Safe Formulas in the General Theory of Stable Models

Joohyung Lee[1], Vladimir Lifschitz[2], and Ravi Palla[1]

[1]School of Computing and Informatics, Arizona State University, USA
[2]Department of Computer Sciences, University of Texas at Austin, USA
[1]{joolee,Ravi.Palla}@asu.edu, [2]vl@cs.utexas.edu

Abstract. Safe first-order formulas generalize the concept of a safe rule, which plays an important role in the design of answer set solvers. We show that any safe sentence is equivalent, in a certain sense, to the result of its grounding—to the variable-free sentence obtained from it by replacing all quantifiers with multiple conjunctions and disjunctions. It follows that a safe sentence and the result of its grounding have the same stable models, and that the stable models of a safe sentence can be characterized by a formula of a simple syntactic form.

1 Introduction

The definition of a stable model proposed in [1] is more general than the original definition from [2]: it applies to models of arbitrary first-order sentences. Logic programs referred to in the 1988 definition are identified in this theory with first-order formulas of a special form. For instance, the rule

$$p(x) \leftarrow not\ q(x) \tag{1}$$

is treated as alternative notation for the sentence

$$\forall x(\neg q(x) \rightarrow p(x)). \tag{2}$$

In this example, stable models are the interpretations of the unary predicate constants p and q (in the sense of first-order logic) that make p identically true and q identically false.

This general definition of a stable model involves a syntactic transformation of formulas, which is reviewed in Section 3 below. That transformation is similar to the circumscription operator [3]—it turns a first-order sentence into a stronger second-order sentence. There is an important difference, however, between stable models and models of circumscription. Two sentences may be equivalent (that is, have the same models), but have different *stable* models. For instance, formula (2) is equivalent to

$$\forall x(\neg p(x) \rightarrow q(x)),$$

but the stable models of these two formulas are not the same. The equivalent transformations of formulas that preserve their stable models are studied in [4].

They are represented there by a subsystem of classical logic called $\mathbf{SQHT}^{=}$ ("static quantified logic of here-and-there with equality"). This deductive system includes all axioms and inference rules of intuitionistic logic with equality, the decidable equality axiom

$$x = y \vee x \neq y \tag{3}$$

and two other axiom schemas, but it does not include the general law of the excluded middle $F \vee \neg F$.

In [5], the new approach to stable models is used to define the semantics of an answer set programming language with choice rules and counting, called RASPL-1. The meaning of a RASPL-1 program is defined in terms of the stable models of a first-order sentence associated with the program, which is called its "FOL-representation." For instance, the FOL-representation of the RASPL-1 rule

$$p \leftarrow \{x : q(x)\}\, 1 \tag{4}$$

is the formula

$$\neg\exists xy(q(x) \wedge q(y) \wedge x \neq y) \rightarrow p. \tag{5}$$

In this paper, we continue one line of research from [5], the study of safe sentences and their stable models. It extends the familiar concept of a safe rule, which plays an important role in the design of answer set solvers [6, Section 2.1]. For instance, rule (1) is not safe, and for this reason it is not allowed in the input of any of the existing systems for computing stable models. Rule (4) is safe, and we expect that it will be accepted by a future implementation of RASPL-1.

According to Proposition 1 below, stable models of a safe sentence (without function symbols) have what can be called the "small predicate property": the relation represented by any of its predicate constants can hold for a tuple of arguments only if each member of the tuple is represented by an object constant. We show, furthermore, that any safe sentence is equivalent, in a certain sense, to the result of its grounding—to the variable-free sentence obtained from it by replacing all quantifiers with multiple conjunctions and disjunctions (Proposition 2). We derive from these two facts that a safe sentence and the result of its grounding have the same stable models (Proposition 3). This theorem leads us to the conclusion that stable models of a safe sentence can be characterized by a sentence of a simple syntactic structure—not just first-order, but universal and, moreover, "almost variable-free" (Proposition 4).

The discussion of stable models of safe sentences here is more general than in [5], because it is not limited to Herbrand models. This may be essential for future applications of stable models to knowledge representation. The theorem about stable Herbrand models stated in [5] is now extended to arbitrary stable models (Proposition 5).

A preliminary report on this work appeared in [7].

2 Review: Safe Sentences

We consider first-order formulas that may contain object constants and equality but no function constants of arity > 0. The propositional connectives

$$\bot \quad \wedge \quad \vee \quad \rightarrow$$

will be treated as primitive; $\neg F$ is shorthand for $F \rightarrow \bot$, $F \leftrightarrow G$ is shorthand for $(F \rightarrow G) \wedge (G \rightarrow F)$, and $\top$ is shorthand for $\bot \rightarrow \bot$. A *sentence* is a formula without free variables.

