The Stability Formula

Vladimir Lifschitz and Stephen Roller
{vl, roller}@cs.utexas.edu

May 19, 2011

Abstract

We show that the stable models of non-disjunctive logic programs may be expressed using a second-order logical formula syntactically similar to program completion.

1 Introduction

Since its introduction in [Gelfond and Lifschitz, 1988], stable model semantics have repeatedly been shown to be useful in many areas, including both industrial applications [Watson, 1998] and theoretical constructs. Within a classroom setting, however, stable model semantics can often appear remote or unintuitive at first to students.

In this paper, we present the stability formula operator, SF. The stability formula operator is a novel definition of the stable models for logic programs meeting certain syntactic requirements. We show equivalence with the SM operator introduced in [**?**], a known definition of stable model semantics.

Given the many existing definitions of stable models [**?**], another specialized definition may not necessarily be interesting in its own right. However, the terms of the *stability formula* of a program bear a notable syntactic similarity to the *completion formulas* found in program completion semantics [**?**]. When taught side-by-side, the similarity of the stability formula and completion formulas can aid students in understanding the semantic differences.

Stability formulas are only defined for flat, non-disjunctive programs. That is, we only consider programs of the form

$$\Pi = \bigwedge_{i=1}^{n} \tilde{\forall}(F_i \rightarrow P_i(\mathbf{t})),$$

where F_i does not contain implication and $\tilde{\forall}$ is understood as universal closure over all variables. Furthermore, for the sake of simplicity, we consider all predicates to be *intensional predicates* in the sense of [**?**], but this limitation may be straightforwardly addressed within the framework of this proof.

2 The Stability Formula operator

The Stability Formula operator, SF, applied to a logic program, Π, is defined as

$$\bigwedge_{P \in \mathbf{P}} \forall \mathbf{x}(P(\mathbf{x}) \leftrightarrow \forall \mathbf{p}(\Pi^{\diamond}(\mathbf{p}) \rightarrow p(\mathbf{x}))), \tag{1}$$

where $\mathbf{P}$ is a tuple of all predicates appearing in Π, $\mathbf{p}$ is a tuple of corresponding distinct predicate variables, and $\Pi^{\diamond}$ is defined as the result of replacing of each predicate constant P appearing within Π with its corresponding variable p wherever P does not appear in the scope of negation.

Example Consider the simple logic program containing just two rules,

$$\forall x(\neg P(x) \rightarrow Q(x)) \wedge P(a). \tag{2}$$

Since there are two predicates in (2), we will need to construct a term of the stability formula for each of the predicates P and Q. The term for P is

$$\forall x(P(x) \leftrightarrow \forall pq(\Pi^{\diamond}(p, q) \rightarrow p(x)))$$

and the term for Q is

$$\forall x(Q(x) \leftrightarrow \forall pq(\Pi^{\diamond}(p, q) \rightarrow q(x))).$$

Recall that $\Pi^{\diamond}(p, q)$ is defined as replacing every instance of P and Q with p and q whenever the instance is outside the scope of negation. In this example, $\Pi^{\diamond}(p, q)$ is

$$\forall x(\neg P(x) \rightarrow q(x)) \wedge p(a),$$

and so our stability formula may be rewritten as

$$\begin{aligned} \forall x(P(x) \leftrightarrow \forall pq(\forall y(\neg P(y) \rightarrow q(y)) \wedge p(a) \rightarrow p(x))) \wedge \\ \forall x(Q(x) \leftrightarrow \forall pq(\forall y(\neg P(y) \rightarrow q(y)) \wedge p(a) \rightarrow q(x))). \end{aligned} \tag{3}$$

Note that we chose to rename the variable x in (2) to y to improve readability.

We may simplify (3) to only contain first-order quantifiers. The left conjunctive term is equivalent to

$$\forall x(P(x) \leftrightarrow \forall pq(\forall y(\neg P(y) \rightarrow q(y)) \wedge p(a) \rightarrow p(x)))$$

or, equivalently,

$$\forall x(P(x) \leftrightarrow \forall p(\forall y[\neg P(y) \rightarrow \exists q(q(y))] \wedge p(a) \rightarrow p(x))).$$

