly infinite. The set U of undefined atoms is instead finite, and is $\{p, q, r\}$. TF_U is empty. P^* is:
$r \vee r$
$p \vee q$
$q \vee p$
$r \vee p$
The only s-generator, that characterizes the unique stable model, is $\{r, q\}$. The stable model is $M = \{r, q\} \cup T$. The corresponding set of abductive hypotheses is $(\{p\} \cup F)*$. On this Example, the abductive procedure of [4] would give $\{q\}*$ as an abductive explanation of success of query $? - p$. This means, the procedure assumes that there exists a stable model of the program where p is true and q is false, which is incorrect. This kind of incorrectness might be avoided by checking candidate abductive hypotheses w.r.t. the s-generators.

References

[1] C. Bell, A. Nerode, R. Ng, and V. S. Subrahmanian. *Implementing Stable Semantics by Linear Programming.* A. Nerode (ed.), *Logic Programming and Non-Monotonic Reasoning*, Proceedings of the Second International Workshop. The MIT Press, 1993.

[2] S. Costantini and G.A. Lanzarone. *Metalevel Negation and Non-Monotonic Reasoning.* Methods of Logic in Computer Science 1, 1994. (abstract in: Proceedings of the Workshop on Non-Monotonic Reasoning and Logic Programming, Austin, TX, November 1-2, 1990).

[3] J. Doyle. *A Truth Maintenance System.* Artificial Intelligence 12, 1979.

[4] K. Eshghi and R.A. Kowalski. *Abduction Compared with Negation as Failure.* G. Levi and M. Martelli (eds.), *Logic Programming*, Proceedings of the Sixth International Conference. The MIT Press, 1989.

[5] C. Elkan. *A Rational Reconstruction of Nonmonotonic Truth Maintenance Systems.* Artificial Intelligence 43, 1990.

[6] K. Eshghi. *Computing Stable Models by Using the ATMS.* Proceedings of AAAI'90, 1990.

[7] M. Gelfond. *On Stratified Autoepistemic Theories.* Proceedings of AAAI'87, 1987.

[8] M. Gelfond and V. Lifschitz. *The Stable Model Semantics for Logic Programming.* K. Bowen and R.A. Kowalski (eds.), *Logic Programming*, Proceedings of the Fifth Symposium, The MIT Press, 1988.

[9] N. Iwayama and K. Satoh. *Computing Abduction by using the TMS* K. Furukawa (ed.), *Logic Programming*, Proceedings of the Eigth International Conference, The MIT Press, 1991.

[10] A.C. Kakas and P. Mancarella. *Generalized Stable Models: A Semantics for Abduction.* Proceedings of ECAI'90, 1990.

[11] R.A. Kowalski. *Problems and Promises of Computational Logic.* Proceedings of the Symposium on Computational Logic, Springer Verlag, 1990.

[12] N. Leone, M. Romeo, P. Rullo, and D. Sacca'. *Effective Implementation of Negation in Database Logic Query Languages.* P. Atzeni (ed.) LOGIDATA+: Deductive Databases with Complex Objects. Lecture Notes in Computer Science 701, Springer-Verlag, Berlin, 1993.

[13] J.W. Lloyd. *Foundations of Logic Programming* (Second, Extended Edition). Springer-Verlag, Berlin, 1987.

[14] S.G. Pimentel and J.L. Cuadrado. *A Truth-Maintenance System based on Stable Models.* Proceedings of NACLP'89, 1989.

[15] H. Przymusinska and T. Przymusinski. *Semantic Issues in Deductive Databases and Logic Programs.* R.B. Banerji (ed.) *Formal Techniques in Artificial Intelligence, a Sourcebook*, Elsevier Science Publisher B.V. (North Holland), 1990.

[16] T. Przymusinski. *On the Declarative and Procedural Semantic of Logic Programs.* Journal of Automated Reasoning, 4, 1988.

[17] T. Przymusinski. *Every Logic Program has a Natural Stratification and an Iterated Fixpoint Model.* Proceedings of the Symposium on Principles of Database Systems, ACM SIGACT-SIGMOD, 1989.

[18] W.L. Rodi and S.G. Pimentel. *A Non-Monotonic Assumption-based TMS Using Stable Bases.* J. A. Allen, R. Fikes and E. Sandewall (eds.) *Principles of Knowledge Representation and Reasoning*, Proceedings of the Second International Conference. Morgan Kauffman, San Mateo, CA,1991.

[19] D. Sacca' and C. Zaniolo. *Stable Models and Non-determinism in Logic Programs with Negation.* Proceedings of the Symposium on Principles of Database Systems. ACM SIGACT-SIGMOD, 1990.

[20] J. Stuber. *Computing Stable Models by Program Transformation.* P. Van Hentenryck (ed.), *Logic Programming*, Proceedings of the Eleventh International Conference. The MIT Press, 1994.

[21] V.S. Subrahmanian, D. Nau and C.Vago. *WFS + Branch and Bound = Stable Models* CS-TR-2935 UMIACS-TR-92_82, July 1992 (submitted to IEEE Transactions on Knowledge and Data Engineering).

[22] H. Tamaki and T. Sato. *Unfold/Fold Transformation of Logic Programs.* K. Clark and S.-Å. Tarnlund (eds.), Proceedings of the Second International Conference of Logic Programming, 1984.

[23] A. Van Gelder. *The Alternating Fixpoint of Logic Programs with Negation.* Proceedings of the Symposium on Principles of Database Systems. ACM SIGACT-SIGMOD, 1989.

[24] A. Van Gelder, K.A. Ross and J.S. Schlipf. *The Well-Founded Semantics for General Logic Programs.* Journal of the ACM Vol. 38 N. 3, 1990 (abstract in Proceedings of the Symposium on Principles of Database Systems, ACM SIGACT-SIGMOD, 1988).