On the Progression Semantics and Boundedness of Answer Set Programs

Yan Zhang and Yi Zhou
Intelligent Systems Laboratory
School of Computing and Mathematics
University of Western Sydney, Australia
Email: {yan,yzhou}@scm.uws.edu.au

Abstract

In this paper, we propose a progression semantics for first-order answer set programs. Based on this new semantics, we are able to define the notion of boundedness for answer set programming. We prove that boundedness coincides with the notions of recursion-free and loop-free under program equivalence, and is also equivalent to first-order definability of answer set programs on arbitrary structures.

Introduction

In recent research on Answer Set Programming (ASP), the concept of first-order answer set programs has been developed, whereas second-order logic was used to define the underlying semantics (Ferraris, Lee, & Lifschitz 2007; Lin & Zhou 2007). While it provides a precise mathematic representation and also generalizes the traditional propositional ASP, this semantics, however, does not reveal much information about the expressiveness of first-order answer set programming. For instance, it is unclear whether we can provide a complete characterization for the first-order definability (expressiveness) of first-order ASP.

In this paper, we propose a progression semantics for first-order answer set programs. Intuitively, this semantics may be viewed as a generalization of Gelfond-Lifschitz transformation (Baral 2003) to the first-order case. We show that this new semantics is equivalent to the general stable model semantics proposed by Ferraris, Lee and Lifschitz (2007). Using the proposed progression semantics, we are able to define the notion of boundedness for first-order answer set programs. We prove that boundedness coincides with the notions of recursion-free and loop-free under program equivalence, and is also equivalent to first-order definability of answer set programs on arbitrary structures. We believe that results in this aspect will establish a foundation for the further study of the expressiveness and related properties of first-order ASP. We also address the differences between our work developed in this paper and earlier work in Datalog, relevant results in situation calculus and Wallace's first-order based semantics for normal logic programs.

The paper is organized as follows. Section 2 proposes the progression semantics and investigates its relationship to existing semantics. Section 3 defines the notion of boundedness for first-order answer set programs and discusses its basic properties. Section 4 further defines notions of recursion-free and loop free programs, and provides the main theorem of this paper stating that the three notions of boundedness, recursion-free and loop-free are actually equivalent, and they all coincide with the concept of first-order definability of answer set programs on arbitrary structures. Section 5 presents technical details of proving this theorem. Section 6 discusses related work, specifically addresses differences between our work developed in this paper and earlier work in datalog, and relevant results in situation calculus and Wallace's first-order alike semantics for normal logic programs. Finally, section 7 concludes this paper with some discussions.

Progression on Answer Set Programs

In this section, we will provide a progression based semantics for first-order answer set programs, which may be viewed as a generalization of Gelfond-Lifschitz's transformation for propositional answer set programs (Baral 2003), but different from existing semantics of first-order logic programs such as the ones presented in (Ferraris, Lee, & Lifschitz 2007; Lin & Zhou 2007).

Logical preliminaries

We start with necessary logic notions and concepts. We consider a second-order language without function symbols but with equality. A *vocabulary* τ is a set that consists of *relation symbols* (or *predicates*) including the *equality* symbol $=$ and *constant symbols* (or *constants*). Each predicate is associated with a natural number, called its *arity*. Given a vocabulary, *term*, *atom*, *substitution*, (first-order and second-order) *formula* and (first-order and second-order) *sentence* are defined as usual. In particular, an atom is called an *equality* atom if it has the form $t_1 = t_2$, where t_1 and t_2 are terms. Otherwise, it is called a *proper atom*.

A *structure* $\mathcal{A}$ of vocabulary τ (or a τ-*structure*) is a tuple $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, P_1^{\mathcal{A}}, \cdots, P_n^{\mathcal{A}})$, where A is a nonempty set called the *domain* of $\mathcal{A}$ (sometimes we use $\mathsf{Dom}(\mathcal{A})$ to denote $\mathcal{A}$'s domain), $c_i^{\mathcal{A}}$ $(1 \leq i \leq m)$ is an element in A for every constant c_i in τ, and $P_j^{\mathcal{A}}$ $(1 \leq j \leq n)$ is a k-ary *relation* over A for every k-ary predicate P_j in τ. $P_j^{\mathcal{A}}$ is also

called the *interpretations* of P_j in $\mathcal{A}$. A structure is *finite* if its domain is a finite set. In this paper, we consider both infinite and finite structures.

Let $\mathcal{A}$ be a structure of τ and $A = \mathsf{Dom}(\mathcal{A})$. An *assignment* in $\mathcal{A}$ is a function η from the set of variables to A. An assignment can be extended to a corresponding function from the set of terms to A by mapping $\eta(c)$ to $c^{\mathcal{A}}$, where c is an arbitrary constant. Let $P(\overrightarrow{x})$ be an atom, η an assignment in structure $\mathcal{A}$. We also use $P(\overrightarrow{x})\eta \in \mathcal{A}$ to denote $\eta(\overrightarrow{x}) \in P^{\mathcal{A}}$. The *satisfaction relation* $\models$ between a structure $\mathcal{A}$ and a formula ϕ associated with an assignment η, denoted by $\mathcal{A} \models \phi[\eta]$, is defined as usual. Let $\overrightarrow{x}$ be the set of free variables occurring in a formula ϕ. Then, the satisfaction relation only relies on the assignment of $\overrightarrow{x}$. In this case, we write $\mathcal{A} \models \phi(\overrightarrow{x}/\overrightarrow{a})$ to denote the satisfaction relation, where $\overrightarrow{a}$ is a tuple of elements in A. In particular, if ϕ is a sentence, then the satisfaction relation is independent from the assignment. In this case, we simply write $\mathcal{A} \models \phi$ for short.

Given a structure $\mathcal{A}$ of τ and an assignment η in $\mathcal{A}$, if Q is a predicate in τ, then we use $\mathcal{A} \cup \{Q(\overrightarrow{x})\eta\}$ to denote a new structure of τ which is obtained from $\mathcal{A}$ by expanding the interpretation of predicate Q in $\mathcal{A}$ (i.e. $Q^{\mathcal{A}}$) to $Q^{\mathcal{A}} \cup \{\eta(\overrightarrow{x})\}$.

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two structures of τ sharing the same domain, i.e. $\mathsf{Dom}(\mathcal{A}_1) = \mathsf{Dom}(\mathcal{A}_2)$, and for each constant c in τ, $c^{\mathcal{A}_1} = c^{\mathcal{A}_2}$. By $\mathcal{A}_1 \subseteq \mathcal{A}_2$, we simply mean that for each predicate $P \in \tau$, $P^{\mathcal{A}_1} \subseteq P^{\mathcal{A}_2}$. By $\mathcal{A}_1 \subset \mathcal{A}_2$, we mean that $\mathcal{A}_1 \subseteq \mathcal{A}_2$ but not $\mathcal{A}_2 \subseteq \mathcal{A}_1$. We write $\mathcal{A}_1 \cup \mathcal{A}_2$ to denote the structure of τ where the domain of $\mathcal{A}_1 \cup \mathcal{A}_2$ is the same as $\mathcal{A}_1$ and $\mathcal{A}_2$'s domain, each constant c is interpreted in the same way as in $\mathcal{A}_1$ and $\mathcal{A}_2$, and for each predicate P in τ, $P^{\mathcal{A}_1 \cup \mathcal{A}_2} = P^{\mathcal{A}_1} \cup P^{\mathcal{A}_2}$.

The progression semantics

A *rule* r is of the following form:

$$\alpha \leftarrow \beta_1, \ldots, \beta_m, \mathsf{not}\, \gamma_1, \ldots, \mathsf{not}\, \gamma_l, \qquad (1)$$

where α is a proper atom or the falsity $\bot$, β_i $(1 \leq i \leq m)$, and γ_j $(1 \leq j \leq l)$ are atoms. We say that α is the *head* of r, denoted by $Head(r)$; $\{\beta_1, \ldots, \beta_m\}$ the *positive body* of r, denoted by $Pos(r)$; and $\{\mathsf{not}\, \gamma_1, \ldots, \mathsf{not}\, \gamma_l\}$ the *negative body* of r, denoted by $Neg(r)$. In addition, we use $Body(r)$ to denote $Pos(r) \cup Neg(r)$.