We may choose q to be the whole universe, so the formula is simplified

$$\forall x(P(x) \leftrightarrow \forall p(p(a) \rightarrow p(x))),$$

or, more concisely,

$$\forall x(P(x) \leftrightarrow x = a).$$

Thus, (3) is equivalent to

$$\begin{aligned}&\forall x(P(x) \leftrightarrow x = a) \wedge \\ &\quad \forall x(Q(x) \leftrightarrow \forall pq(\forall y(\neg P(y) \to q(y)) \wedge p(a) \to q(x)))\end{aligned}$$

and equivalently,

$$\forall x(P(x) \leftrightarrow x = a) \wedge \forall x(Q(x) \leftrightarrow \forall q(\forall y(\neg P(y) \to q(y)) \to q(x))).$$

We notice the left conjunctive term allows us to further simplify the stability formula to

$$\forall x(P(x) \leftrightarrow x = a) \wedge \forall x(Q(x) \leftrightarrow \forall q(\forall y(y \neq a \to q(y)) \to q(x))),$$

which is equivalent to

$$\forall x(P(x) \leftrightarrow x = a) \wedge \forall x(Q(x) \leftrightarrow x \neq a). \tag{4}$$

3 The SM Operator

Before we may present the definition of the SM operator, we should first define some special second-order notation. If p and q are predicate constants of the same arity, then $p \leq q$ stands for $\forall \mathbf{x}(p(\mathbf{x}) \to q(\mathbf{x}))$, where $\mathbf{x}$ is a tuple of distinct object variables. If $\mathbf{p}$ and $\mathbf{q}$ are tuples of predicate constants, $(p_1, \ldots, p_n)$ and $(q_1, \ldots, q_n)$, then $(\mathbf{p} \leq \mathbf{q})$ is shorthand for

$$(p_1 \leq q_1) \wedge \ldots \wedge (p_n \leq q_n)$$

and the formula $(\mathbf{p} < \mathbf{q})$ is shorthand for

$$(\mathbf{p} \leq \mathbf{q}) \wedge \neg(\mathbf{q} \leq \mathbf{p}).$$

The SM operator, as introduced in [**?**], is defined as

$$\Pi \wedge \neg\exists \mathbf{p}((\mathbf{p} < \mathbf{P}) \wedge \Pi^{\diamond}(\mathbf{p})). \tag{5}$$

Informally, we may say that (5) represents the minimal models satisfying Π. The SM operator is limited to logic programs without implication in the bodies of rules. Although it itself a specialization of a more general definition [**?**], its simplicity is particularly convenient for the purpose of this paper.

Example Applying the SM operator to our (2) produces

$$\begin{aligned}&\forall x(\neg P(x) \to Q(x)) \wedge P(a) \wedge \\ &\quad \neg\exists pq[((p, q) < (P, Q)) \wedge \forall x(\neg P(x) \to q(x)) \wedge p(a)].\end{aligned}$$

The upper portion of this formula says P must contain the element a and Q must contain every element P lacks. The lower portion indicates the extents of P and Q are minimal. Clearly, this formula is equivalent to (4).

4 Equivalence of SF and SM

Theorem 1 *For any non-disjunctive logic program* Π*,* $SF[\Pi]$ *is equivalent to* $SM[\Pi]$*.*

Before seeing the proof of equivalence, it may help one to have an intuitive understanding of the proposition. When we say that, a set X is minimal subject to a certain condition, this can be understood in two ways. One is that the condition is not satisfied for any proper subset of X. The other is that each set satisfying the condition is a superset of X, or, in other words, that X is the intersection of all sets satisfying that condition.

Each of the formulas, $SM[\Pi]$ and $SF[\Pi]$, is a minimality condition: $SM[\Pi]$ of the first kind; $SF[\Pi]$ of the second. This theorem shows that for non-disjunctive programs, these two views of minimality are equivalences to each other.

4.1 A Few Lemmas

We begin with several necessary lemmas. We use the following notation throughout:

- Π is a non-disjunctive logic program of the form specified in Section 1.
- $\mathbf{P}$ is the tuple of predicates appearing in Π.
- $\mathbf{p}$, $\mathbf{p}'$ and $\mathbf{p}''$ are tuples of distinct predicate variables corresponding to the members of $\mathbf{P}$.
- $\mathbf{x}$ is a tuple of one or more object variables.
- $\mathbf{t}$ and $\mathbf{u}$ are tuples of one or more object constants.
- P is a member of $\mathbf{P}$, p is a member of $\mathbf{p}$, etc.
- F, G and H are arbitrary formulas without implication.

Lemma 1 *The formula*

$$(\mathbf{p} \leq \mathbf{p}') \wedge F^{\diamond}(\mathbf{p}) \rightarrow F^{\diamond}(\mathbf{p}')$$

is logically valid.