Recall that a traditional rule—an implication of the form

$$(L_1 \wedge \cdots \wedge L_n) \rightarrow A, \tag{6}$$

not containing equality, where $L_1, \ldots, L_n$ are literals and A is an atom—is considered safe if every variable occurring in it occurs in one of the positive literals in the antecedent. The definition of a safe formula from [5], reproduced below, generalizes this condition to arbitrary sentences in prenex form. The assumption that the formula is in prenex form is not a significant limitation in the general theory of stable models, because all steps involved in the standard process of converting a formula to prenex form are equivalent transformations in $\mathbf{SQHT}^=$ [8]. For instance, formula (5) is equivalent in this system to its prenex form

$$\exists xy(\neg(q(x) \wedge q(y) \wedge x \neq y) \rightarrow p). \tag{7}$$

To every quantifier-free formula F we assign a set $\mathrm{RV}(F)$ of its *restricted variables* as follows:[1]

- For an atomic formula F,
  - if F is an equality between two variables then $\mathrm{RV}(F) = \emptyset$;
  - otherwise, $\mathrm{RV}(F)$ is the set of all variables occurring in F;
- $\mathrm{RV}(\bot) = \emptyset$;
- $\mathrm{RV}(F \wedge G) = \mathrm{RV}(F) \cup \mathrm{RV}(G)$;
- $\mathrm{RV}(F \vee G) = \mathrm{RV}(F) \cap \mathrm{RV}(G)$;
- $\mathrm{RV}(F \rightarrow G) = \emptyset$.

We say that a variable x is restricted in F if x belongs to $\mathrm{RV}(F)$. It is clear, for instance, that a variable is restricted in the antecedent of (6) iff it occurs in one of the positive literals among $L_1, \ldots, L_n$.

Consider a sentence F in prenex form:

$$Q_1 x_1 \cdots Q_n x_n M \tag{8}$$

(each Q_i is $\forall$ or $\exists$; $x_1, \ldots, x_n$ are distinct variables; the matrix M is quantifier-free). We say that F is *safe* if every occurrence of each of the variables x_i in M is contained in an occurrence of a subformula $G \rightarrow H$ that satisfies two conditions:

[1] Some clauses of this definition are similar to parts of the definition of an allowed formula in [9]. That paper was written before the invention of the stable model semantics, and long before the emergence of answer set programming.

(a) the subformula is positive[2] in M if Q_i is $\forall$, and negative in M if Q_i is $\exists$;
(b) x_i is restricted in G.

Consider, for instance, the universal closure of a formula of the form (6). If each of its variables occurs in a positive literal in the antecedent then the matrix (6) plays the role of the implication $G \to H$ from the definition of a safe sentence.

To give another example, sentence (7) is safe because the antecedent of the implication in it can be taken as $G \to H$, with $q(x) \wedge q(y) \wedge x \neq y$ as G and $\bot$ as H.

3 Review: Stable Models

The definition of the "stable model operator" SM in [1] uses the following notation from [10]. Let $\mathbf{p}$ be a list of distinct predicate constants $p_1, \dots, p_n$, and let $\mathbf{u}$ be a list of distinct predicate variables $u_1, \dots, u_n$ of the same length as $\mathbf{p}$. By $\mathbf{u} = \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \leftrightarrow p_i(\mathbf{x}))$, where $\mathbf{x}$ is a list of distinct object variables of the same arity as the length of p_i, for all $i = 1, \dots n$. By $\mathbf{u} \leq \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \to p_i(\mathbf{x}))$ for all $i = 1, \dots n$, and $\mathbf{u} < \mathbf{p}$ stands for $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{u} = \mathbf{p})$. For instance, if p and q are unary predicate constants then $(u, v) < (p, q)$ is

$$\begin{array}{l} \forall x(u(x) \to p(x)) \wedge \forall x(v(x) \to q(x)) \\ \quad \wedge \neg(\forall x(u(x) \leftrightarrow p(x)) \wedge \forall x(v(x) \leftrightarrow q(x))). \end{array}$$

For any first-order sentence F, SM[F] stands for the second-order sentence

$$F \wedge \neg \exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u})), \tag{9}$$

where $\mathbf{p}$ is the list $p_1, \dots, p_n$ of all predicate constants occurring in F, $\mathbf{u}$ is a list $u_1, \dots, u_n$ of distinct predicate variables, and $F^*(\mathbf{u})$ is defined recursively:

- $p_i(t_1, \dots, t_m)^* = u_i(t_1, \dots, t_m)$;
- $(t_1 = t_2)^* = (t_1 = t_2)$;
- $\bot^* = \bot$;
- $(F \wedge G)^* = F^* \wedge G^*$;
- $(F \vee G)^* = F^* \vee G^*$;
- $(F \to G)^* = (F^* \to G^*) \wedge (F \to G)$;
- $(\forall x F)^* = \forall x F^*$;
- $(\exists x F)^* = \exists x F^*$.

