FLP Semantics Without Circular Justifications for General Logic Programs

Yi-Dong Shen
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing 100190, China
ydshen@ios.ac.cn

Kewen Wang
School of Computing and Information Technology
Griffith University
Brisbane, QLD 4111, Australia
k.wang@griffith.edu.au

Abstract

The FLP semantics presented by (Faber, Leone, and Pfeifer 2004) has been widely used to define answer sets, called FLP answer sets, for different types of logic programs such as logic programs with aggregates, description logic programs (dl-programs), Hex programs, and logic programs with first-order formulas (general logic programs). However, it was recently observed that the FLP semantics may produce unintuitive answer sets with circular justifications caused by self-supporting loops. In this paper, we address the circular justification problem for general logic programs by enhancing the FLP semantics with a level mapping formalism. In particular, we extend the Gelfond-Lifschitz three step definition of the standard answer set semantics from normal logic programs to general logic programs and define for general logic programs the first FLP semantics that is free of circular justifications. We call this FLP semantics the *well-justified FLP semantics*. This method naturally extends to general logic programs with additional constraints like aggregates, thus providing a unifying framework for defining the well-justified FLP semantics for various types of logic programs. When this method is applied to normal logic programs with aggregates, the well-justified FLP semantics agrees with the conditional satisfaction based semantics defined by (Son, Pontelli, and Tu 2007); and when applied to dl-programs, the semantics agrees with the strongly well-supported semantics defined by (Shen 2011).

Introduction

Answer set programming (ASP) is a major logic programming paradigm for knowledge representation and reasoning. In ASP, the semantics of a logic program is defined by a set of intended models, called stable models or answer sets. Such a semantics is thus called an answer set semantics. We may use different ways to define answer sets of a logic program. As summarized by (Lifschitz 2010), there have been thirteen different definitions of answer sets in the literature. These definitions agree with the standard answer set semantics for normal logic programs (Gelfond and Lifschitz 1988) in the sense that they define the same set of answer sets.

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In this paper, we are devoted to a currently widely used definition of answer sets, called the FLP semantics (Faber, Leone, and Pfeifer 2004; Faber, Pfeifer, and Leone 2011). Let Π be a normal logic program with rules of the form $A_0 \leftarrow A_1, \cdots, A_m, \ not\ A_{m+1}, \cdots, not\ A_n$, where each A_i is an atom. Informally, given an interpretation I the FLP reduct of Π w.r.t. I, denoted $f\Pi^I$, consists of all ground rules of Π whose bodies are satisfied by I. Then, I is an FLP answer set of Π if I is a minimal model of $f\Pi^I$. Observe that the FLP reduct can easily be extended to more general classes of logic programs, provided that the satisfaction relation for normal logic programs is extended to those classes of logic programs. Due to this reason, the FLP semantics has recently been extended, as a unifying formalism, to a variety of logic programs including logic programs with aggregates or abstract constraint atoms (c-atoms) (Faber, Pfeifer, and Leone 2011), description logic programs (dl-programs) (Eiter et al. 2008; 2005), Hex programs (Eiter et al. 2005), disjunctive dl-programs (Lukasiewicz 2010), and logic programs with first-order formulas (Bartholomew, Lee, and Meng 2011).

However, when applied to the above mentioned classes of logic programs, the FLP semantics may produce unintuitive answer sets with circular justifications caused by self-supporting loops. This problem was observed in (Shen and You 2009; Liu et al. 2010) for logic programs with aggregates, in (Shen 2011) for dl-programs, and in (Shen and Wang 2011) for disjunctive dl-programs.

Logic programs with first-order formulas, also called *general logic programs*, were recently introduced by (Bartholomew, Lee, and Meng 2011) and have now received increasing attention. A general logic program consists of rules of the form $H \leftarrow B$, where H and B are arbitrary first-order formulas. Normal logic programs can be viewed as a special form of general logic programs, where the negation not is identified with $\neg$, each rule head H with an atom, and each rule body B with a conjunction of literals. Answer sets of a general logic program are defined by the FLP semantics. We observe that this FLP semantics for general logic programs also incurs circular justifications. To illustrate, consider the following general logic program:

$$\begin{array}{lll} \Pi_1: & p(2) \leftarrow p(2) \wedge (\neg p(-1) \vee p(1)). & r_1 \\ & p(-1) \leftarrow \neg p(-1) \vee p(1) \vee p(2). & r_2 \\ & p(1) \leftarrow p(-1). & r_3 \end{array}$$

Let $I = \{p(-1), p(1)\}$ be an interpretation. Since the body of rule r_1 is not satisfied by I, the FLP reduct of Π_1 w.r.t. I is $f\Pi_1^I = \{r_2, r_3\}$. I is a minimal model of $f\Pi_1^I$ and thus is an answer set of Π_1 under the FLP semantics. Observe that this FLP answer set has circular justifications caused by the following self-supporting loop:

$$p(1) \Leftarrow p(-1) \Leftarrow \neg p(-1) \vee p(1) \vee p(2) \Leftarrow p(1).$$

That is, the evidence of the truth of $p(1) \in I$ is supported by $p(-1) \in I$ (via r_3), while the truth of $p(-1)$ is supported by $\neg p(-1) \vee p(1) \vee p(2)$ (via r_2). Since $\neg p(-1)$ and $p(2)$ are false in I, the truth of $\neg p(-1) \vee p(1) \vee p(2)$ is supported by $p(1)$. As a result, $p(1)$ in I is circularly supported (justified) by itself.