Proof: y structural induction.

Case 1: $F = P_i(\mathbf{t})$: Assume the antecedent. Then $F^{\diamond}(\mathbf{p})$ is $p_i(\mathbf{t})$. From $(\mathbf{p} \leq \mathbf{p}')$ and $p_i(\mathbf{t})$, we have $p'_i(\mathbf{t})$, which is equal to $F^{\diamond}(\mathbf{p}')$.

Case 2: $F = \bot$*,* $F = \top$*, or* $F = (\mathbf{t} = \mathbf{u})$ Then $F^{\diamond}(\mathbf{p}) = F = F^{\diamond}(\mathbf{p}')$.

Case 3: $F = G \wedge H$*.* By the inductive hypothesis, we have

$$(\mathbf{p} \leq \mathbf{p}') \wedge G^{\diamond}(\mathbf{p}) \rightarrow G^{\diamond}(\mathbf{p}') \tag{6}$$

and

$$(\mathbf{p} \leq \mathbf{p}') \wedge H^{\diamond}(\mathbf{p}) \rightarrow H^{\diamond}(\mathbf{p}'). \tag{7}$$

We also have

$$F^{\diamond}(\mathbf{p}) = G^{\diamond}(\mathbf{p}) \wedge H^{\diamond}(\mathbf{p}). \tag{8}$$

From our assumptions, (6), (7), and (8), we derive

$$G^{\diamond}(\mathbf{p}') \wedge H^{\diamond}(\mathbf{p}'),$$

which is equal to $F^{\diamond}(\mathbf{p}')$.

Case 4: $F = G \vee H$. Similar to the conjunctive case.

Case 5: $F = \neg G$. Then $F^{\diamond}(\mathbf{p}) = \neg G = F^{\diamond}(\mathbf{p}')$. ■

Lemma 2 *Let D stand for*

$$\bigwedge_{p \in \mathbf{p}} \forall \mathbf{x}(p(\mathbf{x}) \leftrightarrow p'(\mathbf{x}) \wedge p''(\mathbf{x})), \tag{9}$$

where $\mathbf{p}$, $\mathbf{p}'$ and $\mathbf{p}''$ are same-size tuples of distinct predicate variables. Then the formula

$$D \wedge \Pi^{\diamond}(\mathbf{p}') \wedge \Pi^{\diamond}(\mathbf{p}'') \rightarrow \Pi^{\diamond}(\mathbf{p}) \tag{10}$$

is logically valid.

Proof: irst, assume D. Thus, from (9), we may derive

$$(\mathbf{p} \leq \mathbf{p}') \tag{11}$$

and

$$(\mathbf{p} \leq \mathbf{p}''). \tag{12}$$

Next, assume

$$\Pi^{\diamond}(\mathbf{p}') \wedge \Pi^{\diamond}(\mathbf{p}'')$$

or, equivalently,

$$\bigwedge_{i=1}^{n} \tilde{\forall}(F_i^{\diamond}(\mathbf{p}') \rightarrow p_i'(\mathbf{t})) \wedge \bigwedge_{i=1}^{n} \tilde{\forall}(F_i^{\diamond}(\mathbf{p}'') \rightarrow p_i''(\mathbf{t})). \tag{13}$$

Similarly, we expand our goal, $\Pi^{\diamond}(\mathbf{p})$ as

$$\bigwedge_{i=1}^{n} \tilde{\forall}(F_i^{\diamond}(\mathbf{p}) \rightarrow p_i(\mathbf{t})). \tag{14}$$

For the ith term of (14), assume

$$F_i^{\diamond}(\mathbf{p}). \tag{15}$$

By Lemma 1, (11) and (15), $F_i^{\diamond}(\mathbf{p}')$. Similarly, from (12) and (15), $F_i^{\diamond}(\mathbf{p}'')$. Thus from (13),

$$p_i'(\mathbf{t}) \wedge p_i''(\mathbf{t}),$$

or equivalently, by our definition of D,

$$p_i(\mathbf{t}). \tag{16}$$

Thus we have shown our goal, (14). ■

Lemma 3 Π *entails*

$$\forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \rightarrow (\mathbf{P} \leq \mathbf{p})) \leftrightarrow \forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \rightarrow \neg(\mathbf{p} < \mathbf{P})). \tag{17}$$