[2] A subexpression of a formula is said to be *positive* in it if the number of implications that contain that subexpression in the antecedent is even, and *negative* if it is odd. This should not be confused with the distinction between positive and negative literals.

An interpretation of the signature $\sigma(F)$ consisting of the object and predicate constants occurring in F is a *stable model* of F if it satisfies SM[F].

For instance, if F is

$$p(a) \wedge \forall x(p(x) \rightarrow q(x)) \tag{10}$$

then $F^*(u, v)$ is

$$u(a) \wedge \forall x((u(x) \rightarrow v(x)) \wedge (p(x) \rightarrow q(x)))$$

and SM[F] is

$$\begin{array}{l} p(a) \wedge \forall x(p(x) \rightarrow q(x)) \\ \quad \wedge \neg\exists uv(((u, v) < (p, q)) \wedge u(a) \wedge \forall x((u(x) \rightarrow v(x)) \wedge (p(x) \rightarrow q(x)))). \end{array}$$

This formula is equivalent to the first-order formula

$$\forall x(p(x) \leftrightarrow x = a) \wedge \forall x(q(x) \leftrightarrow p(x)). \tag{11}$$

Consequently, the stable models of (10) can be characterized as the interpretations satisfying (11).

4 The Small Predicate Property

Proposition 1 below shows that all stable models of a safe sentence have the small predicate property: the relation represented by any of its predicate constants p_i can hold for a tuple of arguments only if each member of the tuple is represented by an object constant occurring in F. To make this idea precise, we will use the following notation: for any finite set $\mathbf{c}$ of object constants, $in_{\mathbf{c}}(x_1, \ldots, x_m)$ stands for the formula

$$\bigwedge_{1 \leq j \leq m} \bigvee_{c \in \mathbf{c}} x_j = c.$$

The small predicate property can be expressed by the conjunction of the sentences

$$\forall \mathbf{x}(p_i(\mathbf{x}) \rightarrow in_{\mathbf{c}}(\mathbf{x}))$$

for all predicate constants p_i occurring in F, where $\mathbf{x}$ is a list of distinct variables. We will denote this sentence by $SPP_{\mathbf{c}}$. By $c(F)$ we denote the set of all object constants occurring in F.

Proposition 1 *For any safe sentence F,* SM[F] *entails* $SPP_{c(F)}$.

For instance, in application to the prenex form of (10) this proposition asserts that SM[F] entails

$$\forall x(p(x) \rightarrow x = a) \wedge \forall x(q(x) \rightarrow x = a).$$

Corollary 1 *For any safe sentence F that does not contain object constants,* SM[F] *entails the formulas* $\forall \mathbf{x} \neg p_i(\mathbf{x})$ *for all predicate constants p_i of arity > 0.*

Indeed, $SPP_\emptyset$ is equivalent to the conjunction of all these formulas.

We will show now how to prove a more general form of Proposition 1, in which the safety assumption is weakened. We call a sentence (8) in prenex form *semi-safe* if every strictly positive[3] occurrence of each of the variables x_i in M is contained in a subformula $G \to H$ such that x_i is restricted in G. For instance, the sentence

$$\forall x(\neg p(x) \to q),$$

representing the rule

$$q \leftarrow not\ p(x),$$

is not safe, but semi-safe because the occurrence of x is not strictly positive. The statement of Proposition 1 holds for all semi-safe sentences.

The notation that we use in the proof involves *predicate expressions* of the form

$$\lambda \mathbf{x} F(\mathbf{x}), \tag{12}$$

where $F(\mathbf{x})$ is a formula. If e is (12) and $G(p)$ is a formula containing a predicate constant p of the same arity as the length of $\mathbf{x}$ then $G(e)$ stands for the result of replacing each atomic part of the form $p(\mathbf{t})$ in $G(p)$ with $F(\mathbf{t})$, after renaming the bound variables in $G(p)$ in the usual way, if necessary. For instance, if $G(p)$ is $p(a) \vee p(b)$ then $G(\lambda y(x = y))$ is $x = a \vee x = b$. Substituting a tuple $\mathbf{e}$ of predicate expressions for a tuple $\mathbf{p}$ of predicate constants is defined in a similar way.

