Two New Definitions of Stable Models of Logic Programs with Generalized Quantifiers

Joohyung Lee and Yunsong Meng

School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, USA

Abstract. We present alternative definitions of the first-order stable model semantics and its extension to incorporate generalized quantifiers by referring to the familiar notion of a reduct instead of referring to the SM operator in the original definitions. Also, we extend the FLP stable model semantics to allow generalized quantifiers by referring to an operator that is similar to the SM operator. For a reasonable syntactic class of logic programs, we show that the two stable model semantics of generalized quantifiers are interchangeable.

1 Introduction

Most versions of the stable model semantics involve grounding. For instance, according to the FLP semantics from [1; 2], assuming that the domain is $\{-1, 1, 2\}$, program

$$\begin{array}{rcl} p(2) & \leftarrow & \textit{not}\ \text{SUM}\langle x\!:\!p(x)\rangle < 2 \\ p(-1) & \leftarrow & \text{SUM}\langle x\!:\!p(x)\rangle > -1 \\ p(1) & \leftarrow & p(-1) \end{array} \tag{1}$$

is identified with its ground instance w.r.t the domain:

$$\begin{array}{rcl} p(2) & \leftarrow & \textit{not}\ \text{SUM}\langle\{-1\!:\!p(-1), 1\!:\!p(1), 2\!:\!p(2)\}\rangle < 2 \\ p(-1) & \leftarrow & \text{SUM}\langle\{-1\!:\!p(-1), 1\!:\!p(1), 2\!:\!p(2)\}\rangle > -1 \\ p(1) & \leftarrow & p(-1)\ . \end{array} \tag{2}$$

As described in [1], it is straightforward to extend the definition of satisfaction to ground aggregate expressions. For instance, set $\{p(-1), p(1)\}$ does not satisfy the body of the first rule of (2), but satisfies the bodies of the other rules. The FLP reduct of program (2) relative to $\{p(-1), p(1)\}$ consists of the last two rules, and $\{p(-1), p(1)\}$ is its minimal model. Indeed, $\{p(-1), p(1)\}$ is the only FLP answer set of program (2).

On the other hand, according to the semantics from [3], program (2) is identified with some complex propositional formula containing nested implications:

$$\begin{array}{l} \Big(\neg\big((p(2)\rightarrow p(-1)\vee p(1)) \wedge (p(1)\wedge p(2)\rightarrow p(-1)) \wedge (p(-1)\wedge p(1)\wedge p(2)\rightarrow\bot)\big) \rightarrow p(2)\Big) \\ \wedge \Big(\big(p(-1)\rightarrow p(1)\vee p(2)\big) \rightarrow p(-1)\Big) \\ \wedge \Big(p(-1) \rightarrow p(1)\Big)\ . \end{array}$$

Under the stable model semantics of propositional formulas [3], this formula has two answer sets: $\{p(-1), p(1)\}$ and $\{p(-1), p(1), p(2)\}$. The relationship between the FLP and the Ferraris semantics was studied in [4; 5].

Unlike the FLP semantics, the definition from [3] is not applicable when the domain is infinite because it would require the representation of an aggregate expression to involve "infinite" conjunctions and disjunctions. This limitation was overcome in the semantics presented in [4; 6], which extends the first-order stable model semantics from [7; 8] to incorporate aggregate expressions. Recently, it was further extended to formulas involving generalized quantifiers [9], which provides a unifying framework of various extensions of the stable model semantics, including programs with aggregates, programs with abstract constraint atoms [10], and programs with nonmonotonic dl-atoms [11].

In this paper, we revisit the first-order stable model semantics and its extension to incorporate generalized quantifiers. We provide an alternative, equivalent definition of a stable model by referring to grounding and reduct instead of the SM operator. Our work is inspired by the work of Truszczynski [12], who introduces infinite conjunctions and disjunctions to account for grounding quantified sentences. Our definition of a stable model can be viewed as a reformulation and a further generalization of his definition to incorporate generalized quantifiers. We define grounding in the same way as done in the FLP semantics, but define a reduct differently so that the semantics agrees with the one by Ferraris [3]. As we explain in Section 3.3, our reduct of program (2) relative to $\{p(-1), p(1)\}$ is

$$\begin{array}{rcl} \bot & \leftarrow & \bot \\ p(-1) & \leftarrow & \text{SUM}\langle\{-1\!:\!p(-1), 1\!:\!p(1), 2\!:\!\bot\rangle\}\!>\!-1 \\ p(1) & \leftarrow & p(-1)\ , \end{array} \tag{3}$$

which is the program obtained from (2) by replacing each maximal subformula that is not satisfied by $\{p(-1), p(1)\}$ with $\bot$. Set $\{p(-1), p(1)\}$ is an answer set of program (1) as it is a minimal model of the reduct. Likewise the reduct relative to $\{p(-1), p(1), p(2)\}$ is

$$\begin{array}{rcl} p(2) & \leftarrow & \top \\ p(-1) & \leftarrow & \text{SUM}\{\langle -1\!:\!p(-1), 1\!:\!p(1), 2\!:\!p(2)\rangle\}\!>\!-1 \\ p(1) & \leftarrow & p(-1) \end{array}$$

and $\{p(-1), p(1), p(2)\}$ is a minimal model of the program. The semantics is more direct than the one from [3] as it does not involve the complex translation into a propositional formula.