Proof: ($\Rightarrow$) Assume $(\mathbf{P} \leq \mathbf{p})$. Then,

$$\neg(\mathbf{p} \leq \mathbf{P}) \vee (\mathbf{P} \leq \mathbf{p}),$$

which is equivalent to

$$\neg((\mathbf{p} \leq \mathbf{P}) \wedge \neg(\mathbf{P} \leq \mathbf{p})),$$

which is the definition of $\neg(\mathbf{p} < \mathbf{P})$.
($\Leftarrow$) Assume Π. Take $\mathbf{p}'$ such that it is the intersection of $\mathbf{P}$ and an arbitrary $\mathbf{p}$. That is, let $\mathbf{p}'$ be defined such that

$$\bigwedge_{P \in \mathbf{P}} \forall \mathbf{x}(p'(\mathbf{x}) \leftrightarrow P(\mathbf{x}) \wedge p(\mathbf{x})). \tag{18}$$

We assume the right-hand side of (17) and take $\mathbf{p}'$ as $\mathbf{p}$, giving

$$\Pi^\diamond(\mathbf{p}') \rightarrow \neg(\mathbf{p}' < \mathbf{P}),$$

which is equivalent to

$$\Pi^\diamond(\mathbf{p}') \rightarrow \neg(\mathbf{p}' \leq \mathbf{P}) \vee (\mathbf{P} \leq \mathbf{p}'),$$

and equivalently,

$$\Pi^\diamond(\mathbf{p}') \rightarrow \neg\left(\bigwedge_{P \in \mathbf{P}} \forall \mathbf{x}(p(\mathbf{x}) \wedge P(\mathbf{x}) \rightarrow P(\mathbf{x}))\right) \vee (\mathbf{P} \leq \mathbf{p}'). \tag{19}$$

Next, we notice that each of the implications within the left disjunctive term of (19) are trivially true, making the entire left disjunctive term universally false. Thus, (19) is equivalent to

$$\Pi^\diamond(\mathbf{p}') \rightarrow (\mathbf{P} \leq \mathbf{p}'),$$

and equivalently,

$$\Pi^\diamond(\mathbf{p}') \rightarrow \left(\bigwedge_{P \in \mathbf{P}} \forall \mathbf{x}(P(\mathbf{x}) \rightarrow P(\mathbf{x}) \wedge p(\mathbf{x}))\right),$$

which may be simplified to

$$\Pi^\diamond(\mathbf{p}') \rightarrow \left(\bigwedge_{P \in \mathbf{P}} \forall \mathbf{x}(P(\mathbf{x}) \rightarrow p(\mathbf{x}))\right),$$

or simply,

$$\Pi^\diamond(\mathbf{p}') \rightarrow (\mathbf{P} \leq \mathbf{p}). \tag{20}$$

Next, assume (18), $\Pi^\diamond(\mathbf{P})$ and $\Pi^\diamond(\mathbf{p})$. Recall that by definition of the diamond operator, $\Pi^\diamond(\mathbf{P}) = \Pi$. Then applying Lemma 2, with $\mathbf{P}$ as $\mathbf{p}'$, $\mathbf{p}$ as $\mathbf{p}''$ and $\mathbf{p}'$ as $\mathbf{p}$, gives

$$\Pi^\diamond(\mathbf{p}'). \tag{21}$$

Finally, from (20) and (21),

$$(\mathbf{P} \leq \mathbf{p}). \quad \blacksquare$$

Lemma 4 Π *entails*

$$\left(\bigwedge_{i=1}^{n} \forall \mathbf{x}(P_i(\mathbf{x}) \rightarrow \forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \rightarrow p_i(\mathbf{x})))\right) \leftrightarrow (\neg \exists \mathbf{p}((\mathbf{p} < \mathbf{P}) \wedge \Pi^\diamond(\mathbf{p}))) \, . \tag{22}$$

Proof: Assume Π. We begin by noting that the left hand side is equivalent to

$$\forall \mathbf{p} \left(\bigwedge_{i=1}^{n} \forall \mathbf{x}(P_i(\mathbf{x}) \rightarrow (\Pi^\diamond(\mathbf{p}) \rightarrow p_i(\mathbf{x})))\right),$$

which is equivalent to

$$\forall \mathbf{p} \left(\bigwedge_{i=1}^{n} \forall \mathbf{x}(\Pi^\diamond(\mathbf{p}) \rightarrow (P_i(\mathbf{x}) \rightarrow p_i(\mathbf{x})))\right),$$

which is also equivalent to

$$\forall \mathbf{p} \left(\Pi^\diamond(\mathbf{p}) \rightarrow \bigwedge_{i=1}^{n} \forall \mathbf{x}(P_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))\right). \tag{23}$$

