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 "reductive" approach to understand an aggregate as shorthand for 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

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

has no answer sets.[1]

The difficulty led to several interesting extensions of the stable model semantics to account for aggregates, such as an extended definition of reduct [2], an extension of T_P operator with "conditional satisfaction" [3], and an extension of the standard approximating operator Φ_P to Φ_P^{aggr} [4]. On the other hand, a few reasonable "reductive" semantics of aggregates were also developed. In [5] and [6], the authors

[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.

defined translations of aggregates into nested expressions, which are somewhat complex than the naive approach above. Instead of translating into nested expressions, Ferraris [7] proposed to identify an aggregate with a conjunction of implications under his extension of the answer set semantics for arbitrary propositional formulas. The extended semantics is essentially a reformulation of the equilibrium logic [8], and was generalized to arbitrary first-order formulas in [9].

While most semantics 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.

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

According to [2; 7], the program has one answer set, $\{p(-1), p(1), p(2)\}$, while it has no answer sets according to [3; 5].

In this paper we make further developments on the reductive semantics. We note that the semantics by Pelov, Denecker, Bruynooghe [5] and the semantics by Ferraris [7] are closely related to each other in terms of propositional formula representations of aggregates, yielding a few interesting alternative characterizations. Furthermore we show that the semantics by Faber, Leone and Pfeifer [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 define loop formulas for programs with aggregates guided by the reductive approach: such loop formulas contain aggregates, and when these aggregates are turned into corresponding propositional formulas, the resulting formulas are the same as the loop formulas for a propositional theory corresponding to the program, as defined in [10].

In this note we do not consider weight constraints [11]. They do not cover arbitrary aggregates and the translation into nested expressions is well studied in [12].

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

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

(There is no clause for negation here, $\neg F$ is treated 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 $\mathrm{SM}[F]$ 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).

Ferraris *et al.* [9] shows that this definition, if restricted to the syntax of traditional logic programs, is equivalent to the traditional definition of an answer set and, if restricted to the syntax of arbitrary propositional formulas, is equivalent to the definition of an answer set given by Ferraris [7]. (According to [7], the *reduct* F^X of a formula F relative to a set X of atoms is the formula obtained from F by replacing each maximal subformula that is not satisfied by X with $\bot$. Set X is an *answer set* of F if X is a minimal set satisfying F^X.)

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*, *times*, *min* and *max* are defined for multisets of numbers; *min* and *max* do not allow the empty set in their domains.

An *aggregate expression* is of the form

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

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;
– $\succeq$ 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

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

[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*.

($l \geq 0$; $n \geq m \geq 0$), 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 $Ground(\Pi)$ we denote the program obtained from Π by 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.[3] We consider Herbrand interpretations of $\sigma(\Pi)$ only. Consider any aggregate expression (3) occurring in $Ground(\Pi)$ and any Herbrand interpretation I of $\sigma(\Pi)$. Let S_I be the multiset consisting of all $\mathbf{c}^1$ in the Herbrand universe where [4]

- $\mathbf{c}$ 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 I of ground atoms of $\sigma(\Pi)$ satisfies the aggregate expression if S_I is in the domain of *op*, and $op(S_I) \succeq b$.

The *FLP reduct* of Π relative to I is obtained from $Ground(\Pi)$ by removing every rule whose body is not satisfied by I. Set I 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 I. 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 [7] 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.

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

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

if

[3] The syntax in [2] requires $F(\mathbf{x})$ to be a conjunction of atoms.

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

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

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

Let $E = \text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b$ be an aggregate expression occurring in $Ground(\Pi)$, 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 $Fer_\Pi(E)$ 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, $Fer_{\Pi_1}(E_1)$ is

$$\begin{aligned}&(p(-1) \vee p(1) \vee p(2)) \ \wedge\ (p(-1) \rightarrow p(1) \vee p(2)) \ \wedge\ (p(1) \rightarrow p(-1) \vee p(2)) \\ &\quad \wedge\ (p(-1) \wedge p(1) \rightarrow p(2)) \ \wedge\ (p(-1) \wedge p(2) \rightarrow p(1))\ .\end{aligned}$$

By $Fer(\Pi)$ we denote propositional formula obtained from $Ground(\Pi)$ by replacing every aggregate expression E in it by $Fer_\Pi(E)$. The *Ferraris answer sets* of Π are defined as the answer sets of $Fer(\Pi)$ in the sense of Section 2.1. For example, the Ferraris answer sets of Π_1 are the answer sets of the following formula $Fer(\Pi_1)$: [5]

$$\begin{aligned}p(2)\ \wedge\ (&\underline{[(p(-1) \vee p(1) \vee p(2)) \wedge (p(-1) \rightarrow p(1) \vee p(2)) \wedge (p(1) \rightarrow p(-1) \vee p(2))} \\ &\underline{\wedge (p(-1) \wedge p(1) \rightarrow p(2)) \wedge (p(-1) \wedge p(2) \rightarrow p(1))]} \rightarrow p(-1)) \\ \wedge\ (&\underline{[p(1) \wedge p(2) \rightarrow p(-1)]} \rightarrow p(1)).\end{aligned} \tag{6}$$

Applying the definition of an answer set in [7] (Section 2.1), 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 [6] 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 by Son, Pontelli and Tu [3]. Lemma 6 from [6] shows that these definitions are equivalent to the definition by Pelov, Denecker, Bruynooghe [4], which is in terms of translation into nested expressions. Thus we group them together and review only the last definition.[6]

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."

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.

[5] We underline the parts of a formula that correspond to aggregates.
[6] We ignore some differences in the syntax, and allow disjunctions in the head.

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 $Ground(\Pi)$, 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 $Ground(\Pi)$, by $SPT\text{-}PDB_\Pi(E)$ 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, $SPT\text{-}PDB_{\Pi_1}(E_1)$ 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 $Pos(\Pi)$ 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, $Pos(\Pi)$ contains no negation in front of aggregate expressions.

