Some Negative Results on First-order Definability of Answer Set Programs

Yin Chen
Department of Computer Science
South China Normal University, China
Email: gzchenyin@gmail.com

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

Abstract
A logic program with variables is first-order definable if the set of its (finite) answer sets can be captured by a first-order sentence, otherwise this program is first-order indefinable. In this paper, we study the problem of first-order indefinability of answer set programs. By extending traditional Ehrenfeucht-Frassï¿½ games approach in finite model theory, we provide an Ehrenfeucht-Frassï¿½ game characterization for the first-order indefinability of answer set programs. We then define two notions named the 0-1 property and unbounded cycles or paths under the answer set semantics, from which we develop two sufficient conditions that may be effectively used in proving a program's first-order indefinability.

Key words: Logic programming, answer set programming, computational aspects of knowledge representation

Introduction
Answer Set Programming (ASP) is an important programming paradigm for declarative problem solving. In recent years, it has demonstrated profound applications in many areas such as semantic web, robotic planning and bioinformatics. Recent work on ASP has extended the traditional ASP framework by allowing variables in program rules, while the semantics of such programs is defined via second-order logic (Ferraris, Lee, & Lifschitz 2010; Lin & Zhou 2010). Consequently, such extended ASP programs have significantly increased the expressive power compared to propositional answer set programs (Baral 2003).

Nevertheless, computing the extended ASP is difficult due to its inherited second-order logic semantics. One related issue is the first-order definability (indefinability) problem. A logic program with variables is first-order definable if the set of its (finite) answer sets can be captured by a first-order sentence, otherwise it is first-order indefinable. Results on first-order definability, both positive and negative, will have important impacts to current ASP research. First, results about first-order definability and indefinability provide a theoretic foundation to characterize the expressiveness of first-order ASP and hence to establish its close connections to classical first-order and second-order logics. Second, knowing whether a program is first-order definable or not will be also critical to apply suitable strategies in developing practical first-order ASP solvers. For instance, for first-order definable programs, we can directly ground their corresponding first-order sentences (e.g. (Wittocx, Marien, & Denecker 2008)) and then use an SAT solver to compute their models. Such SAT-based ASP solver can avoid computing programs' loop formulas and are believed to be more efficient generally\(^1\). Finally, as evident from previous research (Cosmadakis 1989; Ebbinghaus & Flum 1999), exploring first-order definability for a problem like this is a challenging task. To achieve certain results, especially negative results, very often new concepts and techniques have to be developed, which may also be useful for other related research.

In this paper, we focus on the negative results of first-order definability of answer set programs. In particular, by extending the traditional Ehrenfeucht-Frassï¿½ games approach in finite model theory, we provide an Ehrenfeucht-Frassï¿½ game characterization for the first-order indefinability of answer set programs. Based on this result, we further propose new notions named the 0-1 property and unbounded cycles and paths under answer set semantics, from which we develop two sufficient conditions that may be effectively used in proving a program's first-order indefinability.

Basic concepts and definitions
We consider a second-order language with equality but without function symbols. A vocabulary consists of a finite set of constant symbols and a finite nonempty set of relation symbols including equality =. We denote the sets of constant symbols of a vocabulary \(\tau\) by \(C(\tau)\) and relation symbols by \(P(\tau)\) respectively. Given a vocabulary, term, atom, and (first-order or second-order) formula are defined as usual. An atom is called an equality atom if it is an atom of the form \(t_1 = t_2\), where \(t_1\) and \(t_2\) are terms, and a proper atom otherwise.

A finite structure \(A\) of vocabulary \(\tau\) is a tuple \((A, c_1^A, \ldots, c_m^A, R_1^A, \ldots, R_n^A)\), where \(A\) is a finite set called the domain of \(A\), each \(c_i^A\) \((i = 1, \ldots, m)\) is an element in \(A\) which corresponds to a constant symbol \(c_i\) in \(C(\tau)\), and each \(R_i^A\) \((i = 1, \ldots, n)\) is a \(k\)-ary relation on \(A\) which corresponds to a \(k\)-ary relation symbol \(R_i\) in \(R(\tau)\). Sometimes,
we use $\text{Dom}(A)$ to denote the domain of structure $A$. In this paper we will focus on finite structures.

Given two vocabularies $\tau_1$ and $\tau_2$ where $C(\tau_2) \subseteq C(\tau_1)$ and $R(\tau_2) \subseteq R(\tau_1)$ (in this case, we also denote $\tau_2 \subseteq \tau_1$), and a finite structure $\mathcal{A}$ of $\tau_1$. We say that the restriction of $A$ on $\tau_2$, denoted by $A|_{\tau_2}$, is a structure of $\tau_2$ where it has the same domain of $A$, and for each constant $c$ and relation symbol $R$ in $\tau_2$, $c^A$ and $R^A$ are in $A|_{\tau_2}$. On the other hand, if we are given a structure $\mathcal{A}'$ of $\tau_2$, a structure $\mathcal{A}$ of $\tau_1$ is an expansion of $\mathcal{A}'$ to $\tau_1$, if $\mathcal{A}$ has the same domain of $\mathcal{A}'$ and retains all $c^{A'}$ and $R^{A'}$ for all constants $c$ and relation symbols $R$ in $\tau_2$.

Consider structure $\mathcal{A} = (A, c_1^A, \ldots, c_m^A, R_1^{A}, \ldots, R_k^A)$ and $S \subseteq A$ where $\{c_1^A, \ldots, c_m^A\} \subseteq S$. Structure $\mathcal{A} \upharpoonright S$ is called a substructure of $\mathcal{A}$ generated from $S$, if $\mathcal{A} \upharpoonright S = (S, c_1^A, \ldots, c_m^A, R_1^{A|S}, \ldots, R_k^{A|S})$, where for any tuple $\tau$ (see below for such notation) from $S, \tau \in R_i^{A|S}$ iff $\tau \in R_i^A, 1 \leq i \leq n$.

We usually write a tuple $(t_1, \ldots, t_n)$ as the form $\bar{t}$, where $\{t_1, \ldots, t_n\}$ is either a set of terms or a set of elements from $\text{Dom}(A)$. For two tuples $\bar{t} = (t_1, \ldots, t_m)$ and $\bar{s} = (s_1, \ldots, s_n)$, we may simply write $\bar{t} \subseteq \bar{s}$ if $n \leq m$ and $\{t_1, \ldots, t_m\} \subseteq \{s_1, \ldots, s_n\}$.

