On Reductive Semantics of Aggregates in Answer Set Programming

Joohyung Lee and Yunsong Meng

Computer Science and Engineering
Arizona State University, Tempe, AZ, USA
{joolee, Yunsong.Meng}@asu.edu

Abstract. Several proposals of the semantics of aggregates are based on different extensions of the stable model semantics, which makes it difficult to see how they are related to each other. In this note, building upon a reductive approach to designing aggregates, we provide some reformulations of the existing semantics in terms of propositional formulas, which help us compare the semantics and understand their properties by turning to their propositional formula representations. We also present a generalization of semantics of aggregates without involving grounding, and define loop formulas for programs with aggregates guided by the reductive approach.

1 Introduction

Defining a reasonable semantics of aggregates under the stable model semantics has turned out to be a non-trivial task. An obvious attempt to identify an aggregate with a nested expression [1] in the form of disjunctions over conjunctions leads to unintuitive results. For instance, one would expect $\{p(0), p(1)\}$ to be the only answer set of the following program.

$$p(1) \qquad\qquad p(0) \leftarrow \text{SUM}\langle\{x : p(x)\}\rangle = 1.$$

Assuming that the domain is $\{0, 1\}$, one may try to identify $\text{SUM}\langle\{x : p(x)\}\rangle = 1$ with the disjunction over two "solutions,"—one in which only $p(1)$ is true, and the other in which both $p(0)$ and $p(1)$ are true—each of which makes the aggregate true. However, the resulting program

$$p(1) \qquad\qquad p(0) \leftarrow (p(0), p(1)) \ ; \ (\mathit{not}\ p(0), p(1))$$

has no answer sets.[1]

The difficulty has led to several interesting extensions of the stable model semantics, such as an extended definition of reduct that accounts for aggregates [2], an extension of the stable model semantics to arbitrary propositional formulas [3] which is essentially a reformulation of the equilibrium logic [4], an extension of T_P operator with "conditional" satisfaction [5]. On the other hand, the semantics from [6] is in

[1] One might also wonder about dropping negative literals in forming each conjunct, but this does not work either. For instance, consider $p(0) \leftarrow \text{SUM}\langle\{x : p(x)\}\rangle \neq 0$, which intuitively has no answer sets.

terms of a translation that turns an aggregate into a nested expression, it involves somewhat sophisticated notion of "local power set".

Since the semantics of aggregates are based on different extensions of the stable model semantics, it is not clear to see how they are related to each other. While most of them agree on monotone and anti-monotone aggregates, they have subtle differences in understanding arbitrary aggregates which are neither monotone nor anti-monotone. For example, consider the following program Π_1.

$$\begin{array}{rl} & p(2) \\ p(-1) & \leftarrow \text{SUM}\langle\{x : p(x)\}\rangle \geq 2 \\ p(1) & \leftarrow \text{SUM}\langle\{x : p(x)\}\rangle \leq 2. \end{array} \tag{1}$$

According to [2; 3], the program has one answer set, $\{p(-1), p(1), p(2)\}$, while it has no answer sets according to [5; 6].

In this note we study their relationships by turning each of them into a uniform framework of propositional formulas. Among many semantics, we limit our discussions to some representative semantics: by Ferraris [3], by Faber, Leone and Pfeifer [2], by Son, Pontelli and Tu [5], and by Pelov, Denecker and Bruynooghe [6]. We do not consider weight constraints [7] since they do not cover arbitrary aggregates. It is shown in [8] that they can be turned into nested expressions.

Among the semantics above, the semantics from [3] and the semantics from [6] are defined in terms of propositional formulas (under the stable model semantics). Here we present a few alternative characterizations following the style of each other definition. Furthermore we show that the semantics from [2] can also be reformulated in terms of propositional formulas. Such uniform characterization helps us compare the semantics and understand their properties by turning to their propositional formula representations.

We also provide a simple reductive approach to understanding the meaning of aggregates as a straightforward extension of the stable model semantics from [9] to take into account aggregates. Unlike the other semantics, this definition applies to non-ground programs without involving grounding, so that it allows non-Herbrand models as well. We also define loop formulas for programs with aggregates under the semantics from [3].

2 Background

2.1 Answer sets of First-Order Formulas

We review the definition of an answer set from [9]. Let $\mathbf{p}$ be a list of predicate constants $p_1, \dots, p_n$, and let $\mathbf{u}$ be a list of distinct predicate variables $u_1, \dots, u_n$ of the same length as $\mathbf{p}$. By $\mathbf{u} = \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \leftrightarrow p_i(\mathbf{x}))$, where $\mathbf{x}$ is a list of distinct object variables of the same length as the arity of p_i. By $\mathbf{u} \leq \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))$ for all $i = 1, \dots n$, and $\mathbf{u} < \mathbf{p}$ stands for $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{u} = \mathbf{p})$.

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

$$F \wedge \neg \exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u})), \tag{2}$$

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

- $p_i(t_1, \ldots, t_m)^* = u_i(t_1, \ldots, t_m)$;
- $(t_1 = t_2)^* = (t_1 = t_2)$;
- $\bot^* = \bot$;
- $(F \odot G)^* = (F^* \odot G^*) \wedge (F \odot G)$ where $\odot \in \{\wedge, \vee, \rightarrow\}$;
- $(QxF)^* = QxF^* \wedge QxF$ where $Q \in \{\forall, \exists\}$.

