Stable Models of Formulas with Generalized Quantifiers

Joohyung Lee and Yunsong Meng
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA
{joolee, Yunsong.Meng}@asu.edu

Abstract

Applications of answer set programming motivated various extensions of the stable model semantics, for instance, to incorporate aggregates and abstract constraint atoms, to facilitate interface with external information sources, such as ontology descriptions, and to integrate with other computing paradigms, such as constraint solving. This paper provides a uniform view on some of these extensions by viewing them as special cases of formulas with generalized quantifiers. The stable models of such formulas are defined by extending the first-order stable model semantics by Ferraris, Lee and Lifschitz. The generality of the formalism introduced here enables a systematic approach to study the individual extensions of the stable model semantics, and allows them to be combined in a single language. We show that important theorems in answer set programming, such as the splitting theorem, the theorem on completion, and the theorem on strong equivalence, can be naturally extended to formulas with generalized quantifiers, which in turn can be applied to the individual extensions of the stable model semantics.

Introduction

Applications of answer set programming motivated various recent extensions of the stable model semantics, for instance, to incorporate aggregates (Faber, Pfeifer, and Leone 2011; Ferraris 2005; Son and Pontelli 2007) and abstract constraint atoms (Marek and Truszczynski 2004), and to facilitate interface with external information sources, such as ontology descriptions (Eiter et al. 2008). While the extensions were driven by different motivations and applications, a common underlying issue is to extend the stable model semantics to incorporate "complex atoms," such as aggregates, abstract constraint atoms and dl-atoms.

HEX programs (Eiter et al. 2005) provide an elegant solution to incorporate such different extensions in a uniform framework via "external atoms." The idea is to define the meaning of external atoms in terms of external functions. For example, aggregate $\text{COUNT}\langle x.p(x,a)\rangle \geq 3$ is associated with binary external function $f_{\#count}$ such that, for any Herbrand interpretation I, $f_{\#count}(I,a,3) = 1$ iff $|\{c \mid c \in |I|,\ I \models p(c,a)\}| \geq 3$. Once the notion of satisfaction is extended to cover external atoms, the stable models of HEX programs are defined as minimal models of the "FLP-reduct" (Faber, Pfeifer, and Leone 2011). The adoption of the FLP reduct instead of the traditional Gelfond-Lifschitz reduct was a key idea to incorporate external atoms in HEX programs. HEX programs are well studied (Eiter et al. 2006; Eiter et al. 2008; Eiter et al. 2011), and are implemented in the system DLV-HEX.[1]

However, the FLP semantics diverges from the traditional stable model semantics in some essential ways. For example, consider a program

$$p(a) \leftarrow \mathit{not}\ \text{COUNT}\langle x.p(x)\rangle < 1, \tag{1}$$

and another program which rewrites the first program as

$$\begin{array}{l} p(a) \leftarrow \mathit{not}\ q \\ q \leftarrow \text{COUNT}\langle x.p(x)\rangle < 1, \end{array} \tag{2}$$

where the second rule defines q in terms of the COUNT aggregate. One may expect this transformation to modify the collection of answer sets in a "conservative" way. That is, each answer set of (2) is obtained from an answer set of (1) in accordance with the definition of q.[2] However, this is not the case under the FLP semantics: the former has $\emptyset$ as the only FLP answer set while the latter has both $\{p(a)\}$ and $\{q\}$ as the FLP answer sets. [3]

Related to this issue is the anti-chain property that is ensured by the FLP semantics: no FLP answer set is a proper subset of another FLP answer set. Consequently, the semantics does not allow choice rules (Niemelä and Simons 2000), which are useful in the "generate-and-test" organization of a program (Lifschitz 2002). Also, Bartholomew, Lee, and Meng (2011) note that the extensions of the FLP semantics to complex formulas as in (Truszczyński 2010; Bartholomew, Lee, and Meng 2011) encounter some unintuitive cases.

On the other hand, these issues do not arise with the stable model semantics from (Ferraris 2005; Lee, Lifschitz, and Palla 2008). According to (Ferraris 2005), which defines the semantics of aggregates by reduction to propositional formulas under the stable model semantics, program (1) has $\{p(a)\}$ and $\emptyset$ as the answer sets, and program (2) has $\{p(a)\}$ and $\{q\}$

[1] http://www.kr.tuwien.ac.at/research/systems/dlvhex/

[2] Indeed, this is what happens in expressing a rule with nested expressions like $p \leftarrow \mathit{not\ not}\ p$ into $p \leftarrow \mathit{not}\ q$, $q \leftarrow \mathit{not}\ p$ by defining q as *not* p.

[3] See the related discussion in http://www.cs.utexas.edu/users/vl/tag/aggregates.

as the answer sets. According to (Lee, Lifschitz, and Palla 2008), choice rule $\{q(x)\} \leftarrow p(x)$ is identified with

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

under the first-order stable model semantics from (Ferraris, Lee, and Lifschitz 2007; Ferraris, Lee, and Lifschitz 2011). In the same paper (Lee, Lifschitz, and Palla 2008), the reductive approach to defining aggregates in (Ferraris 2005) was extended to first-order formulas, but was limited to the COUNT aggregate. Extensions to cover arbitrary aggregates in the first-order case were done in (Lee and Meng 2009; Ferraris and Lifschitz 2010). But even then they do not account for other complex atoms like dl-atoms and external atoms.

So one wonders if it is possible to combine the versatility of HEX programs and the properties of the first-order stable model semantics. In this paper, we answer this question positively by extending the first-order stable model semantics to formulas with generalized quantifiers—a generalization of the standard quantifiers $\forall$ and $\exists$ (Mostowski 1957). The choice is natural as we show in the paper, those complex atoms essentially represent relations among sets of individuals, which can be naturally represented by generalized quantifiers. It was already hinted in (Ferraris and Lifschitz 2010) that aggregates may be viewed in terms of generalized quantifiers. Following up on that suggestion, we present an alternative approach to HEX programs by understanding complex atoms in terms of generalized quantifiers. Our semantics avoids the above issues with the FLP semantics, and allows natural extensions of several important theorems about the first-order stable model semantics from (Ferraris, Lee, and Lifschitz 2011), such as the splitting theorem, the theorem on completion and the theorem on strong equivalence, to formulas with generalized quantifiers. Those theorems in turn apply to the individual extensions of the stable model semantics, which saves efforts in re-proving the theorems for the specific extensions. It also allows us to combine different individual extensions in a single language as in the following example.

Example 1 *We consider an extension of nonmonotonic dl-programs $(\mathcal{T}, \Pi)$ that allows aggregates. For instance, a knowledge base $\mathcal{T}$ specifies that every married man has a spouse who is a woman and similarly for married woman:*

$$\begin{array}{rcl} \mathit{Man} \sqcap \mathit{Married} & \sqsubseteq & \exists \mathit{Spouse}.\mathit{Woman} \\ \mathit{Woman} \sqcap \mathit{Married} & \sqsubseteq & \exists \mathit{Spouse}.\mathit{Man}. \end{array}$$

The following program Π counts the number of people who are eligible for an insurance discount:

$$\begin{array}{l} \mathit{discount}(x) \leftarrow \mathit{not\ accident}(x), \\ \quad \#dl[\mathit{Man} \uplus \mathit{mm}, \mathit{Married} \uplus \mathit{mm}, \mathit{Woman} \uplus \mathit{mw}, \\ \qquad\qquad \mathit{Married} \uplus \mathit{mw}; \exists \mathit{Spouse}.\top](x). \\ \mathit{discount}(x) \leftarrow \mathit{discount}(y), \mathit{family}(y, x), \\ \qquad\qquad \mathit{not\ accident}(x). \\ \mathit{numOfDiscount}(z) \leftarrow \text{COUNT}\langle x.\mathit{discount}(x)\rangle = z. \end{array}$$

The first rule describes that everybody who has a spouse and has no accident is eligible for a discount. The second rule describes that everybody who has no accident and has a family member who has a discount is eligible for a discount.