At this point, we notice that the right side of the consequent is the definition of $(P_i \leq p_i)$, so we may rewrite (23) as

$$\forall \mathbf{p} \left(\Pi^\diamond(\mathbf{p}) \rightarrow \bigwedge_{i=1}^{n} (P_i \leq p_i)\right). \tag{24}$$

Similarly, we notice that the consequent of this sentence is the definition of $(\mathbf{P} \leq \mathbf{p})$, thus (24) is equivalent to

$$\forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \rightarrow (\mathbf{P} \leq \mathbf{p})). \tag{25}$$

From our original assumption, Π, we notice that we apply Lemma 3 and rewrite (25) equivalently as

$$\forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \rightarrow \neg(\mathbf{p} \leq \mathbf{P})). \tag{26}$$

From here, we rewrite implication as disjunction and apply De Morgan's laws, thus (26) is equivalent to

$$\forall \mathbf{p} \neg((\mathbf{p} < \mathbf{P}) \wedge \Pi^\diamond(\mathbf{p})),$$

which is equivalent to the right hand side of (22). $\blacksquare$

Lemma 5 *The formula*

$$(SF[\Pi] \wedge \Pi^\diamond(\mathbf{p}) \wedge F) \to F^\diamond(\mathbf{p}). \tag{27}$$

is logically valid.

Proof: y induction on F. Assume $\Pi^\diamond(\mathbf{p})$, F and $SF[\Pi]$. Recall from Section 1 that $SF[\Pi]$ is defined as

$$\bigwedge_{i=1}^{n} \forall \mathbf{x}(\mathbf{p}(\Pi^\diamond(\mathbf{p}) \to p_i(\mathbf{x})) \leftrightarrow P_i(\mathbf{x})). \tag{28}$$

Case 1: $F = P(\mathbf{t})$. From $SF[\Pi]$ and (27), taking $\mathbf{x}$ to be $\mathbf{t}$, we derive

$$\forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \to p(\mathbf{t})).$$

From this sentence and the assumption $\Pi^\diamond(\mathbf{p})$, we derive $p(\mathbf{t})$, which is $F^\diamond(\mathbf{p})$.

Case 2: $F = \neg G$. Then $F = \neg G = F^\diamond(\mathbf{p})$.

Case 3: $F = \bot, F = \top$, *or* $F = (\mathbf{t} = \mathbf{u})$. Then, just as in case 2, F will be equal to $F^\diamond(\mathbf{p})$.

Case 4: $F = G \wedge H$. By the inductive hypothesis,

$$(\Pi^\diamond(\mathbf{p}) \wedge G) \to G^\diamond(\mathbf{p}) \tag{29}$$

and

$$(\Pi^\diamond(\mathbf{p}) \wedge H) \to H^\diamond(\mathbf{p}). \tag{30}$$

From (29) and (30), we derive

$$(\Pi^\diamond(\mathbf{p}) \wedge G \wedge H) \to (G^\diamond(\mathbf{p}) \wedge H^\diamond(\mathbf{p})),$$

which is, of course, equal to

$$(\Pi^\diamond(\mathbf{p}) \wedge F) \to F^\diamond(\mathbf{p}). \tag{31}$$

Thus from (31) and our assumptions, $\Pi^\diamond(\mathbf{p})$ and F, we derive $F^\diamond(\mathbf{p})$.

Case 5: $F = G \vee H$. Similar to the conjunctive case, we may assume the inductive hypotheses, (29) and (30).

We now consider two cases: If G, then from (29), we derive $G^\diamond(\mathbf{p})$, and consequently $G^\diamond(\mathbf{p}) \vee H^\diamond(\mathbf{p})$; if H, then from (30), we derive $H^\diamond(\mathbf{p})$, and consequently $G^\diamond(\mathbf{p}) \vee H^\diamond(\mathbf{p})$. ■

Lemma 6 *The formula*

$$SF[\Pi] \to \Pi$$

is logically valid.

Proof: Assume $SF[\Pi]$, that is, (1). We wish to show Π follows. Take any rule of Π:

$$\tilde{\forall}(F_j \rightarrow P_j(\mathbf{t})). \tag{32}$$

Now we also assume F_j and need to show $P_j(\mathbf{t})$ in order to show Π is entailed.

