$W_{s,c}$-Stable Semantics for propositional theories

Mauricio Osorio, Juan Carlos Nieves  
Universidad de las Americas  
CENTIA  
Sta. Catarina Martir, Cholula, Puebla.  
78220, México  
josorio@mail.udlap.mx

Abstract

We define new semantics that extend the STABLE semantics in different directions. In particular, we introduce two semantics for general propositional theories that we call $w_s - stable$ and $w_c - stable$. Our semantics have the main properties: $w_s - stable$ and $w_c - stable$ are defined over arbitrary propositional theories. They agree with the stable model semantics when the given theory has at least one stable model. Our proposed semantics are invariant over intuitionistic equivalent theories. Finally, intuitionistic consistent theories have at least one $w_s - stable$ model and one $w_c - stable$ model. Our results naturally generalize to universal theories.

Keywords: Logic programming, intuitionistic logic, stable models.

1 Introduction

Applications using logic programming are wide, just to mention a recent one, Kowalski and Sadri in [1] propose an approach to agents (KS-agents) within an extended logic programming framework. Grosen in [2] observe that, logic programs, as a core declarative knowledge representation, has wide current presence and major representational advantages compared to other rules approaches. The stable semantics is among the most important semantics for logic programming, and has proved to has appealing and enduring features [3].

Answer Set Programming (ASP) (Stable Logic Programming or A-Prolog) is the realization of much theoretical work on Nonmonotonic Reasoning and AI applications of Logic Programming in the last 12 years. It is based on view of program statements as constraints on the solution of a given problem. Subsequently, each model of the program encodes a solution to the program itself. For instance, an ASP program encoding a planing scenario has as many models as valid plans. This schema is similar to that underlying the applications
of SAT algorithms to AI, and in fact the range of applicability of these two techniques is similar. However, thanks to the inherent causal aspect of Answer Set semantics, we can represent default assumptions constraints, uncertainly and non determinism in a direct way. Several ASP systems are now available, among them are DeReS, dlv, smodels and XSB. Several others can be found through the library of Logic Programming Systems and Test Cases. These systems support provably correct inferences and are at least as fast and scalable as SAT checkers. These are exciting results for NMR community and they are attracting the attention of researchers from fields such as planning, product configuration, verification of distributed systems, combinatorial problems and logical cryptanalysis.

The term Answer Set Programming, that we use now, was coined by Vladimir Lifschitz. It nicely captures two key ideas behind the approach: solution or answers are sets and logic programming with the answer-set semantics (a generalization of stable models semantics) is an instantiation of the general approach.

However, it is well known that the standard semantics usually is too strong and sometimes a program do not possess a stable model [4, 5, 6]. As an example, in [7], the authors show that STABLE has some problems in modeling partial-order programs [8, 5].

Some authors have even argued that STABLE does not even define the intended models [9]. From the point of view of structural properties, Dix has suggested the notion of well-behaved semantics [6]. It turns out that STABLE is not well-behaved [6]. Some proposals to improve the stable semantics can be found in [5, 10]. It is also possible that there is not a best semantics as [11] comments.

In this paper we propose a solution to one problem of stable: That some programs do not have stable models. We introduce two semantics for general propositional theories that we call \( w_s - \text{stable} \) and \( w_c - \text{stable} \). Our semantics have the main properties: First, \( w_s - \text{stable} \) and \( w_c - \text{stable} \) are defined over arbitrary propositional theories. Second, they agree with the stable model semantics when the given theory has at least one stable model. Third, our proposed semantics are invariant over intuitionistic equivalent theories. Finally, intuitionistic consistent theories have at least one \( w_s - \text{stable} \) model and one \( w_c - \text{stable} \).

Our results naturally generalize to universal theories. The reason of this claim is because in logic programming is standard to work on the grounded form of a program.

Our paper is structured as follows: In section 2 we introduce our background. In section 3 we introduce our proposed semantics. Finally, in section 4 we present our conclusions.

2 Background

A signature \( \mathcal{L} \) in a finite set of elements that we call atoms. By \( \mathcal{L}_P \) we understand it to mean the signature of P, i.e. the set of atoms that occurs in P. A
 literal is an atom or the negation of an atom \(a\) that we denote as \(\neg a\). Given a set of atoms \(\{a_1, \ldots, a_n\}\), we write \(\neg\{a_1, \ldots, a_n\}\) to denote the set \(\{\neg a_1, \ldots, \neg a_n\}\).

