First-Order Extension of the FLP Stable Model Semantics via Modified Circumscription

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

Abstract

We provide reformulations and generalizations of both the semantics of logic programs by Faber, Leone and Pfeifer and its extension to arbitrary propositional formulas by Truszczyński. Unlike the previous definitions, our generalizations refer neither to grounding nor to fixpoints, and apply to first-order formulas containing aggregate expressions. In the same spirit as the first-order stable model semantics proposed by Ferraris, Lee and Lifschitz, the semantics proposed here are based on syntactic transformations that are similar to circumscription. The reformulations provide useful insights into the FLP semantics and its relationship to circumscription and the first-order stable model semantics.

1 Introduction

The stable model semantics is the mathematical basis of answer set programming, and is one of the most well-studied knowledge representation formalisms. Lifschitz [2010] surveys thirteen different definitions of a stable model presented in the literature. These definitions are equivalent to each other when they are applied to normal logic programs, but are not necessarily so for more general classes of programs. However, each of them deserves its own attention, as it provides useful insights into the stable model semantics and answer set programming.

The semantics defined by Faber, Leone and Pfeifer [2004; 2011] (called the FLP semantics) deserves special attention as it provides a simple satisfactory solution to the semantics of aggregates, and is implemented in the system DLV (`http://www.dlvsystem.com`). It is also a basis of HEX programs, which are an extension of answer set programs towards integration of external sources of information that have possibly heterogeneous semantics [Eiter *et al.*, 2005]. Also, Dao-Tran *et al.* [2009] remark that the FLP semantics provides a more natural basis for their *Modular Logic Programs (MLP)* than the traditional Gelfond-Lifschitz semantics [Gelfond and Lifschitz, 1988].

The idea of the FLP semantics is based on an interesting modification to the traditional definition of a reduct by Gelfond and Lifschitz [1988]. The FLP-reduct of a program Π relative to a set X of atoms is obtained from Π by simply removing all rules whose bodies are not satisfied by X. After then the same minimality condition as in the original definition of an answer set applies. For example, consider the following program Π_1:

$$\begin{array}{ll} p \leftarrow \mathit{not}\ q & \qquad r \leftarrow p \\ q \leftarrow \mathit{not}\ p & \qquad r \leftarrow q\ . \end{array} \tag{1}$$

The FLP-reduct of Π_1 relative to $X = \{p, r\}$ is

$$p \leftarrow \mathit{not}\ q \qquad\qquad r \leftarrow p, \tag{2}$$

and X is minimal among the sets of atoms that satisfy (2), and hence is an answer set of Π_1. Theorem 3.6 from [Faber *et al.*, 2011] asserts that this definition of an answer set is equivalent to the traditional definition when it is applied to the syntax of usual disjunctive programs. Indeed, the GL-reduct of Π_1 relative to X [Gelfond and Lifschitz, 1988] is

$$\begin{array}{ll} p \leftarrow & \qquad r \leftarrow p \\ & \qquad r \leftarrow q \end{array} \tag{3}$$

and, again, $\{p, r\}$ is minimal among the sets of atoms that satisfy (3).

The FLP semantics was recently extended to *arbitrary* propositional formulas by Truszczyński [2010], based on the definition of a reduct that is similar to the one proposed by Ferraris [2005]. However, his extension is still limited as it allows neither variables nor aggregates.

In this paper, we extend the FLP semantics and its extension by Truszczyński, both syntactically and semantically. We consider the syntax of arbitrary first-order formulas allowing aggregates. Instead of referring to grounding and fixpoints, our generalized semantics are given in terms of modifications to circumscription [McCarthy, 1980], in the same spirit as the first-order stable model semantics by Ferraris, Lee and Lifschitz [Ferraris *et al.*, 2007; 2011]. Such uniform characterizations in the form of modified circumscription tell us how the FLP semantics and its extension by Truszczyński are related to circumscription and to the first-order stable model semantics.

In Section 2 we review the original FLP semantics, and present our generalization. Likewise, in Section 3 we review and generalize the extension of the FLP semantics by Truszczyński. In Section 4 we compare these generalizations with the first-order stable model semantics.

2 FLP Semantics

2.1 Review: Original FLP Semantics

A *disjunctive rule* is an expression of the form

$$A_1\,;\dots;\,A_l \leftarrow\; A_{l+1},\dots,A_m, \textit{not}\ A_{m+1},\dots,\textit{not}\ A_n \quad (4)$$

($n \geq m \geq l \geq 0$) where each A_i is an atomic formula (possibly containing equality and 0-place connectives $\top$ and $\bot$). A *disjunctive program* is a set of disjunctive rules. The FLP semantics for a disjunctive program in [Faber *et al.*, 2011] is given in terms of grounding and fixpoints. By $\sigma(\Pi)$ we denote the signature consisting of the object, function and predicate constants occurring in Π. We denote by *Ground*(Π) the ground instance of Π, that is, the program obtained from Π by replacing every occurrence of object variables with every ground term that can be constructed from $\sigma(\Pi)$, and then replacing equality $t = t'$ with $\top$ or $\bot$ depending on whether term t is the same symbol as term t'. Given a set X of ground atoms of $\sigma(\Pi)$, the reduct of Π relative to X, denoted by $\Pi^{\underline{X}}$, is obtained from *Ground*(Π) by removing every rule whose body is not satisfied by X. Set X is called an ***FLP-answer set*** of Π if X is minimal among the sets of atoms that satisfy $\Pi^{\underline{X}}$ (viewed as a formula in propositional logic). For example, for Π_1 and $X = \{p, r\}$ in the introduction, $\Pi_1^{\underline{X}}$ is (2) and X is an FLP-answer set of Π_1.