The quantifier rank $qr(\varphi)$ of a first-order formula $\varphi$ is the maximum number of nested quantifiers occurring in $\varphi$: $qr(\varphi) = 0$ if $\varphi$ is atomic, $qr(\varphi_1 \lor \varphi_2) = qr(\varphi_1 \land \varphi_2) = \text{max}(qr(\varphi_1), qr(\varphi_2)), qr(\neg \varphi) = qr(\varphi)$, and $qr(\exists x \varphi) = qr(\forall x \varphi) + 1$.

With a fixed vocabulary $\tau$, we consider two finite structures $A$ and $B$, and $m \in \mathbb{N}$. $A$ and $B$ are $m$-equivalent, denoted by $A \equiv_m B$, if for any first-order sentence $\varphi$ with $qr(\varphi) \leq m, A \models \varphi$ iff $B \models \varphi$. $A$ and $B$ are called isomorphic, denoted as $A \cong B$, if there is a one-to-one and onto mapping $h: \text{Dom}(A) \rightarrow \text{Dom}(B)$ such that for every constant $c \in \tau, h(c^A) = c^B$, and for every relation symbol $R \in \tau$ and every tuple $\pi$ from $\text{Dom}(A), h(\pi) \in R^B$.

If $\varphi$ is a first-order or second-first sentence, we use $\text{Mod}(\varphi)$ to denote the collection of all finite structures that satisfy $\varphi$. Let $D$ be a finite set. We use $\text{Mod}(\varphi)/D$ to denote the collection of all finite structures that satisfy $\varphi$ and whose domains are $D$.

First-order answer set programs

Syntax and semantics

A rule is of the form:

$$a \leftarrow b_1, \ldots, b_k, \text{not } c_1, \ldots, \text{not } c_l,$$

where $a$ is an atomic rule or the falsity $\bot$, and $b_1, \ldots, b_k, c_1, \ldots, c_l (k, l \geq 0)$ are atoms. Here $a$ is called the head of the rule and $\{b_1, \ldots, b_k, \text{not } c_1, \ldots, \text{not } c_l\}$ the body of this rule.

A (first-order) answer set program (or simply called program) $\Pi$ is a finite set of rules. Every relation symbol occurring in the head of some rules of $\Pi$ is called intentional predicate, and all other relation symbols in $\tau$ are extensional predicates. The extensional predicates and individual constants occurring in $\Pi$ form the extensional vocabulary of $\Pi$. We use notions $\tau(\Pi)$ to denote the vocabulary containing all of relation symbols and constants in $\Pi$, $\tau_{int}(\Pi)$ the vocabulary containing all intentional predicates in $\Pi$, and $\tau_{ext}(\Pi)$ the vocabulary containing all extensional predicates and constants in $\Pi$. We also use notions $\mathcal{P}(\Pi), \mathcal{P}_{int}(\Pi)$ and $\mathcal{P}_{ext}(\Pi)$ to denote the sets all predicates, intentional and extensional predicates in $\Pi$ respectively. A proper atom $P(\mathcal{T})$ is extensional (intentional) if $P$ is extensional (intentional).

Sometimes, we simply call a relation $R_A$ in a structure $\mathcal{A}$ an intentional (extensional) relation if $R_A$ is the interpretation of an intentional (extensional, resp.) predicate of the underlying program $\Pi$.

Now we present the semantics of first-order answer set programs. For each rule $r$ of form (1), we use $\bar{r}$ to denote the sentence

$$\forall \pi (\text{Body}_r \supset a),$$

where $\pi$ is the tuple of all variables occurring in $r$, and $\text{Body}_r$ the formula $b_1 \land \cdots \land b_k \land \neg c_1 \land \cdots \land \neg c_l$. By $\Pi$, we denote the sentence $\bigwedge_{r \in \mathcal{P}} \bar{r}$.

Let $\varphi$ be a formula, $\mathcal{P} = \{P_1, \ldots, P_k\}$ and $\mathcal{P}' = \{P'_1, \ldots, P'_k\}$ two sets of relation symbols where $P_i$ and $P'_i$ are of the same arity. By $\varphi[+P/P']$, we mean the formula that is obtained from $\varphi$ by replacing each relation symbol in $\varphi$ positively occurring in $\varphi$ by the corresponding relation symbol in $P'$. For instance, if $\varphi \equiv \forall x (P(x) \land \neg Q(x))$, then $\varphi[+P'/P'] \equiv \forall x (P'(x) \land \neg Q(x))$. Note that here we do not replace the negative occurrence of $Q$ in $\varphi$. Let $P$ and $Q$ be two predicate symbols or variables of the same arity. $P < Q$ stands for the formula $\forall \pi (P(\pi) \supset Q(\pi)) \land \neg \forall \pi (Q(\pi) \supset P(\pi))$. For the given $\mathcal{P} = \{P_1, \ldots, P_k\}$ and $\mathcal{P}' = \{P'_1, \ldots, P'_k\}$ where all $P_i$ and $P'_i$ have the same arity, we use $\mathcal{P} < \mathcal{P}'$ to denote formula $\bigwedge_{i=1}^n P_i < P'_i$.

Consider two vocabularies $\tau_1$ and $\tau_2$ where $\tau_2 \subseteq \tau_1$. Let $\psi$ be a first-order or second-order sentence on $\tau_1$ and $\mathcal{A}$ a finite structure of $\tau_2$. We specify $\text{Mod}(\psi)_\tau$ as follows:

$$\text{Mod}(\psi)_\tau = \{ \mathcal{A}' \mid \mathcal{A}' \in \text{Mod}(\psi) \text{ and } \mathcal{A}' \text{ is an expansion of } \mathcal{A} \text{ to } \tau_1 \}.$$

Definition 1 (Answer set program semantics) Given a first-order answer set program $\Pi$ and a structure $\mathcal{A}$ of $\tau_{ext}(\Pi)$. A structure $\mathcal{A}'$ of $\tau(\Pi)$ is an answer set of $\Pi$ based on $\mathcal{A}$ iff $\mathcal{A}' \in \text{Mod}(\psi)_{\tau_{int}(\Pi)}$, where $\psi = \Pi \land \neg \exists \mathcal{P}'(\mathcal{P}' < \mathcal{P}_{\tau_{int}(\Pi)} \land \Pi(\mathcal{P}_{\tau_{int}(\Pi)}[P]/\mathcal{P}))$. We also use $\exists(\Pi, \mathcal{A})$ to denote the collection of all answer sets of $\Pi$ based on $\mathcal{A}$. A structure $\mathcal{A}'$ of $\tau(\Pi)$ is an answer set of $\Pi$ if there is some structure $\mathcal{A}$ of $\tau_{ext}(\Pi)$ such that $\mathcal{A}' \in \exists(\Pi, \mathcal{A})$.