A *normal logic program*, also called *(first-order) answer set program* or simply *program*, is a finite set of rules. Given a program Π, predicates that occur in the head of one of the rules in Π are said to be *intentional*; all other predicates are said to be *extensional*. For a given program Π, we use $\tau(\Pi)$ to denote the vocabulary of Π; $\tau_{ext}(\Pi)$ to denote all the extensional predicates in Π together with all the constants in Π; $\tau_{int}(\Pi)$ to denote all the intentional predicates of Π. Clearly, $\tau(\Pi) = \tau_{ext}(\Pi) \cup \tau_{int}(\Pi)$. In addition, $\tau_{int}(\Pi)$ contains no constants. We also use Ω_Π to denote the set of all intentional predicates of Π. Although Ω_Π is the same as $\tau_{int}(\Pi)$, we use two notations to make a difference because the former denotes a set of predicates whilst the latter presents a vocabulary.

Without loss of generality, we may assume that all rules are presented in a *normalized form*. That is, each intentional predicate Q is associated with a tuple of distinguishable variables $\overrightarrow{x_Q}$ so that the head of each rule is of the form $Q(\overrightarrow{x_Q})$. For instance, if for some rule with an intentional predicate Q of its head, there is a constant c occurring in Q, i.e. $Q(x_1, \cdots, x_{i-1}, c, x_{i+1}, \cdots, x_n)$, we simply introduce a new variable x_i to replace c: $Q(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n)$, and add atom $x_i = c$ in the body of this rule. We say that a variable x is a *local variable* of a rule r if it does not occur in the head of r. For convenience in our proofs, we assume that the sets of local variables in rules are pairwise disjoint.

The semantics of first-order answer set programs is defined through a generalization of stable model semantics on propositional answer set programs, and can be specified by a second-order sentence, as shown in (Ferraris, Lee, & Lifschitz 2007; Lin & Zhou 2007). Now we propose a new semantics of first-order answer set programs based on the notion of progression.

Definition 1 *(**Evaluation stage**) Let Π be a program and $\Omega_\Pi = \{Q_1, \ldots, Q_n\}$ the set of all the intentional predicates of Π. Consider a structure $\mathcal{M}$ of $\tau(\Pi)$. The t-*th simultaneous evolution stage *of Π based on $\mathcal{M}$, denoted as $\mathcal{M}^t(\Pi)$, is a structure of $\tau(\Pi)$ defined inductively as follows:*

$\mathcal{M}^0(\Pi) = (\mathsf{Dom}(\mathcal{M}), c_1^{\mathcal{M}^0}, \cdots, c_r^{\mathcal{M}^0}, P_1^{\mathcal{M}^0}, \cdots, P_s^{\mathcal{M}^0}, Q_1^{\mathcal{M}^0}, \cdots, Q_n^{\mathcal{M}^0})$, *where $c_i^{\mathcal{M}^0} = c_i^{\mathcal{M}}$ for each constant c_i of τ $(1 \leq i \leq r)$, $P_j^{\mathcal{M}^0} = P_j^{\mathcal{M}}$ for each extensional predicate P_j in $\tau_{ext}(\Pi)$ $(1 \leq j \leq s)$, and $Q_k^{\mathcal{M}^0} = \emptyset$ for each intentional predicate Q_k in Ω_Π $(1 \leq k \leq n)$;*

$\mathcal{M}^{t+1}(\Pi) = \mathcal{M}^t(\Pi) \cup \{Q_i(\overrightarrow{x})\eta \mid$ *there exists* $Q_i(\overrightarrow{x}) \leftarrow \beta_1, \ldots, \beta_m, \mathsf{not}\, \gamma_1, \ldots, \mathsf{not}\, \gamma_l \in \Pi$ *and an assignment η such that for all j $(1 \leq j \leq m)$, $\beta_j\eta \in \mathcal{M}^t(\Pi)$, and for all k $(1 \leq k \leq l)$, $\gamma_k\eta \notin \mathcal{M}\}$.*

Let us take a closer look at Definition 1. First, it is easy to see that $\mathcal{M}^0(\Pi)$ is just taking all extensional relations as the initial input, while all relations corresponding to intentional predicates in $\tau_{int}(\Pi)$ are set to be empty in $\mathcal{M}^0(\Pi)$.

Second, for each intentional predicate $Q \in \Omega_\Pi$, if we use $Q^{\mathcal{M}^t(\Pi)}$ to denote the relation that corresponds the interpretation of Q in structure $\mathcal{M}^t(\Pi)$, then for the sequence $\mathcal{M}^0(\Pi)$, $\mathcal{M}^1(\Pi)$, $\mathcal{M}^2(\Pi)$, $\cdots$, as specified above, we have a sequence $Q^{\mathcal{M}^0(\Pi)}$, $Q^{\mathcal{M}^1(\Pi)}$, $Q^{\mathcal{M}^2(\Pi)}$, $\cdots$, for each $Q \in \Omega_\Pi$. We also use $Q^i(\Pi, \mathcal{M})$ to denote $Q^{\mathcal{M}^i(\Pi)}$. It is easy to see that sequence $Q^0(\Pi, \mathcal{M})$, $Q^1(\Pi, \mathcal{M})$, $Q^2(\Pi, \mathcal{M})$, $\cdots$, always increases, that is, $Q^j(\Pi, \mathcal{M}) \subseteq Q^i(\Pi, \mathcal{M})$ for $j < i$. So a convergence for the sequence of $Q^0(\Pi, \mathcal{M})$, $Q^1(\Pi, \mathcal{M})$, $Q^2(\Pi, \mathcal{M})$, $\cdots$, always exists. We call $Q^\infty(\Pi, \mathcal{M}) = \bigcup_{1 \leq j \leq \infty} Q^j(\Pi, \mathcal{M})$ the *intended value* of Q on $\mathcal{M}$ with respect to Π. Consequently, the convergence of the sequence $\mathcal{M}^0(\Pi)$, $\mathcal{M}^1(\Pi)$, $\mathcal{M}^2(\Pi)$, $\cdots$, also exists: $\mathcal{M}^\infty(\Pi) = \bigcup_{0 \leq j \leq \infty} \mathcal{M}^j(\Pi)$.

If $Q(a_1, \ldots, a_n) \in \mathcal{M}^\infty(\Pi)$, then we say that $Q(a_1, \ldots, a_n)$ is a *link* of $\mathcal{M}$ with respect to Π. In addition, the *evolution time* of $Q(a_1, \ldots, a_n)$ on $\mathcal{M}$ with respect to Π is the least number t such that $Q(a_1, \ldots, a_n) \in \mathcal{M}^t(\Pi)$. In

particular, if $Q(a_1, \ldots, a_n)$ is not a link of $\mathcal{M}$, we treat the evolution time of $Q(a_1, \ldots, a_n)$ is ∞.

Definition 2 ***(Progression semantics)*** *Let Π be a first-order answer set program and $\mathcal{M}$ a structure of $\tau(\Pi)$. $\mathcal{M}$ is called a* stable model *(or* answer set*) of Π iff $\mathcal{M}^\infty(\Pi) = \mathcal{M}$. For convenience, we use $AS(\Pi)$ to denote the set of all answer sets of Π.*

Two programs are said to be *equivalent* if they have the same set of stable models.

Intuitively, the progression semantics for answer set programs may be viewed as a generalization of Gelfond-Lifschitz transformation (Baral 2003) to the first-order case. First, we guess a structure $\mathcal{M}$. Then, we evaluate the intended values of all intentional predicates based on the guessed structure. Finally, if all the intended values are the same as the ones specified in the guessed structure $\mathcal{M}$, then $\mathcal{M}$ is a stable model (answer set) of the underlying program.

