First-Order Extension of the FLP Semantics

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. Similar to the first-order stable model semantics by Ferraris, Lee and Lifschitz, the reformulations presented 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.

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 (2011) (called the FLP semantics) deserves special attention since it provides a simple satisfactory solution to the semantics of aggregates, and is implemented in the system DLV[1]. It is also a basis of some extensions of the answer set semantics to integrate with external sources that have possibly heterogeneous semantics (e.g., HEX programs (Eiter *et al.* 2005)). Dao-Tran *et al.* (2009) remark that the FLP semantics provides a more natural ground 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 of 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. Then the same minimality condition as in the original definition of an answer set applies. For example, consider the following program Π_1:

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

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

$$\begin{array}{l} p \leftarrow \mathit{not}\ q \\ r \leftarrow p, \end{array} \tag{2}$$

and X is minimal among the sets of atoms satisfying (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. For example, the GL-reduct of Π_1 relative to X (Gelfond and Lifschitz 1988) is

$$\begin{array}{l} p \\ r \leftarrow p \\ r \leftarrow q \end{array} \tag{3}$$

and, again, $\{p, r\}$ is minimal among the sets of atoms satisfying (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, this extension is still limited; 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 translations into second-order classical logic, in the same spirit as the first-order stable model semantics by Ferraris, Lee and Lifschitz (Ferraris *et al.* 2007; 2011). This allows us to show how the FLP semantics and its extension are related to circumscription by McCarthy (McCarthy 1980; 1986) and to the first-order stable model semantics. Interestingly a simple modification of the definition of circumscription yields the first-order extension of the FLP semantics. Truszczyński's extension resembles the FLP semantics on the one hand, and the first-order stable model semantics on the other hand.

[1] http://www.dlvsystem.com

FLP Semantics

Review: Original FLP Semantics

A *disjunctive rule* is an expression of the form

$$A_1 ; \ldots ; A_l \leftarrow A_{l+1}, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n \tag{4}$$

($n \geq m \geq l \geq 0$) 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 is defined in terms of grounding and reduct. 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 Π^X, is obtained from *Ground*(Π) by removing all rules 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 Π^X (viewed as a formula in propositional logic). For example, for Π_1 and $X = \{p, r\}$ in the introduction, Π_1^X is (2) and X is an FLP-answer set of Π_1.

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 that allows rules of a more general form than (4). We assume the following set of primitive propositional connectives and quantifiers for 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

$$H \leftarrow B \tag{5}$$

where H and B are arbitrary formulas in first-order logic. Rule (4) is a special case of (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 $p_1, \ldots, p_n$, and let $\mathbf{u}$ be a list of distinct predicate variables $u_1, \ldots, u_n$. By $\mathbf{u} \leq \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))$ for all $i = 1, \ldots, n$ where $\mathbf{x}$ is a list of distinct object variables of the same length as the arity of p_i, and by $\mathbf{u} < \mathbf{p}$ we denote $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{p} \leq \mathbf{u})$. For instance, if p and q are unary predicate constants then $(u, v) < (p, q)$ is

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

For any formula G, 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$[\Pi; \mathbf{p}]$ we denote the second-order formula

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

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

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

for all rules $H \leftarrow B$ in Π, where $\mathbf{x}$ is the list of all (free) variables in $H \leftarrow B$.[2]

We will often simply write FLP$[\Pi]$ instead of FLP$[\Pi; \mathbf{p}]$ when $\mathbf{p}$ is the list of all predicate constants occurring in Π, and call a model of FLP$[\Pi]$ 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 implications are allowed even in the head and in the body, the outermost implications (the rule arrow explicitly shown in (5)) are distinguished from them due to the presence of '$B\wedge$' in the definition of $\Pi^{\triangle}$. If we drop '$B\wedge$' from (7), then (6) becomes exactly CIRC$[\Pi^{FOL}; \mathbf{p}]$ (McCarthy 1980; 1986). Interestingly, this small difference accounts for many differences in the properties of the two semantics, one leading to stable models and the other leading to minimal models. For instance, classically equivalent transformation preserves the minimal models, but not FLP-stable models. For example, $p \leftarrow \mathit{not}\ q$ and $q \leftarrow \mathit{not}\ p$ are classically equivalent to each other (when we identify them with their FOL-representations), but their FLP-stable models are different.