Example 1 We consider a problem $\Pi_T$ consisting of the following rules:

$$T(x,y) \leftarrow E(x,y), \text{not } E(x,x), \text{not } E(y,y),$$

$$T(x,y) \leftarrow T(x,z), T(z,y),$$

where $\tau_{ext}(\Pi_T) = \{E\}$ and $\tau_{int}(\Pi_T) = \{T\}$. Now given a structure of $\tau_{ext}(\Pi_T) = (A, E^A)$, where $A = \{a, b, c, d\}$ and $E^A = \{(a,a), (a,b), (b,c), (c,d)\}$, according to Definition 1, it can be shown that the unique answer set of $\Pi_T$ based on $A$ is $A' = (A, E^A, T^A)$, where $T^A' = \{T\}$. 

2
In Definition 1, minimization only applies on intentional predicates while extensional predicates are viewed as the initial input of the program. This is different from the previous first-order answer set semantics such as Ferraris et al and Lin and Zhou’s semantics (Ferraris, Lee, & Lifschitz 2010; Lin & Zhou 2010), where no separation between intentional and extensional predicates was made.

By separating intentional and extensional predicates in a program, the program itself may be viewed as a generic description of certain system or agent’s behaviours, while the extensional predicates just provide various instantiations of the system or agent’s initial inputs. Consequently, the class of programs that contain the same rules but with different extensional predicate inputs share many essential properties so that our study on these properties such as as first-order definability can be simplified.

Although such difference, it is actually not hard to show that under the context of finite structures, our semantics presented above coincides with both Ferraris et al and Lin and Zhou’s semantics. Details on this issue are referred to our full version of this paper (Chen, Zhang, & Zhou 2009).

First-order definability for answer set programs

Now we are ready to present a formal definition of first-order definability for an answer set program.

Definition 2 (First-order definability) A program $\Pi$ is called first-order definable iff there exists a first-order sentence $\psi$ on vocabulary $\tau(\Pi)$ such that for every structure $A$ of $\tau_{ext}(\Pi)$, $Mod(\psi)_{\tau(\Pi)}^A = \exists(\Pi, A)$. In this case, we say that $\psi$ defines $\Pi$.

Consider program $\Pi = \{ P(x) \leftarrow Q(x), \text{not } R(x) \}$. According to Definition 2, $\Pi$ can be defined by the sentence $\forall x \exists y \exists z (P(x) \equiv (Q(x) \land \neg R(x)))$.

Proposition 1 A program $\Pi$ is first-order definable iff there exists a first-order sentence $\psi$ such that $Mod(\psi) = \bigcup_{A \in \mathcal{S}(\tau_{ext}(\Pi))} \exists(\Pi, A)$, where $\mathcal{S}(\tau_{ext}(\Pi))$ is the class of all structures of $\tau_{ext}(\Pi)$.

Ehrenfeucht-Frassé games for first-order answer set programs

In this section we extend traditional Ehrenfeucht-Frassé game approach in finite model theory (Ebbinghaus & Flum 1999) to the context of answer set programs so that this approach may be used as a tool to prove the first-order indefinability for a given program.

Given two $\tau$-structures $A = (A, c^A_1, \ldots, c^A_n, R^A_1, \ldots, R^A_m)$ and $B = (B, c^B_1, \ldots, c^B_m, R^B_1, \ldots, R^B_n)$, and $\pi \in A^* \land B \in B^*$, an Ehrenfeucht-Frassé game, which is played on $(A, \pi)$ and $(B, \bar{\pi})$, is played by two players named spoiler and duplicator. Each round of the game spoiler starts by picking an element from the opposite universe. For $k \geq 0$, let $e_k$ (or $f_k$) be the element of $A$ (or $B$ resp.) at round $k$. By default, we denote $e_{k+1}$ (or $f_{k+1}$) to be constant $c_i$’s the interpretation in $A$ (or $B$ resp.) where $i = 1, \ldots, m$. We say that duplicator wins round $k$ ($k \geq 0$) iff the following conditions hold:

1. there is a bijective map $h: \vec{\pi} \rightarrow \vec{\bar{\pi}}$, where $h(\vec{\pi}) = \vec{\bar{\pi}}$;
2. for any tuple $\vec{\tau} \subseteq \vec{\pi}$, $\vec{\tau} \in R^A_1$ iff $h(\vec{\tau}) \in R^B_i$.

For a fixed $k \geq 0$, the Ehrenfeucht-Frassé game of length $k$ is played for $k$ rounds. We say that the duplicator wins the game if he has a strategy to win every round. As a special case, when $\vec{\pi} = \vec{\bar{\pi}} = \vec{0}$, we also say that the duplicator wins the Ehrenfeucht-Frassé game of length $k$ on $A$ and $B$. The following result is due to (Ebbinghaus & Flum 1999).

Theorem 1 (Ebbinghaus & Flum 1999) The duplicator wins the Ehrenfeucht-Frassé game of length $k$ played on $A$ and $B$, iff $A \equiv_k B$.

Now we present a theorem that can be used to gain a negative result of the first-order definability for a given logic program.

Theorem 2 Let $\Pi$ be a logic program. $\Pi$ is not first-order definable if and only if for every $k \geq 0$, there are two structures $A^k$ and $B^k$ of vocabulary $\tau(\Pi)$ such that:

1. $A^k \in \exists(\Pi, A^k|\tau_{ext}(\Pi))$, $B^k \in \exists(\Pi, B^k|\tau_{ext}(\Pi))$;
2. the duplicator wins the Ehrenfeucht-Frassé game of length $k$ on $A^k$ and $B^k$. 

