Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

A Well-Founded Semantics for Basic Logic Programs with Arbitrary Abstract Constraint Atoms

Yisong Wang
Guizhou University
Guiyang, China

Fangzhen Lin
Hong Kong University of
Science and Technology
Hong Kong

Mingyi Zhang
Guizhou Academy of Sciences
Guiyang, China

Jia-Huai You
University of Alberta
Edmonton, Canada

Abstract

Logic programs with abstract constraint atoms proposed by Marek and Truszczynski are very general logic programs. They are general enough to capture aggregate logic programs as well as recently proposed description logic programs. In this paper, we propose a well-founded semantics for basic logic programs with arbitrary abstract constraint atoms, which are sets of rules whose heads have exactly one atom. We show that similar to the well-founded semantics of normal logic programs, it has many desirable properties such as that it can be computed in polynomial time, and is always correct with respect to the answer set semantics. This paves the way for using our well-founded semantics to simplify these logic programs. We also show how our semantics can be applied to aggregate logic programs and description logic programs, and compare it to the well-founded semantics already proposed for these logic programs.

Introduction

Logic programming based on the stable model/answer set semantics, also known as answer set programming (ASP), has emerged as a practical paradigm for declarative problem solving (Niemelä 1999; Marek and Truszczynski 1999; Brewka, Eiter, and Truszczynski 2011). Since the stable model semantics was first proposed for normal logic programs by Gelfond and Lifschitz in 1988, various extensions have been put forward for theoretical and/or practical reasons. These include disjunctive logic programs (Gelfond and Lifschitz 1991), nested logic programs (Lifschitz, Tang, and Turner 1999), general logic program (Ferraris 2011), aggregate programs (Faber, Pfeifer, and Leone 2011), description logic programs (Eiter et al. 2008) and logic programs with abstract constraint atoms (Marek and Truszczynski 2004; Son, Pontelli, and Tu 2007). In this paper we consider logic programs with abstract constraint atoms. These programs are general enough to account for the semantics of aggregate logic programs as well as description logic programs, as aggregate atoms in the former and dl-atoms in the latter can be modeled naturally in terms of abstract constraint atoms (Son, Pontelli, and Tu 2007; Wang et al. 2012).

While ASP assumes that solutions are given by answer sets, well-founded models (Van Gelder, Ross, and Schlipf 1991) have been found to be very useful as well. First, computing the well-founded model of a normal logic program is tractable. This compares to the NP-completeness of computing an answer set. Secondly, the well-founded model of a normal logic program is “correct” under the answer set semantics in the sense that all atoms in the well-founded model (called *well-founded atoms*) are in every answer set and all atoms whose negation are in the well-founded model (called *unfounded atoms*) are not in any answer set of the given logic program. Thus to date all ASP solvers would first compute the well-founded model of a given program and use it to simplify the program (Simons, Niemel*ä*, and Soininen 2002). Because of this and other reasons, despite the dominance of the answer set semantics in logic programming, work on well-founded semantics continues to draw interests in the ASP research community. To date, the well-founded semantics has been extended to disjunctive logic programs (Wang and Zhou 2005), aggregates programs (Pelov, Denecker, and Bruynooghe 2007; Alviano et al. 2011), description logic programs (Eiter et al. 2011) and nondisjunctive hybrid MKNF knowledge base for the semantic web (Knorr, Alferes, and Hitzler 2011).

In this paper we consider logic programs with abstract constraint atoms. As a first step, we consider rules whose heads have exactly one atom. Rules of this form are called *basic*, and sets of these rules are *basic logic programs* (Son, Pontelli, and Tu 2007). For these basic logic programs with abstract constraint atoms, we propose a well-founded model semantics, and show that it has many desirable properties as those for normal logic programs. Specifically, we show that the well-founded model of a basic logic program always exists, is unique, and can be computed in polynomial time. Furthermore, the well-founded atoms are in every answer set while the unfounded ones are not in any of the answer sets of the logic program. Thus, just like in the case of normal logic programs, one can use our well-founded models to simplify basic logic programs with abstract constraint atoms. In particular, they can be used to simplify aggregate logic programs and description logic programs.

The rest of this paper is organized as follows. We briefly review logic programs with abstract constraint atoms in the next section. Then we define the notion of unfounded sets and the associated well-founded model semantics. We then prove some interesting properties about our proposed well-founded model semantics, discuss related work, and then conclude the paper.

Preliminaries

We assume an underlying propositional language $\mathcal{L}$. A *literal* is either an atom (called *a positive literal*) or an expression of the form "*not* a" (called *a negative literal*), where a is an atom. The *complement* of a literal l, denoted by $\overline{l}$, is defined as the literal *not* a (resp. a) if $l = a$ (resp. $l = not\, a$).

Let I be a set of literals. In the following, we denote by I^+ the set of positive literals in I, and I^- the set of atoms whose negated form are in I, i.e., $I^- = \{a | not\, a \in I\}$. A set of literals I is said to be *consistent* if $I^+ \cap I^- = \emptyset$. A *partial interpretation* is a consistent set of literals. A partial interpretation I is said to be *total* if $I^+ \cup I^-$ is exactly the set of the atoms of $\mathcal{L}$.