Our study shows that the key reason behind the circular justification problem for general logic programs is that the FLP semantics is unable to build a level mapping on its answer sets. Such a level mapping is defined implicitly by rules of a logic program such that answers at upper levels $i > 0$ are derived from answers at lower levels $j < i$ by applying rules in the way that *if* the rule bodies hold *then* infer the rule heads. We would like to stress that it is such a level mapping on answer sets that makes an *if-then* rule $H \leftarrow B$ in a logic program essentially different from an implication $B \supset H$ in classical logic. In fact, for normal logic programs, (Fages 1994) shows that the standard answer set semantics has a level mapping on its answer sets. Since the FLP semantics agrees with the standard answer set semantics for normal logic programs, FLP answer sets are free of circular justifications for normal logic programs.

In this paper, we address the circular justification problem for general logic programs by enhancing the FLP semantics with a level mapping formalism. Observe that for a normal logic program Π under the standard answer set semantics, each answer set I together with its level mapping is defined by a fixpoint, which is derived from the Gelfond-Lifschitz reduct Π^I by iteratively applying the van Emden-Kowalski one-step provability operator (van Emden and Kowalski 1976). Inspired by this, we define each FLP answer set I together with a level mapping by a fixpoint, which is derived in a similar way from the FLP reduct $f\Pi^I$. To achieve this, we first extend the van Emden-Kowalski provability operator from a positive logic program to a general logic program. Next we extend the Gelfond-Lifschitz three step definition of the standard answer set semantics from a normal logic program to a general logic program by replacing the Gelfond-Lifschitz reduct Π^I with the FLP reduct $f\Pi^I$ and applying the extended van Emden-Kowalski provability operator to derive a fixpoint from $f\Pi^I$. An answer set I is then defined in terms of the fixpoint. We show that such answer sets are FLP answer sets with a level mapping and thus are free of circular justifications. Therefore, we call such answer sets *well-justified FLP answer sets*, and call the new semantics the *well-justified FLP semantics*. This method naturally extends to general logic programs with additional constraints like aggregates, thus providing a unifying framework for defining the well-justified FLP semantics for various types of logic programs. We show that when this method is applied to normal logic programs with aggregates, the well-justified FLP semantics agrees with the conditional satisfaction based semantics defined by (Son, Pontelli, and Tu 2007); and when applied to dl-programs, the semantics agrees with the strongly well-supported semantics defined by (Shen 2011).

Due to the page limit, we omit the proof of all theorems in this paper.

A First-Order Logic Language

First-Order Logic

We consider a first-order logic language $\mathcal{L}_\Sigma$ with equality over a *signature* $\Sigma = (\mathcal{P}, \mathcal{F})$, where $\mathcal{P}, \mathcal{F}$ are a countable set of *predicate* and *function* symbols, respectively. Let $\mathcal{C} \subseteq \mathcal{F}$ be the set of 0-ary function symbols called *constants*, and $\mathcal{V}$ be a countable set of *variables*. A *term* is a variable or a function $f(t_1, \cdots, t_m)$, where f is an m-ary function symbol ($m \geq 0$) and each t_i is a term. An *atom* is of the form $p(t_1, \cdots, t_n)$, where p is an n-ary predicate symbol and each t_i is a term. Let $\mathcal{N}_\Sigma$ denote the set of ground terms of Σ, and $\mathcal{H}_\Sigma$ the set of ground atoms of Σ. *Formulas* are constructed as usual from atoms using connectives $\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$, $\exists$, $\forall$, (, and). *Literals* are of the form A or $\neg A$, where A is an atom. Formulas are *closed* if they contain no free variables (i.e., all variables are quantified either by $\forall$ or $\exists$). A (first-order) *theory* is a set of closed formulas. *Ground* terms/atoms/formulas contain no variables.

An *interpretation* of $\mathcal{L}_\Sigma$ is a pair $I = \langle U, .^I \rangle$, where U is a *domain*, and $.^I$ a *mapping* which assigns a relation $p^I \subseteq U^n$ to each n-ary predicate symbol $p \in \mathcal{P}$, and a function $f^I : U^m \rightarrow U$ to each m-ary function symbol $f \in \mathcal{F}$. A *variable assignment* B for I is a mapping which assigns an element $X^B \in U$ to each variable $X \in \mathcal{V}$. The interpretation of a term t, denoted $t^{I,B}$, is defined as usual, where B is omitted when t is ground. *Satisfaction* of a formula F in I relative to B is defined as usual. I is a *model* of F if I satisfies F for every variable assignment. I is a model of (or I satisfies) a theory O if I is a model of all formulas in O. A theory is *consistent* or *satisfiable* if it has a model. A theory O *entails* a formula F, denoted $O \models F$, if all models of O are models of F.

An *SNA interpretation* I is an interpretation $\langle U, .^I \rangle$ of $\mathcal{L}_\Sigma$ that employs the following *standard name assumption*: (1) The signature Σ includes a countably infinite set of constants and the 2-ary predicate symbol $\approx$ that expresses equality, (2) $U = \mathcal{N}_\Sigma$ and $t^{\mathcal{I}} = t$ for each $t \in \mathcal{N}_\Sigma$, and (3) $\approx^I$ is a congruence relation over U, which is reflexive, symmetric and transitive, and allows the replacement of equals by equals.

For an SNA interpretation I, each variable assignment over the domain U is a substitution of variables over $\mathcal{N}_\Sigma$. Moreover, since $p^I \subseteq \mathcal{N}_\Sigma^n$ for each n-ary predicate symbol $p \in \mathcal{P}$, there is a one-to-one correspondence between SNA interpretations of $\mathcal{L}_\Sigma$ and subsets of $\mathcal{H}_\Sigma$. Therefore, we may well define an SNA interpretation I as a subset of $\mathcal{H}_\Sigma$, i.e., $I \subseteq \mathcal{H}_\Sigma$. Then, an SNA interpretation I satisfies a ground atom A if $A \in I$; I satisfies $\neg A$ if $A \notin I$.