The paper is organized as follows. We first review the syntax and the classical semantics of formulas with generalized quantifiers (GQ-formulas). Next we define stable models of formulas with generalized quantifiers and then show the individual extensions of the stable model semantics, such as logic programs with aggregates, abstract constraint atoms, and nonmonotonic dl-atoms, can be viewed as special cases of GQ-formulas. We extend important theorems in answer set programming, such as the splitting theorem, the theorem on completion, and the theorem on strong equivalence, to formulas with generalized quantifiers.

Preliminaries

Syntax of Formulas with Generalized Quantifiers

We follow the definition of a formula with generalized quantifiers from (Westerståhl 2008, Section 5) (that is to say, with Lindström quantifiers (Lindström 1966) without the isomorphism closure condition).

As in first-order logic, a *signature* σ is a set of symbols consisting of *function constants* and *predicate constants*. Each symbol is assigned a nonnegative integer, called the arity. Function constants with arity 0 are called *object constants*, and predicate constants with arity 0 are called *propositional constants*. A *term* is an object variable or $f(t_1, \ldots, t_n)$, where f is a function constant in σ of arity n, and t_i are terms. An *atomic formula* is an expression of the form $p(t_1, \ldots, t_n)$ or $t_1 = t_2$, where p is a predicate constant in σ of arity n.

We assume a set $\mathbf{Q}$ of symbols for generalized quantifiers. Each symbol in $\mathbf{Q}$ is associated with a tuple of nonnegative integers $\langle n_1, \ldots, n_k\rangle$ ($k \geq 0$, 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 is a GQ-formula;
- if $F_1, \ldots, F_k$ are GQ-formulas and Q is a generalized quantifier of type $\langle n_1, \ldots, n_k\rangle$ in $\mathbf{Q}$, then

$$Q[\mathbf{x}_1] \ldots [\mathbf{x}_k](F_1(\mathbf{x}_1), \ldots, F_k(\mathbf{x}_k)) \qquad (3)$$

  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 formula F is *bound* if it belongs to a subformula of F that has the form $Q[\mathbf{x}_1] \ldots [\mathbf{x}_k](F_1(\mathbf{x}_1), \ldots, F_k(\mathbf{x}_k))$ 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 sentence is a formula with no free variables.

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

Semantics of Formulas with Generalized Quantifiers

As in first-order logic, an interpretation I of a signature σ consists of a nonempty set U, called the *universe* of I, and

a mapping c^I for each constant c in σ. For each function constant f of σ whose arity is n, f^I is an element of U if n is 0, and is a function from U^n to U otherwise. For each predicate constant p of σ whose arity is n, p^I is an element of $\{\mathbf{t}, \mathbf{f}\}$ if n is 0, and is a function from U^n to $\{\mathbf{t}, \mathbf{f}\}$ otherwise. 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 $\{\mathbf{t}, \mathbf{f}\}$, where $\mathcal{P}(U^{n_i})$ denotes the power set of U^{n_i}.

Example 2 *Besides the standard connectives and quantifiers, the following are other examples of generalized quantifiers.*

- *type* $\langle 1\rangle$ *quantifier* $Q_{\leq 2}$ *such that* $Q^U_{\leq 2}(R) = \mathbf{t}$ *iff the cardinality of* R *is* ≤ 2*;* [4]
- *type* $\langle 1\rangle$ *quantifier* $Q_{majority}$ *such that* $Q^U_{majority}(R) = \mathbf{t}$ *iff the cardinality of* R *is greater than the cardinality of* $U \setminus R$*;*
- *type* $\langle 2, 1, 1\rangle$ *reachability quantifier* Q_{reach} *such that* $Q^U_{reach}(R_1, R_2, R_3) = \mathbf{t}$ *iff there are some* $u, v \in U$ *such that* $R_2 = \{u\}$*,* $R_3 = \{v\}$ *and* (u, v) *belongs to the transitive closure of* R_1*.*

By σ^I we mean the signature obtained from σ by adding new object constants ξ^*, called *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 $I(\xi^*) = \xi$. For any term t of σ^I that does not contain variables, we define recursively the element t^I of the universe that is assigned to t by I. If t is an object constant then t^I is an element of U. For other terms, t^I is defined by the equation

$$f(t_1, \dots, t_n)^I = f^I(t_1^I, \dots, t_n^I)$$

for all function constants f of arity $n > 0$.

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 $(\mathbf{x}_i.F_i(\mathbf{x}_i))^I = \{\boldsymbol{\xi} \in U^{n_i} \mid (F_i(\boldsymbol{\xi}^*))^I = \mathbf{t}\}$.

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

- $Q^U_\forall(R) = \mathbf{t}$ iff $R = U$;
- $Q^U_\exists(R) = \mathbf{t}$ iff $R \cap U \neq \emptyset$;
- $Q^U_\wedge(R_1, R_2) = \mathbf{t}$ iff $R_1 = R_2 = \{\epsilon\}$;[5]
- $Q^U_\vee(R_1, R_2) = \mathbf{t}$ iff $R_1 = \{\epsilon\}$ or $R_2 = \{\epsilon\}$;
- $Q^U_\rightarrow(R_1, R_2) = \mathbf{t}$ iff $R_1 = \emptyset$ or $R_2 = \{\epsilon\}$;
- $Q^U_\neg(R) = \mathbf{t}$ iff $R = \emptyset$;
- $Q^U_\bot() = \mathbf{f}$.

We say that an interpretation I *satisfies* a sentence F, or is a *model* of F, and write $I \models F$, if $F^I = \mathbf{t}$. A sentence F is *logically valid* if every interpretation satisfies F.

Example 3 *Let* I_1 *be an interpretation whose universe is* $\{1, 2, 3, 4\}$ *and let* p *be a unary predicate constant such that* $p(\xi^*)^{I_1} = \mathbf{t}$ *iff* $\xi \in \{1, 2, 3\}$*. We check that* I_1 *satisfies formula* $F = \neg Q_{\leq 2}[x]\ p(x) \rightarrow Q_{majority}[y]\ p(y)$ *("if* p *does not contain at most two elements in the universe, then* p *contains a majority"). Let* I_2 *be another interpretation with the same universe such that* $p(\xi^*)^{I_2} = \mathbf{t}$ *iff* $\xi \in \{1\}$*. It is clear that* I_2 *also satisfies* F*.*

We say that a generalized quantifier Q is *monotone in the* i*-th argument position* if the following holds for any interpretation I: if $Q^U(R_1, \dots, R_k) = \mathbf{t}$ and $R_i \subseteq R'_i \subseteq U^{n_i}$, then $Q^U(R_1, \dots, R_{i-1}, R'_i, R_{i+1}, \dots, R_k) = \mathbf{t}$. Similarly, we say that Q is *anti-monotone in the* i*-th argument position* if the following holds for any interpretation I: if $Q^U(R_1, \dots, R_k) = \mathbf{t}$ and $R'_i \subseteq R_i \subseteq U^{n_i}$, then $Q^U(R_1, \dots, R_{i-1}, R'_i, R_{i+1}, \dots, R_k) = \mathbf{t}$. We call an argument position of Q *monotone (anti-monotone)* if Q is monotone (anti-monotone) in the argument position.