An *abstract constraint atom (c-atom)* is an expression of the form (D, C), where D is a finite set of atoms and $C \subseteq 2^D$ (the powerset of D). The elements in C are called *candidate solutions* of the c-atom. In the following, given a c-atom $A = (D, C)$, we use A_d and A_c to refer to its first and second components, D and C, respectively. The *complement* of a c-atom A, written $\overline{A}$, is then a c-atom such that $\overline{A}_d = A_d$ and $\overline{A}_c = 2^{A_d} \setminus A_c$. A c-atom is said to be *elementary* if it is of the form $(\{a\}, \{\{a\}\})$, where a is an atom. In the following, an elementary c-atom of the above form will be identified with a, and simply written as a. An *abstract constraint literal (c-literal)* is an expression of either the form A or the form *not* A, where A is a c-atom.

A *logic program with abstract constraint atoms* (*logic program* or *program*, for simplicity) is a finite set of *rules* of the form

$$A \leftarrow A_1, \ldots, A_k, not\, A_{k+1}, \ldots, not\, A_n \qquad (1)$$

where A and A_i's are c-atoms. Given a rule r of the form (1), we define $Hd(r) = A$ and $Bd(r) = Pos(r) \cup not\, Neg(r)$ where $Pos(r) = \{A_1, \ldots, A_k\}$, $Neg(r) = \{A_{k+1}, \ldots, A_n\}$ and $not\, S = \{not\, A | A \in S\}$ for a set S of c-atoms. A rule of the form (1) is *basic* (resp. *positive*) if A is elementary (resp. $k = n$). A logic program P is *basic* (resp. *positive*) if every rule in P is basic (resp. positive). By HB_P we denote the set of atoms occurring in the logic program P.

Definition 1 *Let M be a set of atoms and A a c-atom. We say that M* classically satisfies *A iff $M \cap A_d \in A_c$; M* classically satisfies *"not A" iff M does not satisfy A; M* classically satisfies *a set L of c-literals iff it satisfies each c-literal in L.*

For an atom p, the above definition implies that a set of atoms M classically satisfies p (resp. *not* p) iff $p \in M$ (resp. $p \notin M$). A set M of atoms *classically satisfies* a rule r iff M classically satisfies $Bd(r)$ implies M classically satisfies $Hd(r)$, M is a *classical model* of a logic program P iff M classically satisfies every rule in P.

A c-atom A is *monotonic* if for every M, if M classically satisfies A, then M' also classically satisfies A for any M' such that $M \subseteq M'$, otherwise it is *nonmonotonic*. A c-atom is *anti-monotonic* iff M classically satisfies A implies M' classically satisfies A for any M' such that $M' \subseteq M$.

Let M and S be two sets of atoms. The set S *conditionally satisfies* a c-atom A wrt M, denoted by $S \models_M A$, iff S classically satisfies A and $I \in A_c$ for every I with $S \cap A_d \subseteq I \subseteq M \cap A_d$.

Let P be a basic positive program and M a classical model of P. We define the operator $T_{(P,M)}$ as follows:

$$T_{(P,M)}(S) = \{Hd(r) | r \in P \text{ such that } S \models_M Bd(r)\}.$$

The operator is monotonic in the sense that if $S_1 \subseteq S_2 \subseteq M$ then $T_{(P,M)}(S_1) \subseteq T_{(P,M)}(S_2) \subseteq M$. Thus for any classical model M of P, the least fixpoint $T_{(P,M)}$ exists, denoted by $lfp(T_{(P,M)})$, which can be iteratively evaluated as below:

- $T^0_{(P,M)} = \emptyset$,
- $T^{n+1}_{(P,M)} = T_{(P,M)}(T^n_{(P,M)})$ where $n \geq 0$.

A classical model M of a basic positive program P is an *answer set* of P if M is the least fixpoint of $T_{(P,M)}$, i.e., $M = lfp(T_{(P,M)})$. There are two different answer set semantics for basic logic programs. One is defined by reduct, and the other is by complement. Formally, let P be a basic logic program and $M \subseteq HB_P$, the *reduct* of P wrt M, written P^M, is the positive program obtained from P by

- eliminating each rule r if M classically satisfies B for some $B \in Neg(r)$;
- eliminating *not* B from the remaining rules where B is a c-atom.

The *complement* of a basic logic program P, written $\overline{P}$, is the positive program obtained from P by replacing each *not* B with $\overline{B}$. It is clear that both P^M and $\overline{P}$ are positive and basic. For a basic logic program P and M a set of atoms, we say that M is a *c-answer set* (resp. *r-answer set*) of P iff M is an answer set of $\overline{P}$ (resp. P^M). By $rAS(P)$ (resp. $cAS(P)$) we denote the set of all r-answer sets (resp. c-answer sets) of P. It was shown that both the above answer set semantics generalize that of normal logic programs (cf. Proposition 3 of (Son, Pontelli, and Tu 2007)).

To study the characterization of such logic programs, a compact representation of c-atoms (called *local power set*) was proposed in (Shen, You, and Yuan 2009), which plays an important role in this paper. Formally, let S and J be two disjoint sets of atoms, by $S \uplus J$ we denote the set $\{S' | S \subseteq S' \subseteq S \cup J\}$ which is called the *S-prefixed powerset of* J. Let A be a c-atom, S and J two subsets of A_d. The S-prefixed powerset of J is *maximal in* A if $S \uplus J \subseteq A_c$ and there is no other sets S' and J' such that $S' \uplus J' \subseteq A_c$ and $S \uplus J \subset S' \uplus J'$. By A^*_c we denote the set of all maximal S-prefixed powerset of J in A for two sets S and J. The A^*_c is unique (cf. Theorem 3.2 (Shen, You, and Yuan 2009)). The *abstract representation* of a c-atom A is denoted by $A^* = (A_d, A^*_c)$.