Example 1 Consider the following program Π_G:

$$GoShopping(x, y) \leftarrow Friends(x, y),$$
$$GoShopping(x, y) \leftarrow GoShopping(x, z), Likes(z, y), \texttt{not}\ Hate(x, y).$$

Note that $GoShopping$ is the only intentional predicate in program Π_G. We consider a finite structure $\mathcal{M}$, where

$$\mathsf{Dom}(\mathcal{M}) = \{alice, carol, jane, sue\},$$
$$Friends^{\mathcal{M}} = \{(alice, carol), (jane, sue)\},$$
$$Likes^{\mathcal{M}} = \{(carol, sue)\},$$
$$Hate^{\mathcal{M}} = \{(alice, jane), (jane, alice)\},$$
$$GoShopping^{\mathcal{M}} = \{(alice, carol), (jane, sue), (alice, sue)\}.$$

Then from Definition 1, we obtain the following sequence:

$$GoShopping^0(\Pi_G, \mathcal{M}) = \emptyset,$$
$$GoShopping^1(\Pi_G, \mathcal{M}) = \{(alice, carol), (jane, sue)\},$$
$$GoShopping^2(\Pi_G, \mathcal{M}) = \{(alice, carol), (jane, sue), (alice, sue)\},$$
$$GoShopping^3(\Pi_G, \mathcal{M}) = GoShopping^2(\Pi_G, \mathcal{M}).$$

So $GoShopping^\infty(\Pi_G, \mathcal{M}) = \{(alice, carol), (jane, sue), (alice, sue)\}$. From Definition 2, we can see that $\mathcal{M}$ is also a stable model of Π_G. □

Relationship to other semantics

Now we show that our proposed progression based semantics actually coincides with existing semantics of first-order answer set programs, such as Ferraris *et al.*'s general stable model semantics (Ferraris, Lee, & Lifschitz 2007) under the restriction to normal logic programs. To this aim, we will first provide a translational semantics for first-order normal logic programs based on second-order logic. We then observe that such translational semantics coincides with current existing first-order answer set programming semantics. Finally, we prove our progression based semantics and translational semantics coincides.

Given a normal logic program Π, let $\Omega_\Pi = \{Q_1, \ldots, Q_n\}$ be the set of all intentional predicates of Π. Let $\Omega^*_\Pi = \{Q^*_1, \ldots, Q^*_n\}$ be a new set of predicates corresponding to Ω_Π, where each Q^*_i in Ω^*_Π has the same arity of predicate Q_i in Ω_Π. Given a rule r in Π of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_m, \texttt{not}\ \gamma_1, \ldots, \texttt{not}\ \gamma_l,$$

by $\widehat{r}$, we denote the universal closure of the following formula $\beta_1 \wedge \ldots \wedge \beta_m \wedge \neg\gamma_1 \wedge \ldots \wedge \neg\gamma_l \rightarrow \alpha$; by r^*, we denote the universal closure of the following formula $\beta^*_1 \wedge \ldots \wedge \beta^*_m \wedge \neg\gamma_1 \wedge \ldots \wedge \neg\gamma_l \rightarrow \alpha^*$, where $\alpha^* = Q^*(\overrightarrow{x})$ if $\alpha = Q(\overrightarrow{x})$ and

$$\beta^*_i, (1 \le i \le m) = \begin{cases} Q^*_j(\overrightarrow{t_j}) & \text{if } \beta_i = Q_j(\overrightarrow{t_j}) \text{ and } Q_j \in \Omega_\Pi \\ \beta_i & \text{otherwise.} \end{cases}$$

By $\widehat{\Pi}$, we denote the first-order sentence $\bigwedge_{r\in\Pi} \hat{r}$; by Π^*, we denote the first-order sentence $\bigwedge_{r\in\Pi} r^*$. Let Π be a normal logic program. The stable model semantics of Π, denoted by $SM(\Pi)$, is the following second-order sentence:

$$\widehat{\Pi} \wedge \neg\exists\Omega^*_\Pi((\Omega^*_\Pi < \Omega_\Pi) \wedge \Pi^*),$$

where $\Omega^*_\Pi < \Omega_\Pi$ is the abbreviation of the formula
$\bigwedge_{1\le i\le n} \forall\overrightarrow{x}(Q^*_i(\overrightarrow{x}) \rightarrow Q_i(\overrightarrow{x})) \wedge$
$\neg\bigwedge_{1\le i\le n} \forall\overrightarrow{x}(Q_i(\overrightarrow{x}) \rightarrow Q^*_i(\overrightarrow{x}))$.
A structure $\mathcal{M}$ of $\tau(\Pi)$ is called a *stable model* of Π if it is a model of $SM(\Pi)$. We call this *translational semantics*.

Our definition of second-order sentence $SM(\Pi)$ is somewhat different from Ferraris *et al.*'s (Ferraris, Lee, & Lifschitz 2007), in the sense that in Ferraris *et al.*'s $SM(\Pi)$, program Π is allowed to be an arbitrary first-order sentence, and there is no distinction between intentional and extensional predicates. Nevertheless, it is not difficult to observe that these two semantics are actually coincident by restricting to normal logic programs. In particular, for a given normal logic program Π and for each extensional predicate $P \in \tau_{ext}(\Pi)$, we introduce a new predicate P' which has the same arity of P, and add choice rules to Π:

$$P'(\overrightarrow{x}) \leftarrow \texttt{not}\ P(\overrightarrow{x}),$$
$$P(\overrightarrow{x}) \leftarrow \texttt{not}\ P'(\overrightarrow{x}).$$

In this way, the newly formed program, say Π', will have no extensional predicate. Then it can be showed that there is a one-to-one correspondence between the stable models of Π' under our above semantics definition and the stable models of Π under Ferraris *et al.*'s semantics.

We have noted that in their later version (Ferraris, Lee, & Lifschitz 2010), Ferraris *et al.* have modified their generalized stable model semantics for first-order answer set programs where intentional and extensional predicates were explicitly defined. Under their new semantics, it is obvious that without introducing choice rules, these two semantics are coincident as well by restricting to normal logic programs.

Theorem 1 *Let Π be a program and $\mathcal{M}$ a structure of $\tau(\Pi)$. $\mathcal{M}$ is a model of $SM(\Pi)$ iff $\mathcal{M}^\infty(\Pi) = \mathcal{M}$.*

Proof: In order to prove this theorem, we introduce an alternative semantics for first-order logic programs and show that it is equivalent to both the progression semantics and the translational semantics described above.

Let Π be a program and $\mathcal{M}$ a structure of $\tau(\Pi)$. We say that $\mathcal{M}$ is a *stable model* of Π iff

1. for every assignment η and every rule r of form (1) in Π, if for all i ($1 \leq i \leq m$), $\beta_i\eta \in \mathcal{M}$ and for all j ($1 \leq j \leq l$), $\gamma_j\eta \notin \mathcal{M}$, then $\alpha\eta \in \mathcal{M}$.
2. there does not exist a structure $\mathcal{M}'$ of $\tau(\Pi)$ such that
   (a) $\mathsf{Dom}(\mathcal{M}') = \mathsf{Dom}(\mathcal{M})$,
   (b) for each constant c in $\tau(\Pi)$, $c^{\mathcal{M}'} = c^{\mathcal{M}}$,
   (c) for each $P \in \tau_{ext}(\Pi)$, $P^{\mathcal{M}'} = P^{\mathcal{M}}$,
   (d) for all $Q \in \tau_{int}(\Pi)$, $Q^{\mathcal{M}'} \subseteq Q^{\mathcal{M}}$, and for some $Q \in \tau_{int}(\Pi)$, $Q^{\mathcal{M}'} \subset Q^{\mathcal{M}}$,
   (e) for every assignment η and every rule r of form (1) in Π, if for all i ($1 \leq i \leq m$), $\beta_i\eta \in \mathcal{M}'$ and for all j ($1 \leq j \leq l$), $\gamma_j\eta \notin \mathcal{M}$, then $\alpha\eta \in \mathcal{M}'$.