For any finite set $\mathbf{c}$ of object constants, by $\mathbf{e_c}$ we denote the list of predicate expressions

$$\lambda \mathbf{x}(p_i(\mathbf{x}) \wedge in_{\mathbf{c}}(\mathbf{x}))$$

for all predicate constants p_i.

The following two lemmas can be proved by induction on F. The first of them is stated as Lemma 10 in [11].

Lemma 1 *For any formula F,*

$$((\mathbf{u} \leq \mathbf{p}) \wedge F^*(\mathbf{u})) \to F$$

is logically valid.

Lemma 2 *For any quantifier-free formula F and any finite set $\mathbf{c}$ of object constants containing $c(F)$,*

$$F^*(\mathbf{e_c}) \to in_{\mathbf{c}}(\mathrm{RV}(F))$$

is logically valid.

About a variable x occurring in a quantifier-free formula F we say that it is *semi-safe* in F if every strictly positive occurrence of x in F belongs to a subformula $G \to H$ such that x is restricted in G. It is clear that a sentence in prenex form is semi-safe iff all variables in its matrix are semi-safe. By $\mathrm{NS}(F)$ we will denote the set of the variables of F that are *not* semi-safe.

[3] We say that an occurrence of a subformula or a variable in a formula F is *strictly positive* if that occurrence is not in the antecedent of any implication.

Lemma 3 *For any quantifier-free formula F and any finite set $\mathbf{c}$ of object constants containing $c(F)$,*

$$(F \wedge in_{\mathbf{c}}(\mathrm{NS}(F))) \rightarrow F^*(\mathbf{e_c}) \tag{13}$$

is logically valid.

Proof. By induction on F. We only consider the case when F is $G \rightarrow H$; the other cases are straightforward. By the induction hypothesis,

$$(H \wedge in_{\mathbf{c}}(\mathrm{NS}(H))) \rightarrow H^*(\mathbf{e_c}) \tag{14}$$

is logically valid. By Lemma 1, since $\mathbf{e_c} \leq \mathbf{p}$,

$$G^*(\mathbf{e_c}) \rightarrow G \tag{15}$$

is logically valid. By Lemma 2,

$$G^*(\mathbf{e_c}) \rightarrow in_{\mathbf{c}}(\mathrm{RV}(G)) \tag{16}$$

is logically valid. Assume the antecedent of (13)

$$(G \rightarrow H) \wedge in_{\mathbf{c}}(\mathrm{NS}(G \rightarrow H)). \tag{17}$$

Assume $G^*(\mathbf{e_c})$; our goal is to derive $H^*(\mathbf{e_c})$. By (15), G; by the first conjunctive term of (17), H. By (16),

$$in_{\mathbf{c}}(\mathrm{RV}(G)). \tag{18}$$

Note that $\mathrm{NS}(H) \subseteq \mathrm{NS}(G \rightarrow H) \cup \mathrm{RV}(G)$. Consequently, from the second conjunctive term of (17) and (18),

$$in_{\mathbf{c}}(\mathrm{NS}(H)). \tag{19}$$

From H, (19) and (14), $H^*(\mathbf{e_c})$. □

Lemma 4 *For any semi-safe sentence F and any finite set $\mathbf{c}$ of object constants containing $c(F)$, F entails $F^*(\mathbf{e_c})$.*

Proof. Immediate from Lemma 3. □

Proposition 1, Stronger Form For any semi-safe sentence F, $\mathrm{SM}[F]$ entails $SPP_{c(F)}$.

Proof. Assume F and $\neg SPP_{c(F)}$; we will derive

$$\exists \mathbf{u}(\mathbf{u} < \mathbf{p} \wedge F^*(\mathbf{u})).$$

To this end, we will prove

$$(\mathbf{e}_{c(F)} < \mathbf{p}) \wedge F^*(\mathbf{e}_{c(F)}).$$

By Lemma 4, it is sufficient to prove the first conjunctive term, that is,

$$\bigwedge_{p\in\mathbf{p}}\Big(\forall\mathbf{x}\Big(p(\mathbf{x})\wedge in_{c(F)}(\mathbf{x})\to p(\mathbf{x})\Big)\Big)$$
$$\wedge\neg\bigwedge_{p\in\mathbf{p}}\forall\mathbf{x}\Big(p(\mathbf{x})\to\Big(p(\mathbf{x})\wedge in_{c(F)}(\mathbf{x})\Big)\Big). \tag{20}$$

The first conjunctive term of (20) is logically valid, and the second is equivalent to $\neg SPP_{c(F)}$. □