Example 1 [Example 4 of (Son, Pontelli, and Tu 2007)] The aggregate program P consists of

$$p(1), \quad p(-1) \leftarrow p(2), \quad p(2) \leftarrow \text{SUM}(\{X | p(X)\}) \geq 1.$$

The aggregate atom $\text{SUM}(\{X|p(X)\})$ can be represented by the c-atom A with $A_d = \{p(-1), p(1), p(2)\}$, $A_c = \{\{p(1)\}, \{p(2)\}, \{p(1), p(2)\}, \{p(2), p(-1)\}, \{p(1), p(-1), p(2)\}\}$. One can easily check that $A_c^* = \{\{p(1)\} \uplus \{p(2)\}, \{p(2)\} \uplus \{p(1), p(-1)\}\}$. The interested readers can verify that the basic logic program has neither c-answer set nor r-answer set.

Unfounded Sets and Well-founded Models

Following the approach of (Van Gelder, Ross, and Schlipf 1991), we define the notion of unfounded sets first, and then use it to define the set of negative literals that can be derived according to the well-founded semantics.

Unfounded sets

Definition 2 *Let A be a c-atom and I a partial interpretation. We say that I* satisfies *A iff for some $S \uplus J \in A_c^*$, $S \subseteq I^+$ and $A_d \setminus (S \cup J) \subseteq I^-$; I* falsifies *A iff $S \cap I^- \neq \emptyset$ or $(A_d \setminus (S \cup J)) \cap I^+ \neq \emptyset$ for any $S \uplus J \in A_c^*$; I* satisfies *(resp.* falsifies*) not A iff I falsifies (resp. satisfies) A; I* satisfies *(resp.* falsifies*) a set of c-literals L iff I satisfies each (resp. falsifies some) c-literal in L.*

For a literal l, the above definition implies that a partial interpretation I satisfies (resp. falsifies) l iff $l \in I$ (resp. $\bar{l} \in I$). By $I \models \alpha$ (resp. $I \Vdash \alpha$) we mean the partial interpretation I satisfies (resp. falsifies) α, and by $I \not\models \alpha$ (resp. $I \not\Vdash \alpha$) we mean I does not satisfy (resp. falsify) α where α is a literal, c-literal, a set of literals or a set of c-literals.

A partial interpretation I *satisfies* a rule r iff I satisfies $Bd(r)$ implies I satisfies $Hd(r)$. A partial interpretation I is a *(partial) model* of a logic program P iff I satisfies every rule of P. A partial model of a logic program P is a *total model* of P if I is total.

Lemma 1 *Let P be a basic logic program and I a total interpretation of P. Then I is a total model of P iff I^+ is a classical model of P.*

Definition 3 *Let P be a basic program and I a partial interpretation. A set of atoms U is an* unfounded set *of P wrt I iff, for any $a \in U$ and any $r \in P$ with $Hd(r) = a$, at least one of the following two conditions holds:*

(c-i) $I \Vdash not\,Neg(r)$;

(c-ii) *there exists $A \in Pos(r)$ s.t, for any $S \uplus J \in A_c^*$, either $U \cap S \neq \emptyset$ or $I \Vdash S \cup not\,(A_d \setminus (S \cup J))$.*

It is easy to see that $\emptyset$ is an unfounded set of any basic logic program P wrt any partial interpretation. Please note that, the c-atom representation of an atom a is $A = (\{a\}, \{\{a\}\})$ where $A_c^* = \{\{a\} \uplus \emptyset\}$. It follows that the unfounded set defined here is a generalization to that of normal logic programs (Van Gelder, Ross, and Schlipf 1991).

Lemma 2 *Let P be a basic logic program, I a partial interpretation, and U_1 and U_2 two sets of atoms. If U_1 and U_2 are unfounded sets of P wrt I then $U_1 \cup U_2$ is also an unfounded set of P wrt I.*

The above lemma implies that the greatest (under set inclusion) unfounded set of a basic logic program P wrt a partial interpretation I always exists, that is the union of all unfounded sets of P wrt I. We present an algorithm to compute the greatest unfounded set of a basic logic program wrt a partial interpretation as shown in Algorithm 1. The following proposition justifies its soundness and completeness.

Algorithm 1: MaxUnfoundedSet(P, I)

```
Data: A basic logic program $P$ and a partial
      interpretation $I$
Result: The greatest unfounded set of $P$ wrt $I$
$U \leftarrow HB_P$;
while true do
    $U' = \{p \in U | \exists r \in P$ with $Hd(r) = p$ such that
      (a) $I \not\Vdash not\,Neg(r)$ and,
      (b) for any $A \in Pos(r)$, there is $S \uplus J \in A_c^*$ s.t
          $U \cap S = \emptyset$ and $I \not\Vdash S \cup not\,(A_d \setminus (S \cup J))\}$;
    if $U' \neq \emptyset$ then
        $U \leftarrow U \setminus U'$;
    else
        break;
return $U$;
```

Proposition 1 *Let P be a basic logic program and I a partial interpretation. The result returned by Algorithm 1 is the greatest unfounded set of P wrt I.*

Proof: As each **while**-loop iteration removes at least one atom from U, the algorithm terminates. When it terminates, for any $p \in U$ and any $r \in P$ with $Hd(r) = p$, either

(a) $I \Vdash not\,Neg(r)$, or

(b) for some $A \in Pos(r)$ and any $S \uplus J \in A_c^*$, either $U \cap S \neq \emptyset$ or $I \Vdash S \cup not\,(A_d \setminus (S \cup J))$.