(Motik and Rosati 2010) show that using SNA interpretations preserves satisfiability of first-order logic; i.e., a first-order formula is satisfiable if and only if it is satisfiable in a

model that employs the standard name assumption. Therefore, in the sequel we consider only SNA interpretations, which are expressed as a subset I of $\mathcal{H}_\Sigma$. We denote I^- for $\mathcal{H}_\Sigma \setminus I$, and $\neg I^-$ for $\{\neg A \mid A \in I^-\}$.

For convenience, in the sequel by function symbols we refer to m-ary function symbols with $m > 0$.

General Logic Programs

We extend the above first-order language $\mathcal{L}_\Sigma$ with *rules* of the form $H \leftarrow B$, where H and B are formulas. Such a rule r expresses an *if-then* statement, saying that *if* the logic property B holds *then* infer H. We use $body(r)$ to refer to B, and $head(r)$ to H. When $body(r)$ is empty, we omit $\leftarrow$.

A *general logic program* Π consists of a finite set of rules. Π is a *normal logic program* if each rule r is of the form $A_0 \leftarrow A_1 \wedge ... \wedge A_m \wedge \neg A_{m+1} \wedge ... \wedge \neg A_n$, where each A_i is an atom without equality and function symbols. We use $pos(r)$ and $neg(r)$ to denote $A_1 \wedge ... \wedge A_m$ and $\neg A_{m+1} \wedge ... \wedge \neg A_n$, respectively. A *positive logic program* is a normal logic program without negative literals.

Let $\mathcal{C}_\Pi \subseteq \mathcal{C}$ be a non-empty, finite set of constants including all constants occurring in Π. A *closed instance* of a rule is obtained by replacing every free variable in the rule with a constant in $\mathcal{C}_\Pi$. The *grounding* of Π w.r.t. $\mathcal{C}_\Pi$, denoted $ground(\Pi)$, is the set of closed instances of all rules in Π. Since both Π and $\mathcal{C}_\Pi$ are finite, $ground(\Pi)$ is finite.

An interpretation I *satisfies* a closed instance r of a rule if it satisfies $head(r)$ or it does not satisfy $body(r)$. I is a *model* of a logic program Π if it satisfies all rules in $ground(\Pi)$. A *minimal* model is a subset-minimal model.

From the definition of models, a logic program Π is viewed as shorthand for $ground(\Pi)$, where each rule in Π is viewed as shorthand for the set of its closed instances. The FLP semantics for general logic program is defined as follows.

Definition 1 Let Π be a general logic program and I an interpretation. The *FLP-reduct* of Π w.r.t. I is $f\Pi^I = \{r \in ground(\Pi) \mid I \text{ satisfies } body(r)\}$. I is an *FLP answer set* of Π if I is a minimal model of $f\Pi^I$.

Well-Justified FLP Answer Sets for General Logic Programs

As mentioned in the introduction, rules $H \leftarrow B$ in a logic program differ essentially from implications $B \supset H$ in classical logic because rules define a level mapping on their answer sets such that answers at upper levels are derived from answers at lower levels by applying the rules in the way that if the rule bodies hold then infer the rule heads. However, the FLP semantics defined by Definition 1 is unable to capture such a level mapping. For I to be an answer set of Π, the FLP semantics only requires I to be a minimal model of $f\Pi^I$. This amounts to treating all rules $H \leftarrow B$ in $f\Pi^I$ as implications $B \supset H$ in classical logic because I is a model of the rules $H \leftarrow B$ in $f\Pi^I$ if and only if I is a model of the corresponding implications $B \supset H$ in classical logic. Since classical implications define no level mapping on their models, some minimal models of $f\Pi^I$ may have no level mapping and thus some FLP answer sets may have circular justifications (see the example in the introduction).

Therefore, the key to overcome the circular justification problem is to enhance the FLP semantics with a level mapping formalism whereby the FLP reduct is treated as rules instead of classical implications. To achieve this, let us first look at how the standard answer set semantics builds a level mapping for its answer sets from normal logic programs.

For a normal logic program Π and an interpretation I, the definition of answer sets under the standard answer set semantics consists of three steps (Gelfond and Lifschitz 1988):

1. Eliminate from $ground(\Pi)$ all rules whose bodies contain a negative literal not satisfied by I.
2. Eliminate from the bodies of the remaining rules of $ground(\Pi)$ all negative literals.

   Note: All negative literals removed in the second step are in $\neg I^-$. The remaining part of $ground(\Pi)$ after the two steps of transformation is called the *Gelfond-Lifschitz reduct*, denoted Π^I.
3. Compute a fixpoint $T^\alpha_{\Pi^I}(\emptyset)$ from Π^I via the sequence $\langle T^i_{\Pi^I}(\emptyset)\rangle^\infty_{i=0}$, where $T^0_{\Pi^I}(\emptyset) = \emptyset$ and for $i \geq 0$ $T^{i+1}_{\Pi^I}(\emptyset) = T_{\Pi^I}(T^i_{\Pi^I}(\emptyset))$.

   Note: $T_P(S)$, where P is a positive logic program and S is a set of ground atoms, is the van Emden-Kowalski one-step provability operator (van Emden and Kowalski 1976) defined by $T_P(S) = \{head(r) | r \in ground(P)$ such that $body(r)$ is satisfied by $S\}$.

