Expressing the Stable semantics in terms of the Pstable semantics

Mauricio J. Osorio Galindo and Alejandra López Fernández
Email: osoriomauri@gmail.com

Universidad de las Américas-Puebla

Abstract. We show that with a simple translation for normal programs, we can use the pstable model semantics to get the stable models of a normal program. We present the formal proof of this statement.

1 Introduction

The stable model semantics for logic programming lies within the context of nonmonotonic reasoning, [GL1]. It has been widely used and it is perhaps the most well known semantics for knowledge representation. The pstable model semantics, introduced in [NM05] shares several properties with stable model semantics, but it is closer to classical logic than stable.

We show in Section 3 a simple translation for normal programs, and later in Section 4 we show the proof of the equivalence between these two semantics in the context of normal programs and our translation. Finally, Section 5 concludes the paper and offers future work.

We assume that the reader has familiarity with logic and nonmonotonic reasoning.

2 Background

We introduce some concepts which are needed for further understanding of the work and the results presented on this document. We present the definitions of the stable and pstable model semantics that are the main topics of this paper, more basic definitions can be found in [NM05].

2.1 Normal program

As we have already said in the introduction we have a particular interest in the class of normal programs. These are programs formed by *normal rules* for which the body is a conjuntion of literals.

Definition 1. *[GL1] The programs under consideration are sets of rules of the form:*

$$a \leftarrow l_1 \wedge l_2 \wedge .... \wedge l_n \tag{1}$$

where a is an atom, $l_1, l_2, ..., l_n$ is a set of literals and $n \geq 0$. We say that $l_1 \wedge l_2 \wedge .... \wedge l_n$ is the body of the rule and a is the head of the rule. When every literal in the body is an atom, we say that the rule is positive.

2.2 Reduction

Before we present the stable and pstable model semantics definitions, we show the reduction defined in [NM05].

Definition 2. *Let P be a normal program and M a set of atoms:*

$$\mathcal{R}ed_M(P) = \{a \leftarrow \beta^+ \wedge \neg(\beta^- \cap M) \mid a \leftarrow \beta^+ \wedge \neg\beta^- \in P\} \tag{2}$$

where β^+ and β^- are sets containing, respectively, the positive and negative atoms that occur in the body of the clauses of P.

In other words, with this reduction every atom that is not in the model M and is negated at the body of a rule is deleted.

Definition 3. *Let P be a normal program. We define: $\mathcal{P}os(P)$ as the set of positive rules in P.*

Example 1. Take the following logic program P:

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

Given $M = \{a\}$, it follows that $Red_M(P)$ is the program:

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

Finally $Pos(Red_M(P))$ is the program:

$a.$

2.3 The Stable and Pstable model semantics

The Stable semantics was originally defined in [GL1] as follows:

Definition 4. *Let P be a normal program, for any set M of atoms from P, M is a stable model of P if M is a minimal model of $Pos(Red_M(P))$.*

It is well known that this definition is equivalent to saying that M is a model of P and $Pos(Red_M(P)) \models M$; from now on we use this altenative characterization. We write $P1 \equiv_{stable} P2$ to denote that P_1 and P_2 have the same set of stable models.

The definition of the pstable model semantics was shown in [NM05] and is the following:

Definition 5. *Let P be a normal program, and M a set of atoms. We say that M is a pstable model of P if M is a model of P and $Red_M(P) \models M$.*

3 Relating stable and pstable models semantics

Now we can talk about the relation between stable and pstable models. In order to get stable models using the pstable model semantics, we propose a polynomial transformation, we use $Trans(P)$ to denote this operation. In this section we formally define this transformation and show some examples. Later in Section 4 we will formally prove the soundness and completeness of this transformation; but first we show some definitions.