We first show that this semantics coincides with the translational semantics. It is not difficult to verify that Condition 1 holds iff $\mathcal{M} \models \widehat{\Pi}$. Now we prove that Condition 2 does not hold iff $\mathcal{M} \models \exists\Omega^*_\Pi((\Omega^*_\Pi < \Omega_\Pi) \wedge \Pi^*)$. On the one hand, suppose that there exists such an $\mathcal{M}'$, we construct n new relations in $\mathcal{M}$ on predicates $\Omega^*_\Pi = \{Q^*_1, \ldots, Q^*_n\}$ corresponding to $\Omega_\Pi = \{Q_1, \ldots, Q_n\}$ such that each $Q^* \in \Omega^*_\Pi$ and its corresponding $Q \in \Omega_\Pi$, $Q^{*\mathcal{M}} = Q^{\mathcal{M}'}$. Therefore, $\mathcal{M} \models \Omega^* < \Omega$ according to Condition 2(d). In addition, from Condition 2(e), it is easy to see that $\mathcal{M}$ satisfies Π^* where for each $Q^* \in \Omega^*_\Pi$, $Q^{*\mathcal{M}} = Q^{\mathcal{M}'}$ as specified above, here Q is Q^*'s corresponding predicate in Ω_Π. Hence, $\mathcal{M} \models \exists\Omega^*_\Pi((\Omega^*_\Pi < \Omega_\Pi) \wedge \Pi^*)$. On the other hand, suppose that $\mathcal{M} \models \exists\Omega^*_\Pi((\Omega^*_\Pi < \Omega_\Pi) \wedge \Pi^*)$. We can always construct $\mathcal{M}'$ in such a way: (1) $\mathsf{Dom}(\mathcal{M}') = \mathsf{Dom}(\mathcal{M})$; (2) for each constant c in $\tau(\Pi)$, $c^{\mathcal{M}'} = c^{\mathcal{M}}$; (3) for each $P \in \tau_{ext}(\Pi)$, $P^{\mathcal{M}'} = P^{\mathcal{M}}$; and (4) for each $Q \in \Omega_\Pi$ and its corresponding $Q^* \in \Omega^*_\Pi$, $Q^{\mathcal{M}'} = Q^{*\mathcal{M}}$. Then it is not difficult to observe that $\mathcal{M}'$ satisfies Conditions 2(c)-(e).

Now we show that this semantics also coincides with the progression semantics. Suppose that $\mathcal{M}^\infty(\Pi) = \mathcal{M}$. Then, Condition 1 holds. Otherwise, there exists an assignment η and a rule r such that, for all i ($1 \leq i \leq m$), $\beta_i\eta \in \mathcal{M}$ and for all j ($1 \leq j \leq l$), $\gamma_j\eta \notin \mathcal{M}$ but $\alpha\eta \notin \mathcal{M}$. Since $\beta_i\eta \in \mathcal{M}^\infty(\Pi)$, there exists a bound k such that for all i ($1 \leq i \leq m$), $\beta_i\eta \in \mathcal{M}^k(\Pi)$. Then, $\alpha\eta \in \mathcal{M}^{k+1}(\Pi)$ by the definition. This means that $\alpha\eta \in \mathcal{M}^\infty(\Pi)$. Therefore, $\alpha\eta \in \mathcal{M}$, a contradiction. In addition, Condition 2 must hold as well. Otherwise, let us assume that there exists such an $\mathcal{M}'$. By induction on the evolution stage t, it can be shown that for all t, $\mathcal{M}^t(\Pi) \subseteq \mathcal{M}'$. Therefore, $\mathcal{M}^\infty(\Pi) \subseteq \mathcal{M}'$. Hence, $\mathcal{M}^\infty(\Pi) \subseteq \mathcal{M}' \subset \mathcal{M}$, a contradiction. On the other hand, suppose that a structure $\mathcal{M}$ satisfies both Conditions 1 and 2. Then, it can be shown that $\mathcal{M}^t(\Pi) \subseteq \mathcal{M}$ by induction on the evolution stage t by Condition 1. Hence, $\mathcal{M}^\infty(\Pi) \subseteq \mathcal{M}$. Now we prove $\mathcal{M}^\infty(\Pi) \not\subset \mathcal{M}$. Otherwise, we construct a structure $\mathcal{M}'$ of $\tau(\Pi)$ in the following way: $\mathsf{Dom}(\mathcal{M}') = \mathsf{Dom}(\mathcal{M})$, for each constant $c \in \tau(\Pi)$, $c^{\mathcal{M}'} = c^{\mathcal{M}}$, for each extensional predicate $P \in \tau_{ext}(\Pi)$, $P^{\mathcal{M}'} = P^{\mathcal{M}}$, and for each intentional predicate $Q \in \Omega_\Pi$, $Q^{\mathcal{M}'} = Q^{\mathcal{M}^\infty(\Pi)}$. So $\mathcal{M}'$ satisfies Conditions 2(a)-(e) as well, a contradiction. Hence, $\mathcal{M}^\infty(\Pi) = \mathcal{M}$. □

Boundedness for Answer Set Programming

In this section, we define a notion of boundedness of answer set programs, which can be viewed as a generalization of such a notion in datalog programs (Ajtai & Gurevich 1994; Cosmadakis 1989; Vardi 1988). As we will show in the rest of this paper, the notion of boundedness plays an essential role in studying the expressive power of first-order answer set programs.

Definition 3 *(**Boundedness**) A program Π is* bounded *if there exists a natural number k, such that for all intentional predicates Q of Π and all stable models $\mathcal{M}$ of Π, $Q^\infty(\Pi, \mathcal{M}) = Q^k(\Pi, \mathcal{M})$; or equivalently, $\mathcal{M}^\infty(\Pi) = \mathcal{M}^k(\Pi)$. In this case, k is called a* bound *of Π, and Π is called a k-*bounded program.

Definition 3 is not the same as saying that for all stable models $\mathcal{M}$, there exists a natural number k such that $\mathcal{M}^\infty(\Pi) = \mathcal{M}^k(\Pi)$. It is important to note that the fixed constant k applies on all stable models, i.e., such k is independent from specific structures (stable models). Also, Definition 3 only takes all stable models but not all $\tau(\Pi)$-structures into account. Hence, for a k-bounded program Π, there may exist a $\tau(\Pi)$-structure $\mathcal{M}$ such that $\mathcal{M}^\infty(\Pi) \neq \mathcal{M}^k(\Pi)$. Certainly, here $\mathcal{M}$ is not a stable modal of Π.

Example 2 Consider the following program Π_V:

$$Visits(x, y) \leftarrow Interested(x, y), \texttt{not}\ Busy(x),$$
$$Visits(x, y) \leftarrow Visits(z, y), Attraction(y), \texttt{not}\ Busy(x).$$

In program Π_V, $Visits$ is the only intentional predicate. According to Definition 1, it is easy to verify that for any stable model $\mathcal{M}$ of Π_V, the evaluation time for all intended values of $Visits$ is not more than 2. In other words, program Π_V is a 2-bounded program. □

The following propositions reveal some basic properties of boundedness of answer set programs.

Proposition 1 *Let Π be a program and $\mathcal{M}$ a finite stable model of Π. There exists t such that $\mathcal{M}^t(\Pi) = \mathcal{M}^\infty(\Pi)$.*

Proposition 2 *Boundedness is closed under program equivalence. That is, if two programs Π_1 and Π_2 are equivalent, then Π_1 is bounded iff Π_2 is bounded.*

Boundedness, Recursion-free, Loop-free and First-order Definability

In this section, we investigate the relationships between boundedness and some important notions in first-order ASP, including recursion-free, loop-free and first-order definability. We show that, under program equivalence, boundedness coincides with both recursion-free and loop-free. Moreover, all of them coincide with the concept of first-order definability for answer set programs on arbitrary structures.

Recursion-free and loop-free programs

A program is *recursion-free* if no intentional predicates occur in the positive bodies of any rules in the program. It is possible that the intentional predicates may occur negatively in a recursion-free program.

Example 3 Consider the following program Π_{VP}:

$Visits(x, y) \leftarrow Interested(x, y)$,
$PossVisit(x, y) \leftarrow Attraction(y), \texttt{not}\ Visits(x, y)$.

