Stable Models of Formulas with Generalized Quantifiers (Preliminary Report)

Joohyung Lee and Yunsong Meng

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

Abstract

Applications of answer set programming motivated various extensions of the stable model semantics, for instance, to allow aggregates or to facilitate interface with external ontology descriptions. We present a uniform, reductive view on these extensions by viewing them as special cases of formulas with generalized quantifiers. This is done by extending the first-order stable model semantics by Ferraris, Lee and Lifschitz to account for generalized quantifiers and then by reducing the individual extensions to this formalism.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases answer set programming, stable model semantics, generalized quantifiers

Digital Object Identifier 10.4230/LIPIcs.ICLP.2012.61

1 Introduction

Applications of answer set programming motivated various recent extensions of the stable model semantics, for instance, to allow aggregates [4, 8, 15], or to facilitate interface with external ontology descriptions [3]. While the extensions were driven by different motivations and applications, a common underlying issue is how to extend the stable model semantics to incorporate "complex atoms," such as "aggregate atoms" and "dl-atoms."

Most extensions involve grounding. For instance, assuming that the domain is $\{1, 2, \dots\}$ the rule

$$q(y) \leftarrow \#\textsc{count}\{x.p(x, y)\} \geq 2 \tag{1}$$

can be understood as a schema for ground instances

$$\begin{array}{l} q(1) \leftarrow \#\textsc{count}\{1.p(1,1), 2.p(2,1), \dots\} \geq 2 \\ q(2) \leftarrow \#\textsc{count}\{1.p(1,2), 2.p(2,2), \dots\} \geq 2 \\ \dots \end{array}$$

Here y is called a "global" variable, and x is called a "local" variable. Replacing a global variable by ground terms increases the number of rules; replacing a local variable by ground terms increases the size of each rule.

Instead of involving grounding, in [10], a simple approach to understanding the meaning of the count aggregate in answer set programming was provided by reduction to first-order formulas under the stable model semantics [6, 7]. For instance, rule (1) can be understood as the first-order formula

$$\forall y(\exists x_1 x_2(p(x_1) \wedge p(x_2) \wedge \neg(x_1 = x_2)) \rightarrow q(y)) \ ,$$

in which quantifiers are introduced to account for local variables in aggregates.

Technical Communications of the 28th International Conference on Logic Programming (ICLP'12).
Editors: A. Dovier and V. Santos Costa; pp. 61–71
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

An attempt to extend this approach to handle arbitrary nonmonotone aggregates encounters some difficulty, as the quantifiers $\forall$ and $\exists$, like its propositional counterpart $\wedge$ and $\vee$, are "monotone."

It is hinted in [5] that aggregates may be viewed in terms of generalized quantifiers—a generalization of the standard quantifiers, $\forall$ and $\exists$, introduced by Mostowski [13]. We follow up on that suggestion, and extend the stable model semantics by [7] to allow generalized quantifiers.

It turns out that generalized quantifiers are not only useful in explaining the meaning of arbitrary aggregates, but also useful in explaining other recent extensions of the stable model semantics, such as nonmonotonic dl-programs [3]. This allows us to combine the 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, the ontology description $\mathcal{T}$ specifies that every married man has a spouse who is a woman, and similarly for a married woman:

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

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

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

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

The paper is organized as follows. We first review the syntax and the semantics of formulas with generalized quantifiers (GQ-formulas). Next we define stable models of GQ-formulas, and then show the individual extensions of the stable model semantics, such as logic programs with aggregates and/or nonmonotonic dl-atoms, can be viewed as special cases of GQ-formulas.

2 Preliminaries

2.1 Syntax of Formulas with Generalized Quantifiers

We follow the definition of a GQ-formula from [16, Section 5] (that is to say, with Lindström quantifiers [12] 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 $F_1, \ldots, F_k$ ($k \geq 0$) are GQ-formulas and Q is a generalized quantifier of type $\langle n_1, \ldots, n_k \rangle$ in $\mathbf{Q}$, then

$$Q[\mathbf{x}_1] \ldots [\mathbf{x}_k](F_1(\mathbf{x}_1), \ldots, F_k(\mathbf{x}_k)) \tag{2}$$

  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 $Q[\mathbf{x}_1] \ldots [\mathbf{x}_k](F_1(\mathbf{x}_1), \ldots, F_k(\mathbf{x}_k))$ 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 *GQ-sentence* is a GQ-formula with no free variables. Notice that the distinction between free and bound variables is similar to that of global and local variables informally described in the introduction.