(There is no clause for negation here, because we treat $\neg F$ as shorthand for $F \rightarrow \bot$.)

Let $\sigma(F)$ be the signature consisting of the object, function and predicate constants occurring in F. According to [9], an interpretation of $\sigma(F)$ that satisfies SM[F] is called a *stable model* of F. If F contains at least one object constant, an Herbrand stable model of F is called an *answer set* of F.[2] The answer sets of a logic program Π are defined as the answer sets of the FOL-representation of Π (i.e., the conjunction of the universal closure of implications corresponding to the rules). It turns out that this definition, applied to the syntax of logic programs, is equivalent to the traditional definition of answer sets based on grounding and fixpoint construction [9]. If F is a propositional formula, [9] shows that the definition of a stable model above is equivalent to the definition of a stable model in terms of an extension of reduct given in [3].

Note that for any non-atomic formula F, formula $F^*(\mathbf{u})$ yields the conjunction of two formulas, one is F itself and the other is obtained from F by replacing every maximal proper subformula G of F with G^*. This definition of $F^*(\mathbf{u})$ is slightly different from the one from [9], but it is easy to check that the definition of a stable model remains equivalent. [3]

2.2 Syntax of a Program with Aggregates

An *aggregate function* is a function that maps multisets of objects into numbers, such as *count*, *sum*, *times*, *min* and *max*. For this paper we assume that all numbers are integers. The domain of an aggregate function is defined as usual. For instance, *sum*, *min* and *max* are defined for multi-sets of numbers; *min* and *max* do not allow the empty set in their domains.

An *aggregate expression* is of the form

$$\textsc{op}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b \tag{3}$$

where

- OP is a symbol for an *aggregate function op*;
- $\mathbf{x}$ is a nonempty list of object variables;
- $F(\mathbf{x})$ is an arbitrary quantifier-free formula;

[2] Recall that an *Herbrand interpretation* of a signature σ (containing at least one object constant) is an interpretation of σ such that its universe is the set of all ground terms of σ, and every ground term represents itself. An Herbrand interpretation can be identified with the set of ground atoms (not containing equality) to which it assigns the value *true*.

[3] This reformulation is to motivate an approach to understanding aggregates in Section 4.1.

– $\succeq$ is a symbol for a binary relation over integers, such as $\leq$, $\geq$, $<$, $>$, $=$, $\neq$;
– b is an integer constant.

A *rule (with aggregates)* is an expression of the form

$$A_1 \ ; \ldots ; \ A_l \leftarrow \ E_1, \ldots, E_m, \mathit{not}\ E_{m+1}, \ldots, \mathit{not}\ E_n \tag{4}$$

($l \geq 0$; $n \geq m \geq 0$), where each A_i is an atomic formula (possibly containing equality) and each E_i is an atomic formula or an aggregate expression. A *program (with aggregates)* is a finite set of rules.

Throughout this paper, unless otherwise noted (e.g., Section 4.1), we assume that the program contains no function constants of positive arity. We do not consider symbols OP and $\succeq$ as part of the signature.

We say that an occurrence of a variable v in a rule (4) is *bound* if the occurrence is in an aggregate expression (3) such that v is in $\mathbf{x}$; otherwise it is *free*. We say that v is *free* in the rule if some occurrence of v is free in it. Given a program Π, by $\sigma(\Pi)$ we mean the signature consisting of object and predicate constants that occur in Π. By $\mathit{Ground}(\Pi)$ we denote the program obtained from Π by grounding, i.e., replacing every free occurrence of variables with every object constant in $\sigma(\Pi)$.

2.3 Review: FLP Semantics

The FLP semantics [2] is based on an alternative definition of a reduct and the notion of satisfaction extended to aggregate expressions. Let Π be a program such that $\sigma(\Pi)$ contains at least one object constant.[4] We consider Herbrand interpretations of $\sigma(\Pi)$ only. Consider any aggregate expression (3) occurring in $\mathit{Ground}(\Pi)$. Let S be the multiset consisting of all $\mathbf{c}^1$ in the Herbrand universe where [5]

– $\mathbf{c}$ is a list of object constants of $\sigma(\Pi)$ whose length is the same as the length of $\mathbf{x}$, and
– I satisfies $F(\mathbf{c})$.

A set X of ground atoms of $\sigma(\Pi)$ satisfies the aggregate expression if S is in the domain of op, and $op(S) \succeq b$.

Let X be a set of ground atoms of signature $\sigma(\Pi)$. The *FLP reduct* of Π relative to X is obtained from $\mathit{Ground}(\Pi)$ by removing every rule whose body is not satisfied by X. Set X is an *FLP answer set* of Π if it is minimal among the sets of atoms that satisfy the FLP reduct of Π relative to X. For example, in program Π_1 (Section 1), the FLP reduct of Π_1 relative to $\{p(-1), p(1), p(2)\}$ is Π_1 itself. Set $\{p(-1), p(1), p(2)\}$ is minimal among the sets of atoms that satisfy the reduct, and thus is an FLP-answer set of Π_1.