Proof: ($\Rightarrow$) According to condition 2, since for any $k \geq 0$, the duplicator wins the Ehrenfeucht-Frassé game of length $k$ on $A^k$ and $B^k$, from Theorem 1, we have $A \equiv_k B$. Now we assume that $\Pi$ is first-order definable. Then there exists a first-order sentence $\psi_\Pi$ such that for each structure $A^*$ of $\tau_{ext}(\Pi)$, $Mod(\psi_\Pi)^{A^*} = \exists(\Pi, A^*)$. Without lose of generality, we assume $q(\psi_\Pi) \leq m$. By condition 1, this implies that $A^m \models \psi_\Pi$ and $B^m \models \neg \psi_\Pi$. This contradicts the fact $A^m \equiv_k B^m$. So $\Pi$ must not be first-order definable.

($\Leftarrow$) We suppose that condition 1 or 2 does not hold. Then by Theorem 1, we have the following statement:

Statement 1. For some $k \geq 0$ and all structures $A^k$ and $B^k$ of vocabulary $\tau(\Pi)$, $A^k \exists(\Pi, A^k|\tau_{ext}(\Pi))$ and $A^k \equiv_k B^k$ implies $B^k \exists(\Pi, B^k|\tau_{ext}(\Pi))$.

Since $\exists(\Pi, A^k|\tau_{ext}(\Pi)) \cup \exists(\Pi, B^k|\tau_{ext}(\Pi)) \subseteq \bigcup_{A^* \in C(\tau_{ext}(\Pi))} \exists(\Pi, A^*)$, where $C(\tau_{ext}(\Pi))$ is the class of all finite structures of $\tau_{ext}(\Pi)$, Statement 1 then implies

Statement 2. For some $k$ and all structures $A^k$ and $B^k$ of vocabulary $\tau(\Pi)$, $A^k \exists(\Pi, A^k|\tau_{ext}(\Pi))$ and $A^k \equiv_k B^k$ implies $B^k \exists(\Pi, A^k|\tau_{ext}(\Pi))$.

Then according to Theorem 2.2.12 in (Ebbinghaus & Flum 1999), we know that there exists a first-order sentence $\psi$ such that $Mod(\psi) = \bigcup_{A^* \in C(\tau_{ext}(\Pi))} \exists(\Pi, A^*)$. Finally, from Proposition 1, it is concluded that $\Pi$ is first-order definable.
The program of finding Hamiltonian cycles has been used as a benchmark to test various ASP solvers, e.g., (smo). As an application of Theorem 2, we will show that this program is not first-order definable.

**Theorem 3** The following finding Hamiltonian cycles program \( \Pi_{HC} \) is not first-order definable:

\[
HC(x, y) \leftarrow E(x, y), \text{not} \ OtherRoute(x, y), \\
OtherRoute(x, y) \leftarrow E(x, y), E(x, z), HC(x, z), y \neq z, \\
OtherRoute(x, y) \leftarrow E(x, y), E(z, y), HC(z, y), x \neq z, \\
Reached(y) \leftarrow E(x, y), HC(x, y), \\
Reached(x) \leftarrow \text{not} \ InitialVertex(x), \\
Reached(y) \leftarrow E(x, y), HC(x, y), InitialVertex(x), \\
\text{not} \ Reached(x).
\]

**Proof:** For each \( k \geq 0 \), we consider two structures \( A^k \) and \( B^k \) of \( \pi(\Pi) \), where

\[
\begin{align*}
\text{Dom}(A^k) &= A^k = \{0, 1, \ldots, 2m - 1\}, m \geq 2^{k+1}, \\
E^k &= \{(i, i+1) \mid 0 \leq i \leq (2m-1)\} \cup \{(2m-1, 0)\}, \\
\text{InitialVertex}^k &= \{0\}, HC^k = E^k, \\
\text{OtherRoute}^k &= \emptyset, \\
\text{Reached}^k &= \{0, 1, \ldots, 2m - 1\}, \\
\text{Dom}(B^k) &= \{0, 1, \ldots, 2m - 1\}, \\
E^k &= \{(i, i+1) \mid 0 \leq i < m\} \cup \{(m-1, 0)\} \cup \\
&\quad \{(j, j+1) \mid m \leq j < (2m-1)\} \cup \\
&\quad \{(2m-1, m)\}, \\
\text{InitialVertex}^k &= \{0\}, HC^k = E^k, \\
\text{OtherRoute}^k &= \emptyset, \\
\text{Reached}^k &= \text{Reached}^k = \{0, 1, \ldots, 2m - 1\}.
\end{align*}
\]

Note that if we only consider the extensional relations, \( A^k \) and \( B^k \) may be viewed as two different graphs with \( \text{Dom}(A^k) \) and \( \text{Dom}(B^k) \) being their vertices and \( E^k \) and \( E^k \) being their edges respectively. Furthermore, \( \text{Dom}(A^k), E^k \) is a single cycle of length \( 2m \), and \( \text{Dom}(B^k), E^k \) contains two separate cycles and each has a length \( m \).

From the interpretations of all intentional predicates in \( A^k \), it is easy to see that \( A^k \) is an answer set of \( \Pi_{HC} \). On the other hand, \( B^k \) is not an answer set of \( \Pi_{HC} \) because \( \text{Reached}^k = \{0, 1, \ldots, 2m - 1\} \), while it is observed that for each \( j \geq m \), \( j \) is not reachable under the given \( E^k \) and \( \text{InitialVertex}^k \). So we have \( A^k \in \mathcal{A}(\Pi, A^k)_{\text{ext}}(\Pi_{HC}) \) and \( B^k \notin \mathcal{A}(\Pi, B^k)_{\text{ext}}(\Pi_{HC}) \).