Let M be a subset of $\{1, \dots, k\}$. We say that Q is *monotone in* M if Q is monotone in the i-th argument position for all i in M. It is easy to check that both $Q_\wedge$ and $Q_\vee$ are monotone in $\{1, 2\}$. $Q_\rightarrow$ is anti-monotone in $\{1\}$ and monotone in $\{2\}$; $Q_\neg$ is anti-monotone in $\{1\}$. In Example 2, $Q_{\leq 2}$ is anti-monotone in $\{1\}$ and $Q_{majority}$ is monotone in $\{1\}$. We will see later that monotonicity and anti-monotonicity play important roles in the properties of stable models for formulas with generalized quantifiers.

Stable Models of GQ-Formulas

We now define the stable model operator SM with a list of intensional predicates. 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 $u_1, \dots, u_n$. 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 $i = 1, \dots, n$, where $\mathbf{x}$ is a list of distinct object variables of the same length as the arity of p_i, and by $\mathbf{u} < \mathbf{p}$ we denote $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{p} \leq \mathbf{u})$. For instance, if p and q are unary predicate constants then $(u, v) < (p, q)$ is

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

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

$$F \wedge \neg\exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u})), \tag{4}$$

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 $\mathbf{p}$;

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

[5] ϵ denotes the empty tuple. For any interpretation I, $U^0 = \{\epsilon\}$. For I to satisfy $Q_\wedge[][](F, G)$, both $(\epsilon.F)^I$ and $(\epsilon.G)^I$ have to be $\{\epsilon\}$, which means that $F^I = G^I = \mathbf{t}$.

•

$$(Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1),\dots,F_k(\mathbf{x}_k)))^* = \\ Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1^*(\mathbf{x}_1),\dots,F_k^*(\mathbf{x}_k)) \\ \wedge Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1),\dots,F_k(\mathbf{x}_k)). \quad (5)$$

When F is a sentence, the models of $\mathrm{SM}[F;\mathbf{p}]$ 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 $\mathrm{SM}[F]$ in place of $\mathrm{SM}[F;\mathbf{p}]$ when $\mathbf{p}$ is the list of all predicate constants occurring in F, and call $\mathbf{p}$-stable models simply stable models.

Proposition 1 *Let M be a subset of $\{1,\dots,k\}$ and let $Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1),\dots,F_k(\mathbf{x}_k))$ be a formula such that no predicate constant from $\mathbf{p}$ occurs in F_j for all $j \in \{1,\dots,k\} \setminus M$.*

(a) If Q is monotone in M, then

$$\mathbf{u} \le \mathbf{p} \to ((Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1),\dots,F_k(\mathbf{x}_k)))^* \\ \leftrightarrow Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1^*(\mathbf{x}_1),\dots,F_k^*(\mathbf{x}_k)))$$

is logically valid.

(b) If Q is anti-monotone in M, then

$$\mathbf{u} \le \mathbf{p} \to ((Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1),\dots,F_k(\mathbf{x}_k)))^* \\ \leftrightarrow Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1(\mathbf{x}_1),\dots,F_k(\mathbf{x}_k)))$$

is logically valid.

Proposition 1 allows us to simplify the formula $F^*(\mathbf{u})$ in (4) without affecting the models of (4). In formula (5), if Q is monotone in all argument positions, we can drop the second conjunctive term in view of Proposition 1 (a). If Q is anti-monotone in all argument positions, we can drop the first conjunctive term in view of Proposition 1 (b). For instance, recall that each of $Q_\wedge$, $Q_\vee$, $Q_\forall$, $Q_\exists$ is monotone in all its argument positions, and $Q_\neg$ is anti-monotone in $\{1\}$. If F is a standard first-order formula, then (5) can be equivalently rewritten as

- $(\neg F)^* = \neg F$;
- $(F \wedge G)^* = F^* \wedge G^*$; $(F \vee G)^* = F^* \vee G^*$;
- $(F \to G)^* = (F^* \to G^*) \wedge (F \to G)$;
- $(\forall x F)^* = \forall x F^*$; $(\exists x F)^* = \exists x F^*$.

This is almost the same as the definition of F^* given in (Ferraris, Lee, and Lifschitz 2011), except for $(\neg F)^*$.[6] The only propositional connective which is neither monotone nor anti-monotone in all argument positions is $Q_\to$, for which the simplification does not apply. Comparing the two definitions, we note that even in the case when F is a standard first-order formula, our definition of $\mathrm{SM}[F;\mathbf{p}]$ is a bit more concise.

Example 3 continued *For formula F considered earlier,* $\mathrm{SM}[F]$ *is*

$$F \wedge \neg\exists u(u < p \wedge F^*(u)), \quad (6)$$

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

$$\neg Q_{\le 2}[x]\ p(x) \to\ Q_{majority}[y]\ u(y). \quad (7)$$

I_1 considered earlier satisfies (6): it satisfies F and for any proper subset u of p, it satisfies the antecedent of (7) but not the consequent. Thus it is a stable model of F. On the other hand, we can check that I_2 does not satisfy (6).

[6] $\neg F$ is understood as $F \to \bot$ in (Ferraris, Lee, and Lifschitz 2011), but this difference does not affect stable models. When $\neg$ is a primitive propositional connective as above,

$$\mathbf{u} \le \mathbf{p} \to ((F \to \bot)^*(\mathbf{u}) \leftrightarrow (\neg F)^*(\mathbf{u}))$$

is logically valid.

Aggregates as GQ-Formulas

Formulas with Aggregates

The following definition of a formula with aggregates is from (Ferraris and Lifschitz 2010), which extends (Lee and Meng 2009). By a *number* we understand an element of some fixed set $\mathbf{Num}$. For example, $\mathbf{Num}$ is $\mathbf{Z} \cup \{+\infty, -\infty\}$, where $\mathbf{Z}$ is the set of integers. A *multiset* is usually defined as a set of elements along with a function assigning a positive integer, called *multiplicity*, to each of its elements. An *aggregate function* is a partial function from the class of multisets to $\mathbf{Num}$. We assume that the signature σ is a superset of the background signature σ_{bg} that contains symbols for all numbers. We assume that the interpretation of symbols in the background signature is fixed. That is, each number is interpreted as itself. An *expansion* I of I_{bg} to σ is an interpretation of σ such that

- the universe of I is the same as the universe of I_{bg}, and
- I agrees with I_{bg} on the constants in σ_{bg}.