While the FLP semantics in [1] was defined in the context of logic programs with aggregates, it can be straightforwardly extended to allow other "complex atoms." Indeed, the FLP reduct is the basis of the semantics of HEX programs [13]. In [14], the FLP reduct was applied to provide a semantics of nonmonotonic dl-programs [11]. In [5], the FLP semantics of logic programs with aggregates was generalized to the first-order level. That semantics is defined in terms of the FLP operator, which is similar to the SM operator. This paper further extends the definition to allow generalized quantifiers.

By providing an alternative definition in the way that the other semantics was defined, this paper provides a useful insight into the relationship between the first-order stable model semantics and the FLP stable model semantics for programs with generalized quantifiers. While the two semantics behave differently in the general case, we show that they coincide on some reasonable syntactic class of logic programs. This implies that an implementation of one of the semantics can be viewed as an implementation of the other semantics if we limit attention to that class of logic programs.

The paper is organized as follows. Section 2 reviews the first-order stable model semantics and its equivalent definition in terms of grounding and reduct, and Section 3 extends that definition to incorporate generalized quantifiers. Section 4 provides an alternative definition of the FLP semantics with generalized quantifiers via a translation into second-order formulas. Section 5 compares the FLP semantics and the first-order stable model semantics in the general context of programs with generalized quantifiers.

2 First-Order Stable Model Semantics

2.1 Review of First-Order Stable Model Semantics

This review follows [8], a journal version of [7], which distinguishes between intensional and non-intensional predicates.

A *formula* is defined the same as in first-order logic. A *signature* consists of *function constants* and *predicate constants*. Function constants of arity 0 are also called *object constants*. We assume the following set of primitive propositional connectives and quantifiers:

$$\bot, \top, \wedge, \vee, \rightarrow, \forall, \exists\,.$$

$\neg F$ is an abbreviation of $F \rightarrow \bot$, and $F \leftrightarrow G$ stands for $(F \rightarrow G) \wedge (G \rightarrow F)$. We distinguish between atoms and atomic formulas as follows: an *atom* of a signature σ is an n-ary predicate constant followed by a list of n terms that can be formed from function constants in σ and object variables; *atomic formulas* of σ are atoms of σ, equalities between terms of σ, and the 0-place connectives $\bot$ and $\top$.

The stable models of F relative to a list of predicates $\mathbf{p} = (p_1, \dots, p_n)$ are defined via the *stable model operator with the intensional predicates* $\mathbf{p}$, denoted by SM$[F; \mathbf{p}]$.[1] Let $\mathbf{u}$ be a list of distinct predicate variables $u_1, \dots, u_n$. 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 length as the arity of p_i, for all $i = 1, \dots, n$. By $\mathbf{u} \leq \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))$ 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 any first-order sentence F, expression SM$[F; \mathbf{p}]$ stands for the second-order sentence

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

where $F^*(\mathbf{u})$ is defined recursively:

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ for any list $\mathbf{t}$ of terms;
- $F^* = F$ for any atomic formula F that does not contain members of $\mathbf{p}$;
- $(F \wedge G)^* = F^* \wedge G^*$;

[1] The intensional predicates $\mathbf{p}$ are the predicates that we "intend to characterize" by F.

– $(F \vee G)^* = F^* \vee G^*$;
– $(F \rightarrow G)^* = (F^* \rightarrow G^*) \wedge (F \rightarrow G)$;
– $(\forall x F)^* = \forall x F^*$;
– $(\exists x F)^* = \exists x F^*$.

A model of a sentence F (in the sense of first-order logic) is called $\mathbf{p}$-*stable* if it satisfies $\text{SM}[F; \mathbf{p}]$.

Example 1 *Let F be sentence $\forall x(\neg p(x) \rightarrow q(x))$, and let I be an interpretation whose universe is the set of all nonnegative integers $\mathbf{N}$, and $p^I(n) = \text{FALSE}$, $q^I(n) = \text{TRUE}$ for all $n \in \mathbf{N}$. Section 2.4 of [8] tells us that I satisfies $\text{SM}[F; pq]$.*

2.2 Alternative Definition of First-Order Stable Models via Reduct

For any signature σ and its interpretation I, by σ^I we mean the signature obtained from σ by adding new object constants $\xi^\diamond$, called *object names*, for every element ξ in the universe of I. We identify an interpretation I of σ with its extension to σ^I defined by $I(\xi^\diamond) = \xi$.

In order to facilitate defining a reduct, we provide a reformulation of the standard semantics of first-order logic via "a ground formula w.r.t. an interpretation."

Definition 1. *For any interpretation I of a signature σ, a* ground formula w.r.t. *I is defined recursively as follows.*

– *$p(\xi_1^\diamond, \dots, \xi_n^\diamond)$, where p is a predicate constant of σ and $\xi_i^\diamond$ are object names of σ^I, is a ground formula w.r.t. I;*
– *$\top$ and $\bot$ are ground formulas w.r.t. I;*
– *If F and G are ground formulas w.r.t. I, then $F \wedge G$, $F \vee G$, $F \rightarrow G$ are ground formulas w.r.t. I;*
– *If S is a set of pairs of the form $\xi^\diamond\!: F$ where $\xi^\diamond$ is an object name in σ^I and F is a ground formula w.r.t. I, then $\forall(S)$ and $\exists(S)$ are ground formulas w.r.t. I.*

The following definition describes a process that turns any first-order sentence into a ground formula w.r.t. an interpretation:

Definition 2. *Let F be any first-order sentence of a signature σ, and let I be an interpretation of σ whose universe is U. By $gr_I[F]$ we denote the ground formula w.r.t. I, which is obtained by the following process:*

– $gr_I[p(t_1, \dots, t_n)] = p((t_1^I)^\diamond, \dots, (t_n^I)^\diamond)$;
– $gr_I[t_1 = t_2] = \begin{cases} \top & \text{if } t_1^I = t_2^I \text{, and} \\ \bot & \text{otherwise;} \end{cases}$
– $gr_I[\top] = \top$; $\quad gr_I[\bot] = \bot$;
– $gr_I[F \odot G] = gr_I[F] \odot gr_I[G] \quad (\odot \in \{\wedge, \vee, \rightarrow\})$;
– $gr_I[QxF(x)] = Q(\{\xi^\diamond\!: gr_I[F(\xi^\diamond)] \mid \xi \in U\}) \quad (Q \in \{\forall, \exists\})$.

Definition 3. *For any interpretation I and any ground formula F w.r.t. I, the truth value of F under I, denoted by F^I, is defined recursively as follows.*

- $p(\xi_1^\diamond, \ldots, \xi_n^\diamond)^I = p^I(\xi_1, \ldots, \xi_n)$;
- $\top^I = \text{TRUE}$; $\bot^I = \text{FALSE}$;
- $(F \wedge G)^I = \text{TRUE}$ *iff* $F^I = \text{TRUE}$ *and* $G^I = \text{TRUE}$;
- $(F \vee G)^I = \text{TRUE}$ *iff* $F^I = \text{TRUE}$ *or* $G^I = \text{TRUE}$;
- $(F \rightarrow G)^I = \text{TRUE}$ *iff* $G^I = \text{TRUE}$ *whenever* $F^I = \text{TRUE}$;
- $\forall(S)^I = \text{TRUE}$ *iff the set* $\{\xi \mid \xi^\diamond{:}\, F(\xi^\diamond) \in S$ *and* $F(\xi^\diamond)^I = \text{TRUE}\}$ *is the same as the universe of* I;
- $\exists(S)^I = \text{TRUE}$ *iff the set* $\{\xi \mid \xi^\diamond{:}\, F(\xi^\diamond) \in S$ *and* $F(\xi^\diamond)^I = \text{TRUE}\}$ *is not empty.*

We say that I satisfies F, *denoted* $I \models F$, *if* $F^I = \text{TRUE}$.

Example 1 continued (I). $gr_I[F]$ is $\forall(\{n^\diamond{:}\, (\neg p(n^\diamond) \rightarrow q(n^\diamond)) \mid n \in \mathbf{N}\})$. Clearly, I satisfies $gr_I[F]$.

An interpretation I of a signature σ can be represented as a pair $\langle I^{func}, I^{pred}\rangle$, where I^{func} is the restriction of I to the function constants of σ, and I^{pred} is the set of atoms, formed using predicate constants from σ and the object names from σ^I, which are satisfied by I. For example, interpretation I in Example 1 can be represented as $\langle I^{func},\ \{q(n^\diamond) \mid n \in \mathbf{N}\}\rangle$, where I^{func} maps each integer to itself.

The following proposition is immediate from the definitions:

Proposition 1. *Let σ be a signature that contains finitely many predicate constants, let σ^{pred} be the set of predicate constants in σ, let $I = \langle I^{func}, I^{pred}\rangle$ be an interpretation of σ, and let F be a first-order sentence of σ. Then $I \models F$ iff $I^{pred} \models gr_I[F]$.*

The introduction of the intermediate form of a ground formula w.r.t. an interpretation helps us define a reduct.

Definition 4. *For any ground formula F w.r.t. I, the* reduct *of F relative to I, denoted by $F^{\underline{I}}$, is obtained by replacing each maximal subformula that is not satisfied by I with $\bot$. It can also be defined recursively as follows.*

- $(p(\xi_1^\diamond, \ldots, \xi_n^\diamond))^{\underline{I}} = \begin{cases} p(\xi_1^\diamond, \ldots, \xi_n^\diamond) & \textit{if } I \models p(\xi_1^\diamond, \ldots, \xi_n^\diamond), \\ \bot & \textit{otherwise;} \end{cases}$
- $\top^{\underline{I}} = \top$; $\bot^{\underline{I}} = \bot$;
- $(F \odot G)^{\underline{I}} = \begin{cases} F^{\underline{I}} \odot G^{\underline{I}} & \textit{if } I \models F \odot G \quad (\odot \in \{\wedge, \vee, \rightarrow\}), \\ \bot & \textit{otherwise;} \end{cases}$
- $Q(S)^{\underline{I}} = \begin{cases} Q(\{\xi^\diamond{:}\, (F(\xi^\diamond))^{\underline{I}} \mid \xi^\diamond{:}\, F(\xi^\diamond) \in S\}) & \textit{if } I \models Q(S) \quad (Q \in \{\forall, \exists\}), \\ \bot & \textit{otherwise.} \end{cases}$

The following theorem tells us how first-order stable models can be characterized in terms of grounding and reduct.