It follows that U is an unfounded set of P wrt I.

To show the maximality of U after the termination of the algorithm, it is sufficient to show that, at each **while**-loop iteration, any set U^* with $U \subset U^* \subseteq U \cup U'$ is not an unfounded set of P wrt I. Let $p \in U^* \setminus U$. Note that $p \in U'$. It implies that there exists a rule $r \in P$ with $Hd(r) = p$ satisfying

- $I \not\Vdash not\ B$ for any $B \in Neg(r)$ and,
- for any $A \in Pos(r)$, there exists $S \uplus J \in A_c^*$ such that $(U \cup U') \cap S = \emptyset$ and $I \not\Vdash S \cup not\,(A_d \setminus (S \cup J))$.

Therefore, none of the conditions of Definition 3 applies in this case. Thus U^* is not an unfounded set of P wrt I. ■

As constructing A^* from a c-atom A is in polynomial time (cf. Theorem 3.5 of (Shen, You, and Yuan 2009)), it is not difficult to see that Algorithm 1 runs in polynomial time as well.

Well-founded models

Let P be a basic program and I a partial interpretation. We define the operators T_P, U_P and W_P as follows.

- $T_P(I) = \{Hd(r) | r \in P$ and I satisfies $Bd(r)\}$;
- $U_P(I) =$ the greatest unfounded set of P wrt I;

- $W_P(I) = T_P(I) \cup not\, U_P(I)$.

In terms of the above definition, we can show that

Proposition 2 *Let P be a basic logic program. Then we have that*

(1) T_P is monotonic.

(2) U_P is monotonic.

(3) W_P is monotonic.

By the monotonicity of W_P, the least fixpoint of W_P, denoted by $lfp(W_P)$, always exists and can be iteratively constructed as follows:

- $W_P^0 = \emptyset$,
- $W_P^{n+1} = W_P(W_P^n)$ for $n \geq 0$.

We call $lfp(W_P)$ the *well-founded model* of the basic logic program P, written by $WFS(P)$. The atoms in $[WFS(P)]^+$ are *well-founded* (wrt P) and the atoms in $[WFS(P)]^-$ are *unfounded* (wrt P). The operator T_P is clearly polynomial. It follows that W_P is polynomial as well since U_P is polynomial. Thus the well-founded model of a basic logic program can be computed in polynomial time.

Example 2 Let us consider the following logic programs:

- For the basic logic program P of Example 1, we can check that
  $T_P(\emptyset) = \{p(1)\}$ and $U_P(\emptyset) = \emptyset$;
  $T_P(\{p(1)\}) = \{p(1)\}$ and $U_P(\{p(1)\}) = \emptyset$.
  It follows that the well-founded model of P is $\{p(1)\}$.
- Given the program $P_1 = \{a \leftarrow not\,(\{a\}, \{\emptyset\})\}$, it is easy to see that P_1 has two r-answer sets $\emptyset$ and $\{a\}$, while the unique c-answer set of P_1 is $\emptyset$. Note that for the c-atom $A = (\{a\}, \{\emptyset\})$, $A_c^* = \{\emptyset \uplus \emptyset\}$. The partial interpretation $\emptyset$ neither satisfies $not\, A$ nor falsifies $not\, A$, since $\emptyset$ neither falsifies A nor satisfies A. Thus $T_{P_1}(\emptyset) = \emptyset$, and $U_{P_1}(\emptyset) = \emptyset$. It follows that $W_{P_1}(\emptyset) = \emptyset = lfp(W_{P_1})$, i.e., the well-founded model of P_1 is $\emptyset$.

Properties and Applications

In this section, we prove some properties about our well-founded semantics for basic logic programs, and show how it can be used to simplify basic logic programs.

Some model-theoretic properties

Lemma 3 *Let P be a basic logic program and I a total model of P. We have $U_P(I) = HB_P \setminus lfp(T_{(P^{I^+}, I^+)})$.*

Lemma 4 *Let P be a basic logic program and I a total model of P. We have $lfp(T_{(P^{I^+}, I^+)}) \subseteq T_P(I)$.*

The following theorem shows that the fixpoints of W_P exactly capture the r-answer sets of the basic program P.

Theorem 1 *Let P be a basic logic program and I a total model of P. Then I^+ is an r-answer set of P iff $W_P(I) = I$.*

Proof:(sketch) Let $M = lfp(T_{(P^{I^+}, I^+)})$. Since I is a total model of P, we have $U_P(I) = HB_P \setminus M$ and $M \subseteq T_P(I)$ by Lemmas 3 and 4. We show one direction only.

($\Leftarrow$) By $W_P(I) = I$ we have that $I^+ = T_P(I)$ and $I^- = HB_P \setminus M$, which implies $I^+ = M$, i.e., I^+ is an r-answer set of P. ■

In the above theorem, it is necessary to require I to be total. Otherwise the result would not be true in general. For instance, for the program $P = \{p \leftarrow not\, p\}$ and $I = \emptyset$, we have $W_P(I) = I$ but P has no answer set. On the other hand, for the program $P' = \{p \leftarrow not\, q\}$ and $I' = \{p\}$, we have that I'^+ is an answer set of P' but $W_{P'}(I') = \{p, not\, q\}$.