An *aggregate formula* is defined as an extension of a first-order formula by adding the following clause:

•

$$\mathrm{OP}\langle \mathbf{x}_1.F_1,\dots,\mathbf{x}_n.F_n\rangle \succeq b \quad (8)$$

is *a first-order formula with aggregates*, where

- OP is a symbol for an *aggregate function* (not from σ);
- $\mathbf{x}_1,\dots,\mathbf{x}_n$ are nonempty lists of distinct object variables;
- $F_1,\dots,F_n$ are arbitrary *first-order formulas with aggregates* of signature σ;
- $\succeq$ is a symbol for a comparison operator (may not necessarily be from σ);
- b is a term of σ.

Aggregates as GQ-Formulas

Due to the space limit, we refer the reader to (Ferraris and Lifschitz 2010) for the stable model semantics of formulas with aggregates. We can explain their semantics by viewing it as a special case of stable model semantics presented here. Following (Ferraris and Lifschitz 2010), for any set X of n-tuples ($n \ge 1$), let *msp*(X) ("the multiset projection of X") be the multiset consisting of all ξ_1 such that $(\xi_1,\dots,\xi_n) \in X$ for at least one $(n-1)$-tuple $(\xi_2,\dots,\xi_n)$, with the multiplicity equal to the number of such $(n-1)$-tuples (and to $+\infty$ if there are infinitely many of them). For example, *msp*$(\{(a,a),(a,b),(b,a)\}) = \{\!\{a,a,b\}\!\}$.

We identify expression (8) with the GQ-formula

$$Q_{(\mathrm{OP},\succeq)}[\mathbf{x}_1]\dots[\mathbf{x}_n][y](F_1(\mathbf{x}_1),\dots,F_n(\mathbf{x}_n),y=b), \quad (9)$$

where, for any interpretation I, $Q^U_{(\text{OP},\succeq)}$ is a function that maps $\mathcal{P}(U^{|\mathbf{x_1}|}) \times \cdots \times \mathcal{P}(U^{|\mathbf{x_n}|}) \times \mathcal{P}(U)$ to $\{\mathbf{t}, \mathbf{f}\}$ such that $Q^U_{(\text{OP},\succeq)}(R_1, \ldots, R_n, R_{n+1}) = \mathbf{t}$ iff

- $\text{OP}(\alpha)$ is defined, where α is the join of the multisets $msp(R_1), \ldots, msp(R_n)$,
- $R_{n+1} = \{b^I\}$, where $b^I \in \mathbf{Num}$, and
- $\text{OP}(\alpha) \succeq b^I$.

Example 4 *$\{discount(alice), discount(carol), numOfDiscounts(2)\}$ is a Herbrand stable model of the formula*

$$discount(alice) \wedge discount(carol) \wedge \forall z(\text{COUNT}\langle x.discount(x)\rangle = z \rightarrow numOfDiscounts(z)).$$

The following proposition states that this definition is equivalent to the definition from (Ferraris and Lifschitz 2010).

Proposition 2 *Let F be a first-order sentence with aggregates whose signature is σ, and let $\mathbf{p}$ be a list of predicate constants. For any expansion I of σ_{bg} to σ, I is a $\mathbf{p}$-stable model of F in the sense of (Ferraris and Lifschitz 2010) iff I is a $\mathbf{p}$-stable model of F in our sense.*

Abstract Constraint Atoms as GQ-Formulas

Marek and Truszczynski (2004) viewed propositional aggregates as a special case of abstract constraint atoms. Son, Pontelli, and Tu (2007) generalized this semantics to account for arbitrary abstract constraint atoms. In this section we present an alternative semantics of programs with abstract constraint atoms by reduction to formulas with generalized quantifiers. This tells us that GQ-atoms serve as a first-order counterpart of abstract constraint atoms.

Let σ be a propositional signature, D be a finite list of atoms of σ and C be a subset of the power set $\mathcal{P}(D)$.[7] An *abstract constraint atom* (or c-atom) (Son, Pontelli, and Tu 2007) is of the form $\langle D, C\rangle$. We say that an interpretation of σ satisfies a c-atom $\langle D, C\rangle$ if $I \cap D \in C$.

We view c-atoms as a special case of generalized quantifiers containing no variables, and this provides an alternative semantics of c-atoms that is distinct from (Son, Pontelli, and Tu 2007). An abstract constraint $\langle D, C\rangle$, where D is $(p_1, \ldots, p_n)$, can be viewed as a generalized quantified formula

$$Q_C[\,]\ldots[\,]\,D\,, \tag{10}$$

where, for any interpretation I of σ, Q^U_C is a function that maps $\mathcal{P}(\{\epsilon\}) \times \cdots \times \mathcal{P}(\{\epsilon\})$ to $\{\mathbf{t}, \mathbf{f}\}$ such that $Q^U_C(R_1, \ldots, R_n) = \mathbf{t}$ iff $\{p_i \mid 1 \leq i \leq n,\ R_i = \{\epsilon\}\} \in C$.

Lemma 1 *For any c-atom $\langle D, C\rangle$ of σ, let I be an interpretation of σ. I satisfies $\langle D, C\rangle$ in the sense of (Son, Pontelli, and Tu 2007) iff $I \models (10)$.*

The syntax of a *propositional formula with c-atoms* extends the standard syntax of a propositional formula by treating c-atoms as a base case in addition to standard atoms. Obviously, logic programs with c-atoms in the literature can be viewed as a special case of propositional formulas with c-atoms. The stable model semantics of such a formula is defined by understanding the formula as shorthand for the corresponding GQ-formula as described in Lemma 1.

Example 5 *The following is Example 4 from (Liu et al. 2010). Let F be the formula*

$$a \wedge b \wedge \big(\langle (a,b,c), \{\{a\},\{a,c\},\{a,b,c\}\}\rangle \rightarrow c\big)\,.$$

For new atoms d,e,f, formula $F^*(d,e,f)$ is

$$\begin{aligned} d \wedge e \wedge \Big(&\big(\langle (d,e,f), \{\{d\},\{d,f\},\{d,e,f\}\}\rangle \\ &\wedge \langle (a,b,c), \{\{a\},\{a,c\},\{a,b,c\}\}\rangle \rightarrow f\big) \\ &\wedge \big(\langle (a,b,c), \{\{a\},\{a,c\},\{a,b,c\}\}\rangle \rightarrow c\big)\Big)\,. \end{aligned}$$

Any subset X of $\{a,b,c\}$ is a stable model of F iff X satisfies F and for any proper subset Y of X, $X \cup Y^{abc}_{def}$ does not satisfy $F^*(d,e,f)$. (Here Y^{abc}_{def} is the set obtained from Y by replacing a,b,c with d,e,f.)

We can check that $\{a,b\}$ is the only Herbrand stable model of F. Indeed, $\{a,b\}$ satisfies F and each of $\{a,b\}$, $\{a,b,d\}$, $\{a,b,e\}$ does not satisfy $F^*(d,e,f)$.

A c-atom can be viewed as an abbreviation of a propositional formula. Given a c-atom (10), we define its *propositional formula representation* as

$$\bigwedge_{\overline{C} \in \mathcal{P}(D)\setminus C} \left(\bigwedge_{p \in \overline{C}} p \rightarrow \bigvee_{p \in D \setminus \overline{C}} p \right) \tag{11}$$

as defined by Ferraris (2005).

For any propositional formula F with c-atoms, by $Fer(F)$, we denote the usual propositional formula obtained from F by replacing every c-atom (10) with (11).