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 in the sense of (Faber* et al. *2011) are precisely the Herbrand models of* FLP$[\Pi]$ *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$[\Pi]$*, then I is a subset-minimal model of Π.*[3]

[2] Note that we assume that Π is finite in order to avoid infinite conjunctions in the FOL representation.

[3] 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.

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

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

Proposition 3 *For any finite general program Π, formula* $\text{FLP}[\Pi; \mathbf{p}]$ *is equivalent to*

$$\exists \mathbf{u}(\text{CIRC}[\Pi^{\triangle}(\mathbf{u}); \mathbf{u}] \wedge (\mathbf{u} = \mathbf{p})).$$

Extension: First-Order FLP Semantics for Programs with Aggregates

The semantics given 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. We assume that symbols for aggregate functions are not part of the signature.

An *aggregate expression* of signature σ is of the form [4]

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

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.

We define an *aggregate formula* as an extension of a first-order formula by treating 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.

[4] The syntax of 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.

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. 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. We consider "standard" interpretations only, in which symbols for numbers and comparison operators are evaluated in the standard way.[5] 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 $\boldsymbol{\xi}^*[1]$ 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 $F(\boldsymbol{\xi}^*)$.

(For any list of object constants $\mathbf{c}$, by $\mathbf{c}[1]$ we denote the first element of $\mathbf{c}$.)

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 standard 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 the previous section to a *general program with aggregates*, which allows aggregate formulas in the head and in the body of a rule. 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

$$B \rightarrow H \tag{9}$$

for all rules $H \leftarrow B$ in Π. $\text{FLP}[\Pi; \mathbf{p}]$ is defined the same as (6) except that Π is now understood as a general program with aggregates.

[5] For instance, we assume that, when both x and y are integer constants, $x \leq y$ evaluates to true iff x is less than y, and $x + y$ is the sum of the integers. On the other hand, when x or y are not integers, $x \leq y$ evaluates to false, and $x + y$ has an arbitrary value according to the interpretation.

Example 2 *Consider the following aggregate sentence F:*

$$\begin{array}{l}(\neg(\text{SUM}\langle x\!:\!p(x)\rangle\!<\!2)\rightarrow p(2))\\ \wedge\,(\text{SUM}\langle x\!:\!p(x)\rangle\!\geq\!0\rightarrow p(-1))\\ \wedge\,(p(-1)\rightarrow p(1))\,.\end{array}\tag{10}$$

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

$$F\wedge\neg\exists u(u<p\wedge\Pi^{\triangle}(u))\tag{11}$$

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

$$\begin{array}{l}(\neg(\text{SUM}\langle x\!:\!p(x)\rangle\!<\!2)\wedge\neg(\text{SUM}\langle x\!:\!u(x)\rangle\!<\!2)\rightarrow u(2))\\ \wedge\,(\text{SUM}\langle x\!:\!p(x)\rangle\!\geq\!0\wedge\text{SUM}\langle x\!:\!u(x)\rangle\!\geq\!0\rightarrow u(-1))\\ \wedge\,(p(-1)\wedge u(-1)\rightarrow u(1))\,.\end{array}\tag{12}$$

Below we show how this semantics is related to the original semantics in (Faber *et al.* 2011). Faber *et al.* (2011) defined their semantics for *disjunctive programs with aggregates*, whose rules have the form

$$A_1;\ldots;A_l\leftarrow\ E_1,\ldots,E_m,\mathit{not}\ E_{m+1},\ldots,\mathit{not}\ E_n\tag{13}$$