Since c-answer sets coincide with r-answer sets for basic positive programs, we have the following corollary.

Corollary 2 *Let P be a basic logic program and I a total model of P. Then I^+ is a c-answer set of P iff $W_{\overline{P}}(I) = I$.*

By the above two properties and Lemma 1, we have

Corollary 3 *Let P be a basic logic program, $M \subseteq HB_P$ and $M' = M \cup not\,(HB_P \setminus M)$. We have that*

- *if M is an r-answer set of P then $M' = W_P(M')$.*
- *if M is a c-answer set of P then $M' = W_{\overline{P}}(M')$.*

We can now state the main conclusion of this subsection, which asserts that every well-founded atom is in each answer set and no unfounded atoms can be in any answer sets. This result is the basis for simplifying basic programs.

Theorem 4 *Let P be a basic logic program. We have*

(1) $[WFS(P)]^+ \subseteq (\bigcap rAS(P))$,
$[WFS(P)]^+ \subseteq (\bigcap cAS(P))$.

(2) $[WFS(P)]^- \cap (\bigcup rAS(P)) = \emptyset$,
$[WFS(P)]^- \cap (\bigcup cAS(P)) = \emptyset$.

Proof:(sketch) (1) It is enough to show the former. Since $WFS(P)$ is the least fixpoint of W_P, we have

$$WFS(P) \subseteq \bigcap_{I:W_P(I)=I} W_P(I) = \bigcap_{I:W_P(I)=I} I.$$

For every r-answer set M of P, we can show that $M \cup not\,(HB_P \setminus M)$ is a fixpoint of W_P. It implies

$$WFS(P) \subseteq \bigcap_{M \in rAS(P)} M \cup not\,(HB_P \setminus M). \qquad (2)$$

Thus $WFS^+(P) \subseteq (\bigcap rAS(P))$.

(2) It suffices to show $[WFS(P)]^- \subseteq (HB_P \setminus (\bigcup rAS(P)))$ and $[WFS(P)]^- \subseteq (HB_P \setminus (\bigcup cAS(P)))$. Since every c-answer set of P is also an r-answer set of P, the former implies the latter. By the condition (2), we have

$$[WFS(P)]^- \subseteq \bigcap_{M \in rAS(P)} HB_P \setminus M = HB_P \setminus \bigcup rAS(P).$$

It completes the proof. ■

This theorem implies that, for the purpose of answer sets evaluation, well-founded models can be used to simplify programs.

Simplifying basic logic programs using well-founded models

Let A be a c-atom and I a partial interpretation. The *simplification of A wrt I*, denoted $R(A, I)$, is the c-atom (D, C) where $D = A_d \setminus (I^+ \cup I^-)$ and $C = \{S \setminus I^+ | S \in A_c \text{ and } S \cap I^- = \emptyset\}$. Intuitively, the candidate solutions containing unfounded atoms are discarded, and the well-founded atoms are deleted in remaining candidate solutions.

Let P be a basic logic program. The *simplification of P* under the well-founded model $\mathit{WFS}(P)$, denoted $R(P)$, is the basic logic program obtained from P by

- eliminating every rule r if either $\mathit{Hd}(r) \in [\mathit{WFS}(P)]^+$ or $\mathit{WFS}(P)$ falsifies $\mathit{Bd}(r)$,
- removing every c-literal which is satisfied by $\mathit{WFS}(P)$,
- replacing each remaining c-atom A with $R(A, \mathit{WFS}(P))$.

It is clear that $R(P)$ mentions no atom occurring in $\mathit{WFS}(P)$.

Example 3 [Continued from Example 1] We have that $R(P)$ consists of

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

It is obvious that $[R(A)]_c^* = \{\emptyset \uplus \{p(2)\}, \{p(2)\} \uplus \{p(-1)\}\}$ where A is the c-atom occurring in the body of the above second rule. One can verify that $R(P)$ has no r-answer set.

Lemma 5 *Let P be a basic logic program, A a c-atom and M a set of atoms such that $M \cap W^- = \emptyset$ and $W^+ \subseteq M$ where $W = \mathit{WFS}(P)$. Then we have that M classically satisfies A iff $M \setminus W^+$ classically satisfies $R(A, W)$.*

Lemma 6 *Let P be basic logic program and M a classical model of P such that $[\mathit{WFS}(P)]^- \cap M = \emptyset$. Then we have that $[W_P^i]^+ \subseteq \mathit{lfp}(T_{(P^M, M)})$ for any $i \geq 0$.*

Theorem 5 *Let P be a basic logic program. A set M of atoms is an r-answer set of P iff X is an r-answer set of $R(P)$ where $X = M \setminus [\mathit{WFS}(P)]^+$.*

Proof: (sketch) We prove one direction only. The other can be similarly proved using Lemma 6. Let $W = \mathit{WFS}(P)$.

($\Rightarrow$) It can be proved inductively by showing

$$T^k_{(P^M, M)} \setminus W^+ \subseteq \mathit{lfp}(T_{([R(P)]^X, X)}), \text{ for any } k \geq 0.$$

Base case: It is trivial for $k = 0$.