2.2 Extension: First-Order FLP Semantics

We present a reformulation of the FLP semantics in the first-order case. First, we consider a program that does not contain aggregates, but allows rules of a more general form than (4). We assume the following set of primitive propositional connectives and quantifiers in forming formulas:

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

$\neg F$ is an abbreviation for $F \rightarrow \bot$, symbol $\top$ stands for $\bot \rightarrow \bot$, and $F \leftrightarrow G$ stands for $(F \rightarrow G) \wedge (G \rightarrow F)$.

A *(general) rule* is of the form

$$H \leftarrow B \quad (5)$$

where H and B are arbitrary formulas in first-order logic. Rule (4) is a special case of rule (5) when we identify *not* with $\neg$, the head H with the disjunction of atomic formulas and the body B with the conjunction of atomic formulas, possibly preceded by negation. A *(general) program* is a set of (general) rules.

Let $\mathbf{p}$ be a list of distinct predicate constants $p_1,\dots,p_n$, and let $\mathbf{u}$ be a list of distinct predicate variables $u_1,\dots,u_n$. By $\mathbf{u} \leq \mathbf{p}$ we denote the conjunction of the formulas $\forall\mathbf{x}(u_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))$ for all $i = 1,\dots,n$ where $\mathbf{x}$ is a list of distinct object variables of the same length as the arity of p_i, and by $\mathbf{u} < \mathbf{p}$ we denote $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{p} \leq \mathbf{u})$. For instance, if p and q are unary predicate constants then $(u, v) < (p, q)$ is

$$\begin{array}{l}\forall x(u(x) \rightarrow p(x)) \wedge \forall x(v(x) \rightarrow q(x)) \\ \quad \wedge\neg(\forall x(p(x) \rightarrow u(x)) \wedge \forall x(q(x) \rightarrow v(x))).\end{array}$$

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

Let Π be a finite program whose rules have the form (5). The *FOL-representation* Π^{FOL} of Π is the conjunction of the universal closures of $B \rightarrow H$ for all rules (5) in Π. By FLP[Π; $\mathbf{p}$] we denote the second-order formula

$$\Pi^{FOL} \wedge \neg\exists\mathbf{u}(\mathbf{u} < \mathbf{p} \wedge \Pi^{\triangle}(\mathbf{u})) \quad (6)$$

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

$$\forall\mathbf{x}(B \wedge B(\mathbf{u}) \rightarrow H(\mathbf{u})) \quad (7)$$

for all rules (5) in Π, where $\mathbf{x}$ is the list of all (free) variables in (5).

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

Example 1 *Consider the program* Π_1 *in the introduction. Its FOL-representation* Π_1^{FOL} *is*

$$(\neg q \rightarrow p) \wedge (\neg p \rightarrow q) \wedge (p \rightarrow r) \wedge (q \rightarrow r)$$

and $\Pi_1^{\triangle}(u, v, w)$ *is*

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

Formula

$$\Pi_1^{FOL} \wedge \neg\exists uvw((u, v, w) < (p, q, r) \wedge \Pi_1^{\triangle}(u, v, w))$$

can be equivalently rewritten without second-order variables as $\neg(p \leftrightarrow q) \wedge (r \leftrightarrow p \vee q)$.

Though rule arrows ($\leftarrow$) are treated like usual implications in the FOL-representation of Π, they are distinguished in the definition of $\Pi^{\triangle}(\mathbf{u})$ because of the presence of '$B\wedge$' in (7). If we modify $\Pi^{\triangle}(\mathbf{u})$ by dropping '$B\wedge$' from (7), then (6) becomes exactly CIRC[Π^{FOL}; $\mathbf{p}$] [McCarthy, 1980]. Interestingly, this small change accounts for the differences in the two semantics, one leading to stable models and the other leading to minimal models. For instance, classically equivalent transformations preserve minimal models, but not FLP-stable models. Each of the one-rule programs $\{p \leftarrow \neg q\}$, $\{q \leftarrow \neg p\}$, and $\{p \vee q \leftarrow \top\}$ are classically equivalent to each other (when we identify them with their FOL-representations), but their FLP-stable models are different. Formula FLP[$p \leftarrow \neg q;\ p, q$] is equivalent to $p \wedge \neg q$; formula FLP[$q \leftarrow \neg p;\ p, q$] is equivalent to $q \wedge \neg p$; formula FLP[$p \vee q \leftarrow \top;\ p, q$] is equivalent to $(p \vee q) \vee \neg(p \wedge q)$, which is also equivalent to CIRC[$p \vee q;\ p, q$].