By $SPT\text{-}PDB(\Pi)$ we denote the propositional formula obtained from $Ground(Pos(\Pi))$ by replacing all aggregate expressions E in it by $SPT\text{-}PDB_\Pi(E)$. The *SPT-PDB* answer sets of Π are defined as the answer sets of $SPT\text{-}PDB(\Pi)$ 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 $SPT\text{-}PDB(\Pi_1)$:

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

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 $MLPS\text{-}Fer_\Pi(E)$ 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 [13] 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

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

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 $FLP_{\Pi}(E)$ we denote

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

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

Proposition 2 *For any program Π, the FLP answer sets of Π (Section 2.3) are precisely the answer sets of $FLP(\Pi)$.*

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

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

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

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

The following proposition is a slight extension of Theorem 3 from [7] and describes a class of programs whose 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 $G \to \bot$). 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 yet 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 implications in favor of negations and disjunctions as in classical logic:

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

Clearly, (13) and (11) are classically equivalent to each other, but they are not strongly equivalent. However, interestingly, (13) is strongly equivalent to (7), which in turn provides a simple reformulation of the SPT-PDB semantics, without involving the notion of local power sets. 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 E occurring in $\mathit{Ground}(\Pi)$, formula $\mathit{FLP}_{\Pi}(E)$ entails formula $\mathit{SPT\text{-}PDB}_{\Pi}(E)$, but not the other way around. From this we can conclude the following.

Proposition 5 *[6, 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 [3, 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 larger 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 $\sigma(\Pi)$ [7];
- for every rule (4) in $\mathit{Ground}(\mathit{Pos}(\Pi))$, an edge goes from A_i $(i = 1, \dots, l)$ to B
  - if B is an atom occurring as one of E_j $(j = 1, \dots, m)$, or
  - if there are an aggregate expression E_j $(j = 1, \dots, m)$ and a maximal local power set $\langle B', T\rangle$ of $\overline{\mathcal{I}}(E_j)$ such that B belongs to $HU_{\Pi} \setminus T$.

It is not difficult to check that the FLP dependency graph of Π is the same as the dependency graph of the propositional formula $\mathit{MLPS\text{-}FLP}(\Pi)$ (Section 3.2) according to [10]. 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 $\mathit{Ground}(\mathit{Pos}(\Pi))$, 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 $j = 1, \dots, m$. 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.*

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).

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

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,

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

is an aggregate formula.

We say that an occurrence of a variable v in an aggregate formula H *bound* if the occurrence is in a part of H of the form $\{\mathbf{x} : F(\mathbf{x})\}$ where 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 Section 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.[8] Let I be an interpretation of signature σ. Consider any aggregate expression (3) that has no free variables. Let S_I be the multiset consisting of all $\mathbf{d}^1$ in the universe of I where [9]

- $\mathbf{d}^*$ 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_I is in the domain of op, and $op(S_I) \succeq b^I$.

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

- $(\textsc{op}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b)^* = (\textsc{op}\langle\{\mathbf{x} : F^*(\mathbf{x})\}\rangle \succeq b) \wedge (\textsc{op}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b)$.

By a *stable model* of F we mean a model of SM$[F]$ (under the extended notion of satisfaction).

4.2 Programs with Aggregates as a Special Case

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

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

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

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

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 program containing no free variables and 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 $\mathit{NFES}_E(Y)$ 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 $\mathit{NFES}_{E_1}(\{p(-1), p(1)\})$ is

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

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

$$\bigwedge_{i=1,\dots,m} \mathit{NFES}_{E_i}(Y) \wedge \bigwedge_{i=m+1,\dots,n} \neg E_i\ \wedge \bigwedge_{p\in A\setminus Y} \neg p$$

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

$$\bigwedge Y \to \mathit{ES}_\Pi(Y). \tag{17}$$

This definition extends the definition of a loop formula given in [14], which is limited to programs with monotone aggregates.

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

$$\begin{aligned} p(-1) \wedge p(1) \to\ & (\text{SUM}\{x : p(x) \wedge x\neq -1 \wedge x\neq 1\} \geq 2) \wedge \text{SUM}\{x : p(x)\} \geq 2 \\ & \vee(\text{SUM}\{x : p(x) \wedge x\neq -1 \wedge x\neq 1\} \leq 2) \wedge \text{SUM}\{x : p(x)\} \leq 2. \end{aligned}$$

The *PL representation* of (17) is the propositional formula obtained from (17) by replacing all occurrences of aggregate expressions E in it with $\mathit{MLPS\text{-}Fer}_\Pi(E)$.

A Ferraris dependency graph is defined similar to an FLP dependency (Section 3.4). The Ferraris dependency graph (V, E) of Π is such that

- V is the set of all ground atoms of $\sigma(\Pi)$;
- for every rule (4) in $\mathit{Ground}(\Pi)$, an edge goes from A_i $(i = 1, \dots, l)$ to B
  - if B is an atom occurring as one of E_j $(j = 1, \dots, m)$, or

- if there are an aggregate expression E_j ($j = 1, \dots, m$), a maximal local power set $\langle B', T\rangle$ of $\overline{C}(E_j)$, and $\mathbf{c}$ in $\mathbf{O}_\Pi(E_j) \setminus T$ such that B belongs to $F(\mathbf{c})$.

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 $\mathit{MLPS}\text{-}\mathit{Fer}(\Pi)$ according to [10]. A *loop* is a nonempty set L of ground atoms of $\sigma(\Pi)$ such that the subgraph of the dependency graph of Π induced by L is strongly connected. Again, L is a loop of Π according to this definition iff it is a loop of $\mathit{MLPS}\text{-}\mathit{Fer}(\Pi)$ according to [10].[10]

In (b) and (c) in the following proposition, we refer to loops and loop formulas according to this paper.

Proposition 8 *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) for every loop Y of Π, X satisfies the aggregate loop formula of Y for Π;
(c) for every loop Y of Π, X satisfies the PL representation of the aggregate loop formula of Y for Π;
(d) for every loop Y of $\mathit{MLPS}\text{-}\mathit{Fer}(\Pi)$ according to [10], X satisfies the loop formula of Y for $\mathit{MLPS}\text{-}\mathit{Fer}(\Pi)$ according to [10].

This result can be extended to the general case when $F(\mathbf{x})$ 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]. Indeed, one can define loop formulas and loops under the FLP semantics in a similar way based on the reductive approach.