Definition 6. *Let P be a normal program and L_P the set of atoms in P. We define $\delta_P^{\leftarrow}$ as follows:*

$$\delta_P^{\leftarrow} = \{x' \leftarrow x \mid \forall x \in L_P\} \tag{3}$$

Definition 7. *Let L be a set of literals $\{l_1, ..., l_m\}$. We define the operation L' as the result of adding an apostrophe symbol to each literal of the set L. So $L' = \{l'_1, ..., l'_m\}$*

Definition 8. *Let P be a normal program. We define P' as:*

$$\mathcal{P}' = \{a \leftarrow \beta^+ \wedge \neg\beta^{-\prime} \mid a \leftarrow \beta^+ \wedge \neg\beta^- \in P\} \tag{4}$$

and where β^+ and β^- are sets containing, respectively, the positive and negative atoms that occur in the body of the clauses of P.

Definition 9. *Let P be normal program and L_P the alphabet of P. We define a program R as:*

$$R = \begin{cases} u \leftarrow x' \wedge \neg x. & \forall x \in L_P \\ x \leftarrow \neg\ y \wedge u. & \\ y \leftarrow \neg\ z \wedge u. & \\ z \leftarrow \neg\ x \wedge u. & \end{cases} \tag{5}$$

where u, x, y, z are new atoms (not in L_P).

Definition 10. *Let P be normal program we define $Trans(P)$ as:*

$$Trans(P) = P' \cup \delta_P^{\leftarrow} \cup R. \tag{6}$$

As we will show soon, the resulting program can be used to obtain the stable semantics. We will understand better this transformation with an example.

Example 2. Take the following program P.

$a \leftarrow \neg b.$
$b \leftarrow \neg a.$
$p \leftarrow \neg a.$
$p \leftarrow \neg p.$

Observe that $\{p, b\}$ and $\{p, a\}$ are pstable models of P. But $\{p, b\}$ is the unique stable model of P. We apply the transformation $Trans(P)$ as follows:

$a \leftarrow \neg b'.\ \ b \leftarrow \neg a'.\ \ p \leftarrow \neg a'.\ \ p \leftarrow \neg p'.$
$a' \leftarrow a.\ \ b' \leftarrow b.\ \ p' \leftarrow p.$
$u \leftarrow a' \wedge \neg a.\ \ u \leftarrow b' \wedge \neg b.\ \ u \leftarrow p' \wedge \neg p.$
$x \leftarrow \neg y \wedge u.\ \ y \leftarrow \neg z \wedge u.\ \ z \leftarrow \neg x \wedge u.$

Note that $\{p, b, p', b'\}$ is the unique pstable model of $Trans(P)$. If we intersect this model with L_P we obtain $\{p, b\}$, the unique stable model of P.

4 Proof

We show through a simple transformation, that the pstable model semantics can express the stable model semantics. Let us define $Trans1(P) := P' \cup \delta_P^{\leftarrow}$, it uses only three types of rules, namely:

1. $a \leftarrow \alpha^+.$
2. $b \leftarrow \alpha^+ \wedge \neg \beta^{-\prime}.$
3. $c' \leftarrow c.$

$Trans1(P)$ is very similar to $Trans(P)$ already defined, but in $Trans1(P)$ we do not consider the rules of part R defined in Definition 9 because these only work as a filter of the stable models, these rules filter those models that have an atom x' and do not have the atom x.

Example 3. Take the following program P:

$a \leftarrow \neg a.$

The program $Trans1(P)$ would be:

$P' = a \leftarrow \neg a'.$
$\delta_P^{\leftarrow} = a' \leftarrow a.$

$$R = \begin{cases} u \leftarrow a' \wedge \neg a. \\ x \leftarrow \neg y \wedge u. \\ y \leftarrow \neg z \wedge u. \\ z \leftarrow \neg x \wedge u. \end{cases}$$

The part R filters those models where the atom a' is in the model and a is not. For example $M = \{a'\}$ is a model of $P' \cup \delta_P^{\leftarrow}$ but in R this atom is eliminated as a possible model of $Trans1(P)$.

As the result of our above discussion we hope that the reader could see that the we should concentrate in $Trans1(P)$ for our main proof.