5 Grounding

The process of grounding replaces quantifiers by multiple conjunctions and disjunctions. To make this idea precise, we define, for any sentence F in prenex form and any nonempty finite set $\mathbf{c}$ of object constants, the variable-free formula $\mathrm{Ground}_{\mathbf{c}}[F]$ as follows. If F is quantifier-free then $\mathrm{Ground}_{\mathbf{c}}[F]=F$. Otherwise,

$$\mathrm{Ground}_{\mathbf{c}}[\forall xF(x)]=\bigwedge_{c\in\mathbf{c}}\mathrm{Ground}_{\mathbf{c}}[F(c)],$$

$$\mathrm{Ground}_{\mathbf{c}}[\exists xF(x)]=\bigvee_{c\in\mathbf{c}}\mathrm{Ground}_{\mathbf{c}}[F(c)].$$

As in [4], by $\mathbf{INT}^{=}$ we denote intuitionistic predicate logic with equality, and DE stands for the decidable equality axiom (3). The importance of the logical system $\mathbf{INT}^{=}+\mathrm{DE}$ is determined by the fact that it is a part of $\mathbf{SQHT}^{=}$, so that the provability of a sentence $F\leftrightarrow G$ in this system implies that $\mathrm{SM}[F]$ is equivalent to $\mathrm{SM}[G]$.

Proposition 2 *For any safe sentence F and any nonempty finite set $\mathbf{c}$ of object constants containing $c(F)$, the equivalence*

$$\mathrm{Ground}_{\mathbf{c}}[F]\leftrightarrow F$$

is derivable from $SPP_{\mathbf{c}}$ in $\mathbf{INT}^{=}+\mathrm{DE}$.

Lemma 5 *If any of the sentences $\forall xF(x)$, $\exists xF(x)$ is safe then so is $F(c)$ for any object constant c.*

Proof. Immediate from the fact, easily verified by induction, that if a variable other than x is restricted in a formula $G(x)$ then it is restricted in $G(c)$ as well. □

Lemma 6 *If x is restricted in a quantifier-free formula $F(x)$, and $\mathbf{c}$ is a nonempty finite set of object constants containing $c(F)$, then the formula*

$$F(x)\to in_{\mathbf{c}}(x)$$

is derivable from $SPP_{\mathbf{c}}$ in $\mathbf{INT}^{=}$.

Proof. Immediate by induction on $F(x)$. □

Lemma 7 *For any formula $F(x)$ in prenex form that has no free variables other than x, and for any nonempty finite set $\mathbf{c}$ of object constants containing $c(F)$,*

(a) if the sentence $\forall x F(x)$ is safe then the equivalence

$$\forall x F(x) \leftrightarrow \bigwedge_{c \in \mathbf{c}} F(c)$$

is derivable from $SPP_{\mathbf{c}}$ in $\mathbf{INT}^{=} + \mathrm{DE}$;

(b) if the sentence $\exists x F(x)$ is safe then the equivalence

$$\exists x F(x) \leftrightarrow \bigvee_{c \in \mathbf{c}} F(c)$$

is derivable from $SPP_{\mathbf{c}}$ in $\mathbf{INT}^{=} + \mathrm{DE}$.

Proof. (a) Assume that $\forall x F(x)$ is safe. In $\mathbf{INT}^{=} + \mathrm{DE}$, this formula can be equivalently written as

$$\forall x((in_{\mathbf{c}}(x) \to F(x)) \wedge (\neg in_{\mathbf{c}}(x) \to F(x))),$$

and consequently as

$$\bigwedge_{c \in \mathbf{c}} F(c) \ \wedge \ \forall x(\neg in_{\mathbf{c}}(x) \to F(x)). \tag{21}$$

Consider the maximal subformulas $G(x) \to H(x)$ of $F(x)$, positive in $F(x)$, such that x is restricted in $G(x)$. From Lemma 6 we conclude that for each of these subformulas, the implication

$$G(x) \to in_{\mathbf{c}}(x)$$

is derivable from $SPP_{\mathbf{c}}$ in $\mathbf{INT}^{=}$, and consequently so is

$$\neg in_{\mathbf{c}}(x) \to (G(x) \to H(x)).$$

It follows that, under the assumption $SPP_{\mathbf{c}}$, (21) can be equivalently rewritten as

$$\bigwedge_{c \in \mathbf{c}} F(c) \ \wedge \ \forall x(\neg in_{\mathbf{c}}(x) \to S), \tag{22}$$

where S is the formula obtained from $F(x)$ by replacing each of these maximal subformulas $G(x) \to H(x)$ with $\top$. Since $\forall x F(x)$ is safe, x does not occur in S. It follows that S can be obtained from $F(c)$ in the same way as it was obtained from $F(x)$, that is, by replacing some subformulas that are positive in $F(c)$ with $\top$. Consequently, the formula $F(c) \to S$ is intuitionistically provable, and so is

$$F(c) \to \forall x(\neg in_{\mathbf{c}}(x) \to S).$$

It follows that the second conjunctive term of (22) can be dropped.