Then, I is an answer set of Π under the standard answer set semantics if $I = T^\alpha_{\Pi^I}(\emptyset)$. Note that the derivation sequence $\langle T^i_{\Pi^I}(\emptyset)\rangle^\infty_{i=0}$ defines a level mapping on I such that $A \in I$ is at level $i > 0$ if $A \in T^i_{\Pi^I}(\emptyset)$ but $A \notin T^{i-1}_{\Pi^I}(\emptyset)$. As a result, for any $A \in I$ at level i there is a rule $r \in ground(\Pi)$ whose head is A such that all negative literals in $neg(r)$ are in $\neg I^-$ and all positive literals in $pos(r)$ are in $T^{i-1}_{\Pi^I}(\emptyset)$.

The above Gelfond-Lifschitz three step definition of answer sets for normal logic programs are not applicable to general logic programs, since rule heads and bodies of a general logic program are arbitrary first-order formulas. For example, let $\Pi = \{A \leftarrow A \vee \neg A\}$, where A is a ground atom. Since the rule body $A \vee \neg A$ is a tautology, $I = \{A\}$ is supposed to be an answer set of Π. Apparently, this answer set cannot be obtained following the above three steps.

To deal with arbitrary first-order formulas in rule heads and bodies in a general logic program, we propose to extend the above Gelfond-Lifschitz three step definition of answer sets to general logic programs as follows:

1. We extend the first step to first-order formulas by eliminating from $ground(\Pi)$ all rules whose bodies are not satisfied by I. This yields the FLP reduct $f\Pi^I$.
2. Instead of directly eliminating from $f\Pi^I$ all negative literals in $\neg I^-$, we adapt the second step to first-order formulas by adding the negative literals in $\neg I^-$ as constraints on $f\Pi^I$.
3. We extend the third step to first-order formulas by computing a fixpoint O^α from $f\Pi^I$ under the constrains $\neg I^-$

via the sequence $\langle O^i \rangle_{i=0}^{\infty}$, where $O^0 = \emptyset$ and for $i > 0$ and any rule r in $f\Pi^I$, if $body(r)$ is true in O^{i-1} under the constrains $\neg I^-$, i.e. $O^{i-1} \cup \neg I^- \models body(r)$, then $head(r)$ is in O^i, where $\models$ is the entailment relation.

To formally define the sequence $\langle O^i \rangle_{i=0}^{\infty}$ for the above extension 3, we extend the van Emden-Kowalski one-step provability operator $T_P(S)$, which is applicable only to a positive logic program P parameterized with a set S of ground atoms, to a new operator $T_\Pi(O, N)$, which is applicable to a general logic program Π parameterized with two first-order theories O and N. Intuitively, by applying $T_\Pi(O, N)$ we infer all rule heads from $ground(\Pi)$ whose rule bodies are true in O under the constraints N. Formally we define

$$T_\Pi(O, N) = \{head(r) \mid r \in ground(\Pi) \text{ such that } O \cup N \models body(r)\}.$$

Since the entailment relation $\models$ is monotone, when the constraints N are fixed, $T_\Pi(O, N)$ is monotone w.r.t. O. That is, for any $O_1 \subseteq O_2$, $T_\Pi(O_1, N) \subseteq T_\Pi(O_2, N)$. Therefore, the sequence $\langle T_\Pi^i(\emptyset, N) \rangle_{i=0}^{\infty}$, where $T_\Pi^0(\emptyset, N) = \emptyset$ and for $i \geq 0$ $T_\Pi^{i+1}(\emptyset, N) = T_\Pi(T_\Pi^i(\emptyset, N), N)$, will converge to a fixpoint, denoted $T_\Pi^\alpha(\emptyset, N)$.

Thus, when replacing the constraints N with $\neg I^-$, we obtain a fixpoint $T_\Pi^\alpha(\emptyset, \neg I^-)$; and when further replacing Π with the FLP reduct $f\Pi^I$, we obtain a fixpoint $T_{f\Pi^I}^\alpha(\emptyset, \neg I^-)$. This achieves the above extension 3, where for $i \geq 0$, $O^i = T_{f\Pi^I}^i(\emptyset, \neg I^-)$ and $O^\alpha = T_{f\Pi^I}^\alpha(\emptyset, \neg I^-)$.

The first important result about our extended van Emden-Kowalski one-step provability operator is that when I is a model of a general logic program Π, applying the operator to Π and $f\Pi^I$ derives the same rule heads. This justifies the above extension 1, where the FLP reduct $f\Pi^I$ is used as a simplified form of Π.

Theorem 1 *Let I be a model of a general logic program Π. For any $i \geq 0$, $T_\Pi^i(\emptyset, \neg I^-) = T_{f\Pi^I}^i(\emptyset, \neg I^-)$ and thus $T_\Pi^\alpha(\emptyset, \neg I^-) = T_{f\Pi^I}^\alpha(\emptyset, \neg I^-)$.*

The next result shows that the extended provability operator $T_\Pi(O, N)$ for general logic programs is a proper generalization of the original van Emden-Kowalski operator $T_P(S)$ for positive logic programs.