The following theorem states that our semantics is a proper generalization of the semantics from [Faber *et al.*, 2011].

Theorem 1 *Let* Π *be a finite disjunctive program (consisting of rules of the form (4)) containing at least one object constant. The FLP-answer sets of* Π *are precisely the Herbrand models of* FLP[Π] *whose signature is* $\sigma(\Pi)$.

It is known that the FLP semantics from [Faber *et al.*, 2011] has the anti-chain property: no FLP-answer set is a proper subset of another FLP-answer set. This property is still preserved in our generalized semantics.

Proposition 1 *For any finite general program* Π*, if* I *is an Herbrand interpretation of* $\sigma(\Pi)$ *that satisfies* FLP[Π]*, then* I *is a subset-minimal model of* Π.[1]

[1]We identify an Herbrand interpretation I with the set of ground atoms that are satisfied by I.

Clearly, circumscription can be viewed as a special case of the FLP semantics as in the following.

Proposition 2 *For any first-order sentence F and any list $\mathbf{p}$ of predicate constants,* CIRC$[F;\mathbf{p}]$ *is equivalent to* FLP$[F \leftarrow \top;\ \mathbf{p}]$.

The FLP semantics can be represented by circumscription in the following way.

Proposition 3 *For any finite general program Π, formula* FLP$[\Pi;\mathbf{p}]$ *is equivalent to* $\exists\mathbf{u}(\text{CIRC}[\Pi^{\triangle}(\mathbf{u});\mathbf{u}] \wedge (\mathbf{u}=\mathbf{p}))$.

2.3 Extension: First-Order FLP Semantics for Programs with Aggregates

The semantics presented in the previous section can be extended to allow aggregates by simply extending the notion of satisfaction to cover aggregate expressions. Below we adopt the definitions of an aggregate formula and satisfaction as given in [Lee and Meng, 2009; Ferraris and Lifschitz, 2010].

Following [Ferraris and Lifschitz, 2010], by a *number* we understand an element of some fixed set **Num**. For example, **Num** is $\mathbf{Z} \cup \{+\infty, -\infty\}$, where $\mathbf{Z}$ is the set of integers. An *aggregate function* is a partial function from the class of multisets to **Num**. The domain of an aggregate function is defined as usual. For instance, COUNT is defined for any multisets; SUM, TIMES, MIN and MAX are defined for multisets of numbers; SUM is undefined for multisets containing infinitely many positive integers and infinitely many negative integers.

We assume that the signature σ contains symbols for all numbers, and some collection of *comparison operators* that stands for binary relations over numbers, such as $\leq, \geq, <, >, =$ and $\neq$. We assume that symbols for aggregate functions are not part of the signature.

An *aggregate expression* of signature σ is of the form [2]

$$\text{OP}\langle \mathbf{x} : F(\mathbf{x})\rangle \succeq b \qquad (8)$$

where

- OP is an *aggregate function*;
- $\mathbf{x}$ is a nonempty list of distinct object variables;
- $F(\mathbf{x})$ is a first-order formula;
- $\succeq$ is a comparison operator;
- b is a term (as in first-order logic).

We define an *aggregate formula* as an extension of a first-order formula by including aggregate expressions as a base case like (standard) atomic formulas (including equality and $\bot$). 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 is *bound* if the occurrence is in a part of H of the form $\langle \mathbf{x} : F(\mathbf{x})\rangle$ 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. We consider only the interpretations such that each number and each comparison operator is interpreted as itself. 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 ξ^*, called *names*, for all ξ in the universe of I. We identify an interpretation I of σ with its extension to $\sigma^{|I|}$ defined by $I(\xi^*) = \xi$.

The definition of satisfaction in first-order logic is extended to aggregate sentences as follows. Let I be an interpretation of signature σ. Consider any aggregate expression (8) that has no free variables. Let S_I be the multiset consisting of all $\boldsymbol{\xi}^*[1]$ (i.e. the first element of $\boldsymbol{\xi}^*$), in the universe of I where

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

An interpretation I satisfies the aggregate expression if S_I is in the domain of OP, and $\text{OP}(S_I) \succeq b^I$. With this extension, the recursive definition of satisfaction for an aggregate sentence is given in the same way as in first-order logic. We say that an aggregate sentence F is *logically valid* if every interpretation satisfies it. For instance, an Herbrand interpretation $\{p(a)\}$ satisfies $\text{COUNT}\langle x : p(x)\rangle > 0$ but does not satisfy $\text{SUM}\langle x : p(x)\rangle > 0$ because multiset $\{\!\{a\}\!\}$ is not in the domain of SUM. Consider the aggregate expression

$$\text{SUM}\langle x : p(x)\rangle \geq 0$$

and an Herbrand interpretation $I = \{p(-1), p(1)\}$. S_I is $\{\!\{-1, 1\}\!\}$ and $\text{SUM}(S_I) = 0 \geq 0$, so I satisfies $\text{SUM}\langle x : p(x)\rangle \geq 0$.