Now we consider the Ehrenfeucht-Fraïssé game of length \( k \) played on \( A^k \) and \( B^k \). Without loss of generality, we assume that the game starts with two special points played in each of the graph: \( a_{i1} = 0, a_{i0} = (2m-1) \) from \( A^k \), and their responses \( b_{i1} = 0, b_{i0} = (m-1) \) from \( B^k \) respectively. Intuitively, this means that the two endpoints of the cycle in \( A^k \) have responses of the two endpoints of one cycle in \( B^k \). Then during the game is played, we denote that a point \( a_i \) from \( A^k \) has its response \( b_i \) from \( B^k \), and vice versa. We also define the distance between two points in \( A^k \) or \( B^k \) to be the shortest path between them. Note that in \( B^k \), if one point is in one cycle component and the other is in another cycle component, the distance between these two points is infinity.

Next we prove that the duplicator can play the game in such a way that ensures the following conditions after each round \( i \):

**Condition 1.** If \( d(a_j, a_{i1}) \leq 2^{k-i} \), then \( d(b_j, b_i) = d(a_j, a_{i1}) \).

**Condition 2.** If \( d(a_{i1}, a_i) > 2^{k-i} \), then \( d(b_j, b_i) > 2^{k-i} \).

Now we prove these conditions by induction. The case of \( i = 0 \) immediately follows from our assumption \( d(a_{i1}, a_{i0}) = 2m \geq (2 \times 2^{k+1}) > 2^k \), and \( d(b_{i1}, b_i) = m \geq 2^{k+1} > 2^k \). Suppose \( i \) rounds have been played and the spoiler moves in round \( i+1 \). We consider different cases.

**Case 1.** The spoiler makes his move in \( A^k \).

**Case 1.1.** The spoiler plays close to two previous points with a distance at most \( 2^{k-(i+1)} \) such that no other points are placed between these two points. That is, \( b_{i1} \) falls into an interval \( a_{i1} \leq a_{i+1} < a_i \) such that no previously played point is in this interval, where \( d(a_j, a_{i1}) \leq 2^{k-(i+1)} \) and \( d(a_j, a_i) \leq 2^{k-i} \). That follows \( d(a_j, a_i) \leq 2^{k-i} \). Then according to the inductive assumption, \( d(b_j, b_{i1}) = d(a_j, a_{i1}) \leq 2^{k-i} \). So we can choose a \( b_{i1} \) such that \( d(b_{i1}, b_{i+1}) = d(a_{i1}, b_i) = d(a_{i1}, a_{i+1}) \). The condition is preserved.

**Case 1.2.** The spoiler plays at a distance greater than \( 2^{k-(i+1)} \) to all previous points. Since \( m \geq 2^{k+1} \), with fewer than \( k \) rounds been played, we can find a point in \( B^k \) whose distance to all previous points is greater than \( 2^{k-(i+1)} \) in the following way. Suppose \( a_{i1} \) falls into an interval say \( a_j < a_{i1} \leq a_i \), where no previous point is in this interval, \( d(a_{i1}, a_{i+1}) > 2^{k-(i+1)} \) and \( d(a_{i1}, a_i) > 2^{k-i} \). This follows \( d(a_{i1}, a_i) > 2^{k-i} \). Then from the induction assumption, we have \( d(a_j, a_{i1}) > 2^{k-i} \) as well. We show that the duplicator can always choose a point \( b_{i1} \) such that \( d(b_{i1}, b_{i+1}) > 2^{k-(i+1)} \) and \( d(b_{i1}, b_i) > 2^{k-i} \).

Then there are two cases: (1) both \( b_j \) and \( b_{i1} \) fall into the same cycle of \( B^k \). In this case, the duplicator places \( b_{i1} \) in the middle of interval \( [b_j, b_i] \). Hence it follows \( d(b_{i1}, b_{i+1}) > 2^{k-(i+1)} \) and \( d(b_{i1}, b_i) > 2^{k-i} \).

(2) \( b_j \) and \( b_i \) are in different cycles of \( B^k \). Then the duplicator places \( b_{i1} \) in one of the cycles \( B^k \). Clearly, for all points \( b_{i1} \) in the other cycle of \( B^k \), \( d(b_{i1}, b_{i+1}) = \infty > 2^{k-(i+1)} \). Now we show that the duplicator can choose a position in the cycle for \( b_{i1} \) such that for all points \( b_{i1} \) in the cycle, \( d(b_{i1}, b_i) > 2^{k-(i+1)} \) holds. Since there are less than \( i \) points in the cycle, there must exist two points \( b', b'' \) in the cycle such that no other point falls in the interval \( [b', b''] \) and \( d(b', b'') > 2^{k-i} \). Then the duplicator simply places \( b_{i1} \) in the middle of this interval. This follows that for all points \( b_{i1} \) in the cycle, \( d(b_{i1}, b_i) > 2^{k-(i+1)} \) as well.

**Case 1.3.** The spoiler plays at a distance greater than
$2^{k-(i+1)}$ to some point and at a distance at most $2^{k-(i+1)}$ to some other point. In this case, we can assume that $a_{i+1}$ falls into an interval between $a_i < a_{i+1} < a_i$, such that $d(a_j, a_{i+1}) \leq 2^{k-(i+1)}$ and $d(a_{i+1}, a_i) > 2^{k-(i+1)}$, and no other point falls into this interval.

Again, there are two cases. (1) Suppose $d(a_j, a_i) > 2^{k-i}$. From induction assumption, $d(b_j, b_i) > 2^{k-i}$. (a) if $b_j$ and $b_i$ are in the same cycle of $B^k$, in this case, the duplicator places $b_{i+1}$ in such a way that $d(b_j, b_{i+1}) = d(a_j, a_{i+1})$. Clearly, $d(b_{i+1}, b_i) = d(b_j, b_i) - d(b_j, b_{i+1}) > 2^{k-(i+1)}$. (b) if $b_j$ and $b_i$ are in different cycles of $B^k$, then the duplicator places $b_{i+1}$ in such a way that $d(b_j, b_{i+1}) = d(a_j, a_{i+1})$ where $b_j$ and $b_{i+1}$ are in the same cycle of $B^k$. So it also follows $d(b_{i+1}, b_i) > 2^{k-(i+1)}$.

(2) Suppose $d(a_j, a_i) \leq 2^{k-i}$. From the inductive assumption, we have that $d(b_j, b_i) = d(a_j, a_i) \leq 2^{k-i} < 2^k$. So both $b_j$ and $b_i$ are in the same cycle of $B^k$. Then the duplicator places $b_{i+1}$ such that $d(b_j, b_{i+1}) = d(a_j, a_{i+1})$. This also follows $d(b_{i+1}, b_i) = d(b_j, b_i) - d(b_j, b_{i+1}) = d(a_j, a_i) - d(a_j, a_{i+1}) = d(a_{i+1}, a_i) > 2^{k-(i+1)}$.