A theory is built up using the constants \(\land, \lor, \to, \neg\) and atoms (from \(\mathcal{L}\)). A disjunctive clause is a clause of the form: \(a_1 \lor a_2 \lor \ldots \lor a_n \leftarrow l_1, l_2, \ldots, l_m\) where \(n \geq 1, m \geq 0\), every \(a_j\) is an atom and every \(l_i\) is a literal. Sometimes, we denote a clause \(C\) by \(A \leftarrow B^+, \neg B^-\), where \(A\) contains all the head atoms, \(B^+\) contains all the positive body atoms and \(B^-\) contains all the negative body atoms. We also use \(\text{body}(C)\) to denote \(B^+ \cup \neg B^-\). When \(A\) is a singleton set, the clause can be regarded as a normal clause. A definite clause [12] is a normal clause with \(B^- = \emptyset\). A disjunctive program is a finite set of disjunctive clauses. A normal program is built up with normal clauses. A definite program is a set of definite clauses [12]. Given a signature \(\mathcal{L}\), we write \(\text{Prog}_\mathcal{L}\) to denote the set of all programs defined over \(\mathcal{L}\).

Given a theory \(T\) defined over the signature \(\mathcal{L}\) we define an interpretation over \(T\) as a set \(P \cup \neg P\) such that \(P \cap \neg P = \emptyset\) and \(P \cup \neg P = \mathcal{L}\).

A semantics is a mapping that associated to each theory \(T\) a set of interpretations. Given two theories \(T_1\) and \(T_2\) we define \(T_1 \equiv T_2\) iff for all formula \(w\), \(T_1 \vdash L w\) iff \(T_2 \vdash L w\) where \(\vdash_L\) is the provability relation in the same logic \(L\).

If the set of interpretations that the semantics associates to \(P_1\) is the same as the set of interpretations that the semantics associates to \(P_2\), then we denote this by \(P_1 \equiv_{sem} P_2\).

### 2.1 Stable Models

The Gelfond-Lifschitz transformation (GL-transformation) of a logic program \(P\) w.r.t. an interpretation \(M\) is obtained from \(P\) by deleting

1. each rule that has a negative literal \(\neg B\) in its body with \(B \in M\), and
2. all negative literals in the bodies of the remaining rules.

Clearly the program \(GL(P, M)\) resulting from applying the GL-transformation is negation-free, so \(GL(P, M)\) has at least a model. \(M\) is a stable model iff \(M\) is a minimal model of \(GL(P, M)\).

**Definition 1 (Equivalence Transformation, [13])** Given a semantics \(SEM\), a program transformation \(\rightarrow\) is a \(SEM\)-equivalence transformation iff for all \(P, P'\) with \(P \rightarrow P'\): \(SEM(P) = SEM(P')\).

The following transformations are defined in [14, 15] and generalize the corresponding definitions for normal programs.

**Definition 2 (Basic Transformation Rules)**

A transformation rule is a binary relation on \(\text{Prog}_\mathcal{L}\). The following transformation rules are called basic. Let a program \(P \in \text{Prog}_\mathcal{L}\) be given.

\[\text{This clause represents the formula } l_1 \land l_2 \land \ldots \land l_m \rightarrow a_1 \lor a_2 \lor \ldots \lor a_n.\]
**RED**$$^{+}$$**(R$$^{+}$$):** Replace a rule \( A \leftarrow B^{+}, \neg B^{-} \) by \( A \leftarrow B^{+}, \neg(B^{-} \cap \text{HEAD}(P)) \).

**RED$$^{-}$$**(R$$^{-}$$):** Delete a clause \( A \leftarrow B^{+}, \neg B^{-} \) if there is a clause \( \bar{A} \leftarrow \text{true} \) such that \( \bar{A} \subseteq B^{-} \).

**Subsumption****(Sb):** Delete a clause \( A \leftarrow B^{+}, \neg B^{-} \) if there is another clause \( \bar{A}_{1} \leftarrow B_{1}^{+}, \neg B_{1}^{-} \) such that \( \bar{A}_{1} \subseteq A, B_{1}^{+} \subseteq B^{+}, B_{1}^{-} \subseteq B^{-} \).

**Taut****(Ta):** Delete a rule \( A \leftarrow B^{+}, \neg B^{-} \) with \( A \cap B^{+} = \phi \).

**Failure****(F):** Suppose that \( P \) includes an atom \( a \) and a clause \( q \leftarrow \text{Body} \) such that \( a \notin \text{HEAD}(P) \) and \( a \) is a positive literal in \( \text{Body} \). Then we erase the given clause.