Once we extend the definition of satisfaction to aggregate sentences, we can simply extend the FLP semantics in Section 2.2 to a *general program with aggregates*, whose rules have the form

$$H \leftarrow B \qquad (9)$$

where H and B are aggregate formulas. The *AF-representation* ("Aggregate Formula representation") of a finite general program Π with aggregates is the conjunction of the universal closures of the aggregate formulas $B \rightarrow H$ for all rules (9) in Π. Formula FLP$[\Pi;\mathbf{p}]$ is defined the same as (6) except that Π is now understood as a general program with aggregates.

Example 2 *Consider the following general program Π with aggregates:*

$$\begin{array}{rcl} p(2) & \leftarrow & \neg\,\text{SUM}\langle x : p(x)\rangle < 2 \\ p(-1) & \leftarrow & \text{SUM}\langle x : p(x)\rangle \geq 0 \\ p(1) & \leftarrow & p(-1)\ . \end{array} \qquad (10)$$

The AF-representation of Π, denoted by Π^{AF}, is the following:

$$\begin{array}{l} (\neg(\text{SUM}\langle x{:}p(x)\rangle{<}2) \rightarrow p(2)) \\ \wedge\ (\text{SUM}\langle x{:}p(x)\rangle{\geq}0 \rightarrow p(-1)) \\ \wedge\ (p(-1) \rightarrow p(1))\ . \end{array} \qquad (11)$$

[2]The syntax of an aggregate expression considered in Ferraris and Lifschitz [2010] is more general. The results in this paper can be extended to the general syntax, which we omit for simplicity.

The FLP-stable models of (10) are the models of

$$\Pi^{AF} \wedge \neg\exists u(u < p \wedge \Pi^{\triangle}(u)) \qquad (12)$$

where $\Pi^{\triangle}(u)$ *is*

$$\begin{aligned}&(\neg(\text{SUM}\langle x\!:\!p(x)\rangle < 2) \wedge \neg(\text{SUM}\langle x\!:\!u(x)\rangle < 2) \to u(2))\\ &\wedge (\text{SUM}\langle x\!:\!p(x)\rangle \geq 0 \wedge \text{SUM}\langle x\!:\!u(x)\rangle \geq 0 \to u(-1))\\ &\wedge (p(-1) \wedge u(-1) \to u(1))\ .\end{aligned} \qquad (13)$$

Below we show how this semantics is related to the original semantics by Faber *et al.* [2011]. Their semantics is defined for a special class called *disjunctive programs with aggregates*, whose rules have the form

$$A_1; \dots; A_l \leftarrow\ E_1, \dots, E_m, \textit{not}\ E_{m+1}, \dots, \textit{not}\ E_n \qquad (14)$$

($l \geq 0$; $n \geq m \geq 0$), where each A_i is an atomic formula and each E_i is an atomic formula or an aggregate expression.

As before, the original FLP semantics is defined in terms of grounding and fixpoints. Let us assume that b in every aggregate expression (8) is a constant. We extend the notion *Ground*(Π) to a disjunctive program Π with aggregates by replacing every free occurrence of a variable with every ground term that can be constructed from $\sigma(\Pi)$ in all possible ways.

For any disjunctive program Π with aggregates and any Herbrand interpretation X whose signature is $\sigma(\Pi)$, the *FLP-reduct* of Π relative to X is obtained from *Ground*(Π) 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 [Faber *et al.*, 2011]. For example, in program (10) above, the FLP-reduct of (10) relative to $\{p(-1), p(1)\}$ contains the last two rules only. Set $\{p(-1), p(1)\}$ is minimal among the sets of atoms that satisfy the reduct, and thus is an FLP-answer set of (10). In fact, this is the only FLP-answer set. Also one can check that $\{p(-1), p(1)\}$ is the only Herbrand model of $\sigma(\Pi)$ that satisfies (12) in Example 2.

Theorem 2 *Let* Π *be a finite disjunctive program with aggregates that contains at least one object constant. The FLP-answer sets of* Π *are precisely the Herbrand models of* FLP[Π] *whose signature is* $\sigma(\Pi)$.

3 Truszczyński Semantics

3.1 Review: Original Truszczyński Semantics

Truszczyński [2010] defined an extension of the FLP semantics to arbitrary propositional formulas, similar to the extension of the stable model semantics to arbitrary propositional formulas by Ferraris [2005], whose idea is also related to [Pearce, 1997; Osorio *et al.*, 2004].