There are two intentional predicates $Visits$ and $PossVisit$ in program Π_{VP}. Since none of them positively occurs in the bodies of two rules, Π_{VP} is a recursion-free program. □

From previous definitions, it is easy to show that the following result holds.

Proposition 3 *If Π is a recursion-free program, then $\mathcal{M}^{\infty}(\Pi) = \mathcal{M}^{1}(\Pi)$ for any structure $\mathcal{M}$ of $\tau(\Pi)$.*

Proposition 3 states that for recursion-free programs, the stable models of the program can be verified within one step. It immediately follows that all recursion-free programs are bounded.

Now we introduce the notion of loop-free programs. As shown in (Chen *et al.* 2006), under answer set semantics, a logic program can be captured by its completion together with all its loop formulas on finite structures. This means that all programs can be captured by a set (maybe infinite) of first-order sentences on finite structures. More precisely, given any program, there exists a (maybe infinite) set of first-order sentences, i.e., the completion together with all the loop-formulas of the program, such that the finite answer sets of the program are exactly the same as all the finite models of this set of sentences.

Let Π be a program. The positive dependency graph of Π, denoted by G_{Π}, is an infinite graph (V, E), where V is the set of atoms of $\tau_{int}(\Pi)$, and (α, β) is an edge in E if (a) there exists a rule $r \in \Pi$, and α' and β' in r such that α' is the head of r and β' is one of the positive atoms of intentional predicate in the body of r, and (b) there exists a substitution θ such that $\alpha'\theta = \alpha$ and $\beta'\theta = \beta$. A finite non-empty subset L of V is said to be a *loop* of Π if there exists a cycle in G_{Π} that goes through only and all the nodes in L. A program is called *loop-free* if it has no loops.

Example 4 Consider programs Π_V and Π_{VP} once again in Examples 2 and 3 respectively. It is easy to see that Π_V has a loop $L = \{Visits(x, y), Visits(z, y)\}$. So Π_V is not loop-free. On the other hand, program Π_{VP} in Example 3 is loop-free obviously. □

Proposition 4 *A recursion-free program must be loop-free.*

However, the converse of Proposition 4 does not hold in general. For example, the following program

$Visits(x, y) \leftarrow Friends(x, y)$,
$Friends(x, y) \leftarrow Likes(x, y), \texttt{not}\ Hate(y, x)$.

is loop-free but not recursion-free.

First-order definability

We say that a first-order sentence ϕ of vocabulary $\tau(\Pi)$ *defines* a program Π if the models of ϕ are exactly the stable models of Π. A program Π is said to be *first-order definable* if there exists a first-order sentence that defines Π.

Example 5 Let us consider Π_V again in Example 2. It can be verified that Π_V is defined by the following sentence:
$\forall xy(Visits(x, y) \leftrightarrow (Interested(x, y) \wedge \neg Busy(x) \vee \exists z(z \neq x \wedge Visits(z, y) \wedge Attraction(y) \wedge \neg Busy(x))))$.
□

For each rule r of form (1), we use $Head(r)$ and $\widehat{Body(r)}$ to denote α and the formula $\beta_1 \wedge \cdots \wedge \beta_m \wedge \neg\gamma_1 \wedge \cdots \wedge \neg\gamma_l$ respectively. For a given program Π, for each intentional predicate Q in Ω_{Π}, we specify the formula: $\phi_Q(\overrightarrow{x}) \leftrightarrow \bigvee_{\{r \in \Pi, Head(r) = Q(\overrightarrow{x})\}} \exists \overrightarrow{y}\, \widehat{Body(r)}$, where $\overrightarrow{y}$ are local variables in the rule r. Then the *completion* of program Π, denoted by $Comp(\Pi)$, is defined as follows:

$$\bigwedge_{Q \in \Omega_{\Pi}} \forall \overrightarrow{x} Q(\overrightarrow{x}) \leftrightarrow \phi_Q(\overrightarrow{x}).$$

The following result shows that if a program is loop-free, then it is defined by its completion, on both arbitrary structures and finite structures.

Proposition 5 *If Π is a loop-free program, then $Comp(\Pi)$ defines Π on both arbitrary structures and finite structures.*

The main theorem

Now we present the main theorem of this paper. The following theorem shows that boundedness exactly captures first-order definability for Answer Set Programming if infinite structures are allowed.

Theorem 2 ***(Main theorem)*** *Let Π be a program. The following four statements are equivalent on arbitrary structures.*

1. *Π is bounded.*
2. *Π is equivalent to a recursion-free program.*
3. *Π is equivalent to a loop-free program.*
4. *Π is first-order definable.*

Theorem 2 fails on finite structures. That is, there exists a program which is not bounded but first-order definable on finite structures (Ajtai & Gurevich 1994) as the following example shows.

Example 6 Consider program Π_Q from (Ajtai & Gurevich 1994) as follows:

$Q(x, y) \leftarrow E(x, y)$,
$Q(x, y) \leftarrow Q(x, z), Q(z, y)$,
$Q(x, y) \leftarrow Q(x, x), Q(y, y)$.

It is observed that Π_Q is not bounded on both arbitrary and finite structures. However, Ajtai and Gurevich showed that program Π_Q can be defined by the following first-order sentence on finite structures (Ajtai & Gurevich 1994):

$\forall xx'yy'z($
$(E(x,y) \supset Q(x,y)) \wedge ((Q(x,z) \wedge Q(z,y)) \supset Q(x,y)) \wedge$
$((Q(x,x') \wedge Q(x',x') \wedge Q(y',y') \wedge Q(y',y)) \supset Q(x,y)) \wedge$
$((Q(x,y) \wedge \neg E(x,y)) \supset \exists u(E(x,u) \wedge Q(u,y)) \wedge$
$((Q(x,y) \wedge \neg E(x,y)) \supset \exists v(E(v,y) \wedge Q(x,y))))$.

□

Proof of the Main Theorem

This section presents the proof of the main theorem, i.e., Theorem 2. Clearly, $2 \Rightarrow 3$ follows from Proposition 4; $3 \Rightarrow 4$ follows from Proposition 5. So it suffices to prove $1 \Rightarrow 2$ and $4 \Rightarrow 1$.

Given a program Π, we construct a program Π'_t to simulate the t-th evolution stage of Π. We define Π'_t inductively and show that Π'_t is a normalized program as well. Firstly, set $\Pi'_1 = \Pi$. Since Π is in a normalized form, Π'_1 is normalized too. We now specify Π'_{t+1} by giving Π'_t. A rule r^* is in Π'_{t+1} iff there exists a rule r in Π of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_m, \mathtt{not}\, \gamma_1, \ldots, \mathtt{not}\, \gamma_l,$$

and for all i ($1 \leq i \leq m$), if $\beta_i = Q_i(\overrightarrow{t})$ is an intentional atomic formula, then there exists a rule r_i in Π'_t such that $Head(r_i)\theta_i = \beta_i$, where θ_i is the substitution $\overrightarrow{x_{Q_i}}/\overrightarrow{t}$, and r^* is the following rule:

$$\begin{aligned} &Head(r^*) = Head(r),\\ &Pos(r^*) = Pos(r)\backslash\{\beta_{i_1}, \ldots, \beta_{i_n}\} \cup Pos(r_1)\theta_1 \cup \ldots \\ &\qquad\qquad \cup Pos(r_n)\theta_n,\\ &Neg(r^*) = Neg(r) \cup Neg(r_1)\theta_1 \cup \ldots \cup Neg(r_n)\theta_n, \end{aligned}$$

where $\{\beta_{i_1}, \ldots, \beta_{i_n}\}$ is the set of *all* intentional atomic formulas in $\{\beta_1, \ldots, \beta_m\}$, $r_1, \ldots, r_n$ are the corresponding rules in Π'_t as discussed above, and θ_i are defined accordingly. In addition, we apply necessary substitutions such that the sets of local variables in rules in Π'_{t+1} are pairwise disjoint. It is easy to see that Π'_{t+1} is a normalized program as well. Such process is similar to the *unfolding* in propositional logic programs, e.g. (Brass & Dix 1998).