Recall that P and $Trans1(P)$ have the following types of rules:

P	$Trans1(P)$
$1.a \leftarrow \alpha^+.$	$1.a \leftarrow \alpha^+.$
$2.b \leftarrow \alpha^+ \wedge \neg\beta^-.$	$2.b \leftarrow \alpha^+ \wedge \neg{\beta^-}'.$
	$3.c' \leftarrow c.$

Lemma 1. *Let P be a normal program, and M be a set of atoms. If M is a model of P then $M \cup M'$ is a model of $Trans1(P)$.*

Proof. First assume that M is a model of P.

Both $Trans1(P)$ and P has the rules type 1, so if M models those rules in P then $M \cup M'$ models them in $Trans1(P)$.

For the rules type 3 we have two cases:

- If $c \in M$ then $c' \in M \cup M'$ by construction of $M \cup M'$. So $M \cup M'$ models rules type 3 in $Trans1(P)$.
- If $c \notin M$ then the body of the rule type 3 is false. So $M \cup M'$ models rules type 3 in $Trans1(P)$.

Hence $M \cup M'$ models rules type 3 in $Trans1(P)$.

There are also two cases for the rules type 2:

- If $b \in M \cup M'$ then $M \cup M'$ models the rules type 2 in $Trans1(P)$.
- If $b \notin M \cup M'$, we will prove by contradiction that $M \cup M'$ models the rules type 2 in $Trans1(P)$.

Assume, by contradiction, that the rule of type 2 is false when $b \notin M \cup M'$. This means that $\neg{\beta^-}'$ is true, therefore ${\beta^-}'$ is false.

On the other hand by Definition 8 of P', we know that rules type 2 of $Trans1(P)$ came from rules type 2 of P: $b \leftarrow \alpha^+ \wedge \neg\beta^-$.

M is a model of P, this means that $b \leftarrow \alpha^+ \wedge \neg\beta^-$ was true in P, but $b \notin M$ so $\neg\beta^-$ had to be false and β^- true; this means $\beta^- \subseteq M$, and $\beta^- \cup {\beta^-}' \subseteq M \cup M'$, this is a contradiction. Therefore $M \cup M'$ models the rules type 2 in $Trans1(P)$.

Hence $M \cup M'$ is a model of $Trans1(P)$.

Lemma 2. *Let P be a normal program, and M be a set of atoms. If $Pos(Red_M(P)) \models M$ then $Pos(Red_{M \cup M'}(Trans1(P))) \models M \cup M'$*

Proof. First we assume that $Pos(Red_M(P)) \models M$. Then following Definition 2, when $Red_M(P)$ is applied, the rules with the form $b \leftarrow \alpha^+ \wedge \neg\beta^-$, and where $\beta^- \cap M = \emptyset$ are reduced to $b \leftarrow \alpha^+$.

On the other hand, according to Definition 8, when doing $Trans1(P)$ that type of rule is translated into $b \leftarrow \alpha^+ \wedge \neg{\beta^-}'$ and because $\beta^- \cap M = \emptyset$, also ${\beta^-}' \cap M' = \emptyset$. Then when $Red_{M \cup M'}(Trans1(P))$ is done, this type of rule is replaced by $b \leftarrow \alpha^+$; just like in $Red_M(P)$.

Definition 3 states that when $Pos(Red_M(P))$ is done, the rules with the form: $b \leftarrow \alpha^+ \wedge \neg\beta^-$ and where $\beta^- \subseteq M$, are erased from the program $Red_M(P)$. On the other hand, according to Definition 8, that type of rules are translated in $Trans1(P)$ to $b \leftarrow \alpha^+ \wedge \neg{\beta^-}'$, and because $\beta^- \subseteq M$ also ${\beta^-}' \subseteq (M \cup M')$.

Then when $Pos(Red_{M \cup M'}(Trans1(P)))$ is done this type of rule is also erased, just like in $Pos(Red_M(P))$.