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 $\mathit{SPT}\text{-}\mathit{PDB}(\Pi)$ according to [10]. 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 semantics of aggregates presented in Section 4, which extends the definition of a stable model of a first-order formula to an aggregate formula, using a notion of satisfaction extended from the one used in the FLP semantics. The new semantics is more general than that of RASPL-1 [18]

[10] Due to lack of space we do not provide the definitions in [10]. Note that $\mathit{Fer}(\Pi)$ may contain redundant loops not in $\mathit{MLPS}\text{-}\mathit{Fer}(\Pi)$. For example, consider

$$p(1) \leftarrow \textsc{sum}\langle\{x : p(x)\}\rangle \geq 1 \qquad p(-1) \leftarrow p(1)$$

in that it allows arbitrary aggregates and non-Herbrand stable models, along with built-in functions. On the other hand, it is not fully reductive; it requires the notion of satisfaction be extended to aggregate expressions, while a counting aggregate expression in RASPL-1 was defined as an abbreviation for a first-order formula. Defining formulas that arbitrary 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.[11] In: Proceedings of European Conference on Logics in Artificial Intelligence (JELIA). (2004)
3. 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
4. Pelov, N., Denecker, M., Bruynooghe, M.: Well-founded and stable semantics of logic programs with aggregates. TPLP **7** (2007) 301–353
5. Pelov, N., Denecker, M., Bruynooghe, M.: Translation of aggregate programs to normal logic programs. In: Proc. Answer Set Programming. (2003)
6. Son, T.C., Pontelli, E.: A constructive semantic characterization of aggregates in answer set programming. TPLP **7** (2007) 355–375
7. Ferraris, P.: Answer sets for propositional theories. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2005) 119–131
8. 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
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. Ferraris, P., Lee, J., Lifschitz, V.: A generalization of the Lin-Zhao theorem. Annals of Mathematics and Artificial Intelligence **47** (2006) 79–101
11. 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
12. Ferraris, P., Lifschitz, V.: Weight constraints as nested expressions. Theory and Practice of Logic Programming **5** (2005) 45–74
13. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2** (2001) 526–541
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

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

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
19. Ferraris, P., Lee, J., Lifschitz, V., Palla, R.: Symmetric splitting in the general theory of stable models[12] In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2009) To appear.

7 Proofs

7.1 Proof of Proposition 1

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

$$\bigwedge_{\langle B,T\rangle \text{ is an LPS of } \overline{\mathcal{C}}_{\Pi}(E)} \Big(\bigwedge_{\mathbf{c}\in B} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c}\in \mathbf{O}_{\Pi}(E)\setminus T} F(\mathbf{c})\Big). \tag{18}$$

Proposition 1′ *Formulas (5), (18), (9) are strongly equivalent to each other.*

Proof. It is sufficient to show that for any two subsets B, T of $\mathbf{O}_{\Pi}(E)$ such that B is a subset of T, formula

$$\bigwedge_{B\subseteq S\subseteq T} \Big(\bigwedge_{\mathbf{c}\in S} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c}\in \mathbf{O}_{\Pi}(E)\setminus S} F(\mathbf{c})\Big) \tag{19}$$

is equivalent to

$$\bigwedge_{\mathbf{c}\in B} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c}\in \mathbf{O}_{\Pi}(E)\setminus T} F(\mathbf{c})$$

in the Logic of Here-and-There.

From right to left: Clear from the facts that $B \subseteq S$ and $\mathbf{O}_{\Pi}(E) \setminus T \subseteq \mathbf{O}_{\Pi}(E) \setminus S$.

From left to right: Assume (19). Consider several cases, each of which corresponds to S such that $B \subseteq S \subseteq T$. The assumption characterizing each case is that S contains all $\mathbf{c}$ in T such that $F(\mathbf{c})$ holds. Consequently, we have

$$\bigwedge_{\mathbf{c}\in S} F(\mathbf{c}) \wedge \bigwedge_{\mathbf{c}\in T\setminus S} \neg F(\mathbf{c}). \tag{20}$$

It follows from (19) and (20) that

$$\bigvee_{\mathbf{c}\in \mathbf{O}_{\Pi}(E)\setminus S} F(\mathbf{c}).$$

From this and the second conjunctive term of (20), we conclude that

$$\bigvee_{\mathbf{c}\in \mathbf{O}_{\Pi}(E)\setminus T} F(\mathbf{c}).$$

∎

[12] `http://peace.eas.asu.edu/joolee/papers/splitting.pdf` .

7.2 Proof of Proposition 2

Lemma 3. *Let Π be a program and E an aggregate expression that has no free variables. For any set X of ground atoms, $X \models E$ iff $X \models FLP_\Pi(E)$.*

Proof. *From left to right:* Assume that $X \models E$. Consider any I in $\overline{\mathcal{I}}_\Pi(E)$, and assume that $X \models \bigwedge_{A \in I} A$. Thus $I \subseteq X$. From that $X \models E$ and $I \in \overline{\mathcal{I}}_\Pi(E)$, we conclude that $X \neq I$. Consequently, $I \subset X$. There is an atom A such that $A \in X$ and $A \notin I$, so that $X \models \bigvee_{A \in HU_\Pi \setminus I} A$.

From right to left: Assume $X \models FLP_\Pi(E)$, that is,

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

It follows that $X \notin \overline{\mathcal{I}}_\Pi(E)$. Indeed, it is clear that

$$X \not\models \bigwedge_{A \in X} A \rightarrow \bigvee_{A \in HU_\Pi \setminus X} A.$$

Consequently, $X \models E$. ∎

By $Y^{\mathbf{p}}_{\mathbf{q}}$ we denote the set of ground atoms obtained from Y by substituting the members of new predicate constants $\mathbf{q}$ for the corresponding members of $\mathbf{p}$. We assume that $\mathbf{p}$ contains all predicate constants in the underlying signature.

Lemma 4. *Let Π be a program and E an aggregate expression that has no free variables. For any set X of ground atoms and a subset Y of X, we have that $X \models E$, $Y \models E$ iff $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models FLP_\Pi(E)^*(\mathbf{q})$.*

Proof.

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models FLP_\Pi(E)^*(\mathbf{q})$$

iff

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \bigwedge_{I \in \overline{\mathcal{I}}_\Pi(E)} \Big(\bigwedge_{A \in I} A \rightarrow \bigvee_{A \in HU_\Pi \setminus I} A \Big) \wedge \bigwedge_{I \in \overline{\mathcal{I}}_\Pi(E)} \Big(\bigwedge_{A \in I} A^*(\mathbf{q}) \rightarrow \bigvee_{A \in HU_\Pi \setminus I} A^*(\mathbf{q}) \Big).$$

The latter is equivalent to saying that

$$X \models \bigwedge_{I \in \overline{\mathcal{I}}_\Pi(E)} \Big(\bigwedge_{A \in I} A \rightarrow \bigvee_{A \in HU_\Pi \setminus I} A \Big) \tag{21}$$

and

$$Y^{\mathbf{p}}_{\mathbf{q}} \models \bigwedge_{I \in \overline{\mathcal{I}}_\Pi(E)} \Big(\bigwedge_{A \in I} A^*(\mathbf{q}) \rightarrow \bigvee_{A \in HU_\Pi \setminus I} A^*(\mathbf{q}) \Big). \tag{22}$$

By Lemma 3, (21) holds iff $X \models E$. (22) is equivalent to saying that

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

By Lemma 3 again, this is equivalent to saying that $Y \models E$. ∎

Proof of Proposition 2. Without losing generality let's assume that the program Π contains no negation in front of aggregate expressions, and contains no free variables.