Theorem 1. *Let σ be a signature that contains finitely many predicate constants, let σ^{pred} be the set of predicate constants in σ, let $I = \langle I^{func}, I^{pred}\rangle$ be an interpretation of σ, and let F be a first-order sentence of σ. I satisfies* $\mathrm{SM}[F; \sigma^{pred}]$ *iff I^{pred} is a minimal set of atoms that satisfies $(gr_I[F])^{\underline{I}}$.*

Example 1 continued (II). *The reduct of $gr_I[F]$ relative to I, $(gr_I[F])^{\underline{I}}$, is* $\forall(\{n^\diamond\colon(\neg\bot \rightarrow q(n^\diamond)) \mid n \in \mathbf{N}\})$, *which is equivalent to* $\forall(\{n^\diamond\colon q(n^\diamond) \mid n \in \mathbf{N}\})$. *Clearly, $I^{pred} = \{q(n^\diamond) \mid n \in \mathbf{N}\}$ is a minimal set of atoms that satisfies $(gr_I[F])^{\underline{I}}$.*

2.3 Relation to Infinitary Formulas by Truszczynski

The definitions of grounding and a reduct in the previous section are inspired by the work of Truszczynski [12], where he introduces infinite conjunctions and disjunctions to account for the result of grounding $\forall$ and $\exists$ w.r.t. a given interpretation. Differences between the two approaches are illustrated in the following example:

Example 2 *Consider the formula $F = \forall x\, p(x)$ and the interpretation I whose universe is the set of all nonnegative integers $\mathbf{N}$. According to [12], grounding of F w.r.t. I results in the infinitary propositional formula*

$$\{p(n^\diamond) \mid n \in N\}^\wedge\ .$$

On the other hand, formula $gr_I[F]$ is

$$\forall(\{n^\diamond{:}p(n^\diamond) \mid n \in N\}).$$

Our definition of a reduct is essentially equivalent to the one defined in [12]. In the next section, we extend our definition to incorporate generalized quantifiers.

3 Stable Models of Formulas with Generalized Quantifiers

3.1 Review: Formulas with Generalized Quantifiers

We follow the definition of a formula with generalized quantifiers from [15, Section 5] (that is to say, with Lindström quantifiers [16] without the isomorphism closure condition).

We assume a set $\mathbf{Q}$ of symbols for generalized quantifiers. Each symbol in $\mathbf{Q}$ is associated with a tuple of nonnegative integers $\langle n_1, \dots, n_k\rangle$ ($k \geq 0$, and each n_i is ≥ 0), called the *type*. A *(GQ-)formula (with the set $\mathbf{Q}$ of generalized quantifiers)* is defined in a recursive way:

- an atomic formula (in the sense of first-order logic) is a GQ-formula;
- if $F_1, \dots, F_k$ ($k \geq 0$) are GQ-formulas and Q is a generalized quantifier of type $\langle n_1, \dots, n_k\rangle$ in $\mathbf{Q}$, then

$$Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1), \dots, F_k(\mathbf{x}_k)) \tag{4}$$

  is a GQ-formula, where each $\mathbf{x}_i$ ($1 \leq i \leq k$) is a list of distinct object variables whose length is n_i.

We say that an occurrence of a variable x in a GQ-formula F is *bound* if it belongs to a subformula of F that has the form $Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1),\dots,F_k(\mathbf{x}_k))$ such that x is in some $\mathbf{x}_i$. Otherwise the occurrence is *free*. We say that x is *free* in F if F contains a free occurrence of x. A *(GQ-)sentence* is a GQ-formula with no free variables.

We assume that $\mathbf{Q}$ contains type $\langle\rangle$ quantifiers $Q_\bot$ and $Q_\top$, type $\langle 0,0\rangle$ quantifiers $Q_\land, Q_\lor, Q_\to$, and type $\langle 1\rangle$ quantifiers $Q_\forall, Q_\exists$. Each of them corresponds to the standard logical connectives and quantifiers — $\bot,\top,\land,\lor,\to,\forall,\exists$. These generalized quantifiers will often be written in the familiar form. For example, we write $F\land G$ in place of $Q_\land[][](F,G)$, and write $\forall x F(x)$ in place of $Q_\forall[x](F(x))$.

As in first-order logic, an interpretation I consists of the universe U and the evaluation of predicate constants and function constants. For each generalized quantifier Q of type $\langle n_1,\dots,n_k\rangle$, Q^U is a function from $\mathcal{P}(U^{n_1})\times\cdots\times\mathcal{P}(U^{n_k})$ to $\{\text{TRUE},\text{FALSE}\}$, where $\mathcal{P}(U^{n_i})$ denotes the power set of U^{n_i}.

Example 3 *Besides the standard connectives and quantifiers, the following are some examples of generalized quantifiers.*

- *type* $\langle 1\rangle$ *quantifier* $Q_{\le 2}$ *such that* $Q^U_{\le 2}(R)=\text{TRUE}$ *iff* $|R|\le 2$*;* [2]
- *type* $\langle 1\rangle$ *quantifier* $Q_{majority}$ *such that* $Q^U_{majority}(R)=\text{TRUE}$ *iff* $|R|>|U\setminus R|$*;*
- *type* $\langle 1,1\rangle$ *quantifier* $Q_{(\text{SUM},<)}$ *such that* $Q^U_{(\text{SUM},<)}(R_1,R_2)=\text{TRUE}$ *iff*
  - $\text{SUM}(R_1)$ *is defined,*
  - $R_2=\{b\}$*, where* b *is an integer, and*
  - $\text{SUM}(R_1)<b$*.*

Given a sentence F of σ^I, F^I is defined recursively as follows:

- $p(t_1,\dots,t_n)^I = p^I(t_1^I,\dots,t_n^I)$,
- $(t_1=t_2)^I=(t_1^I=t_2^I)$,
- For a generalized quantifier Q of type $\langle n_1,\dots,n_k\rangle$,

$$(Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1),\dots,F_k(\mathbf{x}_k)))^I = Q^U((\mathbf{x}_1\!:\!F_1(\mathbf{x}_1))^I,\dots,(\mathbf{x}_k\!:\!F_k(\mathbf{x}_k))^I),$$

  where $(\mathbf{x}_i\!:\!F_i(\mathbf{x}_i))^I=\{\boldsymbol{\xi}\in U^{n_i}\mid (F_i(\boldsymbol{\xi}^\diamond))^I=\text{TRUE}\}$.