Inductively, suppose it holds for the case k. For any atom $h \in T^{k+1}_{(P^M, M)}$, there exists a rule $(r : h \leftarrow \mathit{Pos}, \mathit{not}\,\mathit{Neg})$ in P such that M does not classically satisfies B for any $B \in \mathit{Neg}$, and $T^k_{(P^M, M)} \models_M A$ for any $A \in \mathit{Pos}$.

By Theorem 4 we have $h \notin W^+ \cup W^-$ since $h \in M \setminus W^+$ and $M \cap W^- = \emptyset$. By Lemma 5, W does not falsify $\mathit{Bd}(r)$ since M classically satisfies $\mathit{Bd}(r)$. It follows that the rule $(r' : h \leftarrow \mathit{Pos}', \mathit{not}\,\mathit{Neg}')$ belongs to $R(P)$ where $\mathit{Pos}' = \{R(A, W) | A \in \mathit{Pos} \text{ and } W \not\models A\}$ and $\mathit{Neg}' = \{R(B, W) | B \in \mathit{Neg} \text{ and } W \not\models \mathit{not}\,B\}$. Thus the rule $(h \leftarrow \mathit{Pos}')$ is in $[R(P)]^X$. By the inductive assumption and Lemma 5, we have that $\mathit{lfp}(T_{([R(P)]^X, X)}) \models_X A'$ for any $A' \in \mathit{Pos}'$ since $X = M \setminus W^+$ and $T^k_{(P^M, M)} \models_M A$ for any $A \in \mathit{Pos}$. It follows $h \in \mathit{lfp}(T_{([R(P)]^X, X)})$. ■

Recall that if a basic logic program P is positive then its c-answer sets are exactly its r-answer sets. Thus we have

Corollary 6 *Let P be a basic logic program. A set M of atoms is a c-answer set of P iff X is a c-answer set of $R(\overline{P})$ where $X = M \setminus [\mathit{WFS}(\overline{P})]^+$.*

Most of the current ASP solvers first compute the well-founded semantics as a simplification step. By our results, a similar process can be done for basic logic programs. In particular, Theorem 5 and Corollary 6 suggest a way to compute the answer sets of basic logic programs in a way analogue to how SMODELS computes the answer sets for normal programs (Simons, Niemelä, and Soininen 2002).

Related Work

As mentioned in Introduction, our well-founded semantics for basic logic programs can be applied to aggregate programs and description logic programs. In the following, we compare our semantics with the well-founded semantics proposed previously for these logic programs (Faber 2005; Pelov, Denecker, and Bruynooghe 2007; Eiter et al. 2011). We start with aggregate programs. Due to space limitation, we omit the detailed technical definitions of aggregate programs, which can be found, e.g. in (Pelov, Denecker, and Bruynooghe 2007).

Aggregate programs

The aggregate programs proposed in (Pelov, Denecker, and Bruynooghe 2007; Alviano et al. 2011) do not allow default negation "*not*" in front of aggregate atoms, while classical negation "$\neg$" is allowed in (Pelov, Denecker, and Bruynooghe 2007), and aggregate atoms in (Alviano et al. 2011) require to be either monotonic or anti-monotonic. One may suggest that the default negation can be modeled as classical negation and the well-founded semantics in (Pelov, Denecker, and Bruynooghe 2007) can then be applied to aggregate programs with default negation. However, the following example shows a subtle difference.

Example 4 Let P be an aggregate program consisting of:

$$p(0) \leftarrow \mathit{not}\ \text{COUNT}(\{\langle 0 : p(0)\rangle, \langle 1 : p(1)\rangle\}) \neq 1.$$

If we take default negation as classical negation then the corresponding aggregate program P' consists of

$$p(0) \leftarrow \neg\text{COUNT}(\{\langle 0 : p(0)\rangle, \langle 1 : p(1)\rangle\}) \neq 1.$$

In terms of (Pelov, Denecker, and Bruynooghe 2007), we have that the ultimate well-founded model of P' is $(\emptyset, \emptyset)$ which corresponds to the partial interpretation $\{\mathit{not}\,p(0), \mathit{not}\,p(1)\}$. According to (Son, Pontelli, and Tu 2007), while we take P as the logic program with abstract constraint atoms P'' consisting of

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

the well-founded model P'' is $\{\mathit{not}\,p(1)\}$. One can verify that the well-founded model of $\overline{P''}$ is $\{\mathit{not}\,p(0), \mathit{not}\,p(1)\}$.

Faber (2005) proposed a notion of unfounded sets for aggregate programs which allows default negation preceding aggregate atoms. The following example shows the difference between Faber's and ours.

Example 5 Let P be the aggregate program consisting of

$$p(0) \leftarrow \mathit{not}\, \mathrm{COUNT}(\{Y : p(Y)\}) \leq 0.$$

P corresponds to the basic logic program P':

$$\{p(0) \leftarrow \mathit{not}\, (\{p(0)\}, \{\emptyset\})\}.$$

One can check that $\emptyset$ is the unique c-answer set of P', the unique answer set of P according to (Faber 2005), while P' has two r-answer sets $\emptyset$ and $\{p(0)\}$. Let $I = \emptyset$ and $X = \{p(0)\}$. It is not difficult to check that X is an unfounded set for P wrt I in terms of (Faber 2005), but X is not an unfounded set for P' wrt I in terms of the definition in this paper. Actually $\emptyset$ is the unique unfounded set for P wrt I, and the well-founded model of P' is $\emptyset$.