Let X be a set of ground atoms of $\sigma(\Pi)$. If $X \not\models \Pi$, then X is neither an FLP answer set nor an answer set of $FLP(\Pi)$.

Assume $X \models \Pi$. Let

$$A_1; \dots; A_k \leftarrow E_1, \dots, E_l, A_{k+1}, \dots, A_m, not\ A_{m+1}, \dots, not\ A_n \tag{23}$$

be any rule in Π, where all A_i are standard atoms and all E_i are aggregate expressions.

It is sufficient to show that, for any subset Y of X, set Y satisfies the FLP-reduct of (23) relative to X iff

$$\begin{array}{c} X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models A_1^*(\mathbf{q});\ \dots;\ A_k^*(\mathbf{q}) \leftarrow FLP(E_1)^*(\mathbf{q}), \dots, FLP(E_l)^*(\mathbf{q}), \\ A_{k+1}^*(\mathbf{q}), \dots, A_m^*(\mathbf{q}), not\ A_{m+1}, \dots, not\ A_n. \end{array}$$

We consider the following cases.

Case 1: X does not satisfy some A_i for $k+1 \leq i \leq m$ or satisfies some A_i for $m+1 \leq i \leq n$. It is clear that the reduct is empty and X does not satisfy the antecedent of $FLP_\Pi((23))$.

Case 2: X does not satisfy some E_i. Again the reduct is empty. By Lemma 3, $X \not\models FLP_\Pi(E_i)$, and consequently, $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \not\models FLP_\Pi(E_i)^*(\mathbf{q})$ by the monotonicity lemma.

Case 3: X satisfies all $E_i (1 \leq i \leq l)$, A_i $(k+1 \leq i \leq m)$ and does not satisfy any A_j $(m+1 \leq j \leq n)$. The reduct is (23) itself. Under this condition, it is sufficient to prove that Y satisfies

$$A_1; \dots; A_k \leftarrow E_1, \dots, E_l, A_{k+1}, \dots, A_m$$

iff $X \cup Y^{\mathbf{p}}_{\mathbf{q}}$ satisfies

$$A_1^*(\mathbf{q}); \dots; A_n^*(\mathbf{q}) \leftarrow FLP_\Pi(E_1)^*(\mathbf{q}), \dots, FLP_\Pi(E_l)^*(\mathbf{q}), A_{k+1}^*(\mathbf{q}), \dots, A_m^*(\mathbf{q}).$$

The claim follows from Lemma 4. ∎

7.3 Proof of Proposition 3

Lemma 5. *Let F be a ground formula that contains no implications, and let X, Y be sets of ground atoms such that $Y \subseteq X$. We have that $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models F^*(\mathbf{q})$ iff $Y \models F$.*

Proof. By structural induction.

Case 1: F is a ground atom $p(\mathbf{t})$. $Y \models p(\mathbf{t})$ iff $p(\mathbf{t}) \in Y$ iff $q(\mathbf{t}) \in Y^{\mathbf{p}}_{\mathbf{q}}$ iff $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models q(\mathbf{t})$.

Case 2: F is equality, $\top$ or $\bot$. Clear.

Case 3: F is $G \wedge H$, $G \vee H$. Clear from I.H. ∎

Lemma 6. *Let $E = \textsc{op}\langle\{\mathbf{x} : F(\mathbf{p}, \mathbf{x})\}\rangle \succeq b$ be an aggregate expression that has no free variables, and let S be the set of all the lists $\mathbf{c}$ of object constants of $\sigma(\Pi)$ whose length is $|\mathbf{x}|$ such that*

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models F^*(\mathbf{q}, \mathbf{c}).$$

We have that

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \textsc{op}\langle\{\mathbf{x} : F^*(\mathbf{q}, \mathbf{x})\}\rangle \succeq b$$

iff

$$\textsc{op}\langle\{\!\{c^1 : \mathbf{c} \in S\}\!\}\rangle \succeq b.$$

Proof. Clear from the definition. ∎

Lemma 7. *For any aggregate expression $E = \textsc{op}\langle\{\mathbf{x} : F(\mathbf{p}, \mathbf{x})\}\rangle \succeq b$ that contains no free variables,*

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \textsc{op}\langle\{\mathbf{x} : F^*(\mathbf{q}, \mathbf{x})\}\rangle \succeq b$$

iff

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \bigwedge_{C \in \overline{\mathcal{C}}(E)} \Big(\bigwedge_{\mathbf{c} \in C} F^*(\mathbf{q}, \mathbf{c}) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_{\Pi}(E) \setminus C} F^*(\mathbf{q}, \mathbf{c}) \Big)$$

Proof. *From left to right:* Assume $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \textsc{op}\langle\{\mathbf{x} : F^*(\mathbf{q}, \mathbf{x})\}\rangle \succeq b$. Let S be the set of all lists $\mathbf{c}$ of object constants of $\sigma(\Pi)$ whose length is $|\mathbf{x}|$ such that

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models F^*(\mathbf{q}, \mathbf{c}).$$

By Lemma 6, it follows from $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \textsc{op}\langle\{\mathbf{x} : F^*(\mathbf{q}, \mathbf{x})\}\rangle \succeq b$ that $\textsc{op}\langle\{\!\{c^1 : \mathbf{c} \in S\}\!\}\rangle \succeq b$. Consequently, S is not in $\overline{\mathcal{C}}(E)$.

Consider any C in $\overline{\mathcal{C}}(E)$, and assume that

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \bigwedge_{\mathbf{c} \in C} F^*(\mathbf{q}, \mathbf{c})$$

Clearly, C is a subset of S. Furthermore C is a strict subset since S is not in $\overline{\mathcal{C}}(E)$. Consequently,

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \bigvee_{\mathbf{c} \in \mathbf{O}_{\Pi}(E) \setminus C} F^*(\mathbf{q}, \mathbf{c}).$$

From right to left: Assume

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \bigwedge_{C \in \overline{\mathcal{C}}(E)} \Big(\bigwedge_{\mathbf{c} \in C} F^*(\mathbf{q}, \mathbf{c}) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_{\Pi}(E) \setminus C} F^*(\mathbf{q}, \mathbf{c}) \Big).$$

Again let S be the set of all lists $\mathbf{c}$ of object constants of $\sigma(\Pi)$ of length $|\mathbf{x}|$ such that $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models F^*(\mathbf{q}, \mathbf{c})$.