We assume that, for the standard logical connectives and quantifiers Q, functions Q^U have the standard meaning:

- $Q^U_\forall(R)=\text{TRUE}$ iff $R=U$;
- $Q^U_\exists(R)=\text{TRUE}$ iff $R\cap U\ne\emptyset$;
- $Q^U_\land(R_1,R_2)=\text{TRUE}$ iff $R_1=R_2=\{\epsilon\}$;[3]
- $Q^U_\lor(R_1,R_2)=\text{TRUE}$ iff $R_1=\{\epsilon\}$ or $R_2=\{\epsilon\}$;
- $Q^U_\to(R_1,R_2)=\text{TRUE}$ iff R_1 is $\emptyset$ or R_2 is $\{\epsilon\}$;
- $Q^U_\bot()=\text{FALSE}$;
- $Q^U_\top()=\text{TRUE}$.

---

[2] It is clear from the type of the quantifier that R is any subset of U. We will skip such explanation.

[3] ϵ denotes the empty tuple. For any interpretation I, $U^0=\{\epsilon\}$. For I to satisfy $Q_\land[][](F,G)$, both $(\epsilon\!:\!F)^I$ and $(\epsilon\!:\!G)^I$ have to be $\{\epsilon\}$, which means that $F^I=G^I=\text{TRUE}$.

We say that an interpretation I *satisfies* a GQ-sentence F, or is a *model* of F, and write $I \models F$, if $F^I = \text{TRUE}$. A GQ-sentence F is *logically valid* if every interpretation satisfies F. A GQ-formula with free variables is said to be *logically valid* if its universal closure is logically valid.

Example 4 *Program (1) in the introduction is identified with the following GQ-formula F_1:*

$$\begin{array}{l}(\neg Q_{(\text{SUM},<)}[x][y](p(x),\ y=2) \rightarrow p(2))\\ \wedge\ (Q_{(\text{SUM},>)}[x][y](p(x),\ y=-1) \rightarrow p(-1))\\ \wedge\ (p(-1) \rightarrow p(1))\ .\end{array}$$

Consider two Herbrand interpretations of the universe $U = \{-1, 1, 2\}$: $I_1 = \{p(-1), p(1)\}$ and $I_2 = \{p(-1), p(1), p(2)\}$. We have $(Q_{(\text{SUM},<)}[x][y](p(x),\ y = 2))^{I_1} = \text{TRUE}$ since

- $(x : p(x))^{I_1} = \{-1, 1\}$ *and* $(y : y=2)^{I_1} = \{2\}$*;*
- $Q^U_{(\text{SUM},<)}(\{-1, 1\}, \{2\}) = \text{TRUE}$.

Similarly, $(Q_{(\text{SUM},>)}[x][y](p(x),\ y=-1))^{I_2} = \text{TRUE}$ *since*

- $(x : p(x))^{I_2} = \{-1, 1, 2\}$ *and* $(y : y=-1)^{I_2} = \{-1\}$*;*
- $Q^U_{(\text{SUM},>)}(\{-1, 1, 2\}, \{-1\}) = \text{TRUE}$.

Consequently, both I_1 and I_2 satisfy F_1.

3.2 Review: SM-Based Definition of Stable Models of GQ-Formulas

For any GQ-formula F and any list of predicates $\mathbf{p} = (p_1, \dots, p_n)$, formula $\text{SM}[F; \mathbf{p}]$ is defined as

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

where $F^*(\mathbf{u})$ is defined recursively:

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ for any list $\mathbf{t}$ of terms;
- $F^* = F$ for any atomic formula F that does not contain members of $\mathbf{p}$;
- $$\begin{array}{l}(Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1), \dots, F_k(\mathbf{x}_k)))^* =\\ \qquad Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1^*(\mathbf{x}_1), \dots, F_k^*(\mathbf{x}_k)) \wedge\ Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1), \dots, F_k(\mathbf{x}_k)).\end{array}$$

When F is a sentence, the models of $\text{SM}[F; \mathbf{p}]$ are called the $\mathbf{p}$-*stable* models of F: they are the models of F that are "stable" on $\mathbf{p}$. We often simply write $\text{SM}[F]$ in place of $\text{SM}[F; \mathbf{p}]$ when $\mathbf{p}$ is the list of all predicate constants occurring in F, and call $\mathbf{p}$-stable models simply stable models.

As explained in [17], this definition of a stable model is a proper generalization of the first-order stable model semantics.