**Contradictions****(Cn):** Delete a rule \( A \leftarrow B^{+}, \neg B^{-} \) with \( B^{+} \cap B^{-} \neq \phi \).

**Example 3** (Transformation)

Let \( L = \{a, b, c, d, e\} \) and let \( P \) be the program:

- \( a \lor b \leftarrow c, \neg c, \neg d \).
- \( a \lor c \leftarrow b \).
- \( c \lor d \leftarrow \neg e \).
- \( b \leftarrow \neg c, \neg d, \neg e \).

then \( \text{HEAD}(P) = \{a, b, c, d\} \).

We can apply **RED**$$^{+}$$ to get the program \( P_{1} \):

- \( a \lor b \leftarrow c, \neg c, \neg d \).
- \( a \lor c \leftarrow b \).
- \( c \lor d \leftarrow \text{true} \).
- \( b \leftarrow \neg c, \neg d, \neg e \).

If we apply **RED**$$^{+}$$ again, we get program \( P_{2} \):

- \( a \lor b \leftarrow c, \neg c, \neg d \).
- \( a \lor c \leftarrow b \).
- \( c \lor d \leftarrow \text{true} \).
- \( b \leftarrow \neg c, \neg d \).

Now, we can apply **SUB** to get program \( P_{3} \):

- \( a \lor c \leftarrow b \).
- \( c \lor d \leftarrow \text{true} \).
- \( b \leftarrow \neg c, \neg d \).

Let \( D_{\text{suc}} \) be the natural generalization of \( \text{suc} \) to disjunctive programs, formally:

**Definition 4** (\( D_{\text{suc}} \),[16])

Suppose that \( P \) is a program that includes a fact \( a \leftarrow \text{true} \) and a clause \( A \leftarrow \text{Body} \) such that \( a \in \text{Body} \). Then we replace this clause by the clause \( A \leftarrow \text{Body} \setminus \{a\} \).

**Definition 5** (\( C_{S_{2}} \))

Let \( C_{S_{2}} \) be the rewriting system based on the transformations \{\( S_{b}, R^{+}, R^{-}, D_{\text{suc}}, F, T_{u}, C_{n} \)\}. 
2.2 Intuitionistic logic

Intuitionistic logic is based on the concept of proof. Given a property \( P \) and an element \( E \), intuitionistic logic does not accept the decidability whether \( E \) has the property \( P \) or not. Due to Brouwer, intuitionistic logic must exhibit the object of our pertained proof, or at least, indicates a method by which one could in principle find our object. Valid formulas in intuitionistic logic, will be also valid in classical logic. For intuitionistic logic, the concept of truth in not used for explain the meaning and the use of the logical constants or quantifiers. We use the concept of \( \text{proof}_f. \) We denote \( \vdash_f \) to denote provability in intuitionistic logic. Ten axioms are defined in intuitionistic logic:

- \( A \rightarrow (B \rightarrow A) \)
- \( (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \)
- \( A \land B \rightarrow A \land B \rightarrow B \)
- \( A \rightarrow (B \rightarrow (A \land B)) \)
- \( A \rightarrow (A \lor B) \)
- \( B \rightarrow (A \lor B) \)
- \( (A \rightarrow (B \rightarrow C)) \rightarrow ((B \rightarrow C) \rightarrow (A \lor B \rightarrow C)) \)
- \( (A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A) \)
- \( \neg A \rightarrow (A \rightarrow B) \)\(^2\)

**Lemma 6** In intuitionistic logic every consistent theory has a complete and consistent extension

**Proof.** Let \( T \) be a consistent theory, and \( T_1 \) a maximal consistent theory of \( T \), then \( T_1 \) is complete. The proof of this fact is by contradiction. Suppose \( T_1 \) is maximal consistent but it is not complete. Then, there is formula \( \alpha \) such that \( T_1 \vdash \alpha \) and \( T_1 \not\vdash \neg \alpha \). Since \( T_1 \) is maximal consistent, \( T_1 \cup \{ \alpha \} \vdash \neg \alpha \). But then \( T_1 \vdash \alpha \rightarrow \neg \alpha \) and so \( T_1 \vdash \neg \alpha \). \( \blacksquare \)

2.3 Characterization of stable semantics based on intuitionist logic

Pearce showed the following: a formula is entailed by a program in the stable model semantics if and only if it belongs to every intuitionistically complete and consistent extension of the program formed by adding only negated atoms \([17].\)

\(^2\)If we replace this axiom schema by \( \neg \neg A \rightarrow A \) then we obtain classical logic.
2.4 Transformation of Programs

The following lemma is well known for normal programs [18], and recently it has been generalized to disjunctive programs [19].