The following proposition tells us that c-atoms in a formula can be rewritten as propositional formulas under the stable model semantics from (Ferraris 2005).

Proposition 3 *For any propositional formula F with c-atoms and any propositional interpretation X, X is a stable model of F iff X is a stable model of $Fer(F)$.*

Example 5 continued *For the formula F above, $Fer(F)$ is*

$$\begin{aligned} a \wedge b \wedge \Big(&\big((a \vee b \vee c) \wedge (b \rightarrow a \vee c) \wedge (c \rightarrow a \vee b) \\ &\wedge (a \wedge b \rightarrow c) \wedge (b \wedge c \rightarrow a)\big) \rightarrow c\Big). \end{aligned}$$

We check that $\{a,b\}$ is the only Herbrand stable model of $Fer(F)$ in accordance with Proposition 3.

Note that our semantics of logic programs with c-atoms is not equivalent to the one from (Son, Pontelli, and Tu 2007). Lee and Meng (2009) present a propositional formula representation of abstract constraint atoms under the semantics from (Son, Pontelli, and Tu 2007), which is classically equivalent, but not strongly equivalent to (11).

[7] We will often identify a list with a set if there is no confusion.

Nonmonotonic dl-Programs as GQ-Formulas

Review of Nonmonotonic dl-Programs

Let C be a set of object constants, and let $P_{\mathcal{T}}$ and P_{Π} be disjoint sets of predicate constants. A nonmonotonic *dl-program* (Eiter et al. 2008) is a pair $(\mathcal{T}, \Pi)$, where $\mathcal{T}$ is a theory in description logic (DL) of signature $\langle C, P_{\mathcal{T}}\rangle$ and Π is a *generalized* normal logic program of signature $\langle C, P_{\Pi}\rangle$ such that $P_{\mathcal{T}} \cap P_{\Pi} = \emptyset$. We assume that Π contains no variables by applying grounding w.r.t. C. A generalized normal logic program is a set of nondisjunctive rules that can contain queries to $\mathcal{T}$ in the form of "dl-atoms." A *dl-atom* is of the form

$$DL[S_1 op_1 p_1, \ldots, S_k op_k p_k;\ Query](\mathbf{t}) \quad (k \geq 0), \tag{12}$$

where $S_i \in P_{\mathcal{T}}$, $p_i \in P_{\Pi}$, and $op_i \in \{\uplus, \cup, \cap\}$. $Query(\mathbf{t})$ is a *dl-query* as defined in (Eiter et al. 2008). A *dl-rule* is of the form

$$a \leftarrow b_1, \ldots, b_m, \mathit{not}\ b_{m+1}, \ldots, \mathit{not}\ b_n\ , \tag{13}$$

where a is an atom and each b_i is either an atom or a dl-atom. We identify rule (13) with

$$a \leftarrow B, N\ , \tag{14}$$

where B is $b_1, \ldots, b_m$ and N is $\mathit{not}\ b_{m+1}, \ldots, \mathit{not}\ b_n$. An Herbrand interpretation I *satisfies* a ground atom A *relative to* $\mathcal{T}$ if I satisfies A. An Herbrand interpretation I *satisfies* a ground dl-atom (12) *relative to* $\mathcal{T}$ if $\mathcal{T} \cup \bigcup_{i=1}^{k} A_i(I)$ entails $Query(\mathbf{t})$, where $A_i(I)$ is

- $\{S_i(\mathbf{e}) \mid\ p_i(\mathbf{e}) \in I\}$ if op_i is $\uplus$,
- $\{\neg S_i(\mathbf{e}) \mid\ p_i(\mathbf{e}) \in I\}$ if op_i is $\cup$,
- $\{\neg S_i(\mathbf{e}) \mid\ p_i(\mathbf{e}) \notin I\}$ if op_i is $\cap$.

A ground dl-atom A is *monotonic* relative to $\mathcal{T}$ if, for any two Herbrand interpretations I and I' such that $I \subseteq I'$, $I \models_{\mathcal{T}} A$ implies $I' \models_{\mathcal{T}} A$. Similarly, a ground dl-atom A is *anti-monotonic* relative to $\mathcal{T}$ if, for any two Herbrand interpretations I and I' such that $I \subseteq I'$, $I' \models_{\mathcal{T}} A$ implies $I \models_{\mathcal{T}} A$.

Given a dl-program $(\mathcal{T}, \Pi)$ and an Herbrand interpretation I of $\langle C, P_{\Pi}\rangle$, the *weak dl-transform* of Π relative to $\mathcal{T}$, denoted by $w\Pi^I_{\mathcal{T}}$, is the set of rules

$$a \leftarrow B' \tag{15}$$

where $a \leftarrow B, N$ is in Π, $I \models_{\mathcal{T}} B \wedge N$, and B' is obtained from B by removing all dl-atoms in it. Similarly, the *strong dl-transform* of Π relative to $\mathcal{T}$, denoted by $s\Pi^I_{\mathcal{T}}$, is the set of rules (15), where $a \leftarrow B, N$ is in Π, $I \models_{\mathcal{T}} B \wedge N$ and B' is obtained from B by removing all nonmonotonic dl-atoms in it. The only difference between these two definitions is whether monotonic dl-atoms in the positive body remain in the reduct or not.

An Herbrand interpretation I is a *weak (strong, respectively) answer set* of $(\mathcal{T}, \Pi)$ if I is minimal among the sets of atoms that satisfy $w\Pi^I_{\mathcal{T}}(s\Pi^I_{\mathcal{T}}, respectively)$.

Nonmonotonic dl-program as GQ-Formulas

Here we understand dl-programs as a special case of GQ-formulas. Consider a dl-program $(\mathcal{T}, \Pi)$ such that Π is ground. Under the strong answer set semantics, we identify every dl-atom (12) in Π with

$$Q_{(12)}[\mathbf{x}_1] \ldots [\mathbf{x}_k](p_1(\mathbf{x}_1), \ldots, p_k(\mathbf{x}_k)) \tag{16}$$

if it is monotonic relative to $(\mathcal{T}, \Pi)$, and

$$\neg\neg Q_{(12)}[\mathbf{x}_1] \ldots [\mathbf{x}_k](p_1(\mathbf{x}_1), \ldots, p_k(\mathbf{x}_k)) \tag{17}$$

otherwise.

Given an interpretation I, $Q^U_{(12)}$ is a function that maps $\mathcal{P}(U^{|\mathbf{x}_1|}) \times \cdots \times \mathcal{P}(U^{|\mathbf{x}_k|})$ to $\{\mathbf{t}, \mathbf{f}\}$ such that, $Q^U_{(12)}(R_1, \ldots, R_k) = \mathbf{t}$ iff $\mathcal{T} \cup \bigcup_{i=1}^{k} A_i(R_i)$ entails $Query(\mathbf{t})$, where $A_i(R_i)$ is

- $\{S_i(\boldsymbol{\xi}_i) \mid \boldsymbol{\xi}_i \in R_i\}$ if op_i is $\uplus$,
- $\{\neg S_i(\boldsymbol{\xi}_i) \mid \boldsymbol{\xi}_i \in R_i\}$ if op_i is $\cup$,
- $\{\neg S_i(\boldsymbol{\xi}_i) \mid \boldsymbol{\xi}_i \in U^{|\mathbf{x}_i|} \setminus R_i\}$ if op_i is $\cap$.