Example 7 Consider program Π_V in Example 2. It is easy to see that $(\Pi'_V)_2$ consists of the following rules:

$$\begin{aligned} &Visits(x,y) \leftarrow Interested(x,y), \mathtt{not}\ Busy(x),\\ &Visits(x,y) \leftarrow Interested(z,y), Attraction(y),\\ &\qquad\qquad \mathtt{not}\ Busy(z), \mathtt{not}\ Busy(x),\\ &Visits(x,y) \leftarrow Visits(z',y), Attraction(y),\\ &\qquad\qquad \mathtt{not}\ Busy(z), \mathtt{not}\ Busy(x). \end{aligned}$$

□

Lemma 1 *Let Π be a program and k an integer. Then, $\mathcal{M}^k(\Pi) = \mathcal{M}^1(\Pi'_k)$ for any structure $\mathcal{M}$ of $\tau(\Pi)$.*

Proof: We prove this assertion by induction on k. Clearly, this assertion holds when $k = 1$. Suppose that for all $k < t$, this assertion holds. Now we prove that it holds when $k = t$ as well.

We first prove that $\mathcal{M}^t(\Pi) \subseteq \mathcal{M}^1(\Pi'_t)$. Let $(a_1, \ldots, a_n) \in Q^t(\mathcal{M})$, where Q is an intentional predicate of Π. If the evolution time of $Q(a_1, \ldots, a_n)$ is less than t, then $Q(a_1, \ldots, a_n) \in \mathcal{M}^1(\Pi'_t)$ by induction assumption. If the evolution time of $Q(a_1, \ldots, a_n)$ is exactly t, then according to the definition, there exists a rule $r \in \Pi$ of form (1) and an assignment η such that (a) $\overrightarrow{x_Q}\eta = (a_1, \ldots, a_n)$, (b) for all i ($1 \leq i \leq m$), $\beta_i\eta \in \mathcal{M}^{t-1}(\Pi)$, and (c) for all j ($1 \leq j \leq l$), $\gamma_j\eta \notin \mathcal{M}$. By induction assumption, for all i ($1 \leq i \leq m$), $\beta_i\eta \in \mathcal{M}^1(\Pi'_{t-1})$. If β_i is of the form $Q'(\overrightarrow{t})$, where Q' is an intentional predicate, then according to Definition 1, there exists a rule $r_i \in \Pi'_{t-1}$ such that $\beta_i\eta$ can be computed by r_i within one step by assuming $\mathcal{M}$. Therefore, $\alpha\eta$ can be computed by the following rule r^* within one step (note that Π'_k is normalized for all k).

$$\begin{aligned} &Head(r^*) = Head(r),\\ &Pos(r^*) = Pos(r)\backslash\{\beta_{i_1}, \ldots, \beta_{i_n}\} \cup Pos(r_1)\theta_1 \cup \ldots \\ &\qquad\qquad \cup Pos(r_n)\theta_n,\\ &Neg(r^*) = Neg(r) \cup Neg(r_1)\theta_1 \cup \ldots \cup Neg(r_n)\theta_n, \end{aligned}$$

where $\beta_{i_1}, \ldots, \beta_{i_n}$ are the atoms discussed above, and r_i and θ_i are defined accordingly. This shows that $Q(a_1, \ldots, a_n) \in \mathcal{M}^1_t(\Pi'_t)$.

We now prove $\mathcal{M}^1(\Pi'_t) \subseteq \mathcal{M}^t(\Pi)$. Suppose that $Q(a_1, \ldots, a_n)$ can be computed from Π'_t within one step by assuming $\mathcal{M}$, where Q is an intentional predicate of Π. Then there exists a rule $r^* \in \Pi'_t$, and an assignment η such that $Head(r^*)\eta = Q(a_1, \ldots, a_n)$. Suppose that r^* has the form

$$\begin{aligned} &Head(r^*) = Head(r),\\ &Pos(r^*) = Pos(r)\backslash\{\beta_{i_1}, \ldots, \beta_{i_n}\} \cup Pos(r_1)\theta_1 \cup \ldots \\ &\qquad\qquad \cup Pos(r_n)\theta_n,\\ &Neg(r^*) = Neg(r) \cup Neg(r_1)\theta_1 \cup \ldots \cup Neg(r_n)\theta_n, \end{aligned}$$

where $r \in \Pi$, $r_i \in \Pi'_{t-1}$, and the others are defined accordingly. Then, $\beta_{i_j}\eta$ can be computed from r_i within one step by assuming $\mathcal{M}$. So $\beta_{i_j}\eta \in \mathcal{M}^{t-1}(\Pi)$ by induction assumption. Consequently, $\alpha\eta \in \mathcal{M}^t(\Pi)$ since it can be computed through rule r. □

For a given program Π, having specified program Π'_k as described above, we now further define Π_k as the program obtained from Π'_k by deleting all rules whose positive bodies contain some intentional predicates. Clearly, Π_k is a recursion-free program.

Lemma 2 *Let Π be a program and k an integer. Then, $\mathcal{M}^1(\Pi_k) = \mathcal{M}^1(\Pi'_k)$ for any structure $\mathcal{M}$ of $\tau(\Pi)$.*

Proof: This result follows from the definition since for the given structure $\mathcal{M}$ of $\tau(\Pi)$ and for each intentional predicate Q of Π, $Q^0(\mathcal{M})$ is empty in any case. □

Now we show that $1 \Rightarrow 2$ follows from Proposition 3, Lemma 1 and Lemma 2. Let Π be a k-bounded program and $\mathcal{M}$ a structure of $\tau(\Pi)$. Then, $\mathcal{M}$ is an answer set of Π iff $\mathcal{M} = \mathcal{M}^\infty(\Pi)$ (by Definition 2) iff $\mathcal{M} = \mathcal{M}^k(\Pi)$ (since $\mathcal{M}^\infty(\Pi) = \mathcal{M}^k(\Pi)$ by Definition 3) iff $\mathcal{M} = \mathcal{M}^1(\Pi_k)$ (by Lemmas 1 and 2) iff $\mathcal{M} = \mathcal{M}^\infty(\Pi_k)$ (by Proposition 3) iff $\mathcal{M}$ is answer set of Π_k.

Next, we turn into proving $4 \Rightarrow 1$. For this purpose, we need to introduce some background knowledge on least

fixed-point logic. Let τ be a vocabulary and P a new predicate not in τ with the arity n. Let $\phi(\overrightarrow{x}, P)$ be a first-order formula, where $\overrightarrow{x}$ is the tuple of all free variables in ϕ with length n, and P only occurs positively in ϕ (i.e. every occurrence of P in ϕ is in the scope of even numbers of negations[1]). Given a structure $\mathcal{A}$ of τ, the formula $\phi(\overrightarrow{x}, P)$ defines an operator $\Phi(T)$ from n-ary relation T on $\mathsf{Dom}(\mathcal{A})$ to n-ary relation on $\mathsf{Dom}(\mathcal{A})$:

$$\Phi(T) = \{\overrightarrow{a} \in \mathsf{Dom}(\mathcal{A})^n : \mathcal{A} \models \phi(\overrightarrow{x}/\overrightarrow{a}, T)\}.$$

The least-fixed point formula $\phi^\infty(\overrightarrow{x}, P)$ (ϕ^∞ for short) on $\mathcal{A}$ is constructed inductively as follows:

$$\phi^0(\overrightarrow{x}, P) = \emptyset;$$
$$\phi^t(\overrightarrow{x}, P) = \Phi(\bigcup_{r<t} \phi^r(\overrightarrow{x}, P)).$$

Given $\overrightarrow{a} \in Dom(\mathcal{A})^n$, $\mathcal{A} \models \phi^t(\overrightarrow{x}/\overrightarrow{a}, P)$ iff $\overrightarrow{a} \in \phi^t(\overrightarrow{x}, P)$; $\mathcal{A} \models \phi^\infty(\overrightarrow{x}/\overrightarrow{a}, P)$ iff $\overrightarrow{a} \in \phi^\infty(\overrightarrow{x}, P)$. Since P only positively occurs in ϕ, the sequence $\phi^1, \ldots, \phi^t, \ldots$ always increases. Thus, there exists a least ordinal k such that $\phi^k = \phi^t = \phi^\infty$, where $t > k$.