**Lemma 7 (**STABLE is closed under CS₂ transformations,[19])** Let P₁ and P₂ two programs related by any transformation in CS₂. Then P₁ and P₂ have the same STABLE models.

The following result (taken from [19]) suggests that it makes sense to reduce a program by CS₂, because this reduction can be computed efficiently.

**Lemma 8 (CS₂ is polynomial time computable, [19])** Let P be a program and P₁ a reduced form of P under CS₂ (namely, that P₁ is obtained from P by a sequence of reductions from P and P₁ is already irreducible by CS₂). Then P₁ is obtained in polynomial time computable with respect to the size of P.

3 A proposal for a new semantics

First of all, let us consider some definitions:

**Definition 9** Let P be a theory. We denote A an appendix of P any set of literals over the signature L.

**Definition 10** Given an appendix I of P, we use Positive(I) to denote the set of positive literals in I.

**Definition 11** Given a theory P, we define two different partial order among appendices of P as follows: I ≤ₘ J iff Positive(I) ⊆ Positive(J), another partial order I ≤ₖ J is defined iff | Positive(I) | ≤ | Positive(J) |

**Definition 12** We say that a theory P is literal-complete w.r.t. intuitionistic logic iff ∀α ∈ Lₚ, P ⊢₁ α or P ⊢₁ ¬α.

**Definition 13** We say that a theory P is consistent w.r.t. intuitionistic logic I iff there is no formula α such that P ⊢₁ α and P ⊢₁ ¬α.

**Definition 14** Given a program P we define a L – completion of P as (P, I') such that P ∪ I' = P' where I' is an appendix of P, and P' is literal complete and consistent with respect to intuitionistic logic.

**Definition 15** Given x ∈ {s, c} and a program P we define a suitable Lₓ – completion of P' := (P, I') of P iff (P, I') is a Lₓ – completion of P and for any other Lₓ – completion P'' := (P, I'') of P it is false that I'' ≤ₓ I'.

**Definition 16** Given x ∈ {s, c}, A wₓ – stable model for P is the set of provable literals in a suitable Lₓ – completion P.
The following results show that our proposal semantics satisfies suitable properties with respect to the original stable semantics. Hence, it is an adequate generalization of the above semantics.

**Theorem 17** If $P$ admits stable models, then $M$ is a $w_s$-stable model iff $M$ is a stable model.

**Proof.** If $P$ admits a stable model said $M$, then, by [17], $P$ has a suitable $\mathcal{L}_s$-completion $P$ using an appendix $I$ with no positive literals. Therefore $\text{Positive}(I) = \emptyset$, and so there is no suitable $\mathcal{L}_s$-completion $P$ that includes positive literals. ■

**Corollary 18** For stratified normal programs,

$$w_s - \text{stable} \iff \text{models}(P) = \text{stratified} - \text{models}(P).$$

**Proof.** If $P$ is a stratified program, then it has exactly one stable model $M$ which corresponds with the stratified model. The conclusion follows immediately from the last theorem. ■

**Theorem 19** Every consistent theory has at least one $w_s$-stable model.

**Proof.** If $T$ is a consistent theory, then by lemma 6 it has a complete and consistent extension $T'$. Let $I := \{l|l\text{ is a literal and } T' \vdash l\}$. So $(T, I')$ is a $\mathcal{L}$-completion of $T$. Therefore $T$ has a suitable $\mathcal{L}_s$-completion. Hence $T$ has a $w_s$-stable model. ■

**Corollary 20** Every disjunctive program has at least one $w_s$-stable model.

**Proof.** Just note that a disjunctive program is consistent. ■

**Theorem 21** Given two programs such that $P_1 \equiv_I P_2$ then $P_1 \equiv_{w_s,\text{stable}} P_2$.

**Proof.** It is enough to show that if $P_1 \cup I_1$ is a suitable $\mathcal{L}_s$-completion of $P_1$ then $P_2 \cup I_1$ also is a suitable $\mathcal{L}_s$-completion for $P_2$. First, note that $P_2 \cup I_1$ is literal complete, for hypothesis, $P_1 \equiv I P_2$ then $P_1 \cup I_1 \equiv I P_2 \cup I_1$. Given that $P_1 \cup I_1$ is a $\mathcal{L}_s$-completion of $P_1$, then $P_2 \cup I_1$ is a $\mathcal{L}_s$-completion of $P_2$.