(b) Assume that $\exists x F(x)$ is safe. In $\mathbf{INT}^{=} + \mathrm{DE}$, this formula can be equivalently written as

$$\exists x((in_{\mathbf{c}}(x) \wedge F(x)) \vee (\neg in_{\mathbf{c}}(x) \wedge F(x))),$$

and consequently as

$$\bigvee_{c \in \mathbf{c}} F(c) \ \vee \ \exists x(\neg in_{\mathbf{c}}(x) \wedge F(x)). \tag{23}$$

Consider the maximal subformulas $G(x) \to H(x)$ of $F(x)$, negative in $F(x)$, such that x is restricted in $G(x)$. As before, the implications

$$\neg in_{\mathbf{c}}(x) \to (G(x) \to H(x))$$

are derivable from $SPP_{\mathbf{c}}$ in $\mathbf{INT}^{=}$. Consequently, under the assumption $SPP_{\mathbf{c}}$, (23) can be equivalently rewritten as

$$\bigvee_{c \in \mathbf{c}} F(c) \ \vee \ \exists x(\neg in_{\mathbf{c}}(x) \wedge S), \tag{24}$$

where S is the formula obtained from $F(x)$ by replacing each of these maximal subformulas $G(x) \to H(x)$ with $\top$. Since $\exists x F(x)$ is safe, x does not occur in S. It follows that S can be obtained from $F(c)$ in the same way as it was obtained from $F(x)$, that is, by replacing some subformulas that are negative in $F(c)$ with $\top$. Consequently, the formula $S \to F(c)$ is intuitionistically provable, and so is

$$\exists x(\neg in_{\mathbf{c}}(x) \wedge S) \to F(c).$$

It follows that the second disjunctive term of (24) can be dropped. □

Proof of Proposition 2. By induction on the length of the prefix. The base case is trivial. Assume that $QxF(x)$ is safe. Case 1: Q is $\forall$. In view of Lemma 5, from the induction hypothesis we can conclude that

$$\mathrm{Ground}_{\mathbf{c}}[F(c)] \leftrightarrow F(c)$$

is derivable from $SPP_{\mathbf{c}}$ in $\mathbf{INT}^{=} + \mathrm{DE}$ for every $c \in \mathbf{c}$. Consequently

$$\bigwedge_{c \in \mathbf{c}} \mathrm{Ground}_{\mathbf{c}}[F(c)] \leftrightarrow \bigwedge_{c \in \mathbf{c}} F(c)$$

is derivable from $SPP_{\mathbf{c}}$ as well. By the definition of $\mathrm{Ground}_{\mathbf{c}}$, the left-hand side is $\mathrm{Ground}_{\mathbf{c}}[\forall x F(x)]$. By Lemma 7(a), under the assumption $SPP_{\mathbf{c}}$ the right-hand side is equivalent in $\mathbf{INT}^{=} + \mathrm{DE}$ to $\forall x F(x)$. Case 2: Q is $\exists$. Similar, using Lemma 7(b). □

It is interesting that without the decidable equality axiom DE, the statement of Proposition 2 would be incorrect. The formula

$$\forall x(((x = a \vee x = b) \to p(x)) \vee ((x = a \vee x = b) \to q(x)))$$

can serve as a counterexample. Indeed, call this formula F, and assume that

$$\text{Ground}_{\{a,b\}}[F] \leftrightarrow F \tag{25}$$

can be derived from

$$SPP_{\{a,b\}} \tag{26}$$

in $\mathbf{INT}^=$. In this derivation, substitute $\lambda x(x = a)$ for p, and $\lambda x(x = b)$ for q. After this substitution, the right-hand side of (25) becomes

$$\forall x(((x = a \vee x = b) \to x = a) \vee ((x = a \vee x = b) \to x = b)), \tag{27}$$

the left-hand side becomes

$$\begin{array}{l}(((a = a \vee a = b) \to a = a) \vee ((a = a \vee a = b) \to a = b)) \\ \wedge(((b = a \vee b = b) \to b = a) \vee ((b = a \vee b = b) \to b = b)),\end{array} \tag{28}$$

and (26) becomes

$$\forall x((x = a \to (x = a \vee x = b)) \wedge (x = b \to (x = a \vee x = b))). \tag{29}$$

Since (28) and (29) can be proved in $\mathbf{INT}^=$, it follows that (27) is provable in this system also. According to the disjunction property of $\mathbf{INT}^=$, if a disjunction is provable in $\mathbf{INT}^=$ then at least one of its disjunctive terms is provable. Consequently, at least one of the formulas

$$(x = a \vee x = b) \to x = a,\ \ (x = a \vee x = b) \to x = b$$

is provable in $\mathbf{INT}^=$. But this is impossible, because these formulas are not even logically valid.