Next, we assume $\Pi^\diamond(\mathbf{p})$ for some arbitrary $\mathbf{p}$. That is, in addition to our previous assumptions F_j and $SM[\Pi]$, we also assume

$$\bigwedge_{i=1}^{n} \tilde{\forall}(F_i^\diamond(\mathbf{p}) \rightarrow p_i(\mathbf{t})). \tag{33}$$

From these three assumptions, we apply Lemma 5 in order to derive $F_j^\diamond(\mathbf{p})$. From this conclusion and (33), we conclude $p_j(\mathbf{t})$. That is, we have shown that the formula

$$(SM[\Pi] \wedge F_j) \rightarrow \forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \rightarrow p_j(\mathbf{t})) \tag{34}$$

is logically valid. From (1) right-to-left, we notice that the consequent of (34) is equivalent to simply $P_j(\mathbf{t})$. Thus we have shown the logical validity of

$$(SM[\Pi] \wedge F_j) \rightarrow P_j(\mathbf{t}),$$

or, equivalently,

$$SM[\Pi] \rightarrow (F_j \rightarrow P_j(\mathbf{t})). \quad \blacksquare$$

Lemma 7 *The formula*

$$\Pi \rightarrow \left(\bigwedge_{i=1}^{n} \forall \mathbf{x}(\forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \rightarrow p_i(\mathbf{x})) \rightarrow P_i(\mathbf{x})) \right)$$

is logically valid.

Proof: Assume Π and

$$\forall \mathbf{p}(\Pi^\diamond(\mathbf{p}) \rightarrow p_i(\mathbf{x})).$$

If we then take $\mathbf{p}$ to be $\mathbf{P}$, we derive

$$\Pi^\diamond(\mathbf{P}) \rightarrow P_i(\mathbf{x}) \tag{35}$$

However, $\Pi^\diamond(\mathbf{P})$ is equal to Π. Thus (35) is equal to

$$\Pi \rightarrow P_i(\mathbf{x}).$$

Thus from our original assumption, Π, we derive $P_i(\mathbf{x})$. $\blacksquare$

4.2 Proof of Theorem 1

($\Rightarrow$) Assume $SF[\Pi]$, that is, (1). By Lemma 6, then Π. From (1),

$$\bigwedge_{i=1}^{n} \forall \mathbf{x}(P_i(\mathbf{x}) \rightarrow \forall \mathbf{p}(\Pi^{\diamond}(\mathbf{p}) \rightarrow p_i(\mathbf{x}))). \tag{36}$$

Then, by Lemma 4,

$$\neg \exists \mathbf{p}((\mathbf{p} < \mathbf{P}) \wedge \Pi^{\diamond}(\mathbf{p})).$$

Thus we have derived both conjunctive terms of $SM[\Pi]$.

($\Leftarrow$) Assume $SM[\Pi]$. By Lemma 7,

$$\forall \mathbf{x}(\forall \mathbf{p}(\Pi^{\diamond}(\mathbf{p}) \rightarrow p_i(\mathbf{x})) \rightarrow P_i(\mathbf{x})). \tag{37}$$

By Lemma 4, it follows that

$$\bigwedge_{i=1}^{n} \forall \mathbf{x}(\forall \mathbf{p}(\Pi^{\diamond}(\mathbf{p}) \rightarrow p_i(\mathbf{x})) \leftarrow P_i(\mathbf{x})). \tag{38}$$

From (37) and (38), $SF[\Pi]$. ■

5 Conclusion

We have presented a definition of the stability formula operator, a novel definition of stable model semantics for non-disjunctive programs. We have shown its equivalence to the SM operator, a known version of stable model semantics for limited programs. The structure of a stability formula is syntactically similar to program completion, making it excellent for use within a classroom setting.

References

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Ferraris *et al.*, 2011] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription. *Artificial Intelligence*, 175:236–263, 2011.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski, Bowen, and Kenneth, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

[Lifschitz, 2010] Vladimir Lifschitz. Thirteen definitions of a stable model. In *Fields of Logic and Computation: Essays Dedicated to Yuri Gurevich on the Occasion of his 70th Birthday*, pages 488–503. Springer, 2010.

[Lifschitz, 2011] Vladimir Lifschitz. Datalog programs and their stable models[1]. In *Datalog 2.0 Post Workshop Proceedings*. Springer, 2011. To appear.

[Watson, 1998] Richard Watson. An application of action theory to the space shuttle. In Gopal Gupta, editor, *Proceedings First Int'l Workshop on Practical Aspects of Declarative Languages (Lecture Notes in Computer Science 1551)*, pages 290–304. Springer, 1998.

[1] http://www.cs.utexas.edu/users/vl/papers/dpsm.pdf