For any propositional formula F, the FLPT-reduct $F^{\underline{X}}$ relative to a set X of atoms is defined recursively:

- $A^{\underline{X}} = \begin{cases} A & \text{if } X \models A, \\ \bot & \text{otherwise;} \end{cases}$
- $\bot^{\underline{X}} = \bot$;
- $(G \odot H)^{\underline{X}} = \begin{cases} G^{\underline{X}} \odot H^{\underline{X}} & \text{if } X \models G \odot H \quad (\odot \in \{\wedge, \vee\}), \\ \bot & \text{otherwise;} \end{cases}$
- $(G \to H)^{\underline{X}} = \begin{cases} G \to H^{\underline{X}} & \text{if } X \models G, \text{ and } X \models H, \\ \top & \text{if } X \not\models G, \\ \bot & \text{otherwise.} \end{cases}$

Set X is an *FLPT-answer set* of F if X is minimal among the sets of atoms that satisfy $F^{\underline{X}}$.

3.2 Extension: FLPT Semantics for First-Order Formulas with Aggregates

We extend the FLPT semantics to arbitrary first-order formulas that allow aggregates.

For any first-order formula F with aggregates and any finite list of predicate constants $\mathbf{p} = (p_1, \dots, p_n)$, formula FLPT[$F; \mathbf{p}$] is defined as

$$F \wedge \neg\exists \mathbf{u}(\mathbf{u} < \mathbf{p} \wedge F^{\Box}(\mathbf{u})) \qquad (15)$$

where $F^{\Box}(\mathbf{u})$ is defined recursively, as follows:

- $p_i(\mathbf{t})^{\Box} = u_i(\mathbf{t})$ for any tuple $\mathbf{t}$ of terms;
- $F^{\Box} = F$ for any atomic formula F that does not contain members of $\mathbf{p}$;
- $(G \odot H)^{\Box} = G^{\Box} \odot H^{\Box}$, where $\odot \in \{\wedge, \vee\}$;
- $(G \to H)^{\Box} = (G \wedge G(\mathbf{u}) \to H^{\Box}) \wedge (G \to H)$;
- $(QxG)^{\Box} = QxG^{\Box}$, where $Q \in \{\forall, \exists\}$;
- $(\text{OP}\langle \mathbf{x} : G)\rangle \succeq t)^{\Box} =$
  $(\text{OP}\langle \mathbf{x} : G\rangle \succeq t) \wedge (\text{OP}\langle \mathbf{x} : G(\mathbf{u})\rangle \succeq t)$.

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

The following theorem states that our semantics is a proper extension of the Truszczyński semantics to first-order formulas with aggregates. For any formula F, by $\sigma(F)$ we denote the signature consisting of object, function and predicate constants occurring in F.

Theorem 3 *For any propositional formula F, the FLPT-answer sets of F are precisely the interpretations of* $\sigma(F)$ *that satisfy* FLPT[F].

Also the semantics above coincides with our extension of the FLP semantics in Section 2.3 when it is applied to disjunctive programs with aggregates (having rules of the form (14)).

Proposition 4 *For any finite disjunctive program* Π *with aggregates and the AF-representation F of* Π, FLP[$\Pi; \mathbf{p}$] *is equivalent to* FLPT[$F; \mathbf{p}$].

However, the statement of the proposition does not apply to general programs.

Example 3 *For general program* $\Pi = \{p \vee \neg p \leftarrow \top\}$ *and its FOL-representation F, formula* FLP[Π] *has only one model,* $\emptyset$*, and* FLPT[F] *has two models,* $\emptyset$ *and* $\{p\}$.

In comparison with Proposition 1, this example illustrates that, unlike the FLP semantics, the FLPT semantics does not keep the anti-chain property. This has to do with the fact the FLP semantics distinguishes between rule arrows and the other implications, while the FLPT semantics does not. The first-order stable model semantics from [Ferraris *et al.*, 2011] also lacks the anti-chain property. In fact, the FLPT semantics has similar properties with that semantics. For instance, the following theorem is similar to Theorem 2 from [Ferraris *et al.*, 2011]. By *Choice*($\mathbf{p}$) where $\mathbf{p}$ is a list of predicate constants, we denote the conjunction of "choice formulas" $\forall \mathbf{x}(p(\mathbf{x}) \vee \neg p(\mathbf{x}))$ for all predicate constants p in $\mathbf{p}$ where $\mathbf{x}$ is a list of distinct object variables whose length is the same as the arity of p.

Theorem 4 *For any first-order formula F with aggregates and any disjoint lists $\mathbf{p}$, $\mathbf{q}$ of distinct predicate constants, the following formulas are logically valid:*

$$\text{FLPT}[F; \mathbf{pq}] \rightarrow \text{FLPT}[F; \mathbf{p}],$$
$$\text{FLPT}[F \wedge \textit{Choice}(\mathbf{q}); \mathbf{pq}] \leftrightarrow \text{FLPT}[F; \mathbf{p}].$$

For example, FLPT$[\forall x(q(x) \rightarrow p(x) \vee \neg p(x));\ p]$ is equivalent to

$$\text{FLPT}[\forall x(q(x) \rightarrow p(x) \vee \neg p(x)) \wedge \forall x(q(x) \vee \neg q(x));\ p, q].$$

The following theorem is similar to Theorem 3 from [Ferraris *et al.*, 2011].