Clearly S is not in $\overline{\mathcal{C}}(E)$. Indeed if S is in $\overline{\mathcal{C}}(E)$, then

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \not\models \bigwedge_{\mathbf{c} \in S} F^*(\mathbf{q}, \mathbf{c}) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_{\Pi}(E) \setminus S} F^*(\mathbf{q}, \mathbf{c})$$

which contradicts the assumption. Consequently,

$$\textsc{op}\langle\{\!\{c^1 : \mathbf{c} \in S\}\!\}\rangle \succeq b.$$

By Lemma 6, we conclude that

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \textsc{op}\langle\{\mathbf{x} : F^*(\mathbf{q}, \mathbf{x})\}\rangle \succeq b.$$

▮

Proof of Proposition 3. It is clear that I is an *FLP* answer set of Π iff I is an *FLP* answer set of $Pos(\Pi)$. It is sufficient to show that for any semi-positive program Π and any aggregate expression E in $Ground(Pos(\Pi))$,

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models Fer_\Pi(E)^*(\mathbf{q})$$

iff

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models FLP_\Pi(E)^*(\mathbf{q}).$$

In view of Lemma 5, this is equivalent to saying that $Y \models Fer_\Pi(E)$ iff $Y \models FLP_\Pi(E)$. That is,

$$Y \models \bigwedge_{C \in \overline{\mathcal{C}}(E)} \Big(\bigwedge_{\mathbf{c} \in C} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_\Pi(E) \setminus C} F(\mathbf{c})\Big) \tag{24}$$

iff

$$Y \models \bigwedge_{I \in \overline{\mathcal{I}}(E)} \Big(\bigwedge_{A \in I} A \rightarrow \bigvee_{A \in HU_\Pi \setminus I} A\Big). \tag{25}$$

From left to right: Assume (24). Take any I in $\overline{\mathcal{I}}(E)$, and assume that

$$Y \models \bigwedge_{A \in I} A.$$

Clearly, $I \subseteq Y$. Furthermore we will check that $I \subset Y$. Consider $C = \{\mathbf{c} \mid I \models F(\mathbf{c})\}$. Clearly,

$$I \not\models \bigwedge_{\mathbf{c} \in C} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_\Pi(E) \setminus C} F(\mathbf{c}). \tag{26}$$

By Lemma 6, it follows that

$$I \not\models \textsc{op}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b \tag{27}$$

iff

$$\textsc{op}\langle\{\!\{c^1 : \mathbf{c} \in C\}\!\}\rangle \not\succeq b. \tag{28}$$

From the fact that $I \in \overline{\mathcal{I}}(E)$, we conclude (28). Consequently $C \in \overline{\mathcal{C}}(E)$ and by (24), we conclude

$$Y \models \bigwedge_{\mathbf{c} \in C} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_\Pi(E) \setminus C} F(\mathbf{c}). \tag{29}$$

In view of (26) and (29), we conclude that $I \neq Y$ and consequently, $I \subset Y$. Therefore there is an atom A in $HU_\Pi \setminus I$ that belongs to Y, that is,

$$Y \models \bigvee_{A \in HU_\Pi \setminus I} A.$$

From right to left: Assume (25). Take any C in $\overline{\mathcal{C}}(E)$, and assume that

$$Y \models \bigwedge_{\mathbf{c} \in C} F(\mathbf{c}).$$

Consider the set $\mathbf{I}$ of all set I of ground atoms of $\sigma(\Pi)$ such that

$$I \models \bigwedge_{\mathbf{c} \in C} F(\mathbf{c})$$

but

$$I \not\models \bigvee_{\mathbf{c} \in \mathbf{O}_\Pi(E) \setminus C} F(\mathbf{c}).$$

Clearly, for any $I \in \mathbf{I}$, (26) holds. By Lemma 7, it follows that

$$I \not\models \text{OP}\langle \{\mathbf{x} : F(\mathbf{x})\} \rangle \succeq b \tag{30}$$

iff

$$I \not\models \bigwedge_{C \in \overline{\mathcal{C}}(E)} \Big(\bigwedge_{\mathbf{c} \in C} F(\mathbf{c}) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_\Pi(E) \setminus C} F(\mathbf{c}) \Big).$$

In view of (26), we conclude (30). It follows that every I in $\mathbf{I}$ does not satisfy E. Consequently, $\mathbf{I}$ is a subset of $\overline{\mathcal{I}}_\Pi(E)$.

From (25) and, for every I in $\mathbf{I}$

$$I \not\models \bigwedge_{I \in \overline{\mathcal{I}}(E)} \Big(\bigwedge_{A \in I} A \rightarrow \bigvee_{A \in HU_\Pi \setminus I} A \Big),$$

we conclude that Y is not in $\mathbf{I}$. Therefore there is some $\mathbf{c} \in \mathbf{O}_\Pi(E) \setminus C$ such that $Y \models F(\mathbf{c})$, that is,

$$Y \models \bigvee_{\mathbf{c} \in \mathbf{O}_\Pi(E) \setminus C} F(\mathbf{c}).$$

∎

7.4 Proof of Lemma 1

Similar to the proof of Proposition 1.

7.5 Proof of Proposition 4

It is sufficient to prove that for every aggregate expression E occurring in program Π that contains no free variables,

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \bigwedge_{I \in \overline{\mathcal{I}}(E)} \Big(\bigvee_{A \in I} \neg A \vee \bigvee_{A \in HU_\Pi \setminus I} A^*(\mathbf{q}) \Big) \tag{31}$$

iff

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \bigvee_{\langle B,T \rangle \text{ is an LPS of } \mathcal{I}(E)} \Big(\bigwedge_{A \in B} A^*(\mathbf{q}) \wedge \bigwedge_{A \in HU_\Pi \setminus T} \neg A \Big). \tag{32}$$

From right to left: Assume (32). There exists an LPS $\langle B, T \rangle$ of $\mathcal{I}(E)$ such that

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \Big(\bigwedge_{A \in B} A^*(\mathbf{q}) \wedge \bigwedge_{A \in HU_\Pi \setminus T} \neg A \Big). \tag{33}$$

Take any I in $\overline{\mathcal{I}}(E)$. We can check that it is not the case that $B \subseteq I \subseteq T$. Indeed, if $B \subseteq I \subseteq T$, then I is in $\mathcal{I}(E)$, contradiction. Therefore two cases are possible.