**Case 2.** The spoiler makes his move in $B^k$. Arguments for this case are similar to Case 1.

Now based on the above argument, we further show that the duplicator wins the game of length $k$. That is, we show $A^k \equiv_k B^k$. More specifically, we show that after $k$ rounds, for any $a_i, a_j$ from $A^k$ and the corresponding $b_i, b_j$ from $B^k$, the following statements hold:

1. $(a_i, a_j) \in E^{A^k}$ iff $(b_i, b_j) \in E^{B^k}$,
2. $a_i \in \text{InitialVertex}^{A^k}$ iff $b_i \in \text{InitialVertex}^{B^k}$,
3. $(a_i, a_j) \in \text{HC}^{A^k}$ iff $(b_i, b_j) \in \text{HC}^{B^k}$,
4. $(a_i, a_j) \in \text{OtherRoute}^{A^k}$ iff $(b_i, b_j) \in \text{OtherRoute}^{B^k}$, and
5. $a_i \in \text{Reached}^{A^k}$ iff $b_i \in \text{Reached}^{B^k}$.

According to the above result showed from Cases 1 and 2, and the construction of $A^k$ and $B^k$, it is easy to verify that (1)-(5) hold. We omit the detailed proof of this part due to a space limit (see (Chen, Zhang, & Zhou 2009)).

**Sufficient conditions for proving ASP first-order indefinability**

From the proof of Theorem 3, we observed that showing a program to be first-order definable is rather technical. In particular, during an Ehrenfeucht-Fraissé game playing, the winning strategy for the duplicator highly relies on the structures we pick up for the proof. In this sense, the approach demonstrated in the proof of Theorem 3 would be hardly applied as a general approach to show indefinability for other programs.

Preferably, we would like to develop some more general sufficient conditions that will ensure the duplicator to win and is easier to apply to a broad range of program cases. For this purpose, let us taking a closer look at the proof of Theorem 3. We observe that for every integer $k$, the program $\Pi_{HC}$ always has an answer set $A^k$ in which the extensional relations actually form a very large cycle. Then we are able to find another structure $B^k$ where its corresponding extensional relations form a graph with the same size of $A^k$’s but $B^k$ itself is not an answer set of $\Pi$, and we can also show $A^k \equiv_k B^k$.

This seems to indicate two important factors to effectively apply the Ehrenfeucht-Fraissé game technique: (1) both the given program’s intentional and extensional relations have to be considered during the game; and (2) the model structure of the extensional relations also significantly affect the duplicator’s winning strategy in the game. Based on these observations, in the following, we will develop some useful sufficient conditions for proving a program’s first-order indefinability which are easier to use in various situations.

**Programs with the 0-1 property**

Let $A = (A_1, c_1^A, \ldots, c_m^A, R_1^A, \ldots, R_n^A)$ be a structure. A relation $R_k^A$ in $A$ is called 0-relation if $R_k^A = \emptyset$, it is called 1-relation if $R_k^A = A_k$, were $h$ is the arity of $R_k^A$. In general, a relation $R_k^A$ in $A$ is called 0-1 relation if it is either a 0-relation or a 1-relation.

**Definition 3 (The 0-1 property)** Let $\Pi$ be a program. We say that $\Pi$ has the 0-1 property, if for each $k \geq 1$, $\Pi$ has an answer set $A$, where $|\text{Dom}(A)| \geq k$, such that all intentional relations in $A$ are 0-1 relations. In this case, we also call $\Pi$ a 0-1 answer set of $\Pi$ and $\Pi$ a 0-1 program.

**Example 2** We consider program $\Pi_{\text{RChecking}}$ which checks whether each vertex in a graph is reachable from the given initial vertex (vertices):

$$\text{Reachable}(x) \leftarrow \text{InitialVertex}(x),$$
$$\text{Reachable}(y) \leftarrow \text{Reachable}(x), E(x, y),$$
$$\neg \text{not Reachable}(x).$$

We can see that for each $k \geq 0$, there exists an answer set of $\Pi_{\text{RChecking}}$, such that the intentional predicate $\text{Reachable}$’s interpretation in the answer set represents a 1-relation. Hence, $\Pi_{\text{RChecking}}$ satisfies the 0-1 property.

0-1 programs represent an important feature which will ensure the duplicator’s winning strategy in an overall Ehrenfeucht-Fraissé game based on certain local information. In particular, if a program holds the 0-1 property, all we need to consider during an Ehrenfeucht-Fraissé game playing is the underlying program’s extensional relations in relevant structures/answer sets.

**Theorem 4** Let $\Pi$ be a program satisfying the 0-1 property. $\Pi$ is not first-order definable if for each $k \geq 0$, there exists a structure $B$ of $\tau(\Pi)$, such that $B$ is not an answer set of $\Pi$, and $A_{\tau_{\text{ext}}(\Pi)} \equiv_k B_{\tau_{\text{ext}}(\Pi)}$, where $A$ is a 0-1 answer set of $\Pi$, and for each $P \in \tau_{\text{int}}(\Pi)$, $P$ is interpreted either as a 0-relation or 1-relation in $B$ if it is in $A$ respectively.

By Theorem 4, if a program satisfies the 0-1 property, then when we prove the program’s first-order indefinability, we may only apply the Ehrenfeucht-Fraissé game over the restricted structures generated by extensional relations, e.g. $A_{\tau_{\text{ext}}(\Pi)}$ and $B_{\tau_{\text{ext}}(\Pi)}$, instead of the whole structures, which are sometimes simpler. The following proposition shows an application of Theorem 4.