Moreover, $P_2 \cup I_1$ is a suitable completion, for if not, $\exists I'$ such that $I' \leq_1 I_1$ and also $P_1 \cup I'$ would be a suitable $\mathcal{L}_s$-completion of $P_1$. Contradiction. ■

**Theorem 22** Let $P_1$ be a logical program. Given any transformation $T \in \{R, Sb, Douc, Ta, Cn\}$ then $T$ is a $w_s$-stable equivalence transformation.

**Proof.** By cases: For $Sc$ let $\{a, a \land a \rightarrow \beta\} \subseteq P_1$. It is enough prove that $\{a, a \land a \rightarrow \beta\} \equiv_I \{a, a \rightarrow \beta\}$. So $a, a \rightarrow (a \rightarrow \beta)$ and $a \rightarrow (a \rightarrow \beta)$. We use $a, a \rightarrow \beta$ to denote conjunction of literals in the body of any clause and the disjunction of atoms in the head of any clause. ■

\[a \rightarrow (a \rightarrow \beta)\]
\( \alpha \rightarrow \beta, \alpha \vdash_I \alpha \rightarrow (\alpha \rightarrow \beta) \). The result follows immediately. For \( \forall a : \text{Then} \ P_2 = P_1 \setminus \{ \alpha \rightarrow (l_1 \rightarrow (l_2 \rightarrow (\ldots (l_n \rightarrow a)\ldots)) \}. \) So \( \alpha \land \alpha \vdash_I \alpha \). For \( R^- \), let \( P_0 = P \cup \{ b, \alpha \rightarrow (\neg b \rightarrow a) \} \), then \( P_0 = P \cup \{ b \} \). Given that \( \alpha \rightarrow (\alpha \rightarrow a) \) in an axiom, \( \alpha \rightarrow a, \alpha \vdash_I a \) follows. For \( \exists b \), the proof straightforward follows from \( \vdash_I \neg b \rightarrow (b \rightarrow a) \) and two successive applications of \textit{Modus ponens}. \hfill \blacksquare

\textbf{Remark 23} The reader could prove directly that the last result can also be proved if we change \( I \leq_s J \) by \( I \leq_c J \)

\textbf{Lemma 24} Let \( P \) a program. Then \( M \) is a \( w_c \) – stable of \( P \) then \( M \) is a \( w_s \) – stable model.

\textbf{Proof.} Straightforward. \hfill \blacksquare

\textbf{Remark 25} Note that the converse is not true as it is shown in the following example:

Let \( P \) the following program:
\[
\begin{align*}
f & \leftarrow e, \neg f \\
d & \leftarrow c \\
c & \leftarrow \ 
\end{align*}
\]
\[d \leftarrow \neg \]
\[c \leftarrow \ d, \neg e
\]

Note that \( M_1 := \{e, d, \neg f, c\} \) and \( M_2 := \{e, f, \neg c, d\} \) are the \( w_s \) – stable models of \( P \). However, only \( M_1 \) is a \( w_c \) – stable model of \( P \).

Due to lack of space, we present only two examples where the stable models semantics has no models but there are \( w_s \) – stable – models.

\textbf{Example 26} Let \( P \) be the program:
\[
\begin{align*}
a & \leftarrow b \\
b & \leftarrow \neg b \\
\end{align*}
\]
then the unique \( w_s \) – stable is \( \{a, b\} \).

\textbf{Example 27} Let \( P \) be the program:
\[
\begin{align*}
p & \leftarrow \neg p \\
a & \leftarrow b \\
b & \leftarrow a \\
h & \leftarrow \neg d
\end{align*}
\]
then the unique \( w_s \) – stable is \( \{p, h\} \).

4 Conclusion

We defined two new semantics that generalized the stable semantics such that for every (stable) consistent disjunctive program our new semantics agree with the original stable semantics. However our new semantics also give a meaning to programs where the original stable semantics is inconsistent. Furthermore, our
new semantics can give a meaning to any propositional theory. Our approach is based on intuitionistic logic and open new lines of research. Our theorems 17, 19, 21 and 22 show that our semantics is an adequate generalization of the stable semantics.

Acknowledgments

This research is sponsored by the Mexican National Council of Science and Technology, CONACyT (project 35804-A).

References