2.4 Review: Ferraris Semantics

Ferraris semantics [3] is to understand an aggregate as an abbreviation of a propositional formula under the stable model semantics (Section 2.1) in the form of conjunctions over implications. The semantics was given for the propositional case, which can be extended to allow variables as follows.

[4] The syntax in [2] requires $F(\mathbf{x})$ be a conjunction of atoms.
[5] Given a list $\mathbf{t}$ of object constants or variables, by $\mathbf{t}^1$ we represent the first element of $\mathbf{t}$.

Notation: Given a multiset of object constants $\{\!\{c_1, \ldots, c_n\}\!\}$,

$$\text{OP}\langle\{\!\{c_1, \ldots, c_n\}\!\}\rangle \succeq b$$

if

- b is an integer constant,
- multiset $\{\!\{c_1, \ldots, c_n\}\!\}$ is in the domain of op, and
- $op(\{\!\{c_1, \ldots, c_n\}\!\}) \succeq b$.

$\text{OP}\langle\{\!\{c_1, \ldots, c_n\}\!\}\rangle \not\succeq b$ if it is not the case that $\text{OP}\langle\{\!\{c_1, \ldots, c_n\}\!\}\rangle \succeq b$.

Let $E = \text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b$ be an aggregate expression occurring in $\mathit{Ground}(\Pi)$, let $\mathbf{O}_\Pi(E)$ be the set of all lists of object constants of $\sigma(\Pi)$ whose length is the same as $|\mathbf{x}|$, and let $\overline{\mathcal{C}}(E)$ be the set of all subsets $\mathbf{C}$ of $\mathbf{O}_\Pi(E)$ such that $\text{OP}\langle\{\!\{\mathbf{c}^1 : \mathbf{c} \in \mathbf{C}\}\!\}\rangle \not\succeq b$. For instance, in program Π_1 (Section 1), let E_1 be $\text{SUM}\langle\{x : p(x)\}\rangle \geq 2$. Set $\mathbf{O}_{\Pi_1}(E_1)$ is $\{-1, 1, 2\}$, and $\overline{\mathcal{C}}(E_1)$ is $\{\emptyset, \{-1\}, \{1\}, \{-1, 1\}, \{-1, 2\}\}$. Similarly, for $E_2 = \text{SUM}\langle\{x : p(x)\}\rangle \leq 2$, set $\overline{\mathcal{C}}(E_2)$ is $\{\{1, 2\}\}$.

By $\mathit{Fer}_\Pi(E)$ we denote

$$\bigwedge_{\mathbf{C} \in \overline{\mathcal{C}}(E)} \Big(\bigwedge_{\mathbf{c} \in \mathbf{C}} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_\Pi(E) \setminus \mathbf{C}} F(\mathbf{c}) \Big). \tag{5}$$

For instance, $\mathit{Fer}_{\Pi_1}(E_1)$ is

$$\begin{array}{l} (p(-1) \vee p(1) \vee p(2)) \ \wedge \ (p(-1) \rightarrow p(1) \vee p(2)) \ \wedge \ (p(1) \rightarrow p(-1) \vee p(2)) \\ \quad \wedge (p(-1) \wedge p(1) \rightarrow p(2)) \ \wedge \ (p(-1) \wedge p(2) \rightarrow p(1)) \end{array}$$

By $\mathit{Fer}(\Pi)$ we denote propositional formula obtained from $\mathit{Ground}(\Pi)$ by replacing every aggregate expression E in it by $\mathit{Fer}_\Pi(E)$. The *Ferraris answer sets* of Π are defined as the answer sets of $\mathit{Fer}(\Pi)$ in the sense of Section 2.1. For example, the Ferraris answer sets of Π_1 are the answer sets of the following formula $\mathit{Fer}(\Pi_1)$: [6]

$$\begin{array}{l} p(2) \ \wedge \ (\underline{[(p(-1) \vee p(1) \vee p(2)) \wedge (p(-1) \rightarrow p(1) \vee p(2)) \wedge (p(1) \rightarrow p(-1) \vee p(2))} \\ \qquad \underline{\wedge (p(-1) \wedge p(1) \rightarrow p(2)) \wedge (p(-1) \wedge p(2) \rightarrow p(1))]} \rightarrow p(-1)) \\ \quad \wedge \ (\underline{[p(1) \wedge p(2) \rightarrow p(-1)]} \rightarrow p(1)). \end{array} \tag{6}$$

One can check that this formula has only one answer set $\{p(-1), p(1), p(2)\}$.

2.5 Review: SPT-PDB Semantics

Son and Pontelli [10] present two equivalent definitions of aggregates, one in terms of "unfolding" into nested expressions, and the other in terms of "conditional satisfaction." The latter notion was simplified in Son, Pontelli and Tu [5]. Lemma 6 from [10] shows that these definitions are equivalent to the definition by Pelov, Denecker, Bruynooghe [11], which is in terms of translation into nested expressions. Thus we group them together and review only the last definition.[7]

Under the SPT-PDB semantics an aggregate can be identified with a nested expression in the form of disjunctions over conjunctions, but unlike the naive attempt given in the introduction, it involves the notion of a "(maximal) local power set."

[6] We underline the parts of a formula that correspond to aggregates.

[7] We ignore some differences in the syntax, and allow disjunctions in the head.