Theorem 5 *For any first-order formulas F and G with aggregates and any list of predicate constants $\mathbf{p}$,* FLPT$[F \wedge \neg G;\ \mathbf{p}]$ *is equivalent to* FLPT$[F;\ \mathbf{p}] \wedge \neg G$*.*

It follows that the FLPT-stable models of $F \wedge \neg G$ can be characterized as the FLPT-stable models of F that satisfy $\neg G$.

The notion of strong equivalence has turned out to be important in the theory of stable models. Similar to the characterization of strong equivalence in terms of *HT-models* [Lifschitz *et al.*, 2001; Ferraris *et al.*, 2011], Truszczyński [2010] defined "FLP-models," a counterpart of HT-models in the FLPT semantics, and used them to characterize strong equivalence between propositional formulas under the FLPT semantics. In the following, we extend the result to arbitrary first-order formulas with aggregates.[3]

Following the definition of strong equivalence in the first-order stable model semantics in [Ferraris *et al.*, 2011], about first-order formulas with aggregates F and G, we say that F is *FLPT-strongly equivalent* to G if, for any formula H with aggregates, any occurrence of F in H, and any list $\mathbf{p}$ of distinct predicate constants, FLPT$[H; \mathbf{p}]$ is equivalent to FLPT$[H'; \mathbf{p}]$, where H' is obtained from H by replacing the occurrence of F by G. The following theorem, which is similar to Theorem 9 from [Ferraris *et al.*, 2011], is a proper extension of Theorem 7 from [Truszczyński, 2010].

Theorem 6 *Let F and G be first-order formulas with aggregates, let $\mathbf{p}^{FG}$ be the list of all predicate constants occurring in F or G and let $\mathbf{u}$ be a list of distinct predicate variables. The following conditions are equivalent to each other.*

- *F and G are FLPT-strongly equivalent to each other;*
- *Formula*

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

  is logically valid.

[3]Due to lack of space, we do not present the extension of FLP models, but instead present an alternative characterization in terms of $F^{\Box}$.

As a special case, Theorem 6 can be applied to checking strong equivalence under the FLP semantics between the programs whose rules have the form (14).

4 Comparing FLP, FLPT and the First-Order Stable Model Semantics

In [Ferraris *et al.*, 2011] the stable models are defined in terms of the SM operator with *intensional* predicates: for any first-order sentence F and any list of intensional predicates $\mathbf{p} = (p_1, \dots, p_n)$, formula SM$[F; \mathbf{p}]$ is defined as

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

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

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ for any list $\mathbf{t}$ of terms;
- $F^* = F$ for any atomic formula F that does not contain members of $\mathbf{p}$;
- $(G \odot H)^* = G^* \odot H^*$, where $\odot \in \{\wedge, \vee\}$;
- $(G \rightarrow H)^* = (G^* \rightarrow H^*) \wedge (G \rightarrow H)$;
- $(QxG)^* = QxG^*$, where $Q \in \{\forall, \exists\}$;
- $(\text{OP}\langle \mathbf{x} : G \rangle \succeq b)^* =$
  $(\text{OP}\langle \mathbf{x} : G^* \rangle \succeq b) \wedge (\text{OP}\langle \mathbf{x} : G \rangle \succeq b)$.

We often simply write SM$[F]$ in place of SM$[F; \mathbf{p}]$ when $\mathbf{p}$ is the list of all predicate constants occurring in F, and call a model of SM$[F]$ simply a *stable* model of F.

Disregarding aggregate expressions, the main difference among the FLP, the FLPT, and the first-order stable model semantics has to do with the treatment of an implication. It is known that they coincide for programs whose rules have the form (4) [Faber *et al.*, 2011, Theorem 3.6], [Truszczyński, 2010, Theorem 3]. However, this is not the case for more general classes of programs (having rules of the form (5)), or for arbitrary formulas. In fact, no one is stronger than another, as the following example shows.

Example 4 *For propositional signature $\{p\}$ and program $\Pi_1 = \{p \leftarrow p \vee \neg p\}$, whose FOL-representation is $F_1 = p \vee \neg p \rightarrow p$, each of* FLP$[\Pi_1]$ *and* FLPT$[F_1]$ *has $\{p\}$ as the only model, and* SM$[F_1]$ *has no models.*

Formula F_1 is strongly equivalent (in the sense of [Ferraris et al.*, 2011]), but not FLPT-strongly equivalent to $F_2 = (p \rightarrow p) \wedge (\neg p \rightarrow p)$. Again,* SM$[F_2]$ *has no models. Neither does* FLP$[\Pi_2]$ *nor* FLPT$[F_2]$*, where Π_2 is the program corresponding to F_2.*

For program $\Pi_3 = \{p \vee \neg p \leftarrow \top\}$, whose FOL-representation is $F_3 = \top \rightarrow p \vee \neg p$, both SM$[F_3]$ *and* FLPT$[F_3]$ *have two models, $\emptyset$ and $\{p\}$, while* FLP$[\Pi_3]$ *has only one model, $\emptyset$.*