($l\geq 0$; $n\geq m\geq 0$), where each A_i is an atomic formula and each E_i is an atomic formula or an aggregate expression. For example, the following is a disjunctive program with aggregates, whose AF-representation is (10):

$$\begin{array}{rcl}p(2) & \leftarrow & \mathit{not}\ \text{SUM}\langle x:p(x)\rangle<2\\ p(-1) & \leftarrow & \text{SUM}\langle x:p(x)\rangle\geq 0\\ p(1) & \leftarrow & p(-1)\,.\end{array}\tag{14}$$

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 I whose signature is $\sigma(\Pi)$, the *FLP-reduct* of Π relative to I is obtained from *Ground*(Π) 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 (14) above, the FLP-reduct of (14) 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 (14). 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 (11) in Example 2. The following theorem tells us that our semantics is a proper generalization of the semantics from (Faber *et al.* 2011).

Theorem 2 *Let Π be a finite disjunctive program with aggregates that contains at least one object constant. The FLP-answer sets of Π in the sense of (Faber* et al. *2011) are precisely the Herbrand models of* FLP[Π] *whose signature is $\sigma(\Pi)$.*

Truszczyński Semantics

Review: 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 proposed by Ferraris (2005).

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

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

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

As noted in (Truszczyński 2010), this definition of a reduct is similar to the definition of a reduct by Ferraris (2005), except for the case $G\rightarrow H$.

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 $\mathbf{p}=(p_1,\ldots,p_n)$, formula FLPT$[F;\mathbf{p}]$ is defined as

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

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

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

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 asserts that our semantics is a proper generalization of the answer set semantics by Truszczyński which covers 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 the previous section when it is applied to disjunctive programs with aggregates.

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$[\Pi]$ *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 notion of strong equivalence is important. Similar to the relationship between *HT-models* and strong equivalence under the stable model semantics (Ferraris *et al.* 2011), Truszczyński (2010) related the FLPT-reduct to "FLP-models," and used them to characterize the strong equivalence between propositional formulas under the FLPT semantics. In the following we extend the result to arbitrary first-order formulas with aggregates.[6]

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.

Theorem 4 *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*

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

is logically valid.

This theorem is a proper extension of Theorem 7 from (Truszczyński 2010), which does not consider aggregates. As a special case, Theorem 4 can be applied to checking strong equivalence under the FLP semantics between the programs whose rules have the form (13).

Theorem 4 is similar to Theorems 9 from (Ferraris *et al.* 2011), which is about strong equivalence under the stable model semantics.

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

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 finite list of intensional predicates $\mathbf{p} = (p_1, \ldots, p_n)$, 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 $Q \in \{\forall, \exists\}$;
- $(\text{OP}\langle \mathbf{x} : G(\mathbf{x})\rangle \succeq b)^* =$
  $(\text{OP}\langle \mathbf{x} : G^*(\mathbf{x})\rangle \succeq b) \wedge (\text{OP}\langle \mathbf{x} : G(\mathbf{x})\rangle \succeq b)$.

We often simply write SM$[F]$ instead 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 FLP, FLPT and SM 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 shown in the following example.

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 models as* SM$[F_3]$.

Note that, in the examples above, when "choice formula" $p \vee \neg p$ is in the body, FLPT yields a formula that is equivalent to the one that FLP yields, and when $p \vee \neg p$ is in the head, FLPT yields a formula that is equivalent to the one that SM yields. This is not a coincidence; below we describe the classes of formulas for which each pair of the semantics coincide.

But before that, we remark that FLPT-strong equivalence involves some unintuitive results. We would expect that

$F \wedge G$ and $F \wedge (F \to G)$ have the same FLPT-stable models. Indeed, they are strongly equivalent to each other in the sense of (Ferraris *et al.* 2011), but not FLPT-strongly equivalent. The two formulas may not even have the same FLPT-stable models.