Given a set A of some sets, a pair $\langle B, T\rangle$ where $B, T \in A$ and $B \subseteq T$ is called a *local power set (LPS)* of A if every S such that $B \subseteq S \subseteq T$ belongs to A as well. A local power set is called *maximal* if there is no other local power set $\langle B', T'\rangle$ of A such that $B' \subseteq B$ and $T \subseteq T'$.

Let $E = \text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b$ be an aggregate expression occurring in $\mathit{Ground}(\Pi)$, let HU_Π be the set of all ground atoms that can be constructed from $\sigma(\Pi)$. Let $\mathcal{I}_\Pi(E)$ be the set of all Herbrand interpretations I of $\sigma(\Pi)$ such that $I \models E$ (satisfaction as defined in Section 2.3). For instance, in Example Π_1, HU_{Π_1} is $\{p(-1), p(1), p(2)\}$, $\mathcal{I}_{\Pi_1}(E_1)$ is $\{\{p(2)\}, \{p(1), p(2)\}, \{p(-1), p(1), p(2)\}\}$ and $\mathcal{I}_{\Pi_1}(E_2)$ is

$$\{\emptyset, \{p(-1)\}, \{p(1)\}, \{p(2)\}, \{p(-1), p(1)\}, \{p(-1), p(2)\}, \{p(-1), p(1), p(2)\}\}.$$

The maximal local power sets of $\mathcal{I}_{\Pi_1}(E_1)$ are

$$\langle\{p(2)\}, \{p(1), p(2)\}\rangle, \quad \langle\{p(1), p(2)\}, \{p(-1), p(1), p(2)\}\rangle,$$

and the maximal local power sets of $\mathcal{I}_{\Pi_1}(E_2)$ are

$$\langle\emptyset, \{p(-1), p(1)\}\rangle, \quad \langle\emptyset, \{p(-1), p(2)\}\rangle, \quad \langle\{p(-1)\}, \{p(-1), p(1), p(2)\}\rangle.$$

For any aggregate expression E occurring in $\mathit{Ground}(\Pi)$, by $\mathit{SPT\text{-}PDB}_\Pi(E)$ we denote

$$\bigvee_{\langle B,T\rangle \text{ is a maximal LPS of } \mathcal{I}_\Pi(E)} \Big(\bigwedge_{A \in B} A \wedge \bigwedge_{A \in HU_\Pi \setminus T} \neg A \Big). \tag{7}$$

For instance, $\mathit{SPT\text{-}PDB}_{\Pi_1}(E_1)$ is $(p(2) \wedge \neg p(-1)) \vee (p(1) \wedge p(2))$.

The SPT-PDB semantics eliminates the negation in front of an aggregate expression using an equivalent transformation. Let $\mathit{Pos}(\Pi)$ be a program obtained from Π by replacing *not* E_i in each rule (4) where $E_i = \text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b$ with $\text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \not\succeq b$. Clearly, $\mathit{Pos}(\Pi)$ contains no negation in front of aggregate expressions.

By $\mathit{SPT\text{-}PDB}(\Pi)$ we denote the propositional formula obtained from $\mathit{Ground}(\mathit{Pos}(\Pi))$ by replacing all aggregate expressions E in it by $\mathit{SPT\text{-}PDB}_\Pi(E)$. The *SPT-PDB* answer sets of Π are defined as the answer sets of $\mathit{SPT\text{-}PDB}(\Pi)$ in the sense of Section 2.1. For example, the *SPT-PDB* answer sets of Π_1 are the answer sets of the following formula $\mathit{SPT\text{-}PDB}(\Pi_1)$:

$$\begin{array}{l} p(2) \ \wedge \ (\underline{[(p(2) \wedge \neg p(-1)) \vee (p(1) \wedge p(2))]} \rightarrow p(-1)) \\ \wedge \ (\underline{[\neg p(2) \vee \neg p(1) \vee p(-1)]} \rightarrow p(1)). \end{array} \tag{8}$$

3 Comparison of the Semantics of Aggregates

3.1 A Reformulation of Ferraris Semantics

The propositional formula representation of an aggregate according to Ferraris semantics can be written in a more compact way by considering maximal local power sets as in the SPT-PDB semantics.

For an aggregate expression that contains no free variables, by $\mathit{MLPS\text{-}Fer}_{\Pi}(E)$ we denote

$$\bigwedge_{\langle B,T\rangle \text{ is a maximal LPS of } \overline{\mathcal{C}}_{\Pi}(E)} \Big(\bigwedge_{\mathbf{c}\in B} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c}\in \mathbf{O}_{\Pi}(E)\setminus T} F(\mathbf{c}) \Big). \tag{9}$$

One can check that formulas (5) and (9) are strongly equivalent [12] to each other, which provides another characterization of Ferraris answer sets. We define MLPS-Ferraris answer sets of Π same as the Ferraris answer sets of Π except that we refer to (9) in place of (5).