Example 4 continued (I). For GQ-sentence F_1 considered earlier, $\text{SM}[F_1]$ is

$$F_1 \wedge \neg\exists u(u < p \wedge F_1^*(u))\ , \tag{5}$$

where $F_1^*(u)$ is equivalent to the conjunction of F_1 and

$$
\begin{aligned}
&(\neg Q_{(\text{SUM},<)}[x][y](p(x), y\!=\!2) \rightarrow u(2)) \\
&\wedge ((Q_{(\text{SUM},>)}[x][y](u(x), y\!=\!-1) \wedge\ Q_{(\text{SUM},>)}[x][y](p(x), y\!=\!-1)) \rightarrow u(-1)) \\
&\wedge (u(-1) \rightarrow u(1))\ .
\end{aligned}
$$

The equivalence can be explained by Proposition 1 from [9], which simplifies the transformation for monotone and antimonotone GQs. I_1 and I_2 considered earlier satisfy (5) and thus are stable models of F_1.

3.3 Reduct-Based Definition of Stable Models of GQ-Formulas

The reduct-based definition of stable models presented in Section 2.2 can be extended to GQ-formulas as follows.

Let I be an interpretation of a signature σ. As before, we assume a set $\mathbf{Q}$ of generalized quantifiers, which contains all propositional connectives and standard quantifiers.

Definition 5. *A ground GQ-formula w.r.t. I is defined recursively as follows:*

- $p(\xi_1^\diamond, \dots, \xi_n^\diamond)$*, where p is a predicate constant of σ and $\xi_i^\diamond$ are object names of σ^I, is a ground GQ-formula w.r.t. I;*
- *for any $Q \in \mathbf{Q}$ of type $\langle n_1, \dots, n_k\rangle$, if each S_i is a set of pairs of the form $\boldsymbol{\xi}^\diamond\!: F$ where $\boldsymbol{\xi}^\diamond$ is a list of object names from σ^I whose length is n_i and F is a ground GQ-formula w.r.t. I, then*

  $$Q(S_1, \dots, S_k)$$

  is a ground GQ-formula w.r.t. I.

The following definition of grounding turns any GQ-sentence into a ground GQ-formula w.r.t. an interpretation:

Definition 6. *Let F be a GQ-sentence of a signature σ, and let I be an interpretation of σ. By $gr_I[F]$ we denote the ground GQ-formula w.r.t. I that is obtained by the process similar to the one in Definition 2 except that the last two clauses are replaced by the following single clause:*

- $gr_I[Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1), \dots, F_k(\mathbf{x}_k))] = Q(S_1, \dots, S_k)$
  where $S_i = \{\boldsymbol{\xi}^\diamond\!: gr_I[F_i(\boldsymbol{\xi}^\diamond)] \mid \boldsymbol{\xi}^\diamond$ is a list of object names from σ^I whose length is $n_i\}$.

For any interpretation I and any ground GQ-formula F w.r.t. I, the satisfaction relation $I \models F$ is defined recursively as follows.

Definition 7. *For any interpretation I and any ground GQ-formula F w.r.t. I, the satisfaction relation $I \models F$ is defined similar to Definition 3 except that the last five clauses are replaced by the following single clause:*

- $Q(S_1, \dots, S_k)^I = Q^U(S_1^I, \dots, S_k^I)$ *where* $S_i^I = \{\boldsymbol{\xi} \mid \boldsymbol{\xi}^\diamond\!: F(\boldsymbol{\xi}^\diamond) \in S_i,\ F(\boldsymbol{\xi}^\diamond)^I = \text{TRUE}\}$.

Example 4 continued (II). For Herbrand interpretation $I_1 = \{p(-1), p(1)\}$, formula $gr_{I_1}[F_1]$ is [4]

$$
\begin{array}{l}
(\neg Q_{(\text{SUM},<)}(\{-1\!:\!p(-1), 1\!:\!p(1), 2\!:\!p(2)\}, \{-1\!:\!\bot, 1\!:\!\bot, 2\!:\!\top\}) \rightarrow p(2)) \\
\wedge\, (Q_{(\text{SUM},>)}(\{-1\!:\!p(-1), 1\!:\!p(1), 2\!:\!p(2)\}, \{-1\!:\!\top, 1\!:\!\bot, 2\!:\!\bot\}) \rightarrow p(-1)) \\
\wedge\, (p(-1) \rightarrow p(1))\ .
\end{array} \tag{6}
$$

I_1 satisfies $Q_{(\text{SUM},<)}(\{-1 : p(-1), 1 : p(1), 2 : p(2)\}, \{-1 : \bot, 1 : \bot, 2 : \top\})$ because $I_1 \models p(-1)$, $I_1 \models p(1)$, $I_1 \not\models p(2)$, and

$$Q^U_{(\text{SUM},<)}(\{-1, 1\}, \{2\}) = \text{TRUE}.$$

I_1 satisfies $Q_{(\text{SUM},>)}(\{-1\!:\!p(-1), 1\!:\!p(1), 2\!:\!p(2)\}, \{-1\!:\!\top, 1\!:\!\bot, 2\!:\!\bot\})$ because

$$Q^U_{(\text{SUM},>)}(\{-1, 1\}, \{-1\}) = \text{TRUE}.$$

Consequently, I_1 satisfies (6).