Theorem 2 *Let I be a model of a normal logic program Π. For any $i \geq 0$, $T_{\Pi^I}^i(\emptyset) = T_\Pi^i(\emptyset, \neg I^-)$ and thus $T_{\Pi^I}^\alpha(\emptyset) = T_\Pi^\alpha(\emptyset, \neg I^-)$.*

It follows immediately from Theorems 2 and 1:

Corollary 1 *A model I of a normal logic program Π is an answer set under the standard answer set semantics if and only if $I = T_{\Pi^I}^\alpha(\emptyset)$ if and only if $I = T_\Pi^\alpha(\emptyset, \neg I^-)$ if and only if $I = T_{f\Pi^I}^\alpha(\emptyset, \neg I^-)$.*

The conditions listed in Corollary 1 of an answer set of a normal logic program do not apply to a general logic program because for a general logic program Π, $T_{f\Pi^I}^\alpha(\emptyset, \neg I^-)$ (resp. $T_\Pi^\alpha(\emptyset, \neg I^-)$) would be a first-order theory instead of a set of ground atoms. However, these conditions suggest that answer sets of a general logic program Π can be defined by requiring that each $A \in I$ be true in the fixpoint $T_{f\Pi^I}^\alpha(\emptyset, \neg I^-)$ under the constraints $\neg I^-$; i.e., for each $A \in I$, $T_{f\Pi^I}^\alpha(\emptyset, \neg I^-) \cup \neg I^- \models A$. Formally, we have

Definition 2 Let I be a model of a general logic program Π. I is an *answer set* of Π if for each $A \in I$, $T_{f\Pi^I}^\alpha(\emptyset, \neg I^-) \cup \neg I^- \models A$.

Such answer sets are minimal models.

Theorem 3 *If I is an answer set of a general logic program Π, then I is a minimal model of Π and is also a minimal model of $f\Pi^I$.*

It is immediate from Theorem 3 and Definition 1:

Corollary 2 *If a model I is an answer set of a general logic program Π, then I is an FLP answer set of Π.*

As a result, answer sets under Definition 2 are FLP answer sets enhanced with a level mapping, which is built from the FLP reduct $f\Pi^I$ via the sequence $\langle T_{f\Pi^I}^i(\emptyset, \neg I^-) \rangle_{i=0}^{\infty}$, where for each $A \in I$, A is at level $i > 0$ if $T_{f\Pi^I}^i(\emptyset, \neg I^-) \cup \neg I^- \models A$ but $T_{f\Pi^I}^{i-1}(\emptyset, \neg I^-) \cup \neg I^- \not\models A$. The enhancement of the level mapping makes the resulting FLP answer sets free of circular justifications. Due to this, we call answer sets (resp. the semantics) under Definition 2 *well-justified FLP answer sets* (resp. the *well-justified FLP semantics*).

For normal logic programs, by Corollary 1 the well-justified FLP semantics agrees with the standard answer set semantics that agrees with the FLP semantics.

Example 1 Consider the general logic program Π_1 in the introduction. $I = \{p(-1), p(1)\}$ is a model of Π_1 and is also an FLP answer set. $\neg I^-$ contains $\neg p(2)$. $T_{f\Pi_1^I}^0(\emptyset, \neg I^-) = \emptyset$, $T_{f\Pi_1^I}^1(\emptyset, \neg I^-) = T_{f\Pi_1^I}(T_{f\Pi_1^I}^0(\emptyset, \neg I^-), \neg I^-) = \emptyset$, and thus $T_{f\Pi_1^I}^\alpha(\emptyset, \neg I^-) = \emptyset$. Neither $p(-1) \in I$ nor $p(1) \in I$ can be proved true in $T_{f\Pi_1^I}^\alpha(\emptyset, \neg I^-)$ under the constraints $\neg I^-$, so I is not a well-justified FLP answer set of Π_1.

Next, we briefly discuss computational properties of the well-justified FLP semantics. Since it is undecidable in general to determine if a first-order theory is satisfiable, it is undecidable to determine if a general logic program has either an FLP answer set or a well-justified FLP answer set. Therefore, we restrict to the propositional case under Herbrand interpretations. By a propositional program we refer to a general logic program that contains no variables, no function symbols, and no equalities. The Herbrand base HB_Π of a propositional program Π is the set of atoms $p(a_1, ..., a_n)$, where p is a predicate symbol occurring in Π and each a_i is a constant in $\mathcal{C}_\Pi$. Any $I \subseteq \text{HB}_\Pi$ is a Herbrand interpretation of Π, where $I^- = \text{HB}_\Pi \setminus I$.

Since it is NP-complete to determine if a propositional theory is satisfiable, it is easy to show the following result.

Theorem 4 *Let Π be a propositional program and I a Herbrand interpretation. It is NP-complete to determine if I is an FLP answer set of Π. It is also NP-complete to determine if I is a well-justified FLP answer set of Π.*

Well-Justified FLP Answer Sets for General Logic Programs with Aggregates

COUNT, SUM, TIMES, MIN and MAX are commonly used aggregate functions, which map a finite set of elements in a domain to a value in a range. For simplicity, we assume the range is a set of (positive and negative) integers. We adopt the formulation of aggregates as given in (Bartholomew, Lee, and Meng 2011) with slight modification. An *aggregate atom* over the signature Σ of our logic language $\mathcal{L}_\Sigma$ is of the form $\text{OP}\langle(D, X) : F(X)\rangle \succeq b$, where OP is an aggregate function symbol (not included in Σ); $D \subseteq \mathcal{N}_\Sigma$ is the domain of OP; X is an aggregate variable (can easily be extended to a list of aggregate variables), which takes on values from D; $F(X)$ is a first-order formula in $\mathcal{L}_\Sigma$ without free variables (X is not treated as a free variable; it is bounded by D); $\succeq$ is a comparison operator, such as $\leq, \geq, <, >$, etc., which defines a binary relation over integers; and b is an integer. When the context is clear, the parameter D can be omitted.

A *general logic program* Π *with aggregate atoms* is a finite set of rules of the form $H \leftarrow B$, where H and B are first-order formulas in $\mathcal{L}_\Sigma$ extended with aggregate atoms.

Let I be an interpretation of $\mathcal{L}_\Sigma$ and A be an aggregate atom $\text{OP}\langle(D, X) : F(X)\rangle \succeq b$. Let $S^I = \{a \mid a \in D$ such that I satisfies $F(a)\}$. I *satisfies* the aggregate atom A if $\text{OP}(S^I) \succeq b$ holds; I *satisfies* $\neg A$ if I does not satisfy A.