Unlike Proposition 1, Proposition 2 will not hold if we replace "safe" in its statement with "semi-safe." For instance, take F to be $\forall x \neg\neg p(x, a)$. The equivalence

$$\neg\neg p(a, a) \ \leftrightarrow\ \forall x \neg\neg p(x, a)$$

is not entailed by the small predicate property

$$\forall xy(p(x, y) \to (x = a \wedge y = a))$$

even classically. (Consider an interpretation with a non-singleton universe in which $p(x, y)$ is defined as $x = a \wedge y = a$.)

Proposition 3 *For any safe sentence F and any nonempty finite set* $\mathbf{c}$ *of object constants containing $c(F)$,* $\text{SM}[\text{Ground}_{\mathbf{c}}[F]]$ *is equivalent to* $\text{SM}[F]$.

In the proof we use the following terminology, which generalizes the concept of a negative literal. A formula F is *negative* if every occurrence of every predicate constant in F belongs to the antecedent of an implication. For any sentence F and any negative sentence G, $\text{SM}[F \wedge G]$ is equivalent to $\text{SM}[F] \wedge G$ [11, Proposition 2].

Proof of Proposition 3. By Proposition 2 proved above, the equivalence

$$\text{Ground}_{\mathbf{c}}[F] \wedge SPP_{\mathbf{c}} \leftrightarrow F \wedge SPP_{\mathbf{c}}$$

is provable in $\mathbf{INT}^= + \text{DE}$. Consequently

$$\text{SM}[\text{Ground}_{\mathbf{c}}[F] \wedge \mathit{SPP}_{\mathbf{c}}] \text{ is equivalent to } \text{SM}[F \wedge \mathit{SPP}_{\mathbf{c}}].$$

Since $\mathit{SPP}_{\mathbf{c}}$ is negative, it follows that

$$\text{SM}[\text{Ground}_{\mathbf{c}}[F]] \wedge \mathit{SPP}_{\mathbf{c}} \text{ is equivalent to } \text{SM}[F] \wedge \mathit{SPP}_{\mathbf{c}}.$$

In view of Proposition 1 and the fact that $c(F) \subseteq \mathbf{c}$, the conjunctive term SM$[F]$ in the second conjunction entails its other conjunctive term $\mathit{SPP}_{\mathbf{c}}$, and the latter can be dropped. Furthermore, $\text{Ground}_{\mathbf{c}}[F]$ is variable-free and consequently safe. It follows by similar reasoning that in the first conjunction the term $\mathit{SPP}_{\mathbf{c}}$ can be dropped also. □

6 Characterizing Stable Models of a Safe Sentence

Proposition 4 *For every safe sentence F there exists a variable-free formula G such that* SM$[F]$ *is equivalent to $G \wedge \mathit{SPP}_{c(F)}$.*

Proof. In view of Proposition 1, we need to find a variable-free formula G such that $\mathit{SPP}_{c(F)}$ entails SM$[F] \leftrightarrow G$.

Case 1: $c(F) = \emptyset$. Under the assumption $\mathit{SPP}_{\emptyset}$, every atomic part of SM$[F]$ that contains a predicate constant or variable of arity > 0 can be equivalently replaced by $\bot$. The result is a second-order propositional formula, so that it is equivalent to a propositional formula.

Case 2: $c(F) \neq \emptyset$ and F is variable-free. The only quantifiers in (9) are the second-order quantifiers $\exists \mathbf{u}$. Clearly $\mathit{SPP}_{c(F)}$ entails

$$u_i \leq p_i \;\rightarrow\; u_i \leq \lambda \mathbf{x} \left(\bigvee_{\mathbf{c}} \mathbf{x} = \mathbf{c} \right)$$

where $\mathbf{c}$ ranges over the tuples of members of $c(F)$ of the same length as $\mathbf{x}$. Consequently it entails also

$$\mathbf{u} < \mathbf{p} \;\rightarrow\; u_i \leq \lambda \mathbf{x} \left(\bigvee_{\mathbf{c}} \mathbf{x} = \mathbf{c} \right)$$

and

$$\mathbf{u} < \mathbf{p} \;\rightarrow\; \bigvee_{C} \left(u_i = \lambda \mathbf{x} \bigvee_{\mathbf{c} \in C} \mathbf{x} = \mathbf{c} \right),$$

where C ranges over all sets of such tuples. It follows that under the assumption $\mathit{SPP}_{c(F)}$ the quantifiers $\exists \mathbf{u}$ can be equivalently replaced by finite disjunctions, with expressions of the form $\lambda \mathbf{x} \bigvee_{\mathbf{c} \in C} \mathbf{x} = \mathbf{c}$ substituted for the variables u_i. The result is a variable-free formula with the required properties.