Proposition 2. *Let σ be a signature that contains finitely many predicate constants, let σ^{pred} be the set of predicate constants in σ, let $I = \langle I^{func}, I^{pred}\rangle$ be an interpretation of σ, and let F be a GQ-sentence of σ. Then $I \models F$ iff $I^{pred} \models gr_I[F]$.*

Definition 8. *For any GQ-formula F w.r.t. I, the reduct of F relative to I, denoted by $F^{\underline{I}}$, is defined in the same way as in Definition 4 by replacing the last two clauses with the following single clause:*

– $$(Q(S_1, \ldots, S_k))^{\underline{I}} = \begin{cases} Q(S_1^{\underline{I}}, \ldots, S_k^{\underline{I}}) & \textit{if } I \models Q(S_1, \ldots, S_k), \\ \bot & \textit{otherwise;} \end{cases}$$

where $S_i^{\underline{I}} = \{\boldsymbol{\xi}^\diamond\!:\!(F(\boldsymbol{\xi}^\diamond))^{\underline{I}} \mid \boldsymbol{\xi}^\diamond\!:\!F(\boldsymbol{\xi}^\diamond) \in S_i\}$.

Theorem 2. *Let σ be a signature that contains finitely many predicate constants, let σ^{pred} be the set of predicate constants in σ, let $I = \langle I^{func}, I^{pred}\rangle$ be an interpretation of σ, and let F be a GQ-sentence of σ. $I \models \text{SM}[F; \sigma^{pred}]$ iff I^{pred} is a minimal set of atoms that satisfies $(gr_I[F])^{\underline{I}}$.*

Example 4 continued (III). Interpretation I_1 considered earlier can be identified with the tuple $\langle I^{func}, \{p(-1), p(1)\}\rangle$ where I^{func} maps every term to itself. The reduct $(gr_{I_1}[F_1])^{\underline{I_1}}$ is

$$
\begin{array}{l}
(\bot \rightarrow \bot) \\
\wedge\, (Q_{(\text{SUM},>)}(\{-1\!:\!p(-1), 1\!:\!p(1), 2\!:\!\bot\}, \{-1\!:\!\top, 1\!:\!\bot, 2\!:\!\bot\}) \rightarrow p(-1)) \\
\wedge\, (p(-1) \rightarrow p(1))\ ,
\end{array}
$$

which is the GQ-formula representation of (3). We can check that $\{p(-1), p(1)\}$ is a minimal model of the reduct.

Extending Theorem 2 to allow an arbitrary list of intensional predicates, rather than σ^{pred}, is straightforward in view of Proposition 1 from [18].

[4] For simplicity, we write $-1, 1, 2$ instead of their object names $(-1)^\diamond, 1^\diamond, 2^\diamond$.

4 FLP Semantics of Programs with Generalized Quantifiers

The FLP stable model semantics [1] is an alternative way to define stable models. It is the basis of HEX programs, an extension of the stable model semantics with higher-order and external atoms, which is implemented in system DLV-HEX. The first-order generalization of the FLP stable model semantics for programs with aggregates was given in [5], using the FLP operator that is similar to the SM operator. In this section we show how it can be extended to allow generalized quantifiers.

4.1 FLP Semantics of Programs with Generalized Quantifiers

A *(general) rule* is of the form

$$H \leftarrow B \tag{7}$$

where H and B are arbitrary GQ-formulas. A *(general) program* is a finite set of rules.

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 $u_1, \dots, u_n$. For any formula G, formula $G(\mathbf{u})$ is obtained from G by replacing all occurrences of predicates from $\mathbf{p}$ with the corresponding predicate variables from $\mathbf{u}$.

Let Π be a finite program whose rules have the form (7). The *GQ-representation* Π^{GQ} of Π is the conjunction of the universal closures of $B \rightarrow H$ for all rules (7) in Π. By FLP$[\Pi; \mathbf{p}]$ we denote the second-order formula

$$\Pi^{GQ} \wedge \neg\exists \mathbf{u}(\mathbf{u} < \mathbf{p} \wedge \Pi^{\triangle}(\mathbf{u}))$$

where $\Pi^{\triangle}(\mathbf{u})$ is defined as the conjunction of the universal closures of

$$B \wedge B(\mathbf{u}) \rightarrow H(\mathbf{u})$$

for all rules $H \leftarrow B$ in Π.

We will often simply write FLP$[\Pi]$ instead of FLP$[\Pi; \mathbf{p}]$ when $\mathbf{p}$ is the list of all predicate constants occurring in Π, and call a model of FLP$[\Pi]$ an *FLP-stable* model of Π.

Example 4 continued (IV). For formula F_1 considered earlier, FLP$[F_1]$ is

$$F_1 \wedge \neg\exists u(u < p \wedge F_1^{\triangle}(u)) \;, \tag{8}$$

where $F_1^{\triangle}(u)$ is

$$\begin{aligned}
&(\neg Q_{(\textsc{sum},<)}[x][y](p(x), y\!=\!2) \wedge \neg Q_{(\textsc{sum},<)}[x][y](u(x), y\!=\!2) \rightarrow u(2)) \\
&\wedge (Q_{(\textsc{sum},>)}[x][y](p(x), y\!=\!-1) \wedge (Q_{(\textsc{sum},>)}[x][y](u(x), y\!=\!-1) \rightarrow u(-1)) \\
&\wedge (p(-1) \wedge u(-1) \rightarrow u(1)) \;.
\end{aligned}$$

I_1 considered earlier satisfies (8) but I_2 does not.

5 Comparing the FLP Semantics and the First-Order Stable Model Semantics

In this section, we show a class of programs with GQs for which the FLP semantics and the first-order stable model semantics coincide.