We assume that $\mathbf{Q}$ contains a type $\langle\rangle$ quantifier $Q_\perp$, 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 propositional connectives and quantifiers, $\perp, \neg, \wedge, \vee, \rightarrow, \forall, \exists$. 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 $Q_\forall[x](F(x))$.

2.2 Models of GQ-Formulas

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 $\langle n_1, \ldots, n_k \rangle$, 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 propositional 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; [1]
- 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

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

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

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

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

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

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

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 propositional connectives and quantifiers Q, functions Q^U have the standard meaning:

- $Q^U_\forall(R) = \mathbf{t}$ iff $R = U$; $\quad Q^U_\exists(R) = \mathbf{t}$ iff $R \cap U \neq \emptyset$;
- $Q^U_\wedge(R_1, R_2) = \mathbf{t}$ iff $R_1 = R_2 = \{\epsilon\}$;[2] $\quad Q^U_\vee(R_1, R_2) = \mathbf{t}$ 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 GQ-sentence F, or is a *model* of F, and write $I \models F$, if $F^I = \mathbf{t}$. 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 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 GQ-sentence $F = \neg Q_{\leq 2}[x]\ p(x) \rightarrow Q_{majority}[y]\ p(y)$ ("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 universe U: if $Q^U(R_1, \ldots, R_k) = \mathbf{t}$ and $R_i \subseteq R'_i \subseteq U^{n_i}$, then $Q^U(R_1, \ldots, R_{i-1}, R'_i, R_{i+1}, \ldots, R_k) = \mathbf{t}$. Similarly, we say that Q is *anti-monotone in the i-th argument position* if the following holds for any universe U: if $Q^U(R_1, \ldots, R_k) = \mathbf{t}$ and $R'_i \subseteq R_i \subseteq U^{n_i}$, then $Q^U(R_1, \ldots, R_{i-1}, R'_i, R_{i+1}, \ldots, R_k) = \mathbf{t}$. We call an argument position of Q *monotone (anti-monotone)* if Q is monotone (anti-monotone) in that argument position.

Let M be a subset of $\{1, \ldots, k\}$. 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 $\{1\}$.

Predicate variables can be added to the language in the usual way as we define the standard second-order logic. Syntactically, n-ary predicate variables are used to form atomic formulas in the same way as n-ary predicate constants. Semantically, these variables range over arbitrary truth-valued functions on U^n.

3 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 $p_1, \ldots, p_n$, and let $\mathbf{u}$ be a list of distinct predicate

[2] 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 $F^I = G^I = \mathbf{t}$.

variables $u_1, \ldots, u_n$. By $\mathbf{u} \leq \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \to p_i(\mathbf{x}))$ for all $i = 1, \ldots, n$, 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

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

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

$$F \wedge \neg\exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u})), \tag{3}$$

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 $\mathbf{p}$;
- $$\begin{aligned}&(Q[\mathbf{x}_1] \ldots [\mathbf{x}_k](F_1(\mathbf{x}_1), \ldots, F_k(\mathbf{x}_k)))^* = \\ &\qquad Q[\mathbf{x}_1] \ldots [\mathbf{x}_k](F_1^*(\mathbf{x}_1), \ldots, F_k^*(\mathbf{x}_k)) \wedge Q[\mathbf{x}_1] \ldots [\mathbf{x}_k](F_1(\mathbf{x}_1), \ldots, F_k(\mathbf{x}_k)).\end{aligned} \tag{4}$$

When F is a GQ-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 $Q[\mathbf{x}_1] \ldots [\mathbf{x}_k](F_1(\mathbf{x}_1), \ldots, F_k(\mathbf{x}_k))$ be a GQ-formula and let M be a subset of $\{1, \ldots, k\}$ such that every predicate constant from $\mathbf{p}$ occurs in some F_j where $j \in M$.

(a) If Q is monotone in M, then

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

is logically valid.

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

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

is logically valid.

Proposition 1 allows us to simplify the formula $F^*(\mathbf{u})$ in (3) without affecting the models of (3). In formula (4), 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 (4) can be equivalently rewritten as

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

This is almost the same as the definition of F^* given in [7], except for the case $(\neg F)^*$, which is a bit more concise.[3] The only propositional connective which is neither monotone nor anti-monotone in all argument positions is $Q_{\rightarrow}$, for which the simplification does not apply.