We say that I is a *strong answer set* of $(\mathcal{T}, \Pi)$ if I satisfies SM$[\Pi; P_{\Pi}]$.

Similarly a weak answer set of $(\mathcal{T}, \Pi)$ is defined by identifying every dl-atom (12) in Π with (17) regardless of A being monotonic or not.

Example 1 continued *The dl-atom*

$$\#dl[\mathit{Man} \uplus \mathit{mm}, \mathit{Married} \uplus \mathit{mm}, \mathit{Woman} \uplus \mathit{mw}, \mathit{Married} \uplus \mathit{mw}; \exists \mathit{Spouse}.\top](\mathit{alice}) \tag{18}$$

is identified with the generalized quantified formula

$$Q_{(18)}[x_1][x_2][x_3][x_4](\mathit{mm}(x_1), \mathit{mm}(x_2), \mathit{mw}(x_3), \mathit{mw}(x_4)) \tag{19}$$

where, for any interpretation I, $Q^U_{(18)}$ is a function that maps $\mathcal{P}(U) \times \mathcal{P}(U) \times \mathcal{P}(U) \times \mathcal{P}(U)$ to $\{\mathbf{t}, \mathbf{f}\}$ such that $Q^U_{(18)}(R_1, R_2, R_3, R_4) = \mathbf{t}$ iff $\mathcal{T} \cup \{\mathit{Man}(c) \mid c \in R_1\} \cup \{\mathit{Woman}(c) \mid c \in R_3\} \cup \{\mathit{Married}(c) \mid c \in R_2 \cup R_4\}$ entails $\exists x \mathit{Spouse}(\mathit{alice}, x)$.

Consider an Herbrand interpretation $I = \{mw(alice)\}$, which satisfies (18). I also satisfies (19) since $(x.mw(x))^I = \{alice\}$ and $\mathcal{T} \cup \{Woman(alice), Married(alice)\}$ entails $\exists x Spouse(alice, x)$.

The following proposition tells us that the definitions of a strong answer set and a weak answer set given here are equivalent to the definitions from (Eiter et al. 2008).

Proposition 4 *For any dl-program $(\mathcal{T}, \Pi)$, an Herbrand interpretation is a strong (weak, respectively) answer set of $(\mathcal{T}, \Pi)$ in the sense of (Eiter et al. 2008) iff it is a strong (weak, respectively) answer set of $(\mathcal{T}, \Pi)$ in our sense.*

Another Semantics of Nonmonotonic dl-programs

Shen (2011) notes that both strong and weak answer set semantics suffer from circular justifications.

Example 6 *(Shen 2011) Consider $(\mathcal{T}, \Pi)$, where $\mathcal{T} = \emptyset$ and Π is the program*

$$p(a) \leftarrow \#dl[c \uplus p, b \cap q;\ c \sqcap \neg b](a)\ , \qquad (20)$$

in which the dl-atom is neither monotonic nor anti-monotonic. The dl-program has two strong (weak, respectively) answer sets: $\emptyset$ and $\{p(a)\}$. According to (Shen 2011), the second answer set is circularly justified:

$$p(a) \Leftarrow \#dl[c \uplus p, b \cap q; c \sqcap \neg b](a) \Leftarrow p(a) \wedge \neg q(a).$$

Indeed, $s\Pi_{\mathcal{T}}^{\{p(a)\}}$ ($w\Pi_{\mathcal{T}}^{\{p(a)\}}$, respectively) is $p(a) \leftarrow$, and $\{p(a)\}$ is its minimal model.

The example suggests that the issue is related to the fact that both strong and weak answer set semantics do not distinguish between two kinds of nonmonotonic dl-atoms: anti-monotonic and non-anti-monotonic ones. The former does not contribute to loops, but the latter does, so that they should participate in enforcing minimality of answer sets (See the later section on loops). This suggests the following alternative definition of the semantics of dl-programs. Instead of removing every nonmonotonic dl-atoms in forming the reduct under strong answer set semantics, we remove only anti-monotonic dl-atoms from the bodies, but leave non-anti-monotonic dl-atoms. In other words, the *dl-transform* of Π relative to $\mathcal{T}$ and an Herbrand interpretation I of $\langle C, P_\Pi \rangle$, denoted by $\Pi_{\mathcal{T}}^I$, is the set of rules (15), where $a \leftarrow B, N$ is in Π, $I \models_{\mathcal{T}} B \wedge N$ and B' is obtained from B by removing all anti-monotonic dl-atoms in it. We say that an Herbrand interpretation I is an *answer set* of $(\mathcal{T}, \Pi)$ if I is minimal among the sets of atoms that satisfy $\Pi_{\mathcal{T}}^I$.

Example 6 continued *$\{p(a)\}$ is not an answer set of $(\mathcal{T}, \Pi)$ according to the new definition. $\Pi_{\mathcal{T}}^{\{p(a)\}}$ is (20) itself, and $\emptyset$, a proper subset of $\{p(a)\}$ satisfies it.*

This new definition can be also characterized in terms of generalized quantifiers. In fact, the characterization is simpler than those for the other two semantics. We simply identify (12) with (16) regardless of the (anti-)monotonicity of the dl-atom.

Proposition 5 *For any dl-program $(\mathcal{T}, \Pi)$, and any Herbrand interpretation X of $\langle C, P_\Pi \rangle$, X is an answer set of $(\mathcal{T}, \Pi)$ as defined here iff X satisfies $\mathrm{SM}[\Pi; \mathbf{p}]$ when we identify every dl-atom (12) in Π with (16).*

The new definition is closely related to another variant of FLP-reduct based semantics of nonmonotonic dl-programs from (Fink and Pearce 2010). The following proposition states that the relationship between the two semantics.

Proposition 6 *For any dl-program $(\mathcal{T}, \Pi)$, and any Herbrand interpretation X of $\langle C, P_\Pi \rangle$, if every occurrence of nonmonotonic dl-atoms is in the positive body of a rule, then X is an answer set of $(\mathcal{T}, \Pi)$ in the sense of (Fink and Pearce 2010) iff X is an answer set of $(\mathcal{T}, \Pi)$ in our sense.*

The following example shows why the condition in the statement is essential.

Example 7 *Consider the dl-program $(\mathcal{T}, \Pi)$ such that $\mathcal{T}$ is empty, and Π is the following single rule program containing a nonmonotonic dl-atom:*

$$p(a) \leftarrow \mathit{not}\ \#dl[C \cap p;\ \neg C](a).$$

While $\emptyset$ and $\{p(a)\}$ are answer sets according to us, only $\emptyset$ is the answer set according to (Fink and Pearce 2010).

Strong Equivalence

Strong equivalence (Lifschitz, Pearce, and Valverde 2001) is an important notion that allows us to substitute one subformula for another subformula without affecting the stable models. The theorem on strong equivalence from (Ferraris, Lee, and Lifschitz 2011) can be extended to GQ-formulas as follows.