Description logic programs

Here, we introduce the basic description logic $\mathcal{ALC}$ (Baader et al. 2007), instead of the description logics $\mathcal{SHIF}$ and $\mathcal{SHOIN}$ described in (Eiter et al. 2008). For the language $\mathcal{ALC}$, we assume a vocabulary $\Psi = (\mathbf{A} \cup \mathbf{R}, \mathbf{I})$, where $\mathbf{A}, \mathbf{R}$ and $\mathbf{I}$ are pairwise disjoint (denumerable) sets of *atomic concepts*, *roles* (including equality $\approx$ and inequality $\not\approx$), and *individuals* respectively. The *concepts* of $\mathcal{ALC}$ are defined as follows:

$$C, D \longrightarrow A|\top|\bot|\neg C|C \sqcap D|C \sqcup D|\forall R.C|\exists R.C$$

where A is an atomic concept and R is a role. The *assertions* of $\mathcal{ALC}$ are of the forms $C(a)$ or $R(b, c)$, where C is a concept, R is a role, and a, b, c are individuals. An *inclusion axiom* of $\mathcal{ALC}$ has the form $C \sqsubseteq D$ where C and D are concepts. A *description knowledge base* (or *ontology*) of $\mathcal{ALC}$ is a set of inclusion axioms and assertions of $\mathcal{ALC}$. The semantics of $\mathcal{ALC}$ is defined by a mapping from description logic to first-order logic and then the resulting formulas are interpreted under classical first-order interpretations.

Let $\Phi = (\mathcal{P}, \mathcal{C})$ be a first-order vocabulary with nonempty finite sets $\mathcal{C}$ and $\mathcal{P}$ of constant symbols and predicate symbols respectively such that $\mathcal{P}$ is disjoint from $\mathbf{A} \cup \mathbf{R}$ and $\mathcal{C} \subseteq \mathbf{I}$. *Atoms* are formed from the symbols in $\mathcal{P}$ and $\mathcal{C}$ as usual.

A *dl-atom* is an expression of the form

$$\mathit{DL}[S_1\ op_1\ p_1, \ldots, S_m\ op_m\ p_m; Q](\vec{t}), \quad (m \geq 0) \tag{3}$$

where

- each S_i is either a concept, a role, or a special symbol in $\{\approx, \not\approx\}$;
- $op_i \in \{\oplus, \odot, \ominus\}$ (we call $\ominus$ the *constraint operator*);
- p_i is a unary predicate symbol in $\mathcal{P}$ if S_i is a concept, and a binary predicate symbol in $\mathcal{P}$ otherwise.
- $Q(\vec{t})$ is a *dl-query*, i.e., either (1) $C(t)$ where $\vec{t} = t$; (2) $C \sqsubseteq D$ where $\vec{t}$ is an empty argument list; (3) $R(t_1, t_2)$ where $\vec{t} = (t_1, t_2)$; (4) $t_1 \approx t_2$ where $\vec{t} = (t_1, t_2)$; or their negations, where C and D are concepts, R is a role, and $\vec{t}$ is a tuple of constants.

A *description logic program* (*dl-program* in short) $\mathcal{K}$ is a pair (O, P) where O is a ontology, which is a decidable first-order theory, such as $\mathcal{SHIF}$ or $\mathcal{SHOIN}$, and P is a finite set of ground dl-rules

$$A \leftarrow B_1, \ldots, B_m, \mathit{not}\, B_{m+1}, \ldots, \mathit{not}\, B_n, \tag{4}$$

where A is an atom, each B_i $(1 \leq i \leq n)$ is an atom or a dl-atom.

Given a dl-program $\mathcal{K} = (O, P)$, the *Herbrand base* of P, denoted by HB_P, is the set of atoms formed from the predicate symbols of $\mathcal{P}$ and the constant symbols in $\mathcal{C}$. An *interpretation* I (relative to P) is a subset of HB_P. Such an I is a *model* of an atom or dl-atom A under O, written $I \models_O A$, if the following holds:

- if $A \in \mathit{HB}_P$, then $I \models_O A$ iff $A \in I$;
- if A is a dl-atom $\mathit{DL}(\lambda; Q)(\vec{t})$ of the form (3), then $I \models_O A$ iff $O(I; \lambda) \models Q(\vec{t})$ where $O(I; \lambda) = O \cup \bigcup_{i=1}^{m} A_i(I)$ and, for $1 \leq i \leq m$,

$$A_i(I) = \begin{cases} \{S_i(\vec{e})|p_i(\vec{e}) \in I\}, & \text{if } op_i = \oplus; \\ \{\neg S_i(\vec{e})|p_i(\vec{e}) \in I\}, & \text{if } op_i = \odot; \\ \{\neg S_i(\vec{e})|p_i(\vec{e}) \notin I\}, & \text{if } op_i = \ominus; \end{cases}$$

where $\vec{e}$ is a tuple of constants over $\mathcal{C}$. While define well-founded semantics for description logic programs in (Eiter et al. 2011), the constraint operator $\ominus$ is not allowed. Thus in what follows, we assume that $\ominus$ does not appear in dl-atoms, unless otherwise stated explicitly.

The notions of unfounded sets and well-founded models of dl-programs are referred to (Eiter et al. 2011). In terms of the translation τ in (Wang et al. 2012), dl-programs can be modeled as basic logic programs.