Proposition 1 *The MLPS-Ferraris answer sets of Π are precisely the Ferraris answer sets of Π.*

For example, in program Π_1, the maximal local power sets of $\overline{\mathcal{C}}_{\Pi_1}(E_1)$ are $\langle \emptyset, \{-1,1\}\rangle$, $\langle \{-1\}, \{-1,2\}\rangle$. The maximal local power set of $\overline{\mathcal{C}}_{\Pi_1}(E_2)$ is $\langle \{1,2\}, \{1,2\}\rangle$. Formula (6) is strongly equivalent to a shorter formula

$$p(2) \ \wedge \ (\underline{p(2) \wedge (p(-1) \rightarrow p(1))} \rightarrow p(-1)) \ \wedge \ (\underline{(p(1) \wedge p(2) \rightarrow p(-1))} \rightarrow p(1)). \tag{10}$$

3.2 A Reformulation of FLP Semantics

The FLP semantics can also be defined by reduction to propositional formulas. For an aggregate expression E that contains no free variables, let $\overline{\mathcal{I}}_{\Pi}(E)$ be the set of all Herbrand interpretations I of $\sigma(\Pi)$ such that $I \not\models E$ (as defined in Section 2.3). Clearly $\overline{\mathcal{I}}_{\Pi}(E)$ and $\mathcal{I}_{\Pi}(E)$ partition HU_{Π}. By $\mathit{FLP}_{\Pi}(E)$ we denote

$$\bigwedge_{I\in \overline{\mathcal{I}}_{\Pi}(E)} \Big(\bigwedge_{A\in I} A \rightarrow \bigvee_{A\in HU_{\Pi}\setminus I} A \Big). \tag{11}$$

By $\mathit{FLP}(\Pi)$ we denote the propositional formula obtained from $\mathit{Ground}(\mathit{Pos}(\Pi))$ by replacing all aggregate expressions E in it by $\mathit{FLP}_{\Pi}(E)$ (Recall the definition of $\mathit{Pos}(\Pi)$ in Section 2.5).

Proposition 2 *For any program Π, the FLP answer sets of Π are precisely the answer sets of $\mathit{FLP}(\Pi)$.*

Formula (11) looks similar to (5). In fact, for program Π_1, formula $\mathit{FLP}(\Pi_1)$ is (6), same as $\mathit{Fer}(\Pi_1)$. However, in general, they do not result in the same formula. For example, consider the following program Π_2.

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

$\mathit{Fer}(\Pi_2)$ is $\neg\neg p(a) \rightarrow p(a)$ while $\mathit{FLP}(\Pi_2)$ is $p(a) \rightarrow p(a)$. Furthermore if we add rule $q(a)$ to Π_2, for the resulting program, say Π_2', formula $\mathit{Fer}(\Pi_2')$ is $(\neg\neg p(a) \rightarrow p(a)) \wedge q(a)$, while $\mathit{FLP}(\Pi_2')$ is

$$[(p(a) \vee q(a)) \wedge (q(a) \rightarrow p(a)) \rightarrow p(a)] \wedge q(a).$$

The following proposition is a slight extension of Theorem 3 from [3] and describes a class of programs under which Ferraris answer sets coincide with FLP answer sets. We call a program *semi-positive* if, for every aggregate expression (3) occurring in it, $F(\mathbf{x})$ is a quantifier-free formula that contains no implications (this, in particular, means that there are no negations since we treat $\neg G$ as shorthand for $G \rightarrow \bot$). For example, Π_1 is semi-positive.

Proposition 3 *For any semi-positive program Π, the Ferraris answer sets of $Pos(\Pi)$ are precisely the FLP answer sets of Π.*

Similar to (9), formula $FLP(\Pi)$ can also be simplified using the notion of maximal local power sets, which provides another characterization of the FLP semantics.

Lemma 1. *Formula (11) is strongly equivalent to*

$$\bigwedge_{\langle B,T\rangle \text{ is a maximal LPS of } \overline{\mathcal{I}}_\Pi(E)} \Big(\bigwedge_{A\in B} A \rightarrow \bigvee_{A\in HU_\Pi \setminus T} A \Big). \tag{12}$$

3.3 A Reformulation of SPT-PDB Semantics

Consider the following formula modified from (11) by simply eliminating implication in favor of negation and disjunction as in classical logic.

$$\bigwedge_{I\in\overline{\mathcal{I}}_\Pi(E)} \Big(\bigvee_{A\in I} \neg A \ \vee \bigvee_{A\in HU_\Pi\setminus I} A \Big) \tag{13}$$

(13) is not strongly equivalent to (11), but, interestingly, is strongly equivalent to (7). This fact provides a simple reformulation of the SPT-PDB semantics. We define *modified FLP answer sets* of Π same as in Section 3.2 except that we refer to (13) in place of (11).

Proposition 4 *For any program Π, the modified FLP answer sets of Π are precisely the SPT-PDB answer sets of Π.*

For instance, the SPT-PDB answer sets of Π_1 are the same as the answer sets of the following formula:

$$\begin{array}{l} p(2) \\ \wedge([\underline{(p(-1)\vee p(1)\vee p(2)) \wedge (\neg p(-1)\vee p(1)\vee p(2)) \wedge (\neg p(1)\vee p(-1)\vee p(2))} \\ \quad \underline{\wedge(\neg p(-1)\vee \neg p(1)\vee p(2)) \wedge (\neg p(-1)\vee \neg p(2)\vee p(1))]} \rightarrow p(-1)) \\ \wedge(\underline{[\neg p(1)\vee \neg p(2)\vee p(-1)]} \rightarrow p(1)). \end{array} \tag{14}$$

The reformulation above is simpler than the SPT-PDB semantics reviewed in Section 2.5 in the sense that it does not involve the notion of local power sets. On the other hand, similar to the Ferraris and the FLP semantics, considering maximal local power sets can yield shorter propositional formula representation as the following lemma tells.