Summarizing $Pos(Red_{M \cup M'}(Trans1(P)))$ has the same rules as $Pos(Red_M(P))$ plus rules of the form $c' \leftarrow c$ that P does not have. Therefore if $Pos(Red_M(P)) \models M$ then $Pos(Red_{M \cup M'}(Trans1(P))) \models M \cup M'$

Lemma 3. *Let P be a normal program, and M be a set of atoms.*

If $Pos(Red_{M \cup M'}(Trans1(P))) \models M \cup M'$ then $Red_{M \cup M'}(Trans1(P)) \models M \cup M'$

Proof. Assume that $Pos(Red_{M \cup M'}(P)) \models M \cup M'$, then by monotonicity it follows that $Red_{M \cup M'}(Trans1(P)) \models M \cup M'$.

Theorem 1. *If M is a stable model of P then $M \cup M'$ is a Pstable model of $Trans1(P)$*

Proof. We assume that M is a stable model of P. By definitions 4 and 5 we can express Theorem 1 as:

If M is a model of P and $Pos(Red_M(P)) \models M$ then $M \cup M'$ is a model of $Trans1(P)$ and $Red_{M \cup M'}(Trans1(P)) \models M \cup M'$.

According to Lemma 1 and Lemma 3 we conclude that $M \cup M'$ is a Pstable model of $Trans1(P)$.

Now we move to prove the other half of the proof, namely that If $M \cup M'$ is a Pstable model of $Trans1(P)$ then M is a stable model of P. We need to prove some partial results first.

Lemma 4. *If $Red_{M \cup M'}(Trans1(P)) \models M$ then $Pos(Red_{M \cup M'}(Trans1(P))) \models M$.*

Proof. Given an atom a $\in M$ we will prove that if $Red_{M \cup M'}(Trans1(P) \models \{a\}$ then $Pos(Red_{M \cup M'}(Trans1(P))) \models a$, this is equivalent to prove that if

$Pos(Red_{M \cup M'}(Trans1(P))) \cup \{\neg a\}$ is consistent then

$Red_{M \cup M'}(Trans1(P)) \cup \{\neg a\}$ is consistent.

We assume that I is a model of $Pos(Red_{M \cup M'}(Trans1(P))) \cup \{\neg a\}$ and we define an interpretation J as follows:

$$J(x) = \begin{cases} I(x) & \text{if } x \in L_P, \\ 1 & x \in L_{P'}. \end{cases}$$

We will prove that J models $Red_{M \cup M'}(Trans1(P)) \cup \{\neg a\}$. First we can see that J models $\{\neg a\}$. Now we want to prove that J models $Red_{M \cup M'}(Trans1(P))$. Given the program $Red_{M \cup M'}(Trans1(P))$ we have these rules:

1. $a \leftarrow \alpha^+$.
2. $b \leftarrow \alpha^+ \wedge \neg\beta^{-'}$.
3. $a' \leftarrow a$.

According to the definition of J, it models the three types of rules. So we have proved that there exists an interpretation that models $Red_{M\cup M'}(Trans1(P)) \cup \{\neg a\}$. Hence $Pos(Red_{M\cup M'}(Trans1(P))) \models M$.

Lemma 5. *If $M \cup M'$ is a pstable model of $Trans1(P)$ then $M \cup M'$ is a stable model of $Trans1(P)$.*

Proof. We want to prove that $M \cup M'$ is a model of $Trans1(P)$ and
$Pos(Red_{M\cup M'}(Trans1(P))) \models M \cup M'$. We assume that $M \cup M'$ is a pstable model of $Trans1(P)$, so we know that $M \cup M'$ is a model of $Trans1(P)$ and
$Red_{M\cup M'}(Trans1(P)) \models M \cup M'$, taking this we can write the proof as follows:
E1 $Red_{M\cup M'}(Trans1(P)) \models M$
E2 $Pos(Red_{M\cup M'}(Trans1(P))) \models M$ by Lemma 4 of E1
E3 $Pos(Red_{M\cup M'}(Trans1(P))), M \models M'$ by rules $x' \leftarrow x$ where $x \in M$
E4 $Pos(Red_{M\cup M'}(Trans1(P))) \models M'$ by CUT of E2 and E3
E5 $Pos(Red_{M\cup M'}(Trans1(P))) \models M \cup M'$ by E4 and E2
We conclude that $M \cup M'$ is a stable model of $Trans1(P)$.