Case 3: $c(F) \neq \emptyset$ and F is not variable-free. The part of Proposition 4 corresponding to Case 2 can be applied to $\text{Ground}_{c(F)}[F]$. Since the formulas F

and $\text{Ground}_{c(F)}[F]$ contain the same object constants, we can assert that, for some variable-free formula G, $SPP_{c(F)}$ entails

$$\text{SM}[\text{Ground}_{c(F)}[F]] \leftrightarrow G.$$

It remains to observe that, by Proposition 3, the left-hand side is equivalent to $\text{SM}[F]$. □

7 Extending a Stable Model

Let I be an interpretation of a set of object and predicate constants, and let X be a superset of the universe of I. By the *extension of I to X* we mean the interpretation of the same constants with the universe X such that each object constant represents the same object under both interpretations, and each predicate constant represents the same set of tuples.

Proposition 5 *For any safe sentence F, any interpretation I of the object and predicate constants from F, and any superset X of the universe of I, the extension of I to X is a stable model of F iff I is a stable model of F.*

Proof. Consider a variable-free formula G such that $\text{SM}[F]$ is equivalent to $G \wedge SPP_{c(F)}$ (Proposition 4). It is clear that I satisfies G iff the extension of I to X satisfies G, and that I satisfies $SPP_{c(F)}$ iff the extension of I to X satisfies $SPP_{c(F)}$. □

In the special case when I is an Herbrand interpretation, this theorem turns into Proposition 1 from [5].

8 Conclusion

The approach to stable models developed in [1] is richer than the traditional view not only syntactically, but also semantically: stable models became now models in the sense of classical logic, not merely sets of ground atoms. But the only models referred to in the definition of RASPL-1 are Herbrand models—sets of ground atoms. That definition exploits the syntactic generality of the new theory of stable models, but not its semantic generality.

We expect, however, that future work on applications of stable models to knowledge representation will demonstrate the usefulness of non-Herbrand stable models. Such models allow us to talk about elements of the universe that are "unnamed," that is, not represented by ground terms. They also allow us to talk about elements of the universe that may have "multiple names" in the language. These additional possibilities may be certainly useful.

In this paper we investigated properties of stable models of safe formulas in a semantically general situation, not limited to Herbrand models, and established a few positive results. We saw, in particular, that grounding a safe sentence preserves its stable models even in this general case. We hope that these theorems will help us in future work on non-Herbrand answer set programming.

Acknowledgements

We are grateful to Paolo Ferraris and anonymous referees for ICLP 2008 for their useful comments on an earlier version of this paper. The first and the third author were partially supported by the National Science Foundation under Grant IIS-0839821. The second author was partially supported by the National Science Foundation under Grant IIS-0712113.

References

1. Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2007) 372–379
2. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R., Bowen, K., eds.: Proceedings of International Logic Programming Conference and Symposium, MIT Press (1988) 1070–1080
3. McCarthy, J.: Circumscription—a form of non-monotonic reasoning. Artificial Intelligence **13** (1980) 27–39,171–172
4. Lifschitz, V., Pearce, D., Valverde, A.: A characterization of strong equivalence for logic programs with variables. In: Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2007)
5. Lee, J., Lifschitz, V., Palla, R.: A reductive semantics for counting and choice in answer set programming. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). (2008) 472–479
6. Leone, N., Faber, W., Pfeifer, G., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic **7** (2006) 499–562
7. Lee, J., Lifschitz, V., Palla, R.: Safe formulas in the general theory of stable models (preliminary report). In: Proceedings of International Conference on Logic Programming (ICLP). (2008) 672–676
8. Lee, J., Palla, R.: Yet another proof of the strong equivalence between propositional theories and logic programs. In: Working Notes of the Workshop on Correspondence and Equivalence for Nonmonotonic Theories. (2007)
9. Topor, R.W., Sonenberg, E.A.: On domain independent databases. In Minker, J., ed.: Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann, San Mateo, CA (1988) 217–240
10. Lifschitz, V.: Computing circumscription. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (1985) 121–127
11. Ferraris, P., Lee, J., Lifschitz, V.: Stable models and circumscription.[4] Artificial Intelligence (2009) To appear.

[4] `http://www.cs.utexas.edu/users/vl/papers/smcirc.pdf` .