Lemma 2. *Formula (13) is strongly equivalent to*

$$\bigwedge_{\langle B,T\rangle \text{ is a maximal LPS of } \overline{\mathcal{I}}_{\Pi}(E)} \Big(\bigvee_{A\in B} \neg A \vee \bigvee_{A\in HU_{\Pi}\setminus T} A \Big). \tag{15}$$

Again note the similarity between (15) and (12). They are classically equivalent, but not strongly equivalent to each other. For instance, the SPT-PDB answer sets of Π_1 are the same as the answer sets of

$$p(2) \wedge (\underline{p(2) \wedge (\neg p(-1) \vee p(1))} \rightarrow p(-1)) \wedge (\underline{\neg p(1) \vee \neg p(2) \vee p(-1)} \rightarrow p(1)).$$

3.4 Relationship between the Semantics

The characterizations of each semantics in terms of the uniform framework of propositional formulas give insights into their relationships. First, it is not difficult to check that for any aggregate expression occurring in $Ground(\Pi)$, formula $FLP_{\Pi}(E)$ entails formula $SPT\text{-}PDB_{\Pi}(E)$, but not the other way around. From this we can conclude the following.

Proposition 5 *[10, Theorem 2] Every SPT-PDB answer set of Π is an FLP answer set of Π.*

Next, it is known that for monotone and antimonotone aggregates the FLP and the SPT-PDB semantics coincide [5, Proposition 9]. Alternatively, this can be easily explained by the fact that for monotone and antimonotone aggregates E, formula (12) and (7) are strongly equivalent to each other.

In order to describe a bigger class of programs in which both the FLP and the SPT-PDB semantics coincide, we need to define the notion of a dependency graph. We define an *FLP dependency graph* as follows. For simplicity we assume that $F(\mathbf{x})$ in every aggregate expression (3) is a conjunction of atoms. Given such a program Π that contains no free variables, the FLP dependency graph (V, E) of Π is such that

- V is the set of all ground atoms of $\sigma(\Pi)$;[8]
- an edge from A_i $(i = 1, \ldots, l)$ to B if there is a rule (4) in $Ground(\Pi)$ such that
  - B is an atom occurring as one of E_j $(j = 1, \ldots m)$, or
  - B belongs to $HU_{\Pi} \setminus T$ for some maximal local power set $\langle B', T\rangle$ of $\overline{\mathcal{I}}(E_j)$ for some aggregate expression E_j where $j = 1, \ldots, m$.

It is not difficult to check that the FLP dependency graph of Π is the same as the dependency graph of the propositional formula $MLPS\text{-}FLP(\Pi)$ (Section 3.2) according to [13]. For example, the FLP dependency graph of Π_1 is the same as the dependency graph of (10). The graph has three vertices $p(-1)$, $p(1)$, $p(2)$, and three edges $\langle p(-1), p(2)\rangle$, $\langle p(-1), p(1)\rangle$, $\langle p(1), p(-1)\rangle$.

We call Π *regular* if, for each rule (4) in $Ground(\Pi)$, the dependency graph of Π has no edges from B to A_i where B is an atom that belongs to $HU_{\Pi} \setminus T$ for some maximal local power set $\langle B', T\rangle$ of $\overline{\mathcal{I}}(E_i)$ for some aggregate expression E_j where $j = 1, \ldots, m$. One can check that program Π_1 is not regular.

Proposition 6 *For any regular program Π, the FLP answer sets of Π are precisely the SPT-PDB answer sets of Π.*

[8] By an *atom* we mean non-equality atomic formulas.

4 Generalized Definition of Aggregates

4.1 Syntax and Semantics of Aggregate Formulas

In this section we provide a general definition of a stable model that applies to arbitrary "aggregate formulas" in the style of the definition in Section 2.1, by extending the notion F^* to aggregate expressions similar to other connectives, and using the extended notion of satisfaction as in the FLP semantics (Section 2.3).

We allow the signature to contain any function constants of positive arity, and allow b in aggregate expression (3) to be any term. We define *aggregate formulas* as an extension of first-order formulas by treating aggregate expressions as a base case in addition to (standard) atomic formulas (including equality) and $\bot$ (falsity). In other words, aggregate formulas are constructed from atomic formulas and aggregate expressions using connectives and quantifiers as in first-order logic. For instance,

$$(\text{SUM}\langle\{x : p(x)\}\rangle \geq 1 \ \vee \ \exists y\, q(y)) \rightarrow r(x)$$

is an aggregate formula.

We say that an occurrence of a variable v in an aggregate formula H *bound* if the occurrence is

- in $\{\mathbf{x} : F(\mathbf{x})\}$ of an aggregate expression (3) occurring in H such that v is in $\mathbf{x}$, or
- in a part of H of the form QvG.

Otherwise it is *free*. We say that v is *free* in H if H contains a free occurrence of v. An aggregate sentence is an aggregate formula with no free variables.