Formula F_3 is strongly equivalent, but not FLPT-strongly equivalent to $F_4 = \neg\neg p \rightarrow p$. Both FLP$[\Pi_4]$ *(Π_4 is the program corresponding to F_4) and* FLPT$[F_4]$ *have only one model, $\emptyset$, while* SM$[F_4]$ *has the same two models as* SM$[F_3]$*.*

For program $\Pi_5 = \{\neg\neg p \leftarrow \top,\ p \vee \neg p \leftarrow \neg\neg p\}$, and its FOL-representation F_5, both FLP$[\Pi_5]$ *and* SM$[F_5]$ *have only one model $\{p\}$, while* FLPT$[F_5]$ *has no models.*

We note that FLPT-strong equivalence involves some unintuitive cases. Formulas $F \wedge G$ and $F \wedge (F \rightarrow G)$ are strongly equivalent under the first-order stable model semantics, but this is not the case under the FLPT semantics. The two formulas may not even have the same FLPT-stable models.

Example 5 *Similar to program Π_5 in the previous example, for propositional signature $\{p\}$ and $F = \neg\neg p$ and $G = p \vee \neg p$, formulas* $\mathrm{SM}[F \wedge G]$ *and* $\mathrm{FLPT}[F \wedge G]$ *are equivalent to each other, having only one model, $\{p\}$. Formula* $\mathrm{SM}[F \wedge (F \rightarrow G)]$ *has the same model as* $\mathrm{SM}[F \wedge G]$*, but* $\mathrm{FLPT}[F \wedge (F \rightarrow G)]$ *has no models.*

We now show the relationships among the three semantics. Roughly speaking, the FLPT semantics is in between the two others in the sense that it treats a "non-strictly positive" occurrence of a subformula same as in the FLP semantics, and treats a "strictly positive" occurrence of a subformula the same way as in the first-order stable model semantics. The definition of a strictly positive occurrence is given below.

First we present a class of programs for which the FLP semantics and the FLPT semantics coincide. Following [Ferraris and Lifschitz, 2010], we say that an aggregate function OP is *monotone* w.r.t. $\succeq$ if for any multisets α, β such that $\alpha \subseteq \beta$,

- if $\mathrm{OP}(\alpha)$ is defined then so is $\mathrm{OP}(\beta)$, and
- for any $n \in \mathbf{Num}$, if $\mathrm{OP}(\alpha) \succeq n$ then $\mathrm{OP}(\beta) \succeq n$.

For an occurrence of a predicate constant or any other subexpression in a formula F with aggregates, we consider two numbers, k and m.

- k: the number of implications in F that contain that occurrence in the antecedent;
- m: the number of aggregate expressions (8) containing that occurrence such that OP is not monotone w.r.t. $\succeq$.

We call an occurrence of a subexpression in F *strictly positive* if $k + m$ for that occurrence in F is 0. For example, in formula $(p \rightarrow q) \rightarrow p$, the second occurrence of p is strictly positive. In $\neg(\mathrm{SUM}\langle x : p(x)\rangle < 2)$, the occurrence of p is not strictly positive (for that occurrence, $k = m = 1$).

The following theorem presents a class of programs for which the FLP semantics and the FLPT semantics coincide.

Theorem 7 *Let Π be a finite general program with aggregates, and let F be the AF-representation of Π. For every rule (9) in Π, if every occurrence of p from $\mathbf{p}$ in H is strictly positive in H, then* $\mathrm{FLP}[\Pi; \mathbf{p}]$ *is equivalent to* $\mathrm{FLPT}[F; \mathbf{p}]$.

The theorem is a generalization of Proposition 4. For example, the FLP and the FLPT semantics coincide on the programs whose heads have the form of a disjunction of atoms, regardless of the form of the formulas in the body. In Example 4, programs Π_1, Π_2 and Π_4 satisfy the condition of Theorem 7, and so does program (10).

Next we show the class of programs for which the FLP semantics and the stable model semantics coincide. We first define two notions. We call an aggregate formula *semi-positive relative to* $\mathbf{p}$ if, for every aggregate expression $\mathrm{OP}\langle \mathbf{x} : G\rangle \succeq b$ in it, every occurrence of every predicate p from $\mathbf{p}$ is strictly positive in G. We say that an aggregate formula F is *canonical* relative to a list $\mathbf{p}$ of predicate constants if

- F is semi-positive relative to $\mathbf{p}$;
- for every occurrence of every predicate constant p from $\mathbf{p}$ in F, we have that $k + m \leq 1$;
- if a predicate constant p from $\mathbf{p}$ occurs in the scope of a strictly positive occurrence of $\exists$ or $\vee$ in F, then the occurrence of p is strictly positive in F.

For any canonical aggregate formula, the following result holds.

Proposition 5 *For any aggregate formula F, if F is canonical relative to $\mathbf{p}$, then formula*

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

is logically valid.