**Proposition 2** $\Pi_{\text{RChecking}}$ is not first-order definable.
Proof: (Sketch) In Example 2, we showed that $\Pi_{RChecking}$ holds the 0-1 property. From Theorem 4, all we need to do is that for each $k$, we construct two structures $A^k$ and $B^k$ such that (1) $A^k$ is an answer set while $B^k$ is not; (2) $Reachable^A$ and $Reachable^B$ are the 1-relations in $A^k$ and $B^k$ respectively, and (3) show $A^k|_{rext(\Pi_{RChecking})} \equiv_k B^k|_{rext(\Pi_{RChecking})}$. Details are referred to (Chen, Zhang, & Zhou 2009). □

Programs with 0-1 unbounded cycles or paths

Theorem 4 can be effective in proving a 0-1 program $\Pi$’s first-order indefinability if the proof of $A^k|_{rext(\Pi)} \equiv_k B^k|_{rext(\Pi)}$ is already clear through the Ehrenfeucht-Fraisse game approach. Nevertheless, as has been revealed in finite model theory, directly using the Ehrenfeucht-Fraisse game approach is technically challenging for general cases (Arora & Fagin 1997). Furthermore, in our first-order indefinability proofs for programs $\Pi_{HC}$ and $\Pi_{RChecking}$, both programs happen to only have one binary extensional predicates, so that we can use graphs to represent the corresponding extensional relations and hence successfully apply Theorem 2. When a program involves more than one binary extensional predicates or extensional predicates with arity bigger than 2, our previous proof methods do not apply any more.

In this subsection, we will develop a more general sufficient condition by which we can effectively prove a broader range of 0-1 programs’ first-order indefinability.

To begin with, we first introduce a useful notion. Let $A$ be a structure, the Gaifman graph of $A$ (Ebbinghaus & Flum 1999) is an undirected graph $G(A) = (A, E^{A})$, where $Dom(A) = A$, and $E^{A}$ is defined as follows:

$$E^{A} = \{ (a, b) \mid a \neq b \text{ and there are a relation } R^{A} \text{ in } A \text{ and } \tau \text{ in } A \text{ such that } \tau \in R^{A} \text{ and } a \text{ and } b \text{ are among } \tau \},$$

We say that $A$ has a cycle (or an acyclic path\footnote{We will simply call it a path.}) if $G(A)$ contains a connected component that is a cycle (or a path, resp.).

Definition 4 (Programs with 0-1 unbounded cycles or paths) A program $\Pi$ has unbounded cycles (or paths) if for every $k > 0$, there is a $\Pi$’s answer set $A$ such that $G(A|_{rext(\Pi)})$ contains a cycle (path, resp.) with length greater than $k$. A program $\Pi$ has 0-1 unbounded cycles (or paths) if $\Pi$ is a 0-1 program, and for every $k > 0$, there is a $\Pi$’s 0-1 answer set $A$ such that $G(A|_{rext(\Pi)})$ contains a cycle (path, resp.) with length greater than $k$. In this case, $A$ is called a 0-1 cyclic (linear, resp.) answer set of $\Pi$.

Programs with 0-1 unbounded cycles or paths are of special interests in relation to first-order indefinability. The following theorem provides a new sufficient condition, which, as will be showed next, completely avoids the Ehrenfeucht-Fraisse game in proving a program’s first-order indefinability.

Theorem 5 A program $\Pi$ is not first-order definable if (1) $\Pi$ has 0-1 unbounded cycles or paths, and (2) for each $\Pi$’s 0-1 cyclic or linear answer set $A$, $G(A|_{rext(\Pi)})$ contains only one cycle or path, while all other connected components of $G(A|_{rext(\Pi)})$ are neither cycles nor paths.

The following Example 3 and Theorem 6 show an application of using Theorem 5 to prove a program’s first-order indefinability.

Example 3 Consider program $\Pi_{TCovered}$ as follows:

$$r_1: T(x, y) \leftarrow E(x, y), \neg E(x, x), \neg E(y, y),$$
$$r_2: T(x, y) \leftarrow T(x, z), T(z, y),$$
$$r_3: Covered(x) \leftarrow D(x, y),$$
$$r_4: Covered(y) \leftarrow D(x, y),$$
$$r_5: \leftarrow D(x, y), \neg E(x, y),$$
$$r_6: \leftarrow E(x, y), \neg Covered(x),$$
$$r_7: \leftarrow E(x, y), \neg Covered(y).$$

Intuitively, program $\Pi_{TCovered}$ computes the transitive closure based on the subgraph of $E$ without self-loops and verifies whether all vertices of the graph are covered by a given subset $D$ of edges of the graph. □

Theorem 6 Program $\Pi_{TCovered}$ in Example 3 is not first-order definable.

Proof: We prove this result by using Theorem 5. For any given $k > 0$, we consider structure $A^k$ as follows:

$$Dom(A^k) = \{0, 1, \cdots, m\}, \text{ where } m \geq k,$$
$$E^{A^k} = \{(i, i + 1) \mid 0 \leq i < m\} \cup \{(m, 0)\},$$
$$D^{A^k} = \{(j, j + 1) \mid 0 \leq j < m\},$$
$$T^{A^k} = \{(i, j) \mid 0 \leq i, j \leq m\},$$
$$Covered^{A^k} = \{0, \cdots, m\}.$$

It is easy to verify that $A^k$ is a 0-1 answer set of $\Pi_{TCovered}$. In fact, for both intentional predicates $T$ and $Covered$, they are interpreted as 1-relations in $A^k$. So $\Pi_{TCovered}$ satisfies the 0-1 property. Furthermore, $G(A^k|_{\{E, D\}})$ is a cycle with length $m$. Since there is no bound on $m$, $\Pi_{TCovered}$ has 0-1 unbounded cycles.

It is also observed that for an arbitrary 0-1 cyclic answer set $B$ of $\Pi_{TCovered}$, $G(B|_{\{E, D\}})$ must be of the same form of $G(A^k|_{\{E, D\}})$ as specified above. So conditions (1) and (2) in Theorem 5 also hold for $\Pi_{TCovered}$. This concludes that $\Pi_{TCovered}$ is not first-order definable. □

Example 4 We consider program $\Pi_Q$ from (Ajtai & Gurevich 1994) as follows:

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

Note that this program is almost the same as traditional datalog program of computing transitive closure of a graph except the extra last rule, and we already know that the property of transitive closure is not first-order definable (Ebbinghaus & Flum 1999). Taking a glance at this program, we may think that $\Pi_Q$ is not first-order definable as well. Let us examine whether this is the case.