Example 3 continued. *For the GQ-sentence F considered earlier,* $\mathrm{SM}[F]$ *is*

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

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

$$\neg Q_{\leq 2}[x]\ p(x) \rightarrow\ Q_{majority}[y]\ u(y). \tag{6}$$

I_1 considered earlier satisfies (5): it satisfies F, and, for any proper "subset" u of p, it satisfies the antecedent of (6) 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 (5), and is not a stable model.

4 Aggregates as GQ-Formulas

4.1 Formulas with Aggregates

The following definition of a formula with aggregates is from [5], which extends the one from [9] to allow nested aggregates. By a *number* we understand an element of some fixed set **Num**. For example, **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 the *multiplicity*, to each of its elements. An *aggregate function* is a partial function from the class of multisets to **Num**. We assume the presence of some fixed background signature σ_{bg} that contains all numbers. Furthermore, we assume that the interpretation I_{bg} of the background signature is fixed, and interpretes each number as itself.

We consider a signature σ as a superset of σ_{bg}. 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 all the constants in σ_{bg}.

First-order formulas with aggregates are defined as an extension of standard first-order formulas by adding the following clause:

-

$$\mathrm{OP}\langle \mathbf{x}_1.F_1, \ldots, \mathbf{x}_n.F_n\rangle \succeq b \tag{7}$$

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

- OP is a symbol for an *aggregate function* (not from σ);
- $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are nonempty lists of distinct object variables;
- $F_1, \ldots, F_n$ 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.

[3] $\neg F$ is understood as $F \rightarrow \bot$ in [7], but this difference does not affect stable models. When $\neg$ is a primitive propositional connective as above,

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

is logically valid.

4.2 Aggregates as GQ-Formulas

Due to the space limit, we refer the reader to [5] for the stable model semantics of formulas with aggregates. We can explain their semantics by viewing it as a special case of the stable model semantics presented here. Following [5], for any set X of n-tuples ($n \geq 1$), let $msp(X)$ ("the multiset projection of X") be the multiset consisting of all ξ_1 such that $(\xi_1, \ldots, \xi_n) \in X$ for at least one $(n-1)$-tuple $(\xi_2, \ldots, \xi_n)$, 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 (7) with the GQ-formula

$$Q_{(\mathrm{OP},\succeq)}[\mathbf{x}_1]\ldots[\mathbf{x}_n][y](F_1(\mathbf{x}_1),\ldots,F_n(\mathbf{x}_n), y = b)\ , \tag{8}$$

where, for any interpretation I, $Q^U_{(\mathrm{OP},\succeq)}$ 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 $Q^U_{(\mathrm{OP},\succeq)}(R_1,\ldots,R_n,R_{n+1}) = \mathbf{t}$ iff

- $\mathrm{OP}(\alpha)$ is defined, where α is the join of the multisets $msp(R_1),\ldots,msp(R_n)$,
- $R_{n+1} = \{n\}$, where n is an element of **Num**, and
- $\mathrm{OP}(\alpha) \succeq n$.

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

$$\begin{aligned} &discount(alice) \wedge discount(carol) \\ &\quad \wedge \forall z(\mathrm{COUNT}\langle x.discount(x)\rangle = z \rightarrow numOfDiscounts(z)). \end{aligned}$$

The following proposition states that this definition is equivalent to the definition from [5].

▶ 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 [5] iff I is a $\mathbf{p}$-stable model of F in our sense.

5 Nonmonotonic dl-Programs as GQ-Formulas

5.1 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* [3] is a pair $(\mathcal{T}, \Pi)$, where $\mathcal{T}$ is a theory in description logic 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 $\mathcal{T}$ using "dl-atoms." A *dl-atom* is of the form

$$DL[S_1 op_1 p_1, \ldots, S_k op_k p_k;\ Query](\mathbf{t}) \quad (k \geq 0), \tag{9}$$

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 [3]. A *dl-rule* is of the form

$$a \ \leftarrow\ b_1, \ldots, b_m, not\ b_{m+1}, \ldots, not\ b_n\ , \tag{10}$$

where a is an atom and each b_i is either an atom or a dl-atom. We identify rule (10) with

$$a \ \leftarrow\ B, N\ , \tag{11}$$

where B is $b_1, \ldots, b_m$ and N is $not\ b_{m+1}, \ldots, not\ b_n$. 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 (9) *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 $I \subseteq I'$ and $I \models_{\mathcal{T}} A$, we have that $I' \models_{\mathcal{T}} A$. Similarly, A is *anti-monotonic* relative to $\mathcal{T}$ if, for any two Herbrand interpretations I and I' such that $I' \subseteq I$ and $I \models_{\mathcal{T}} A$, we have that $I' \models_{\mathcal{T}} A$.