The notion of definability and boundedness can be defined for least fixed-point logic as well. Let $\mathcal{K}$ be a class of τ-structures. We say that a formula $\psi(\overrightarrow{y})$, where $\overrightarrow{y}$ is the tuple of all free variables in ψ with length n, of τ *defines* the fixed-point $\phi^\infty(\overrightarrow{x}, P)$ on $\mathcal{K}$ iff for every $\mathcal{A} \in \mathcal{K}$ and every $\overrightarrow{a} \in \mathsf{Dom}(\mathcal{A})^n$,

$$\mathcal{A} \models \phi^\infty(\overrightarrow{x}/\overrightarrow{a}, P) \text{ iff } \mathcal{A} \models \psi(\overrightarrow{y}/\overrightarrow{a}).$$

We say that the least-fixed point formula $\phi^\infty(\overrightarrow{x}, P)$ is *bounded* on $\mathcal{K}$ if there exists a fixed natural number k such that for all $\mathcal{A} \in \mathcal{K}$ and every $\overrightarrow{a} \in \mathsf{Dom}(\mathcal{A})^n$, $\mathcal{A} \models \phi^\infty(\overrightarrow{x}/\overrightarrow{a}, P)$ iff $\mathcal{A} \models \phi^k(\overrightarrow{x}/\overrightarrow{a}, P)$.

Barwise and Moschovakis (1978) revealed the important correspondence between definability and boundedness on arbitrary structures in least fixed-point logic.

Theorem 3 *(Barwise & Moschovakis 1978) Let $\mathcal{K}$ be a class of τ-structures which is first-order finitely axiomatizable[2]. A least fixed-point formula is bounded on $\mathcal{K}$ iff it is defined by a first-order formula on $\mathcal{K}$.*

We prove $4 \Rightarrow 1$ in Theorem 2 based on Theorem 3. The basic ideas are divided into two steps. First, we show that for each program, we can construct a program with a single intentional predicate to simulate the original program. Then we show that each program with a single intentional predicate can be translated to an equivalent fixed-point formula.

Let Π be a program. Let $\{P_1, \ldots, P_n\}$ be the set of intentional predicates of Π. Suppose that k is the maximal arity among all $P_i, (1 \leq i \leq n)$. Let $0, 1, \ldots, n$ be $n+1$ distinguishable new constants. Construct a new predicate P whose arity is $k+1$. Let Π^S be the program obtained from Π by simultaneously replacing each atom $P_i(\overrightarrow{t_i})$ in Π with $P(\overrightarrow{t_i}, 0, \ldots, 0, i)$, where the number of occurrences of 0 is equal to $k - |\overrightarrow{t_i}|$. We show that Π^S simulates Π.

[1]Here we assume that ϕ is constructed only from connectives of $\neg$, $\wedge$ and $\vee$, while $\rightarrow$ and $\leftrightarrow$ are defined in terms of $\neg$, $\wedge$ and $\vee$.

[2]That is, there exists a first-order sentence ϕ on τ whose models are exactly captured by $\mathcal{K}$.

Lemma 3 *Let Π be a program and Π^S be the program constructed above. Let $\mathcal{M}$ be a structure of $\tau(\Pi)$. We construct a structure $\mathcal{M}^S$ on $\tau_{ext}(\Pi) \cup \{P\}$ such that*

- *the domain of is $\mathcal{M}^S$ is $M \cup \{0, 1, \ldots, n\}$;*
- *for all extensional predicates Q of Π, $Q^{\mathcal{M}^S} = Q^{\mathcal{M}}$;*
- *for all constants c in Π, $c^{\mathcal{M}^S} = c^{\mathcal{M}}$;*
- *for all intentional predicates P_i, $P_i(\overrightarrow{a}) \in \mathcal{M}$ iff $P(\overrightarrow{a}, 0, \ldots, 0, i) \in \mathcal{M}^S$.*

Then, for any integer k, P_i, and $\overrightarrow{a}$ that matches the arity of P_i, $P_i(\overrightarrow{a}) \in \mathcal{M}^k(\Pi)$ iff $P(\overrightarrow{a}, 0, \ldots, 0, i) \in (\mathcal{M}^S)^k(\Pi^S)$.

Proof: This assertion follows from the constructions and definitions by induction on k. □

Lemma 3 shows that Π^S can simulate Π in the sense that every intentional atom $P_i(\overrightarrow{t_i})$ in Π is associated with the intentional atom $P(\overrightarrow{t_i}, 0, \ldots, 0, i)$ in Π^S.

Now we show that each program with a single intentional predicate can be equivalently transferred into a fixed-point formula on a class of axiomatizable structures. Let Π be a program that only contains a single intentional predicate, say P. Then, all the heads of rules in Π are of the form $P(\overrightarrow{x})$ since Π is normalized. Let P^* be a new predicate that has the same arity as P. Let $\psi(\Pi, P^*)$ be the first-order formula obtained from Π and P^* by two steps: (1) construct a program Π^* by replacing every occurrence of $P(\overrightarrow{t})$ in the negative bodies of any rules in Π with $P^*(\overrightarrow{t})$, (2) let $\psi(\Pi, P^*)$ be the formula $\bigvee_{r \in \Pi^*} \exists \overrightarrow{y} \widehat{Body(r)}$, where $\overrightarrow{y}$ is the set of local variables in rule r. Clearly, $\psi(\Pi, P^*)$ is a first-order formula of the vocabulary $\tau(\Pi) \cup \{P^*\}$, where P only occurs positively and $\overrightarrow{x}$ are all the free variables.

Let $\mathcal{M}$ be a $\tau(\Pi)$-structure. By $\mathcal{M}^*$, we denote the structure of the vocabulary $\tau(\Pi) \cup \{P^*\}$ such that

- $\mathsf{Dom}(\mathcal{M}^*) = \mathsf{Dom}(\mathcal{M})$;
- for all $\overrightarrow{a}$, $P^*(\overrightarrow{a}) \in \mathcal{M}^*$ iff $P(\overrightarrow{a}) \in \mathcal{M}$;
- the interpretations of all constants and other predicates are the same as those in $\mathcal{M}$.

The fixed-point formula $\psi(\Pi, P^*)^\infty(\overrightarrow{x}, P)$ simulates the program Π on all answer sets of Π. By induction on k, the following lemma holds.

Lemma 4 *Let Π be a program that has a single intentional predicate P, and $\mathcal{M}$ an answer set of $\tau(\Pi)$. Suppose that $\psi(\Pi, P^*)$ and $\mathcal{M}^*$ are constructed as above. Then, for any integer k and any $\overrightarrow{a}$, $P(\overrightarrow{a}) \in \mathcal{M}^k(\Pi)$ iff $\overrightarrow{a} \in \psi(\Pi, P^*)^k(\overrightarrow{x}, P)$.*

Now we show $4 \Rightarrow 1$. From Lemma 3, it suffices to prove the case in which the program only contains a single intentional predicate. Let Π be such a program, which has a single intentional predicate P and is defined by a first-order sentence ϕ. Let $\mathcal{K} = \{\mathcal{M}^* \mid \mathcal{M} \in AS(\Pi)\}$. Then, $\mathcal{K}$ is first-order axiomatized by $\phi \wedge \forall \overrightarrow{x}(P(\overrightarrow{x}) \leftrightarrow P^*(\overrightarrow{x}))$. By Lemma 4, the fixed-point formula $\psi(\Pi, P^*)^\infty(\overrightarrow{x}, P)$ on $\mathcal{K}$ is defined by the formula $\phi^* \wedge P^*(x)$, where ϕ^* is obtained from ϕ by simultaneously replacing each occurrence of $P(\overrightarrow{t})$ with $P^*(\overrightarrow{t})$. Then, by Theorem 3,

$\psi(\Pi, P^*)^\infty(\overrightarrow{x}, P)$ is bounded on $\mathcal{K}$. Again, by Lemma 4, Π is bounded.