Case 1: There is an element A that belongs to $B \cap (HU_\Pi \setminus I)$. From (33), $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models A^*(\mathbf{q})$. Consequently,

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \Big(\bigvee_{A \in I} \neg A \vee \bigvee_{A \in HU_\Pi \setminus I} A^*(\mathbf{q}) \Big). \tag{34}$$

Case 2: There is an element A that belongs to $I \cap (HU_\Pi \setminus T)$. From (33), $X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models \neg A$. Consequently, we get (34).

From left to right: Assume (31). We will prove that $\langle Y, X \rangle$ is an LPS of $\mathcal{I}(E)$, from which the claim follows since it is clear that

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models (\bigwedge_{A \in Y} A^*(\mathbf{q}) \wedge \bigwedge_{A \in HU_\Pi \setminus X} \neg A).$$

Take any set S of ground atoms such that $Y \subseteq S \subseteq X$. Clearly,

$$X \not\models \bigvee_{A \in S} \neg A$$

and

$$Y \not\models \bigvee_{A \in HU_\Pi \setminus S} A,$$

so that

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \not\models \bigvee_{A \in S} \neg A \vee \bigvee_{A \in HU_\Pi \setminus S} A^*(\mathbf{q}).$$

In view of (31), it follows that S is not in $\overline{\mathcal{I}}(E)$, that is, S is in $\mathcal{I}(E)$. Since this holds for every S such that $Y \subseteq S \subseteq X$, it follows that $\langle Y, X \rangle$ is an LPS of $\mathcal{I}(E)$. ∎

7.6 Proof of Lemma 2

We will prove a more general lemma that involves LPS representation of *SPT-PDB* which is

$$\bigwedge_{\langle B,T\rangle \text{ is an LPS of } \overline{\mathcal{I}}_{\Pi}(E)} \Big(\bigvee_{A\in B} \neg A \vee \bigvee_{A\in HU_{\Pi}\setminus T} A \Big). \tag{35}$$

Lemma 2′. *Formulas (13), (35), (15) are strongly equivalent to each other.*

Proof. It is sufficient to show that for any two subsets B and T of HU_{Π} such that B is a subset of T,

$$\bigwedge_{B\subseteq I\subseteq T} (\bigvee_{A\in I} \neg A \vee \bigvee_{A\in HU_{\Pi}\setminus I} A) \tag{36}$$

is equivalent to

$$\bigvee_{A\in B} \neg A \vee \bigvee_{A\in HU_{\Pi}\setminus T} A$$

in the Logic of Here-and-There.

From right to left: Clear since $B \subseteq I$ and $HU_{\Pi} \setminus T \subseteq HU_{\Pi} \setminus I$.

From left to right: Assume (36). Consider several cases, each of which corresponds to S s.t. $B \subseteq S \subseteq T$. The assumption characterizing each case is that $T \setminus S$ contains all atoms in T s.t. $\neg A$. Due to the assumption

$$\neg \bigvee_{A\in S} \neg A \wedge \bigwedge_{A\in T\setminus S} \neg A \tag{37}$$

follows. From (36),

$$\bigvee_{A\in S} \neg A \vee \bigvee_{A\in HU_{\Pi}\setminus S} A.$$

From this, in view of (37), we conclude

$$\bigvee_{A\in HU_{\Pi}\setminus S} A.$$

Since

$$\bigwedge_{A\in T\setminus S} \neg A$$

we conclude

$$\bigvee_{A\in HU_{\Pi}\setminus T} A.$$

∎

7.7 Proof of Proposition 5

Lemma 8. *Let F and G be aggregate formulas.*

$$\Big(\bigwedge_{I\in\overline{\mathcal{I}}_\Pi(E)}\Big(\bigwedge_{A\in I} A\rightarrow\bigvee_{A\in HU_\Pi\setminus I} A\Big)\wedge F\Big)\rightarrow G$$

is strongly equivalent to the conjunction of

$$\Big(\bigwedge_{I\in\mathbf{I}}\Big(\bigvee_{A\in I}\neg A\vee\bigvee_{A\in HU_\Pi\setminus I} A\Big)\wedge F\Big)\rightarrow\Big(G\vee\bigvee_{I\in\overline{\mathcal{I}}_\Pi(E)\setminus\mathbf{I}}\Big(\bigwedge_{A\in I} A\vee\bigwedge_{A\in HU_\Pi\setminus I}\neg A\Big)\Big) \quad (38)$$

for all subsets $\mathbf{I}$ of $\overline{\mathcal{I}}_\Pi(E)$.

Proof. By induction on cardinality n of $\overline{\mathcal{I}}(E)$.

When $n=1$,

$$[(\bigwedge_{A\in I} A\rightarrow\bigvee_{A\in HU_\Pi\setminus I} A)\wedge F]\rightarrow G$$

is strongly equivalent to the conjunction of

$$[(\bigvee_{A\in I}\neg A\vee\bigvee_{A\in HU_\Pi\setminus I} A)\wedge F]\rightarrow G$$

and

$$F\rightarrow(G\vee\bigwedge_{A\in I} A\vee\bigwedge_{A\in HU_\Pi\setminus I}\neg A).$$

Assume that the statement to prove holds when $n=k$.

Let $n=k+1$, and let I_1 be any element in $\overline{\mathcal{I}}(E)$.

$$[\bigwedge_{I\in\overline{\mathcal{I}}(E)}(\bigwedge_{A\in I} A\rightarrow\bigvee_{A\in HU_\Pi\setminus I} A)\wedge F]\rightarrow G$$

is strongly equivalent to the conjunction of

$$[\bigwedge_{I\in\overline{\mathcal{I}}(E)\setminus I_1}(\bigwedge_{A\in I} A\rightarrow\bigvee_{A\in HU_\Pi\setminus I} A)\wedge(\bigvee_{A\in I_1}\neg A\vee\bigvee_{A\in HU_\Pi\setminus I_1} A)\wedge F]\rightarrow G$$

and

$$[\bigwedge_{I\in\overline{\mathcal{I}}(E)\setminus I_1}(\bigwedge_{A\in I} A\rightarrow\bigvee_{A\in HU_\Pi\setminus I} A)\wedge F]\rightarrow(G\vee\bigwedge_{A\in I_1} A\vee\bigwedge_{A\in HU_\Pi\setminus I_1}\neg A)$$

The first formula by I.H. is strongly equivalent to the conjunction of

$$[\bigwedge_{I\in\mathbf{I}}(\bigvee_{A\in I}\neg A\vee\bigvee_{A\in HU_\Pi\setminus I} A)\wedge(\bigvee_{A\in I_1}\neg A\vee\bigvee_{A\in HU_\Pi\setminus I_1} A)\wedge F]\rightarrow[G\vee\bigvee_{I\in\overline{\mathcal{I}}_\Pi(E)\setminus\mathbf{I}}(\bigwedge_{A\in I} A\vee\bigwedge_{A\in HU_\Pi\setminus I}\neg A)]$$

for all subsets $\mathbf{I}$ of $\overline{\mathcal{I}}_\Pi(E) \setminus \{I_1\}$, which can be viewed as the conjunction of (38) that contains I_1.