Once the satisfaction relation is extended to aggregate atoms, the entailment relation $\models$ is extended accordingly in terms of the extended satisfaction and thus the one-step provability operator $T_\Pi(O, N)$ can be applied to general logic programs with aggregate atoms. Therefore, all results (definitions and theorems) obtained in the above section for general logic programs are applicable to general logic programs with aggregate atoms. By Definition 2, a model I of a general logic program Π with aggregate atoms is a *well-justified FLP answer set* of Π if for every $A \in I$, $T^\alpha_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models A$. By Corollary 2, such answer sets are FLP answer sets enhanced with a level mapping and thus are free of circular justifications.

Example 2 Consider the following logic program with aggregate atoms, which is borrowed from (Bartholomew, Lee, and Meng 2011).

$$\begin{array}{lll} \Pi_2: & p(2) \leftarrow \neg\text{SUM}\langle(\{-1,1,2\}, X) : p(X)\rangle < 2. & r_1 \\ & p(-1) \leftarrow \text{SUM}\langle(\{-1,1,2\}, X) : p(X)\rangle \geq 0. & r_2 \\ & p(1) \leftarrow p(-1). & r_3 \end{array}$$

For an interpretation $I = \{p(-1), p(1)\}$, $\text{SUM}(S^I) = \text{SUM}(\{-1, 1\}) = 0$. So I satisfies the two aggregate atoms $\text{SUM}\langle(\{-1,1,2\}, X) : p(X)\rangle < 2$ and $\text{SUM}\langle(\{-1,1,2\}, X) : p(X)\rangle \geq 0$. The FLP reduct is $f\Pi^I_2 = \{r_2, r_3\}$. I is a minimal model of $f\Pi^I_2$ and thus it is an FLP answer set of Π_2. However, I is not a well-justified FLP answer set because $T^\alpha_{f\Pi^I_2}(\emptyset, \neg I^-) = \emptyset$. One can check that this FLP answer set has the same circular justification as that shown in Π_1 in the introduction.

Many aggregate atoms can be represented in an abstract form as *abstract constraint atoms* (or *c-atoms*) (Marek and Truszczynski 2004). A c-atom is a pair (V, C), where V, the domain of the c-atom, is a finite subset of $\mathcal{H}_\Sigma$, and C, the admissible solutions of the c-atom, is a collection of sets of atoms in V. For instance, the aggregate atom $\text{SUM}\langle\{-1, 1, 2\}, X) : p(X)\rangle < 2$ in Π_2 can be represented as a c-atom (V, C), where $V = \{p(-1), p(1), p(2)\}$ and $C = \{\emptyset, \{p(-1)\}, \{p(1)\}, \{p(-1), p(1)\}, \{p(-1), p(2)\}\}$.

Let I be an interpretation of $\mathcal{L}_\Sigma$. I *satisfies* a c-atom (V, C) if $I \cap V \in C$; I *satisfies* $\neg(V, C)$ if I does not satisfy (V, C). With this extended satisfaction relation, the entailment relation $\models$ and the provability operator $T_\Pi(O, N)$ are directly extended to general logic programs with c-atoms. Therefore, all results, including the definition and properties of well-justified FLP answer sets, for general logic programs with aggregate atoms carry over to general logic programs with c-atoms.

A *positive basic logic program* is a finite set of function and equality free rules of the form $A \leftarrow A_1 \wedge ... \wedge A_m$, where A is a ground atom and each A_i is a c-atom. Note that any ground atom A can be represented as an *elementary c-atom* $(\{A\}, \{\{A\}\})$, and $\neg A$ represented as a c-atom $(\{A\}, \{\emptyset\})$. For any c-atom (V, C), $\neg(V, C)$ can be represented as a c-atom $(V, 2^V \setminus C)$, where 2^V is the power set of V. Therefore, for any normal logic program Π with c-atoms, its grounding $ground(\Pi)$ can be represented in this way by an equivalent positive basic logic program.

(Son, Pontelli, and Tu 2007) define an answer set semantics for positive basic logic programs based on a notion of conditional satisfaction. Let R and S be two sets of ground atoms with $R \subseteq S$. For a c-atom $A = (V, C)$, R *conditionally satisfies* A w.r.t. S, denoted $R \models_S A$, if for every F with $R \cap V \subseteq F \subseteq S \cap V$, $F \in C$; for a ground atom A, $R \models_S A$ if $R \models_S (\{A\}, \{\{A\}\})$. For a positive basic logic program Π, define $\Gamma_\Pi(R, S) = \{A \mid A \leftarrow body(r) \in \Pi$ and $R \models_S body(r)\}$. It is proved that when the second argument S is a model of Π, the sequence $\langle\Gamma^i_\Pi(\emptyset, S)\rangle^\infty_{i=0}$, where $\Gamma^0_\Pi(\emptyset, S) = \emptyset$ and for $i > 0$ $\Gamma^i_\Pi(\emptyset, S) = \Gamma_\Pi(\Gamma^{i-1}_\Pi(\emptyset, S), S)$, is monotone and will converge to a fixpoint $\Gamma^\alpha_\Pi(\emptyset, S)$. A model I of Π is a *conditional satisfaction based answer set* of Π if $I = \Gamma^\alpha_\Pi(\emptyset, I)$.

Since positive basic logic programs are a special form of general logic programs with c-atoms, the following result shows that the well-justified FLP semantics is a proper extension of the conditional satisfaction based semantics.