It is easy to observe that $\Pi_Q$ satisfies the 0-1 property, because any structure $A$ of the following form is a 0-1 answer set of $\Pi_Q$: 

6
Furthermore, it is clear that $G(A|_{E_i})$ contains one unbounded cycle. Nevertheless, it is crucial to also note that not every $\Pi_Q$’s 0-1 answer set’s Gaifman graph is of the same form of $G(A|_{E_i})$. For each $m$, we can construct a structure $B$ as follows:

$$\text{Dom}(B) = \{0, 1, \cdots, 2m - 1\},$$
$$\text{E}^B = \{(i, i + 1) \mid 0 \leq i < m - 1\} \cup \{(m, 0)\} \cup \{(j, j + 1) \mid m \leq i < 2m - 1\} \cup \{(2m - 1, m)\},$$
$$Q^B = \{(i, j) \mid 0 \leq i, j \leq m\}.$$ 

Clearly, $B$ is also a 0-1 answer set of $\Pi_Q$, but $G(B|_{E_i})$ contains two disjoint cycles. Hence we cannot apply Theorem 5 to program $\Pi_Q$ to show its first-order indefinability. In fact, Ajtai and Gurevich showed that program $\Pi_Q$ is indeed first-order definable (Ajtai & Gurevich 1994). □

From Example 4, we can see that Theorem 5 may also be used as a method to reveal crucial clues why a program could be first-order definable.

Proof of Theorem 5

In order to prove Theorem 5, we will need a result in finite model theory (Fagin, Stockmeyer, & Vardi 1995). We first present necessary notions and concepts. Consider a structure $A = (\mathcal{A}, c_1^A, \cdots, c_m^A, R_1^A, \cdots, R_n^A)$. Let $G(A) = (\mathcal{A}, \text{Edge}^A)$ be the Gaifman graph of $A$ and $a$ an element of $A$. The neighborhood $N(a, d)$ of $a$ of radius $d$ is recursively defined as follows:

$$N(a, 1) = \{a, c_1^A, \cdots, c_m^A\},$$
$$N(a, d + 1) = N(a, d) \cup \{c \mid c \in A, \text{ and there is } b \in N(a, d) \text{ such that } (b, c) \in \text{Edge}^A\}.$$ 

Intuitively, $N(a, d)$ may be viewed as a sphere forming from elements of $A$ where each element in $N(a, d)$ has a distance from $a$ not more than $d$. Then we define that the $d$-type of $a$ is the isomorphism type of $A \upharpoonright N(a, d)$. That is, if $B$ is a structure of the same vocabulary of $A$ and $b$ is an element of $\text{Dom}(B)$, then $a$ and $b$ have the same $d$-type iff $A \upharpoonright N(a, d) \cong B \upharpoonright N(b, d)$ under an isomorphism mapping $a$ to $b$. $A$ and $B$ are $d$-equivalent if for every $d$-type $\iota$, they have the same number of points with $d$-type $\iota$.

Theorem 7 (Fagin, Stockmeyer, & Vardi 1995) For every $k > 0$ and for every $d \geq 3^k - 1$, if $A$ and $B$ are $d$-equivalent, then $A \equiv \equiv_k B$.

The following two lemmas will be crucial in our proof of Theorem 5, while the proof of Lemma 1 is also quite technical and tedious. Due to a space limit, detailed proofs of these two lemmas are referred to our full version of this paper (Chen, Zhang, & Zhou 2009).

Lemma 1 If $\Pi$ has unbounded cycles, then for each $k > 0$, there exist two structures $A$ and $B$ of $\tau(\Pi)$ such that (1) $A$ is an answer set of $\Pi$ and $G(A|_{\tau_{ext}(\Pi)})$ contains a path, (2) $G(B|_{\tau_{ext}(\Pi)})$ contains two disjoint cycles, and (3) for each $d > 0$, $A|_{\tau_{ext}(\Pi)}$ and $B|_{\tau_{ext}(\Pi)}$ are $d$-equivalent.

Lemma 2 If $\Pi$ has unbounded paths, then for each $k > 0$, there exist two structures $A$ and $B$ of $\tau(\Pi)$ such that (1) $A$ is an answer set of $\Pi$ and $G(A|_{\tau_{ext}(\Pi)})$ contains a path, (2) $G(B|_{\tau_{ext}(\Pi)})$ contains two disjoint paths, and (3) for each $d > 0$, $A|_{\tau_{ext}(\Pi)}$ and $B|_{\tau_{ext}(\Pi)}$ are $d$-equivalent.

Proof of Theorem 5:

Since $\Pi$ is a 0-1 program and has 0-1 unbounded cycles or paths, from Lemmas 1 and 2, we know that if for any $k > 0$, $\Pi$ has a 0-1 cyclic or linear answer set $A$, and we can always find a structure $B$ of $\tau(\Pi)$ such that $A|_{\tau_{ext}(\Pi)}$ and $B|_{\tau_{ext}(\Pi)}$ are $d$-equivalent for each $d > 0$, where whenever $G(A|_{\tau_{ext}(\Pi)})$ contains one cycle (path), $G(B|_{\tau_{ext}(\Pi)})$ contains two cycles (one cycle and one path, resp.). Since this result holds for any $k > 0$ and $d > 0$, from Theorem 7, by setting $d \geq 3^k - 1$, we then have $A|_{\tau_{ext}(\Pi)} \equiv_k B|_{\tau_{ext}(\Pi)}$.

Now by setting every intentional relation of $B$ to be either 0-relation or 1-relation accordingly as in $A$, it is concluded that $B$ cannot be an answer set of $\Pi$ due to condition (2) of Theorem 5. So by Theorem 4, $\Pi$ is not first-order definable. □

Conclusions

In this paper, we have extended traditional Ehrenfeucht-Fraissé game approach to the context of ASP semantics, from which we provided a precise game theoretic characterization on the first-order definability/indefinability for answer set programs (i.e. Theorem 2). We have further developed new methods for effectively prove a program’s first-order indefinability under certain situations (i.e. Theorems 4 and 5).

We believe that the results and techniques we proposed in this paper may be further generalized to prove more complex programs’ first-order indefinability. For instance, we may extend the 0-1 property and unbounded cycles (paths) concepts to arbitrary sets of predicates in a program and obtain more general sufficient conditions. This is being studied in our current work.

References