By I.H. the second formula is strongly equivalent to the conjunction of

$$[\bigwedge_{I\in\mathbf{I}}(\bigvee_{A\in I}\neg A\vee \bigvee_{A\in HU_\Pi\setminus I} A)\wedge F] \rightarrow [G\vee \bigvee_{I\in\overline{\mathcal{I}}_\Pi(E)\setminus\mathbf{I}} (\bigwedge_{A\in I} A\vee \bigwedge_{A\in HU_\Pi\setminus I}\neg A)\vee(\bigwedge_{A\in I_1} A\vee \bigwedge_{A\in HU_\Pi\setminus I_1}\neg A)]$$

for all subsets $\mathbf{I}$ of $\overline{\mathcal{I}}_\Pi(E) \setminus \{I_1\}$, which can be view as the conjunction of (38) that does not contain I_1. ∎

Proof of Proposition 5. Follows from Lemma 8 since, when $\mathbf{I} = \overline{\mathcal{I}}_\Pi(E)$,

$$\bigwedge_{I\in\mathbf{I}}\Big(\bigvee_{A\in I}\neg A\vee \bigvee_{A\in HU_\Pi\setminus I} A\Big)$$

in the antecedent of (38) is exactly (13). ∎

7.8 Proof of Proposition 6

The proof uses the terminology and the Theorem on Double Negation from [19].

Let $E\wedge F \rightarrow H$ be a rule in a regular program Π, where E is an aggregate expression. We denote formula (13) by *Modified-FLP*$_\Pi(E)$ and (15) by *MLPS-Modified-FLP*$_\Pi(E)$.

On one hand, by Proposition 4, *SPT-PDB*$_\Pi(E) \wedge F \rightarrow H$ is strongly equivalent to *Modified-FLP*$_\Pi(E) \wedge F \rightarrow H$. By Lemma 2, *Modified-FLP*$_\Pi(E) \wedge F \rightarrow H$ is strongly equivalent to *MLPS-Modified-FLP*$_\Pi(E) \wedge F \rightarrow H$. On the other hand, by Proposition 1, *FLP*$_\Pi(E) \wedge F \rightarrow H$ is strongly equivalent to *MLPS-FLP*$_\Pi(E) \wedge F \rightarrow H$.

We will show that for any strongly connected component $\mathbf{c}$ of dependency graph of Π, *MLPS-FLP*$_\Pi(E)$ is contained in a subformula G of [*MLPS-FLP*$_\Pi(E)\wedge F \rightarrow H$] that is negative on $\mathbf{c}$. Consider any strongly connected component $\mathbf{c}$ of the dependency graph.

Case 1: There is an element A in $\mathbf{c}$ such that $A \in HU_\Pi \setminus T$ for some maximal LPS $\langle B, T\rangle$ of $\overline{\mathcal{I}}_\Pi(E)$. Since Π is regular, it follows that no elements in $\mathbf{c}$ appears in H. Thus *MLPS-FLP*$_\Pi(E) \wedge F \rightarrow H$ is negative on $\mathbf{c}$, and so is *MLPS-Modified-FLP*$_\Pi(E) \wedge F \rightarrow H$.

Case 2: There is no elements A in $\mathbf{c}$ such that $A \in HU_\Pi \setminus T$ for some maximal LPS $\langle B, T\rangle$ of $\overline{\mathcal{I}}_\Pi(E)$. In this case, *MLPS-FLP*$_\Pi(E)$ is negative on $\mathbf{c}$. For the same reason, *MLPS-Modified-FLP*$_\Pi(E)$ is negative on $\mathbf{c}$ in *MLPS-Modified-FLP*$_\Pi(E) \wedge F \rightarrow H$.

By the Theorem on Double Negations from [19], it follows that *MLPS-FLP*(Π) and *MLPS-Modified-FLP*(Π) have the same answer sets. ∎

7.9 Proof of Proposition 7

Let $E = \text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b$. It is sufficient to prove that

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models E^*(\mathbf{q})$$

iff

$$X \cup Y^{\mathbf{p}}_{\mathbf{q}} \models [\bigwedge_{C \in \overline{\mathcal{C}}(E)} (\bigwedge_{\mathbf{c} \in C} F(\mathbf{c}) \to \bigvee_{\mathbf{c} \in \mathbf{O}_{\Pi}(E) \setminus C} F(\mathbf{c}))]^*(\mathbf{q}).$$

This follows from Lemma 7 in conjunction with the fact that $E^*(\mathbf{p})$ is equivalent to E, and $Fer_{\Pi}(E)^*(\mathbf{p})$ is equivalent to $Fer_{\Pi}(E)$. ∎

7.10 Proof of Proposition 8

Lemma 9. *For arbitrary ground formula F, under unique name assumption, $NFES_F(Y)$ (as defined in [15]) is equivalent to $NES_F(Y)$ (as defined in [10]).*

Proof. We prove it by induction.
We will only show the case when F is an atom, the other cases are clear.
If F is an atom $p(\mathbf{t})$, $NFES_F(Y)$ is

$$p(\mathbf{t}) \wedge \bigwedge_{p(\mathbf{t}_1) \in Y} \mathbf{t} \neq \mathbf{t}_1. \tag{39}$$

Under the unique name assumption, (39) is equivalent to

- $\bot$ if $p(\mathbf{t}) \in Y$;
- $p(\mathbf{t})$ otherwise,

which is the same as definition of $NES_F(Y)$. ∎

Lemma 10. *For any aggregate expression $E = \text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b$ from a program Π with no free variables and any sets X, Y of ground atoms of $\sigma(\Pi)$, $X \models NFES_E(Y)$ iff $X \models NES_{Fer_{\Pi}(E)}(Y)$.*

Proof. $NFES_E(Y)$ is

$$(\text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b) \wedge (\text{OP}\langle\{\mathbf{x} : NFES_{F(\mathbf{x})}(Y)\}\rangle \succeq b). \tag{40}$$