Theorem 5 *A model I of a positive basic logic program Π is a well-justified FLP answer set if and only if I is a conditional satisfaction based answer set.*

Well-Justified FLP Answer Sets for DL-Programs

In principle, the above method of defining well-justified FLP answer sets can be applied to different types of logic programs, provided that the satisfaction relation of $\mathcal{L}_\Sigma$ is extended to those programs. As another illustration, in this section we briefly describe how to define well-justified FLP answer sets for dl-programs. Other well-known types of logic programs, such as Hex programs (Eiter et al. 2005) and disjunctive dl-programs (Lukasiewicz 2010), can be handled in a similar way. Due to the page limit, we assume familiarity

with dl-programs (Eiter et al. 2008) and with the basics of description logics (DLs) (Baader et al. 2003). DLs are fragments of first-order logic and thus a DL knowledge base can be viewed as a first-order theory.

A dl-program Π relative to an external DL knowledge base L is a normal logic program extended in rule bodies with equalities and *dl-atoms* as an interface to access L. A dl-atom is of the form $DL[S_1op_1p_1, \cdots, S_mop_mp_m; Q](\mathbf{t})$, where each $S_iop_ip_i$ semantically maps a predicate symbol p_i in Π to a concept or role S_i in L via a special interface operator $op_i \in \{\uplus, \cup\!\!\!\!\cdot, \cap\!\!\!\!\cdot\}$, and $Q(\mathbf{t})$ is a DL expression which will be evaluated against L after the predicate mapping.

The semantics of a dl-program Π relative to L is defined in terms of Herbrand interpretations. $ground(\Pi)$ is defined as before except that (1) all those ground instances of rules which contain invalid equalities or inequalities (under the unique name assumption) are removed, and (2) all valid equalities and inequalities are removed. Let I be a Herbrand interpretation of Π and $A = DL[S_1op_1p_1, \cdots, S_mop_mp_m; Q](\mathbf{c})$ be a dl-atom occurring in $ground(\Pi)$. I *satisfies* A if $L \cup \bigcup_{i=1}^{m} A_i(I) \models Q(\mathbf{c})$, where for $1 \leq i \leq m$, $A_i(I)$ is defined by the predicate mapping $S_iop_ip_i$ such that

$$A_i(I) = \begin{cases} \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, & \text{if } op_i = \uplus; \\ \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, & \text{if } op_i = \cup\!\!\!\!\cdot; \\ \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \notin I\}, & \text{if } op_i = \cap\!\!\!\!\cdot. \end{cases}$$

I *satisfies* $\neg A$ if I does not satisfy A.

This extended satisfaction relation to dl-atoms is called *satisfaction under* L, denoted $\models_L$, in (Eiter et al. 2005; 2008). An FLP semantics for dl-programs is defined in (Eiter et al. 2005) using this satisfaction relation; i.e., a Herbrand model I of a dl-program Π is an *FLP answer set* of Π if I is a minimal model of $f\Pi^I$.

Note that dl-programs are a special form of general logic programs extended with dl-atoms (in rule bodies and heads). When applying the above extended satisfaction relation to dl-atoms, the definition and properties of well-justified FLP answer sets for general logic programs carry over to general logic programs with dl-atoms. Therefore, a Herbrand model I of a dl-program Π is a *well-justified FLP answer set* if for every $A \in I$, $T^{\alpha}_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models A$. By Corollary 2, such well-justified FLP answer sets for dl-programs are FLP answer sets enhanced with a level mapping and thus are free of circular justifications.

For a dl-program Π, since the head of each rule in $ground(\Pi)$ is a ground atom, the fixpoint $T^{\alpha}_{f\Pi^I}(\emptyset, \neg I^-)$ is a set of ground atoms. Thus for each A in a Herbrand model I, $T^{\alpha}_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models A$ if and only if $A \in T^{\alpha}_{f\Pi^I}(\emptyset, \neg I^-)$. This leads to the following.

Corollary 3 *A Herbrand model I of a dl-program Π relative to an external DL knowledge base L is a well-justified FLP answer set if and only if $I = T^{\alpha}_{f\Pi^I}(\emptyset, \neg I^-)$.*

(Shen 2011) observes that the FLP semantics defined in (Eiter et al. 2005) for dl-programs incurs circular justifications. Then an alternative semantics without circular justifications, called *the strongly well-supported* semantics, is proposed by extending the Fages' well-supportedness condition (Fages 1994) from normal logic programs to dl-programs. Informally, a Herbrand model I of a dl-program Π is a *strongly well-supported answer set* if there exists a strict well-founded partial order $\prec$ on I such that for any $A \in I$, there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ and a subset $E \subset I$ such that (1) for every $B \in E$, $B \prec A$, and (2) for every F with $E \subseteq F \subseteq I$, F satisfies $body(r)$.

The following result shows that the well-justified FLP semantics is a proper extension of the strongly well-supported semantics from dl-programs to general logic programs with dl-atoms.

Theorem 6 *A Herbrand model I of a dl-program Π relative to an external DL knowledge base L is a well-justified FLP answer set if and only if I is a strongly well-supported answer set.*

Related Work

The FLP semantics is first introduced in (Faber, Leone, and Pfeifer 2004) for normal (and disjunctive) logic programs with aggregates. It is then extended to dl-programs and Hex programs (Eiter et al. 2005; 2008), modular logic programs (Dao-Tran et al. 2009), disjunctive dl-programs (Lukasiewicz 2010), and general logic programs with aggregates (Bartholomew, Lee, and Meng 2011). In the last extension, the FLP semantics defined by Definition 1 is reformulated in terms of a modified circumscription. (Truszczynski 2010) and (Ferraris, Lee, and Lifschitz 2011) also extend the FLP semantics of (Faber, Leone, and Pfeifer 2004) to propositional formulas and first-order formulas, respectively. As illustrated in (Bartholomew, Lee, and Meng 2011), the two extensions do not agree with the FLP semantics defined by Definition 1 for general logic programs. However, we observe that the problem of circular justifications with the FLP semantics persists in the extensions.