The definition of an interpretation is the same as in first-order logic. Consider an interpretation I of a first-order signature σ that may contain any function constants of positive arity. By $\sigma^{|I|}$ we mean the signature obtained from σ by adding distinct new object constants d^*, called *names*, for all d in the universe of I. We identify an interpretation I of σ with its extension to $\sigma^{|I|}$ defined by $I(d^*) = d$.

The notion of satisfaction in first-order logic is extended to aggregate sentences, similar to the definition given in 2.3. The integer constants and built-in symbols, such as $+$, $-$, $\leq$, $\geq$ are evaluated in the standard way, and we consider only those "standard" interpretations.[9] Let I be an interpretation of signature σ. Consider any aggregate expression (3) that has no free variables. Let S be the multiset consisting of all $\mathbf{d}^1$ in the universe of I where [10]

- $\mathbf{d}^*$ is a list of object names of $\sigma^{|I|}$ whose length is the same as the length of $\mathbf{x}$, and
- I satisfies $F(\mathbf{d}^*)$.

An interpretation I satisfies the aggregate expression if S is in the domain of op, and $op(S) \succeq b^I$.

For any aggregate sentence F, expression SM[F] stands for (2) where $F^*(\mathbf{u})$ is extended to aggregate expressions as

[9] We assume that, when x or y is not an integer, $x \leq y$ evaluates to false; $x + y$ is defined by the interpretation.

[10] Given a list $\mathbf{t}$ of object constants or variables, by $\mathbf{t}^1$ we represent the first element of $\mathbf{t}$.

– $(\text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b)^* = (\text{OP}\langle\{\mathbf{x} : F^*(\mathbf{x})\}\rangle \succeq b) \wedge (\text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b)$.

By a *stable model* of F we mean the models of SM$[F]$ (under the extended notion of satisfaction) whose signature is $\sigma(F)$.

4.2 Programs with Aggregates as a Special Case

The *AF-representation* ("Aggregate Formula representation") of (4) is the universal closure of aggregate formula

$$E_1 \wedge \cdots \wedge E_m \wedge \neg E_{m+1} \wedge \cdots \wedge \neg E_n \rightarrow A_1 \vee \cdots \vee A_l. \tag{16}$$

The *AF-representation* of Π is the conjunction of the AF-representation of its rules.

The *stable models* of Π are defined as the stable models of the AF-representation of Π. The following proposition shows that this definition is a proper generalization of the Ferraris semantics.

Proposition 7 *Let Π be a program that contains no function constants of positive arity and let F be its AF-representation. The Herbrand stable models of F whose signature is $\sigma(\Pi)$ are precisely the Ferraris answer sets of Π.*

5 Loop Formulas for Programs with Aggregates

Let Π be a ground program containing no function constants of positive arity such that $F(\mathbf{x})$ in every aggregate expression (3) is a conjunction of atoms. For any aggregate expression $E = \text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b$, formula $NFES_E(Y)$ is defined as the conjunction of E and

$$\text{OP}\langle\{\mathbf{x} \ : \ F(\mathbf{x}) \wedge \bigwedge_{\substack{p_i(\mathbf{t}) \text{ occurs in } F(\mathbf{x}) \\ p_i(\mathbf{t}') \in Y}} \mathbf{t} \neq \mathbf{t}'\}\rangle \succeq b$$

For instance, in Example Π_1, formula $NFES_{E_1}(\{p(-1), p(1)\})$ is $\text{SUM}\{x : p(x) \wedge x \neq -1 \wedge x \neq 1\} \geq 2$.

We define the *external support formula* of a set Y of atoms for Π, denoted by $ES_\Pi(Y)$, as the disjunction of

$$\bigwedge_{i=1,\ldots,m} NFES_{E_i}(Y) \wedge \bigwedge_{i=m+1,\ldots,n} \neg E_i \ \wedge \bigwedge_{p \in A \setminus Y} \neg p$$

for all rules (4) in Π such that $A \cap Y \neq \emptyset$. The *aggregate loop formula* of Y for Π is the aggregate formula

$$\bigwedge Y \rightarrow ES_\Pi(Y). \tag{17}$$

This definition extends the definition of a loop formula given in [14].

For instance, if Y is $\{p(-1), p(1)\}$, the loop formula of Y is

$$p(-1) \wedge p(1) \rightarrow (\text{SUM}\{x : p(x) \wedge x \neq -1 \wedge x \neq 1\} \geq 2) \wedge \text{SUM}\{x : p(x)\} \geq 2 \\ \vee (\text{SUM}\{x : p(x) \wedge x \neq -1 \wedge x \neq 1\} \leq 2) \wedge \text{SUM}\{x : p(x)\} \leq 2.$$

The *propositional loop formula* of Y for Π is obtained from (17) by replacing all occurrences of aggregate expression E with $Fer_\Pi(E)$.

A Ferraris dependency graph is defined the same as an FLP dependency graph (Section 3.4) except that we refer to $\mathbf{O}_\Pi(E)$ instead of HU_Π, and $\overline{\mathcal{C}}(E_j)$ instead of $\overline{\mathcal{I}}(E_j)$. It is not difficult to check that the dependency graph of Π according to this definition is the same as the dependency graph of propositional formula *MLPS-Fer*(Π) according to [13]. A *loop* is a nonempty set L of ground atoms of $\sigma(\Pi)$ such that the subgraph of the dependency graph of F induced by L is strongly connected. Again, L is a loop of Π according to this definition iff it is a loop of *MLPS-Fer*(Π) according to [13].[11]

In the following we refer to loops and loop formulas according to this paper.