Related Work

In this section, we discuss the relationships between our progression semantics for answer set programs and other progression or progression-like semantics for datalog and normal logic programs. We address the main difference between our progression semantics definition and others.

Progression semantics for datalog programs

As showed from Definitions 1 and 2, our progression semantics for first-order answer set programs is defined based on the simultaneous evaluation stage of a given program Π, which may be viewed as a generalization of the stage evaluation for intentional predicates for datalog programs (Ajtai & Gurevich 1994). However, since intentional predicates do not occur in the negative bodies of rules in a datalog program, datalog simultaneous evaluation stage definition is not applicable for defining the progression semantics of answer set programs.

In fact, even for non-standard datalog programs where intentional predicates are allowed in the negative bodies of rules, their evaluation stage definition will not result in the answer set semantics, unlike ours developed in this paper (see Theorem 1). In particular, consider datalog programs under well-founded semantics, named *WF-datalog programs*, where intentional predicates are allowed to occur in the negative bodies of rules. The evaluation stage for intentional predicates for a WF-datalog program is defined quite differently from our Definition 1, in the sense that during each evaluation stage, the interpretations for negative intentional predicates in the rule bodies are *not* fixed by the given structure $\mathcal{M}$, instead, they are assigned by the values obtained from the previous evaluation stage (see Chapter 9 in (Ebbinghaus & Flum 1999) for details).

Situation calculus and Wallace's semantics for normal logic programs

Lin and Reiter have developed a situation calculus semantics to capture the answer set semantics of first-order (normal) logic program (Lin & Reiter 1997). In their approach, a rule of the form (1) in a logic program Π is rewritten as an *effect axiom* under the framework of situation calculus as follows:

$$Poss(A(\overrightarrow{x}), s) \rightarrow ((\beta_1(s) \wedge \cdots \wedge \beta_m(s) \wedge \neg\gamma_1(s) \wedge \cdots \wedge \neg\gamma_l(s)) \rightarrow \alpha(\overrightarrow{x}, do(A(\overrightarrow{x}), s))).$$

Intuitively, this formula interprets the head of the underlying logic program rule as a result by performing some action at a situation where all atoms in the body hold. Here term $do(A(\overrightarrow{x}), s)$ is viewed as a resulting situation from situation s by performing action $A(\overrightarrow{x})$. In this sense, we may think that a concept of progression stage is employed in this semantics.

However, the situation calculus semantics differs from our progression semantics because the former is based on second-order logic[3], while the later is defined on a basis of the fixed-point of a structure sequence.

Besides Lin and Reiter's situation calculus approach, Wallace proposed a so-called *tightened completion semantics* based on classical first-order logic to capture the answer set semantics of logic programs where an idea of progression is presented (Wallace 1993). In Wallace's approach, a program Π is translated into a tightened program Π', where each rule of the form (1) in Π is rewritten as the following form in Π':

$$\alpha(\overrightarrow{x}, s(n)) \leftarrow \beta_1(\overrightarrow{y_1}, n), \cdots, \beta_m(\overrightarrow{y_m}, n), \texttt{not}\ \gamma_1, \cdots, \texttt{not}\ \gamma_l, \quad (2)$$

where $s(n)$ is the successor of the natural number n, and for each predicate $P(\overrightarrow{x})$ in Π, Π' also contains a rule

$$P(\overrightarrow{x}) \leftarrow P(\overrightarrow{x}, n).$$

Wallace showed that the Herbrand models of completion of tightened program Π' precisely capture the stable models of program Π under the restriction to predicates in Π (Wallace 1993). Wallace further extended this result to non-Herbrand universes.

By taking a closer look at rule (2), we can see that the interpretation for a predicate in program Π is progressively generated through the interpretations of predicates occurring in the positive bodies of the rules in the previous stage, while keeping those negative predicate interpretations fixed. This is similar to our evaluation stage definition (see Definition 1).

Nevertheless, the key difference between Wallace's semantics and ours is clear: Wallace's approach introduces the successor function $s(n)$ where n is a natural number to encode the progression process, a consequence of this is: for non-Herbrand universes, a set of induction axioms has to be added into the tightened completion to handle negative atoms:

$$\forall\overrightarrow{x}\neg P(\overrightarrow{x}, 0) \wedge (\forall N \neg P(\overrightarrow{x}, N) \rightarrow \neg P(\overrightarrow{x}, s(N))) \rightarrow \forall M \neg P(\overrightarrow{x}, M). \quad (3)$$

Since Wallace's approach involves the reasoning with natural numbers, though (3) is a first-order sentence, it is not sufficient to precisely represent the underlying progression semantics. We will need Peano axioms for natural numbers whereas a second-order induction axiom is included (Mendelson 1987). In this sense, like Lin and Reiter's the situation calculus semantics, Wallace's approach is also in second-order logic.

Conclusions

The fixed-point style progression semantics proposed in this paper precisely captures current semantics of first-order answer set programs (see Theorem 1). One main technical issue in developing such semantics for answer set programming is the way of handling negation as failure for intentional predicates, while in standard datalog no such handling is needed, and in WF-datalog, it is handled differently. On

[3]Recall that the induction axiom in the situation calculus is a second-order sentence.

the other hand, in both Lin and Reiter and Wallace's approaches (Lin & Reiter 1997; Wallace 1993), the concept of progression is encoded relying on second-order logic.

The progression semantics enables us to define an explicit notion of boundedness, therefore to provide a technical basis to study various fundamental issues in relation to first-order answer set programming. Our main result (Theorem 2) regarding the equivalence among boundedness, recursion-free, loop-free and first-order definability on arbitrary structures sheds new insights for a better understanding of the expressive power of (first-order) normal logic programs under answer set semantics. Also, the techniques introduced to prove the two theorems are useful for studying other related topics in first-order answer set programming.

For future work, an important problem worth pursuing is, given a fixed natural number k, to identify some sufficient conditions, particularly tractable syntactical conditions, of k-bounded programs, whose answer sets could be computed in certain easier ways.

Acknowledgement

We specially thank Fangzhen Lin for his initial inspiration on this work. Our results presented in this paper provide a complete answer to his conjecture on the equivalence among boundedness, loop-free and first-order definability under answer set semantics.

This research is supported in part by an Australian Research Council Discovery Projects grant DP0988396.

References

Ajtai, M., and Gurevich, Y. 1994. Datalog vs. first order logic. *J. of Computer and System Sciences* 49:562–588.

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. MIT Press.

Barwise, J., and Moschovakis, Y. 1978. Global inductive definability. *Journal of Symbolic Logic* 43:521–534.

Brass, S., and Dix, J. 1998. A general framework for semantics of disjunctive logic programs based on partial evaluation. *Journal of Logic Programming* 38:167–213.

Chen, Y.; Lin, F.; Wang, Y.; and Zhang, M. 2006. First-order loop formulas for normal logic programs. In *Proceedings of KR-2006*, 298–307.

Cosmadakis, S. 1989. On the first-order expressibility of recursive queries. In *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on PODS*, 311–323.

Ebbinghaus, H., and Flum, J. 1999. *Finite Model Theory*. 2nd edition, Springer.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2007. A new perspective on stable models. In *Proceedings of IJCAI-2007*, 372–379.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2010. Stable models and circumscription. *Artificial Intelligence (to appear)*.

Lin, F., and Reiter, R. 1997. Rules as actions: A situation calculus semantics for logic programs. *Journal of Logic Programming* 31:299–330.

Lin, F., and Zhou, Y. 2007. From answer set logic programming to circumscription via logic of gk. In *Proceedings of IJCAI-2007*, 441–661.

Mendelson, R. 1987. *Introduction to Mathematical Logic*. 3rd edition, Chapman & Hall.

Vardi, M. 1988. Decidability and undecidability results for boundedness of linear recursive queries. In *Proceedings of the 7th ACM Symp. on Principles of Database Systems*, 341–351.

Wallace, M. 1993. Tight, consistent, and computable completions for unrestricted logic programs. *Journal of Logic Programming* 15:243–273.