From Proposition 5, when F is a first-order formula, it is easy to see that $\mathrm{CIRC}[F; \mathbf{p}]$ and $\mathrm{SM}[F; \mathbf{p}]$ are equivalent to each other. This fact is used in [Kim *et al.*, 2009; Lee and Palla, 2010] to compute circumscriptive action formalisms using ASP solvers. Here we generalize the result to formulas containing aggregates and use it to relate the FLP and the FLPT operators to the SM operator.

Theorem 8 *Let Π be a finite general program with aggregates and let F be the AF-representation of Π. For every rule (9) in Π, if B is canonical relative to $\mathbf{p}$ and every occurrence of p from $\mathbf{p}$ in H is strictly positive in H, then* $\mathrm{FLP}[\Pi; \mathbf{p}]$ *is equivalent to* $\mathrm{SM}[F; \mathbf{p}]$.

Among the programs in Example 4, only Π_2 satisfies the condition of Theorem 8. Program (10) does not satisfy the condition because $\neg(\mathrm{SUM}\langle x : p(x)\rangle < 2)$ is not canonical relative to $\{p\}$. In fact, $\{p(-1), p(1), p(2)\}$ is an Herbrand interpretation that satisfies $\mathrm{SM}[(11)]$, but does not satisfy $\mathrm{FLP}[(10)]$.

Next we show the class of formulas F for which $\mathrm{FLPT}[F; \mathbf{p}]$ coincides with $\mathrm{SM}[F; \mathbf{p}]$.

Theorem 9 *Let F be a semi-positive aggregate formula relative to $\mathbf{p}$ such that every subformula that has a non-strictly positive occurrence in F is canonical relative to $\mathbf{p}$. Then* $\mathrm{FLPT}[F; \mathbf{p}]$ *is equivalent to* $\mathrm{SM}[F; \mathbf{p}]$.

In Example 4, relative to $\{p\}$, formulas F_2 and F_3 satisfy the condition of Theorem 9. In Example 5, relative to $\{p\}$, formula $F \wedge G$ satisfy the condition, but $F \wedge (F \rightarrow G)$ does not. Also formula (11) does not satisfy the condition. Again, $\{p(-1), p(1), p(2)\}$ is an Herbrand interpretation that satisfies $\mathrm{SM}[(11)]$, but it does not satisfy $\mathrm{FLPT}[(11)]$.

5 Conclusion

Our work lifts up the mathematical foundations of the FLP and the FLPT semantics to the first-order level; our semantics are applicable to non-Herbrand models allowing arithmetic and arbitrary functions. The fact that these semantics are uniformly characterized in terms of modifications to circumscription provides us useful insights into the relationships among the FLP semantics, the FLPT semantics, circumscription and the first-order stable model semantics. We present the classes of formulas under which the semantics are interchangeable, which would have interesting ramifications. For instance, the FLP semantics is simpler than the first-order stable model semantics, but involves unintuitive cases if arbitrary formulas are used in the head and in the body. Such cases do not arise if the FLP semantics is applied to the class of programs that satisfy the condition of Theorem 8. For the same class of programs, system F2LP [Lee and Palla, 2009], an implementation of the first-order stable model semantics, can be viewed also as an implementation of the general FLP semantics.

Acknowledgements

We are grateful to anonymous referees for their useful comments on this paper. The authors were partially supported by the National Science Foundation under Grant IIS-0916116 and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the US Army Research Lab.

References

[Dao-Tran *et al.*, 2009] Minh Dao-Tran, Thomas Eiter, Michael Fink, and Thomas Krennwallner. Modular nonmonotonic logic programming revisited. In *Proceedings of International Conference on Logic Programming (ICLP)*, pages 145–159, 2009.

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

[Faber *et al.*, 2004] Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)*, 2004.

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

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

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

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

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

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

[Kim *et al.*, 2009] Tae-Won Kim, Joohyung Lee, and Ravi Palla. Circumscriptive event calculus as answer set programming. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 823–829, 2009.

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

[Lee and Palla, 2009] Joohyung Lee and Ravi Palla. System F2LP – computing answer sets of first-order formulas. In *Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 515–521, 2009.

[Lee and Palla, 2010] Joohyung Lee and Ravi Palla. Situation calculus as answer set programming. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 309–314, 2010.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustin Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.

[Lifschitz, 2010] Vladimir Lifschitz. Thirteen definitions of a stable model. *Fields of Logic and Computation*, volume 6300 of *Lecture Notes in Computer Science*, pages 488–503. Springer, 2010.

[McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39,171–172, 1980.

[Osorio *et al.*, 2004] Mauricio Osorio, Juan A. Navarro, and José Arrazola. Applications of intuitionistic logic in answer set programming. *TPLP*, 4(3):325–354, 2004.

[Pearce, 1997] David Pearce. A new logical characterization of stable models and answer sets. *Non-Monotonic Extensions of Logic Programming (Lecture Notes in Artificial Intelligence 1216)*, pages 57–70. Springer, 1997.

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