About GQ-formulas F and G we say that F is *strongly equivalent* to G if, for any formula H, any occurrence of F in H, and any list $\mathbf{p}$ of distinct predicate and function constants, $\mathrm{SM}[H; \mathbf{p}]$ is equivalent to $\mathrm{SM}[H'; \mathbf{p}]$, where H' is obtained from H by replacing the occurrence of F by G. In this definition, H is allowed to contain object, function and predicate constants that do not occur in F, G; Theorem 1 below shows, however, that this is not essential.

Theorem 1 *Let F and G be GQ-formulas, let $\mathbf{p}$ be the list of all predicate constants occurring in F or G and let $\mathbf{u}$ be a list of distinct predicate variables corresponding to $\mathbf{p}$. Formulas F and G are strongly equivalent to each other iff the formula*

$$(\mathbf{u} \leq \mathbf{p}) \rightarrow (F^*(\mathbf{u}) \leftrightarrow G^*(\mathbf{u}))$$

is logically valid.

Example 8 *The program (2) in the introduction can be identified with the formula F*

$$(\neg \mathrm{COUNT}\langle x.p(x)\rangle < 1 \rightarrow p(a)) \\ \wedge (\mathrm{COUNT}\langle x.p(x)\rangle < 1 \rightarrow q),$$

and is strongly equivalent to the following formula G:

$$(\neg q \rightarrow p(a)) \\ \wedge (\mathrm{COUNT}\langle x.p(x)\rangle < 1 \rightarrow q).$$

One can check that $F^*(u, v)$ and $G^*(u, v)$ are equivalent to each other.

Splitting Theorem

We extend the splitting theorem from (Ferraris et al. 2009) to GQ-formulas.

Let F be a GQ-formula. We say that an occurrence of p in F is *mixed* if there is some generalized quantifier Q that contains the occurrence in its argument position which is neither monotone nor anti-monotone. Let l be the number of generalized quantifiers Q in F such that the occurrence of p belongs to an anti-monotone argument position of Q. If the occurrence is not mixed then we call it *positive* in F if l is even, and *negative* otherwise. The occurrence is *strictly positive* in F if $l = 0$. We call an occurrence of a predicate constant *semi-positive* if it is positive or mixed. Similarly, it is *semi-negative* if it is negative or mixed.

We say that F is *negative on* $\mathbf{p}$ if there is no strictly positive occurrence of a predicate constant from $\mathbf{p}$ in F. An

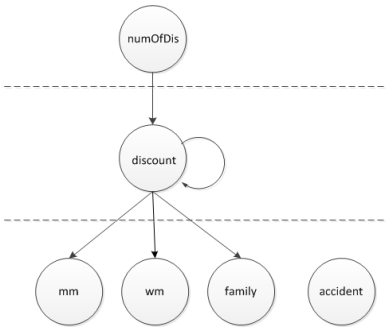

Figure 1: Dependency Graph of the Formula in Example 1

occurrence of a predicate constant or a subformula of F is $\mathbf{p}$-*negated* in F if it is contained in a subformula of F that is negative on $\mathbf{p}$.

The dependency graph of F relative to a list $\mathbf{p}$ of intensional predicates (denoted by $\mathrm{DG}_{\mathbf{p}}[F]$) is a directed graph such that

- the vertices are the members of $\mathbf{p}$, and
- there is an edge from p to q if there is a strictly positive occurrence of a subformula $G = Q[\mathbf{x}_1]\dots[\mathbf{x}_k](F_1, \dots, F_k)$ such that
  - p has a strictly positive occurrence in G, and
  - q has a semi-positive, non-$\mathbf{p}$-negated occurrence in a non-monotone argument position of Q.

Figure 1 shows the dependency graph of F relative to $\{\mathit{discount}, \mathit{family}, \mathit{mm}, \mathit{mw}, \mathit{accident}, \mathit{numOfDiscount}\}$.

A *loop* of F (relative to a list $\mathbf{p}$ of intensional predicates) is a nonempty subset $\mathbf{l}$ of $\mathbf{p}$ such that the subgraph of $\mathrm{DG}_{\mathbf{p}}[F]$ induced by $\mathbf{l}$ is strongly connected. It is clear that the strongly connected components of $\mathrm{DG}_{\mathbf{p}}[F]$ are the maximal loops of F.

Theorem 2 (Splitting Lemma) *Let F be a GQ-sentence, and let $\mathbf{p}$ be a tuple of distinct predicate constants. If $\mathbf{l}^1, \dots, \mathbf{l}^n$ are all the loops of F relative to $\mathbf{p}$ then*

$$\mathrm{SM}[F; \mathbf{p}] \quad \textit{is equivalent to} \quad \mathrm{SM}[F; \mathbf{l}^1] \wedge \dots \wedge \mathrm{SM}[F; \mathbf{l}^n].$$

The following theorem extends the splitting theorem from (Ferraris et al. 2009) to GQ-sentences. The proof uses Theorem 2.

Theorem 3 (Splitting Theorem) *Let F, G be GQ-sentences, and let $\mathbf{p}$, $\mathbf{q}$ be disjoint tuples of distinct predicate constants. If*

- *each strongly connected component of $\mathrm{DG}_{\mathbf{pq}}[F \wedge G]$ is a subset of $\mathbf{p}$ or a subset of $\mathbf{q}$,*
- *F is negative on $\mathbf{q}$, and*
- *G is negative on $\mathbf{p}$*

then

$$\mathrm{SM}[F \wedge G; \mathbf{pq}] \quad \textit{is equivalent to} \quad \mathrm{SM}[F; \mathbf{p}] \wedge \mathrm{SM}[G; \mathbf{q}].$$

Example 1 continued $\mathrm{SM}[F;\ \mathit{discount}, \mathit{numOfDiscount}]$ is equivalent to $\mathrm{SM}[G_1; \mathit{discount}] \wedge \mathrm{SM}[G_2; \mathit{numOfDiscount}]$, where G_1 is the conjunction of the universal closures of the first two implications in F and G_2 is the universal closure of the last implication.

Completion

A GQ-formula F is in *Clark normal form* if it is a conjunction of sentences of the form

$$\forall \mathbf{x}(G \to p(\mathbf{x})), \tag{21}$$

one for each intensional predicate p, where $\mathbf{x}$ is a list of distinct object variables, and G has no free variables other than $\mathbf{x}$. The *completion* (relative to $\mathbf{p}$) of a GQ-formula F in Clark normal form, denoted by $\mathrm{COMP}[F]$, is obtained by replacing each conjunctive term (21) with

$$\forall \mathbf{x}(p(\mathbf{x}) \leftrightarrow G).$$

We say that a GQ-formula is *tight* on $\mathbf{p}$ if its dependency graph relative to $\mathbf{p}$ is acyclic.