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

$$a \leftarrow B' \tag{12}$$

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 (12), 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 transforms is whether monotonic dl-atoms remain in the positive body or not. Both transforms do not retain nonmonotonic dl-atoms.

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 $w\Pi^I_{\mathcal{T}}(s\Pi^I_{\mathcal{T}}, \mathit{respectively})$.

5.2 Nonmonotonic dl-program as GQ-Formulas

We can view 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 (9) in Π with

$$Q_{(9)}[\mathbf{x}_1] \ldots [\mathbf{x}_k](p_1(\mathbf{x}_1), \ldots, p_k(\mathbf{x}_k)) \tag{13}$$

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

$$\neg\neg Q_{(9)}[\mathbf{x}_1] \ldots [\mathbf{x}_k](p_1(\mathbf{x}_1), \ldots, p_k(\mathbf{x}_k)) \tag{14}$$

otherwise. Since $\neg$ is an anti-monotone GQ, prepending $\neg\neg$ in front of the quantified formula in (14) means that, under the strong answer set semantics, every nonmonotonic dl-atom is understood in terms of an anti-monotone GQ.

Given an interpretation I, $Q^U_{(9)}$ 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, $Q^U_{(9)}(R_1, \ldots, R_k) = \mathbf{t}$ 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 $\mathrm{SM}[\Pi; P_\Pi]$.

Similarly, a weak answer set of $(\mathcal{T}, \Pi)$ is defined by identifying every dl-atom (9) in Π with (14) regardless of A being monotonic or not. This means that, under the weak answer set semantics, every dl-atom is understood in terms of an anti-monotone GQ.

Example 1 continued. *The dl-atom*

$$\#dl[\mathit{Man} \uplus \mathit{mm}, \mathit{Married} \uplus \mathit{mm}, \mathit{Woman} \uplus \mathit{mw}, \mathit{Married} \uplus \mathit{mw}; \exists \mathit{Spouse}.\top](\mathit{alice}) \tag{15}$$

is identified with the generalized quantified formula

$$Q_{(15)}[x_1][x_2][x_3][x_4](mm(x_1), mm(x_2), mw(x_3), mw(x_4)) \tag{16}$$

where, for any interpretation I, $Q^U_{(15)}$ 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 $Q^U_{(15)}(R_1,R_2,R_3,R_4)=\mathbf{t}$ iff $\mathcal{T}\cup\{Man(c)\mid c\in R_1\}\cup\{Woman(c)\mid c\in R_3\}\cup\{Married(c)\mid c\in R_2\cup R_4\}$ entails $\exists x Spouse(alice,x)$.

Consider an Herbrand interpretation $I=\{mw(alice)\}$, which satisfies (15). I also satisfies (16) 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 reformulations of the original definitions from [3].

▶ Proposition 3. 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 [3] iff it is a strong (weak, respectively) answer set of $(\mathcal{T},\Pi)$ in our sense.

5.3 Another Semantics of Nonmonotonic dl-programs

Shen [14] notes that both strong and weak answer set semantics suffer from circular justifications.

▶ **Example 5.** [14] Consider $(\mathcal{T},\Pi)$, where $\mathcal{T}=\emptyset$ and Π is the program

$$p(a)\leftarrow \#dl[c\uplus p, b\cap q;\ c\sqcap\neg b](a)\ , \tag{17}$$

in which the dl-atom is neither monotonic nor anti-monotonic. This dl-program has two strong (weak, respectively) answer sets: $\emptyset$ and $\{p(a)\}$. According to [14], 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, $s\Pi^{\{p(a)\}}_{\mathcal{T}}$ ($w\Pi^{\{p(a)\}}_{\mathcal{T}}$, respectively) is $p(a)\leftarrow$, and $\{p(a)\}$ is its minimal model.

As we hinted in the previous section, this kind of circular justifications is related to the treatment that understands every nonmonotonic dl-atom in terms of an anti-monotone GQ, regardless of the nonmonotonic dl-atom's being anti-monotonic or not. In this case, in view of Proposition 1, predicates in a nonmonotonic dl-atom are exempt from the minimality checking. This is different from how we treat nonmonotone aggregates, where we simply identify them with nonmonotone GQs. This observation suggests the following alternative semantics of dl-programs, in which we understand only anti-monotonic dl-atoms in terms of anti-monotone GQs, unlike in the strong and the weak answer set semantics. We say that an Herbrand interpretation I is an *answer set* of $(\mathcal{T},\Pi)$ if I satisfies $\mathrm{SM}[\Pi;P_\Pi]$, where we simply identify every dl-atom (9) in Π with (13).