Theorem 7 *Let $\mathcal{K} = (O, P)$ be a dl-program mentioning no constraint operators. We have $\mathit{WFS}(\mathcal{K}) = \mathit{WFS}(\tau(\mathcal{K}))$.*

To define the well-founded semantic for dl-programs containing constraint operators, Eiter *et al.* propose an approach of translating dl-programs into ones without constraint operators and then the well-founded semantics defined in (Eiter et al. 2011) can be applied. The following example shows some difference between their approach and ours.

Example 6 Let $\mathcal{K} = (O, P)$ where $O = \emptyset$ and

$$P = \{p(a) \leftarrow \mathit{DL}[S \odot p, S \ominus p; \neg S](a)\}.$$

The unique strong answer set of $\mathcal{K}$ is $\{p(a)\}$. In terms of our definition, the well-founded model of $\mathcal{K}$ is $\{p(a)\}$ as the corresponding basic logic program is

$$\{p(a) \leftarrow (\{p(a)\}, \{\emptyset, \{\{p(a)\}\}\})\}$$

According to the proposal in (Eiter et al. 2011), the corresponding dl-program without using constraint operator is $\mathcal{K}' = (O, P')$ where P' consists of

$$p(a) \leftarrow \mathit{DL}[S \odot p, S \odot p'; \neg S](a),$$
$$p'(a) \leftarrow \mathit{not}\, \mathit{DL}[S' \oplus p, S'](a).$$

We have that $T_{\mathcal{K}'}(\emptyset) = \emptyset$ and $U_{\mathcal{K}'}(\emptyset) = \emptyset$. Thus the well-founded model of $\mathcal{K}$ is $\emptyset$ according to (Eiter et al. 2011).

Conclusion and Future Work

In the paper we proposed a notion of unfounded set for basic logic programs with abstract constraint atoms, and based on that a well-founded semantics for such logic programs. We showed that our proposed well-founded semantics has many desirable properties similar to those possessed by normal logic programs, which can be used to simplify basic logic programs and thus form a foundation for answer set computation. The notion of well-founded semantics can also be applied to aggregate programs and description logic programs.

Our next step is to extend the well-founded semantics to disjunctive logic programs with abstract constraint atoms. We expect this to be a challenging task, and to do that some techniques and insights developed for disjunctive logic programs (e.g. (Wang and Zhou 2005; Cabalar, Odintsov, and Pearce 2006)) may prove helpful.

Acknowledgement

We thanks the reviewers for their useful comments. This work is partially supported by the Natural Science Foundation of China under grant 60963009. Yisong Wang is also partially supported by Open Funds of the State Key Laboratory of Computer Science of Chinese Academy of Science under grant SYSKF1106.

References

Alviano, M.; Calimeri, F.; Faber, W.; Leone, N.; and Perri, S. 2011. Unfounded sets and well-founded semantics of answer set programs with aggregates. *Journal of Artificial Intelligence Research* 42:487–527.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY: Cambridge University Press, 2nd edition.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54(12):92–103.

Cabalar, P.; Odintsov, S. P.; and Pearce, D. 2006. Logical foundations of well-founded semantics. In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, 2006*, 25–35. Lake District of the United Kingdom: AAAI Press.

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

Eiter, T.; Lukasiewicz, T.; Ianni, G.; and Schindlauer, R. 2011. Well-founded semantics for description logic programs in the semantic web. *ACM Transactions on Computational Logic (TOCL)* 12(2):11:1–11:41.

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

Faber, W. 2005. Unfounded sets for disjunctive logic programs with arbitrary aggregates. In *Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LPNMR 2005*, volume 3662 of *Lecture Notes in Computer Science*, 40–52. Diamante, Italy: Springer.

Ferraris, P. 2011. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic (TOCL)* 12(4):25:1–25:40.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Knorr, M.; Alferes, J. J.; and Hitzler, P. 2011. Local closed world reasoning with description logics under the well-founded semantics. *Artificial Intelligence* 175(9-10):1528–1554.

Lifschitz, V.; Tang, L. R.; and Turner, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence* 25(3-4):369–389.

Marek, V. W., and Truszczynski, M. 1999. Stable models and an alternative logic programming paradigm. In Apt, K.; Marek, V.; Truszczynski, M.; and Warren, D., eds., *The Logic Programming Paradigm: A 25-Year Perspective*. Berlin: Springer-Verlag. 375–398.

Marek, V. W., and Truszczynski, M. 2004. Logic programs with abstract constraint atoms. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence (AAAI 2004)*, 86–91. San Jose, California, USA: AAAI Press.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25(3-4):241–273.

Pelov, N.; Denecker, M.; and Bruynooghe, M. 2007. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming* 7(3):301–353.

Shen, Y.-D.; You, J.-H.; and Yuan, L.-Y. 2009. Characterizations of stable model semantics for logic programs with arbitrary constraint atoms. *Theory and Practice of Logic Programming* 9(4):529–564.

Simons, P.; Niemel*ä*, I.; and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence* 138(1-2):181–234.

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.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *J. ACM* 38(3):620–650.

Wang, K., and Zhou, L. 2005. Comparisons and computation of well-founded semantics for disjunctive logic programs. *ACM Transactions on Computational Logic* 6(2):295–327.

Wang, Y.; You, J.-H.; Yuan, L. Y.; Shen, Y.-D.; and Zhang, M. 2012. The loop formula based semantics of description logic programs. *Theoretical Computer Science* 415:60–85.