By Lemma 7,

$$X \models \text{OP}\langle\{\mathbf{x} : F(\mathbf{x})\}\rangle \succeq b \wedge (\text{OP}\langle\{\mathbf{x} : NFES_{F(\mathbf{x})}(Y)\}\rangle \succeq b)$$

iff

$$X \models \Big(\bigwedge_{C \in \overline{\mathcal{C}}(E)} (\bigwedge_{\mathbf{c} \in C} F(\mathbf{c}) \to \bigvee_{\mathbf{c} \in \mathbf{O}_{\Pi}(E) \setminus C} F(\mathbf{c}))\Big) \wedge \Big(\bigwedge_{C \in \overline{\mathcal{C}}_{\Pi}(E)} (\bigwedge_{\mathbf{c} \in C} NFES_{F(\mathbf{c})}(Y) \to \bigvee_{\mathbf{c} \in \mathbf{O}_{\Pi}(E) \setminus C} NFES_{F(\mathbf{c})}(Y))\Big).$$

By Lemma 9, the latter is equivalent to saying that

$$X \models \mathit{Fer}_{\Pi}(E) \wedge \bigwedge_{C \in \overline{\mathcal{C}}_{\Pi}(E)} \Big(\bigwedge_{\mathbf{c} \in C} \mathit{NES}_{F(\mathbf{c})}(Y) \rightarrow \bigvee_{\mathbf{c} \in \mathbf{O}_{\Pi}(E) \setminus C} \mathit{NES}_{F(\mathbf{c})}(Y) \Big).$$

Thus

$$X \models \mathit{NES}_{\mathit{Fer}_{\Pi}(E)}(Y).$$

∎

Lemma 11. *Let Π be a program that contains no free variables, and let X, Y be sets of ground atoms of $\sigma(\Pi)$ such that $X \models \Pi$. Then $X \models ES_{\Pi}(Y)$ iff $X \models \neg \mathit{NES}_{\mathit{Fer}(\Pi)}(Y)$.*

Proof. Let r be a rule $A \leftarrow E_1, \dots, E_m, \mathit{not}\ E_{m+1}, \dots, \mathit{not}\ E_n, B$ in Π, where B is a list of standard literals and E_i are aggregate expressions.

$$X \models \neg \bigwedge_{r \in \Pi} \mathit{NES}_{\mathit{Fer}_{\Pi}(r)}(Y)$$

iff

$$\begin{gathered} X \models \textstyle\bigvee_{r \in \Pi} [\neg \mathit{NES}_A(Y) \wedge \mathit{NES}_{\mathit{Fer}_{\Pi}(E_1)}(Y) \wedge \ldots \wedge \mathit{NES}_{\mathit{Fer}_{\Pi}(E_m)}(Y) \wedge \\ \neg \mathit{Fer}_{\Pi}(E_{m+1}) \wedge \ldots \wedge \neg \mathit{Fer}_{\Pi}(E_n) \wedge \mathit{NES}_B(Y)]. \end{gathered}$$

By Lemma 10, the latter is equivalent to

$$X \models \bigvee_{r \in \Pi} [\neg \mathit{NES}_A(Y) \wedge \mathit{NFES}_{E_1}(Y) \wedge \ldots \wedge \mathit{NFES}_{E_m}(Y) \wedge \neg E_{m+1} \wedge \ldots \wedge \neg E_n \wedge \mathit{NES}_B(Y)].$$

Under the assumption that $X \models \Pi$, this is equivalent to

$$X \models \bigvee_{\substack{r \in \Pi \\ A \cap Y \neq \emptyset}} [\neg \mathit{NES}_A(Y) \wedge \mathit{NFES}_{E_1}(Y) \wedge \ldots \wedge \mathit{NFES}_{E_m}(Y) \wedge \neg E_{m+1} \wedge \ldots \wedge \neg E_n \wedge \mathit{NES}_B(Y)] \tag{41}$$

Indeed, if $A \cap Y = \emptyset$, then $\mathit{NES}_A(Y)$ is A, and the fact that

$$X \models \neg A \wedge \mathit{NFES}_{E_1}(Y) \wedge \ldots \wedge \mathit{NFES}_{E_m}(Y) \wedge \neg E_{m+1} \wedge \ldots \wedge \neg E_n \wedge \mathit{NES}_B(Y)$$

follows from the assumption that $X \models \Pi$.

By Lemma 9, (41) is equivalent to

$$X \models \bigvee_{\substack{r \in \Pi \\ A \cap Y \neq \emptyset}} \bigwedge_{i=1,\dots,m} \mathit{NFES}_{E_i}(Y) \wedge \bigwedge_{i=m+1,\dots,n} \neg E_i \wedge \mathit{NFES}_B(Y) \wedge \neg \bigvee_{p \in A \setminus Y} p.$$

∎

Lemma 12. *For any program Π, the dependency graph of Π (in our definition) is identical to the dependency graph of MLPS-Fer(Π) as defined in [10].*

Proof. It is clear from definition of the dependency graph. ∎

We prove a slight extension of Proposition $8'$.

Proposition $8'$. *Let Π be a program that has no free variables and that every $F(\mathbf{x})$ in every aggregate expression (3) occurring in it is an arbitrary quantifier free formula. For any set X of ground atoms of $\sigma(\Pi)$ that satisfies $Ground(\Pi)$, the following conditions are equivalent to each other.*

(a) X is a Ferraris answer set of Π;
(b) for every loop Y of Π, X satisfies the aggregate loop formula of Y for Π;
(c) for every loop Y of Π, X satisfies the PL representation of the aggregate loop formula of Y for Π;
(d) for every loop Y of $MLPS\text{-}Fer(\Pi)$ according to [10], X satisfies the loop formula of Y for $MLPS\text{-}Fer(\Pi)$ according to [10].

Proof. *Between (a) and (d):* By definition, X is an Ferraris answer set of Π iff X is an answer set of $Fer(\Pi)$. By Proposition 1, X is an answer set of $Fer(\Pi)$ iff X is an answer set of $MLPS\text{-}Fer(\Pi)$. According to Theorem 2 from [10], the latter is equivalent to the condition that X is a Herbrand model of Π that satisfies the loop formulas of all loops of $MLPS\text{-}Fer(\Pi)$ according to [10].

Between (b) and (d): By Lemma 11 and Lemma 12, X is a Herbrand model of $\sigma(\Pi)$ that satisfies the loop formulas of all loops of $MLPS\text{-}Fer(\Pi)$ according to [10] iff X satisfies aggregate loop formulas of all loops of Π (in our sense).

Between (b) and (c): Follows from Lemma 7. ∎