It is well known that the following property holds in the stable model semantics.

Lemma 6. *Let P be a normal program, M be a stable model of P, and x be an atom $\in M$ then $\exists x \leftarrow \alpha \in P$ and $M \models \alpha$.*

Definition 11. *Let P be a normal program, we define $HEAD(P)$ as a set with all the atoms at the heads of its rules.*

Lemma 7. *Let P be any normal program and $a \leftarrow \alpha \wedge \neg x'$ be any normal rule. Let $P1 = P \cup \{a \leftarrow \alpha \wedge \neg x'\} \cup \{x' \leftarrow x\}$. Let $P2 = P \cup \{a \leftarrow \alpha \wedge \neg x\} \cup \{x' \leftarrow x\}$. If $x' \notin HEAD(P \cup \{a\})$ then $P1 \equiv_{stable} P2$.*

Proof. Let M be a stable model of P_1 we have two cases:

- $x \in M$ then $x' \in M$ and $Pos(Red_M(P1)) = Pos(Red_M(P2))$ so M is a stable model of $P2$.
- $x \notin M$ then $x' \notin M$ by Lemma 6, so $Pos(Red_M(P1)) = Pos(Red_M(P2))$. Hence M is a stable model of $P2$.

We have proved that every stable model of $P1$ is a stable model of $P2$. The proof that every stable model of $P2$ is a stable model of $P1$ is analogous.

The following proposition is given in [LF1].

Proposition 1. *Let $P1$ be a program and Q be a set of atoms that do not occur in $P1$. Let $P2$ be a program that consists of rules of the form*

$$q \leftarrow F \tag{7}$$

where $q \in Q$, and F does not contain any element of Q in the scope of negation as failure. Then $Z \rightarrow Z \setminus Q$ is a $1-1$ correspondence between the answer sets for $P1 \cup P2$ and the answer sets for $P1$.

Lemma 8. *Let P be a normal program $P \cup \delta_P{}^{\leftharpoonup} \equiv_{stable} Trans1(P)$.*

Proof. Follows by a direct generalization of Lemma 7.

Lemma 9. *If $M \cup M'$ is a stable model of $Trans1(P)$ then M is a stable model of P*

Proof. Suppose that $M \cup M'$ is a stable model of $Trans1(P)$, by Lemma 8 then $M \cup M'$ is a stable model of $P \cup \delta_P{}^{\leftharpoonup}$. But by Proposition 1, $(M \cup M') \setminus M'$ is a stable model of P. Therefore M is a stable model of P

Theorem 2. *If $M \cup M'$ is a Pstable model of $Trans1(P)$ then M is a stable model of P*

Proof. Assume that $M \cup M'$ is a pstable model of $Trans1(P)$ then by Lemma 5 $M \cup M'$ is a stable model of $Trans1(P)$. Then by Lemma 9 M is a stable model of P.

5 Conclusions

The main contributions of this work are Theorem 1 and Theorem 2. We were able to prove that Pstable model semantics can express the Stable model semantics, within the class of normal programs, although we conjecture that the expressiveness of pstable model semantics is even greater than that of Stable.

For further research work, we can explore the possibility to extend the theorem to the context of disjunctive programs.

References

[GL1] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press, URL citeseer.ist.psu.edu/gelfond88stable.html.

[LF1] Selim T. Erdogan and Vladimir Lifschitz. *Definitions in Answer Set Programming: (Extended Abstract). Proceedings ICLP 2003*, pages 483-484.

[NM05] Mauricio Osorio Galindo and Juan Antonio Navarro Pérez and José R. Arrazola Rodríguez and Verónica Borja Macías". Logics with common weak completions. *Journal of Logic and Computation*, 2006. URL doi: 10.1093/logcom/exl013