Theorem 4 *For any GQ-formula F in Clark normal form that is tight on $\mathbf{p}$, $\mathrm{SM}[F; \mathbf{p}]$ is equivalent to the completion of F relative to $\mathbf{p}$.*

Example 1 continued *Let F' be the formula obtained from F by dropping the second implication. The Clark normal form of F' is tight on $\{\mathit{discount}, \mathit{numOfDiscount}\}$. So $\mathrm{SM}[F_3; \mathit{discount}, \mathit{numOfDiscount}]$ equivalent to*

$$\begin{aligned}&\forall x(\mathit{discount}(x) \leftrightarrow \neg \mathit{accident}(x) \wedge \\ &\quad \#dl[\mathit{Man} \uplus \mathit{mm}, \mathit{Married} \uplus \mathit{mm}, \mathit{Woman} \uplus \mathit{mw}, \\ &\qquad\qquad \mathit{Married} \uplus \mathit{mw}; \exists \mathit{Spouse}.\top](x)) \wedge \\ &\forall y(\mathit{numOfDiscount}(y) \leftrightarrow \mathrm{COUNT}\langle x.\mathit{discount}(x)\rangle = y).\end{aligned}$$

Related Work

We refer the reader to (Eiter et al. 2005) for the semantics of HEX programs. It is not difficult to see that an external atom in a HEX program can be represented in terms of a generalized quantifier. Eiter et al. show how dl-atoms can be simulated by external atoms $\#dl[](x)$. The treatment is similar to ours in terms of generalized quantifiers. For another example, rule

$$reached(x) \leftarrow \#reach[edge, a](x)$$

defines all the vertices that are reachable from the vertex a in the graph with $edge$. The external atom $\#reach[edge, a](x)$ can be represented by a generalized quantified formula

$$Q_{reach}[x_1x_2][x_3][x_4](edge(x_1, x_2), x_3 = a, x_4 = x),$$

where Q_{reach} is as defined in Example 2.

In fact, the incorporation of generalized quantifiers in logic programming was considered earlier in (Eiter, Gottlob, and Veith 1997), but the treatment there was to simply view them like negative literals. This approach does not allow recursion through generalized quantified formulas, and often yields an unintuitive result even when we limit attention to standard quantifiers. For instance, according to (Eiter, Gottlob, and Veith 1997), program

$$p(a) \leftarrow \forall x\ p(x) \tag{22}$$

has two answer sets, $\emptyset$ and $\{p(a)\}$. The latter is "unfounded." This is not the case in the first-order stable model semantics (Ferraris, Lee, and Lifschitz 2011; Lin and Zhou 2011), which allows the standard quantifiers, but no other generalized quantifiers. According to our semantics, which properly extends the semantics from (Ferraris, Lee, and Lifschitz 2011) does not have the unintuitive answer set $\{p(a)\}$.

Conclusion

We presented the stable model semantics for formulas containing generalized quantifiers, and showed that several recent extensions of the stable model semantics can be viewed as special cases of this language. We expect that the generality of the formalism is useful in providing a principled way to study and compare the different extensions of the stable model semantics.

Acknowledgements. We are grateful to Michael Bartholomew and the anonymous referees for their useful comments. This work was partially supported by the National Science Foundation under Grant IIS-0916116.

References

[Bartholomew, Lee, and Meng, 2011] Bartholomew, M.; Lee, J.; and Meng, Y. 2011. First-order extension of the FLP stable model semantics via modified circumscription. In *Proceedings of IJCAI*, 724–730.

[Eiter et al., 2005] Eiter, T.; Ianni, G.; Schindlauer, R.; and Tompits, H. 2005. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proceedings of IJCAI*, 90–96.

[Eiter et al., 2006] Eiter, T.; Ianni, G.; Schindlauer, R.; and Tompits, H. 2006. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *Proceedings of ESWC*, 273–287.

[Eiter et al., 2008] Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence* 172(12-13):1495–1539.

[Eiter et al., 2011] Eiter, T.; Fink, M.; Ianni, G.; Krennwallner, T.; and Schüller, P. 2011. Pushing efficient evaluation of hex programs by modular decomposition. In *Proceedings of LPNMR*, 93–106.

[Eiter, Gottlob, and Veith, 1997] Eiter, T.; Gottlob, G.; and Veith, H. 1997. Modular logic programming and generalized quantifiers. In *Proceedings of LPNMR*, 290–309.

[Faber, Pfeifer, and Leone, 2011] Faber, W.; Pfeifer, G.; and Leone, N. 2011. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence* 175(1):278–298.

[Ferraris and Lifschitz, 2010] Ferraris, P., and Lifschitz, V. 2010. On the stable model semantics of firsr-order formulas with aggregates. In *Proceedings of NMR*.

[Ferraris et al., 2009] Ferraris, P.; Lee, J.; Lifschitz, V.; and Palla, R. 2009. Symmetric splitting in the general theory of stable models. In *Proceedings of IJCAI*, 797–803.

[Ferraris, Lee, and Lifschitz, 2007] Ferraris, P.; Lee, J.; and Lifschitz, V. 2007. A new perspective on stable models. In *Proceedings of IJCAI*, 372–379.

[Ferraris, Lee, and Lifschitz, 2011] Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175:236–263.

[Ferraris, 2005] Ferraris, P. 2005. Answer sets for propositional theories. In *Proceedings of LPNMR*, 119–131.

[Fink and Pearce, 2010] Fink, M., and Pearce, D. 2010. A logical semantics for description logic programs. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)*, 156–168.

[Lee and Meng, 2009] Lee, J., and Meng, Y. 2009. On reductive semantics of aggregates in answer set programming. In *Procedings of LPNMR*, 182–195.

[Lee, Lifschitz, and Palla, 2008] Lee, J.; Lifschitz, V.; and Palla, R. 2008. A reductive semantics for counting and choice in answer set programming. In *Proceedings of AAAI*, 472–479.

[Lifschitz, Pearce, and Valverde, 2001] Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2:526–541.

[Lifschitz, 2002] Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence* 138:39–54.

[Lin and Zhou, 2011] Lin, F., and Zhou, Y. 2011. From answer set logic programming to circumscription via logic of GK. *Artificial Intelligence* 175:264–277.

[Lindström, 1966] Lindström, P. 1966. First-order predicate logic with generalized quantifiers. *Theoria* 32:186–195.

[Liu et al., 2010] Liu, L.; Pontelli, E.; Son, T. C.; and Truszczynski, M. 2010. Logic programs with abstract constraint atoms: The role of computations. *Artificial Intelligence* 174(34):295 – 315.

[Marek and Truszczynski, 2004] Marek, V. W., and Truszczynski, M. 2004. Logic programs with abstract constraint atoms. In *Proceedings of AAAI*, 86–91.

[Mostowski, 1957] Mostowski, A. 1957. On a Generalization of Quantifiers. *Fundamenta Mathematicae* 44:12–35.

[Niemelä and Simons, 2000] Niemelä, I., and Simons, P. 2000. Extending the Smodels system with cardinality and weight constraints. *Logic-Based Artificial Intelligence*. Kluwer. 491–521.

[Shen, 2011] Shen, Y.-D. 2011. Well-supported semantics for description logic programs. In *Proceedings of IJCAI*, 1081–1086.

[Son and Pontelli, 2007] Son, T. C., and Pontelli, E. 2007. A constructive semantic characterization of aggregates in answer set programming. *TPLP* 7(3):355–375.

[Son, Pontelli, and Tu, 2007] Son, T. C.; Pontelli, E.; and Tu, P. H. 2007. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research* 29:353–389.

[Truszczyński, 2010] Truszczyński, M. 2010. Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. *Artificial Intelligence* 174(16-17):1285–1306.

[Westerståhl, 2008] Westerståhl, D. 2008. Generalized quantifiers. In *The Stanford Encyclopedia of Philosophy (Winter 2008 Edition)*. http://plato.stanford.edu/archives/win2008/entries/generalized-quantifiers/ .