Example 5 *For propositional signature* $\{p\}$ *and* $F = \neg\neg p$ *and* $G = p \vee \neg p$*, formulas* SM$[F \wedge G]$ *and* FLPT$[F \wedge G]$ *are equivalent to each other, having only one model,* $\{p\}$*. On the other hand,* SM$[F \wedge (F \to G)]$ *has only one model* $\{p\}$*, but* FLPT$[F \wedge (F \to G)]$ *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 same as in the first-order stable model semantics. This is related to the fact that the definition $(F \to G)^{\square}$ is similar to the $(F \to G)^{\triangle}$ on the one hand and is similar to $(F \to G)^{*}$ on the other hand.

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

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.

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

Theorem 5 *Let* Π *be a finite general program with aggregates, and let* F *be the AF-representation of* Π*. For every rule (5) in* Π*, if every occurrence of* p *from* $\mathbf{p}$ *in* H *is strictly positive in* H*, then* FLP$[\Pi; \mathbf{p}]$ *is equivalent to* 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 5. Also program (14) satisfies the same condition.

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 $\text{OP}\langle \mathbf{x} : G \rangle \succeq b$ 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.

Theorem 6 *Let* Π *be a finite general program with aggregates and let* F *be the AF-representation of* Π*. For every rule (5) 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* FLP$[\Pi; \mathbf{p}]$ *is equivalent to* SM$[F; \mathbf{p}]$.

Among the programs in Example 4, only Π_2 satisfies the condition of Theorem 6. For another example, in program (14), $\neg(\text{SUM}\langle x : p(x) \rangle < 2)$ is not canonical relative to $\{p\}$. In fact, $\{p(-1), p(1), p(2)\}$ is an Herbrand interpretation that satisfies SM$[(10)]$, but it does not satisfy FLP$[(14)]$.

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

Theorem 7 *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* FLPT$[F; \mathbf{p}]$ *is equivalent to* SM$[F; \mathbf{p}]$.

In Example 4, relative to $\{p\}$, formulas F_2 and F_3 satisfy the condition of Theorem 7, but formulas F_1 and F_4 do not. In Example 5, relative to $\{p\}$, formula $F \wedge G$ satisfy the condition, but $F \wedge (F \to G)$ does not. Also formula (10) does not satisfy the condition. Again, $\{p(-1), p(1), p(2)\}$ is an Herbrand interpretation that satisfies SM$[(10)]$, but it does not satisfy FLPT$[(14)]$.

Conclusion

We presented a reformulation of the FLP semantics by a simple modification to circumscription, and presented a reformulation of the FLPT semantics using the recursive definition similar to the one used in the first-order stable model semantics. Our reformulations are more general than the original semantics, and provide useful insights into the relationships among the FLP semantics, the FLPT semantics, circumscription and the first-order stable model semantics.

The FLP semantics is a basis of HEX programs that integrate logic programs with external source of information. In HEX programs, the external atoms are treated similar to aggregates. Our work suggests that HEX programs can be generalized without the need to refer to grounding. While in general the FLP semantics does not coincide with the first-order stable model semantics, the mismatch is usually avoided in practice because the FLP semantics is applied to the rules (13) with *complex atoms* instead of the syntax of *complex formulas*, thereby satisfying the condition of the theorem under which the two semantics coincide.

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 Grants IIS-0916116 and by the IARPA SCIL program.

References

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.

Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *IJCAI*, pages 90–96, 2005.

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

Paolo Ferraris and Vladimir Lifschitz. On the stable model semantics of firsr-oder formulas with aggregates. In *NMR*, 2010.

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.

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

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

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

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.

Vladimir Lifschitz. Thirteen definitions of a stable model. In Andreas Blass, Nachum Dershowitz, and Wolfgang Reisig, editors, *Fields of Logic and Computation*, volume 6300 of *Lecture Notes in Computer Science*, pages 488–503. Springer, 2010.

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

John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 26(3):89–116, 1986.

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