This definition of an answer set has a reduct-based characterization as well. Just like we form a strong dl-transform, we first remove the negative body, but instead of removing all nonmonotonic dl-atoms in the positive body, we remove only anti-monotonic dl-atoms from the positive body. In other words, the *reduct* of Π relative to $\mathcal{T}$ and an Herbrand interpretation I of $\langle C,P_\Pi\rangle$, denoted by $\Pi^I_{\mathcal{T}}$, is the set of rules (12), 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. The following proposition shows that this modified definition of a reduct can capture the new answer set semantics of dl-programs.

▶ Proposition 4. For any dl-program $(\mathcal{T}, \Pi)$ and any Herbrand interpretation I of $\langle C, P_\Pi \rangle$, I is an answer set of $(\mathcal{T}, \Pi)$ according to the new definition iff I is minimal among the sets of atoms that satisfy $\Pi^I_{\mathcal{T}}$.

The new semantics does not have the circular justification problem described in Example 5.

Example 5 continued. *$\{p(a)\}$ is not an answer set of $(\mathcal{T}, \Pi)$ according to the new definition. The reduct $\Pi^{\{p(a)\}}_{\mathcal{T}}$ is (17) itself retaining the dl-atom unlike under the strong and the weak answer set semantics. We check that $\emptyset$, a proper subset of $\{p(a)\}$, satisfies it, which means that $\{p(a)\}$ is not an answer set.*

6 Related Work

We refer the reader to [2] 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

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

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

In fact, incorporation of generalized quantifiers in logic programming was considered earlier in [1], but the treatment there was not satisfactory because they understood generalized quantifiers simply as anti-monotone GQs in our sense. Without going into detail, this amounts to modifying our definition of F^* as

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

This approach does not allow recursion through generalized quantified formulas, and often yields an unintuitive result. According to [1], program $p(a) \leftarrow \forall x\ p(x)$ has two answer sets, $\emptyset$ and $\{p(a)\}$. The latter is "unfounded." This is not the case with the semantics that we introduced in this note. According to our semantics, which properly extends the semantics from [7], $\{p(a)\}$ is not an answer set.

7 Conclusion

We presented the stable model semantics for formulas containing generalized quantifiers, and showed that some recent extensions of the stable model semantics with "complex atoms" can be viewed as special cases of this formalism. We expect that the generality of the formalism will be useful in providing a principled way to study and compare the different extensions of the stable model semantics. As we observed, distinguishing among monotone, anti-monotone, and neither monotone nor anti-monotone GQs is essential in defining the semantics of such extensions, whereas the last group of GQs was not considered in the traditional stable model semantics.

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

References

1 Thomas Eiter, Georg Gottlob, and Helmut Veith. Modular logic programming and generalized quantifiers. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 290–309, 1997.

2 Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 90–96, 2005.

3 Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.

4 Wolfgang Faber, Gerald Pfeifer, and Nicola Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298, 2011.

5 Paolo Ferraris and Vladimir Lifschitz. On the stable model semantics of firsr-order formulas with aggregates. In *Proceedings of International Workshop on Nonmonotonic Reasoning (NMR)*, 2010.

6 Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A new perspective on stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 372–379, 2007.

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

8 Paolo Ferraris. Answer sets for propositional theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 119–131, 2005.

9 Joohyung Lee and Yunsong Meng. On reductive semantics of aggregates in answer set programming. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 182–195, 2009.

10 Joohyung Lee, Vladimir Lifschitz, and Ravi Palla. A reductive semantics for counting and choice in answer set programming. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 472–479, 2008.

11 Fangzhen Lin and Yi Zhou. From answer set logic programming to circumscription via logic of GK. *Artificial Intelligence*, 175:264–277, 2011.

12 Per Lindström. First-order predicate logic with generalized quantifiers. *Theoria*, 32:186–195, 1966.

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

14 Yi-Dong Shen. Well-supported semantics for description logic programs. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1081–1086, 2011.

15 Tran Cao Son and Enrico Pontelli. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming*, 7(3):355–375, 2007.

16 Dag Westerståhl. Generalized quantifiers. In *The Stanford Encyclopedia of Philosophy (Winter 2008 Edition)*. 2008. `http://plato.stanford.edu/archives/win2008/entries/generalized-quantifiers/`.