The following definition is from [17]. We say that a generalized quantifier Q is *monotone in the i-th argument position* if the following holds for any universe U: if $Q^U(R_1, \dots, R_k) = \text{TRUE}$ and $R_i \subseteq R'_i \subseteq U^{n_i}$, then

$$Q^U(R_1, \dots, R_{i-1}, R'_i, R_{i+1}, \dots, R_k) = \text{TRUE}.$$

Consider a program Π consisting of rules of the form

$$A_1; \dots; A_l \leftarrow E_1, \dots, E_m, \textit{not } E_{m+1}, \dots, \textit{not } E_n$$

($l \geq 0$; $n \geq m \geq 0$), where each A_i is an atomic formula and each E_i is an atomic formula or a GQ-formula (4) such that all $F_1(\mathbf{x}_1), \dots, F_k(\mathbf{x}_k)$ are atomic formulas. Furthermore we require that, for every GQ-formula (4) in one of $E_{m+1}, \dots, E_n$, Q is monotone in all its argument positions.

Proposition 3. *Let Π be a program whose syntax is described as above, and let F be the GQ-representation of Π. Then* $\text{FLP}[\Pi; \mathbf{p}]$ *is equivalent to* $\text{SM}[F; \mathbf{p}]$.

Example 5 *Consider the following one-rule program:*

$$p(a) \leftarrow \textit{not } Q_{\leq 0}[x]\ p(x)\ . \tag{9}$$

This program does not belong to the syntactic class of programs stated in Proposition 3 since $Q_{\leq 0}[x]\ p(x)$ is not monotone in $\{1\}$. Indeed, both $\emptyset$ and $\{p(a)\}$ satisfy $\text{SM}[\Pi; p]$*, but only $\emptyset$ satisfies* $\text{FLP}[\Pi; p]$.

Conditions under which the FLP semantics coincides with the first-order stable model semantics has been studied in [4; 5] in the context of logic programs with aggregates.

6 Conclusion

We introduced two definitions of a stable model. One is a reformulation of the first-order stable model semantics and its extension to allow generalized quantifiers by referring to grounding and reduct, and the other is a reformulation of the FLP semantics and its extension to allow generalized quantifiers by referring to a translation into second-order logic. These new definitions help us understand the relationship between the FLP semantics and the first-order stable model semantics, and their extensions. For the class of programs where the two semantics coincide, system DLV-HEX can be viewed as an implementation of the stable model semantics of GQ-formulas; A recent extension of system F2LP [19] to allow "complex" atoms may be considered as a front-end to DLV-HEX to implement the generalized FLP semantics.

Acknowledgements

We are grateful to Vladimir Lifschitz for useful discussions related to this paper. We are also grateful to Joseph Babb and the anonymous referees for their useful comments. This work was partially supported by the National Science Foundation under Grant IIS-0916116 and by the South Korea IT R&D program MKE/KIAT 2010-TD-300404-001.

References

1. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: Semantics and complexity. In: Proceedings of European Conference on Logics in Artificial Intelligence (JELIA). (2004)
2. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1) (2011) 278–298
3. Ferraris, P.: Answer sets for propositional theories. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2005) 119–131
4. Lee, J., Meng, Y.: On reductive semantics of aggregates in answer set programming. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2009) 182–195
5. Bartholomew, M., Lee, J., Meng, Y.: First-order extension of the flp stable model semantics via modified circumscription. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2011) 724–730
6. Ferraris, P., Lifschitz, V.: On the stable model semantics of first-order formulas with aggregates. In: Proceedings of International Workshop on Nonmonotonic Reasoning (NMR). (2010)
7. Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), AAAI Press (2007) 372–379
8. Ferraris, P., Lee, J., Lifschitz, V.: Stable models and circumscription. Artificial Intelligence **175** (2011) 236–263
9. Lee, J., Meng, Y.: Stable models of formulas with generalized quantifiers (preliminary report). In: Technical Communications of the 28th International Conference on Logic Programming. (2012)
10. Marek, V.W., Truszczynski, M.: Logic programs with abstract constraint atoms. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). (2004) 86–91
11. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artificial Intelligence **172**(12-13) (2008) 1495–1539
12. Truszczynski, M.: Connecting first-order ASP and the logic FO(ID) through reducts. In: Correct Reasoning: Essays on Logic-Based AI in Honor of Vladimir Lifschitz. (2012) 543–559
13. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2005) 90–96
14. Fink, M., Pearce, D.: A logical semantics for description logic programs. In: Proceedings of European Conference on Logics in Artificial Intelligence (JELIA). (2010) 156–168
15. Westerståhl, D.: Generalized quantifiers. In: The Stanford Encyclopedia of Philosophy (Winter 2008 Edition). (2008) URL = <http://plato.stanford.edu/archives/win2008/entries/generalized-quantifiers/>.

16. Lindström, P.: First-order predicate logic with generalized quantifiers. Theoria **32** (1966) 186–195
17. Lee, J., Meng, Y.: Stable models of formulas with generalized quantifiers. In: Proceedings of International Workshop on Nonmonotonic Reasoning (NMR). (2012) http://peace.eas.asu.edu/joolee/papers/smgq-nmr.pdf.
18. Lee, J., Palla, R.: Reformulating the situation calculus and the event calculus in the general theory of stable models and in answer set programming. Journal of Artificial Intelligence Research (JAIR) **43** (2012) 571–620
19. Lee, J., Palla, R.: System F2LP — computing answer sets of first-order formulas. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2009) 515–521