For normal logic programs with c-atoms, (Shen and You 2009) observe that answer sets under the FLP semantics of (Faber, Leone, and Pfeifer 2004) have circular justifications. They propose a default semantics by translating a propositional logic program to a default logic theory and show that the default semantics agrees with the conditional satisfaction based semantics of (Son, Pontelli, and Tu 2007). (Liu et al. 2010) also indicate the circular justification (self-supportedness) problem with the FLP semantics of (Faber, Leone, and Pfeifer 2004). They propose a computation based semantics for normal logic programs with c-atoms, which proves to coincide with the conditional satisfaction based semantics. For dl-programs, (Shen 2011) observes the circular justification problem with the FLP semantics defined by (Eiter et al. 2005). For disjunctive dl-programs, (Shen and Wang 2011) notice the circular justification problem with the FLP semantics defined by (Lukasiewicz 2010).

Summary and Future Work

In this paper, we observe that the FLP semantics for general logic programs suffers from circular justifications. The key reason behind the circular justification problem is that the FLP semantics is unable to build a level mapping on

its answer sets. Therefore, we address this problem by enhancing the FLP semantics with a level mapping formalism. Inspired by the fact that for a normal logic program Π under the standard answer set semantics, each answer set I together with its level mapping is defined by a fixpoint, which is derived from the Gelfond-Lifschitz reduct Π^I by iteratively applying the van Emden-Kowalski one-step provability operator $T_P(S)$, we define each FLP answer set I together with a level mapping by a fixpoint, which is derived from the FLP reduct $f\Pi^I$ by iteratively applying our extended van Emden-Kowalski provability operator $T_\Pi(O, N)$. We call the new FLP semantics without circular justifications the well-justified FLP semantics. This method naturally extends to general logic programs with additional constraints like aggregates and dl-atoms, thus providing a unifying framework for defining the well-justified FLP semantics for various types of logic programs. When this method is applied to normal logic programs with aggregates, the well-justified FLP semantics agrees with the conditional satisfaction based semantics defined by (Son, Pontelli, and Tu 2007); and when applied to dl-programs, the semantics agrees with the strongly well-supported semantics defined by (Shen 2011). To the best of our knowledge, no existing work in the literature can capture the well-justified FLP semantics for general logic programs.

As future work, it is interesting to extend the well-justified FLP semantics to disjunctive rules of the form $H_1|...|H_k \leftarrow B$, where B and each H_i are a first-order formula and $|$ is an epistemic disjunction operator that is different from the classical disjunction connective $\vee$. It is also interesting to study the relation of the well-justified FLP semantics with major nonmonotonic formalisms such as circumscription (McCarthy 1980) and default logic (Reiter 1980). Finally, efficiently implementing the well-justified FLP semantics presents a challenging work.

Acknowledgments

We would like to thank all anonymous reviewers for their helpful comments. Yi-Dong Shen is supported in part by the National Natural Science Foundation of China (NSFC) grants 60970045 and 60833001. Kewen Wang is partially supported by the Australia Research Council (ARC) grants DP1093652 and P110101042.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Bartholomew, M.; Lee, J.; and Meng, Y. 2011. First-order extension of the FLP stable model semantics via modified circumscription. In *IJCAI*, 724–730.

Dao-Tran, M.; Eiter, T.; Fink, M.; and Krennwallner, T. 2009. Modular nonmonotonic logic programming revisited. In *ICLP*, 145–159.

Eiter, T.; Ianni, G.; Schindlauer, R.; and Tompits, H. 2005. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *IJCAI*, 90–96.

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence* 172(12-13):1495–1539.

Faber, W.; Leone, N.; and Pfeifer, G. 2004. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Logics in Artificial Intelligence: European Workshop*, 200–212.

Faber, W.; Pfeifer, G.; and Leone, N. 2011. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence* 175(1):278–298.

Fages, F. 1994. Consistency of clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science* 1:51–60.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175(1):236–263.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP*, 1070–1080.

Lifschitz, V. 2010. Thirteen definitions of a stable model. In *Fields of Logic and Computation*, 488–503.

Liu, L.; Pontelli, E.; Son, T.; and Truszczynski, M. 2010. Logic programs with abstract constraint atoms: the role of computations. *Artificial Intelligence* 174(3-4):295–315.

Lukasiewicz, T. 2010. A novel combination of answer set programming with description logics for the semantic web. *IEEE Transactions on Knowledge and Data Engineering* 22(11):1577–1592.

Marek, V. W., and Truszczynski, M. 2004. Logic programs with abstract constraint atoms. In *AAAI*, 86–91.

McCarthy, J. 1980. Circumscription and other nonmonotonic formalisms. *Artificial Intelligence* 13(1-2):171–172.

Motik, B., and Rosati, R. 2010. Reconciling description logics and rules. *J. ACM* 57(5).

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81–132.

Shen, Y. D., and Wang, K. W. 2011. Extending logic programs with description logic expressions for the semantic web. In *International Semantic Web Conference*, 633–648.

Shen, Y. D., and You, J. H. 2009. A default approach to semantics of logic programs with constraint atoms. In *LPNMR*, 277–289.

Shen, Y. D. 2011. Well-supported semantics for description logic programs. In *IJCAI*, 1081–1086.

Son, T. C.; Pontelli, E.; and Tu, P. H. 2007. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research* 29:353–389.

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

van Emden, M. H., and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *J. ACM* 23(4):733–742.