Proposition 8 *Let Π be a ground program such that every $F(\mathbf{x})$ in every aggregate expression (3) is a conjunction of atoms. For any set X of ground atoms of $\sigma(\Pi)$ that satisfies Π, the following conditions are equivalent to each other.*

(a) X is a Ferraris answer set of Π;
(b) X satisfies the aggregate loop formulas of all nonempty subsets Y of X;
(c) X satisfies the aggregate loop formulas of all loops Y of Π;
(d) X satisfies the propositional loop formulas of all nonempty subsets Y of X;
(e) X satisfies the propositional loop formulas of all loops Y of Π.

This result can be straightforwardly extended to the general case when F in an aggregate expression (3) is an arbitrary quantifier-free formula, by using the notion of $NFES_F$ that is defined in [15]. The definition of external support formula above is closely related to the definition of unfounded sets under the FLP semantics given in [16].

You and Liu [17] presented the definition of loop formulas under the SPT-PDB semantics. We note that a set of atoms is a loop of Π according to their definition iff it is a loop of *SPT-PDB*(Π) according to [13]. The same can be said about loop formulas.

6 Conclusion

The paper presented several reformulations of the semantics of aggregates in terms of propositional formulas. This gives us new insights into each of the semantics in terms of the underlying general language. Guided by the reduction, we defined the loop formulas of a program with aggregates, which result in the same as loop formulas of the corresponding propositional formula representation.

The reductive approach led us to the general definition of aggregates presented in Section 4, which extends the definition of a stable model of first-order formula to aggregate formulas, using a notion of satisfaction extended from the one used in the FLP semantics. It is more general than RASPL-1 [18] in that it allows arbitrary

[11] Note that *Fer*(Π) may contain redundant loops not in *MLPS-Fer*(Π). For example, consider

$$p(1) \leftarrow \text{SUM}\langle\{x : p(x)\}\rangle \geq 1$$
$$p(-1) \leftarrow p(1)$$

aggregates and non-Herbrand stable models, along with built-in functions. On the other hand, it requires the notion of satisfaction be extended to aggregate expressions. In contrast, a counting aggregate expression in RASPL-1 was defined as an abbreviation for a first-order formula, without extending the notion of satisfaction. Defining the formulas that aggregate expressions (other than counting) stand for is an on-going work.

References

1. Lifschitz, V., Tang, L.R., Turner, H.: Nested expressions in logic programs. Annals of Mathematics and Artificial Intelligence **25** (1999) 369–389
2. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: Semantics and complexity.[12] In: Proceedings of European Conference on Logics in Artificial Intelligence (JELIA). (2004)
3. Ferraris, P.: Answer sets for propositional theories. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2005) 119–131
4. Pearce, D.: A new logical characterization of stable models and answer sets. In Dix, J., Pereira, L., Przymusinski, T., eds.: Non-Monotonic Extensions of Logic Programming (Lecture Notes in Artificial Intelligence 1216), Springer-Verlag (1997) 57–70
5. Son, T.C., Pontelli, E., Tu, P.H.: Answer sets for logic programs with arbitrary abstract constraint atoms. J. Artif. Intell. Res. (JAIR) **29** (2007) 353–389
6. Pelov, N., Denecker, M., Bruynooghe, M.: Translation of aggregate programs to normal logic programs. In: Proc. Answer Set Programming. (2003)
7. Niemelä, I., Simons, P.: Extending the Smodels system with cardinality and weight constraints. In Minker, J., ed.: Logic-Based Artificial Intelligence. Kluwer (2000) 491–521
8. Ferraris, P., Lifschitz, V.: Weight constraints as nested expressions. Theory and Practice of Logic Programming **5** (2005) 45–74
9. Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2007) 372–379
10. Son, T.C., Pontelli, E.: A constructive semantic characterization of aggregates in answer set programming. TPLP **7** (2007) 355–375
11. Pelov, N., Denecker, M., Bruynooghe, M.: Well-founded and stable semantics of logic programs with aggregates. TPLP **7** (2007) 301–353
12. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2** (2001) 526–541
13. Ferraris, P., Lee, J., Lifschitz, V.: A generalization of the Lin-Zhao theorem. Annals of Mathematics and Artificial Intelligence **47** (2006) 79–101
14. Liu, L., Truszczynski, M.: Properties and applications of programs with monotone and convex constraints. J. Artif. Intell. Res. (JAIR) **27** (2006) 299–334
15. Lee, J., Meng, Y.: On loop formulas with variables. In: Proceedings of the International Conference on Knowledge Representation and Reasoning (KR). (2008) 444–453
16. Faber, W.: Unfounded sets for disjunctive logic programs with arbitrary aggregates. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2005) 40–52
17. You, J.H., Liu, G.: Loop formulas for logic programs with arbitrary constraint atoms. In: AAAI. (2008) 584–589
18. Lee, J., Lifschitz, V., Palla, R.: A reductive semantics for counting and choice in answer set programming. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). (2008) 472–479

[12] Revised version: `http://www.wfaber.com/research/papers/jelia2004.pdf` .