A Game-theoretic Characterization on the First-order Indefinability of Answer Set Programs

Yan Zhang and Yi Zhou
School of Computing, Engineering and Mathematics
University of Western Sydney, Australia
Email: {yan,yzhou}@scem.uws.edu.au
Yin Chen
Department of Computer Science
South China Normal University, China
Email: ychen@scnu.edu.cn

Abstract

Under the general theory of stable models [17, 24], a first-order answer set program is semantically equivalent to a second-order sentence. Then a first-order answer set program is called first-order definable on finite structures if the set of finite answer sets of the program can be captured by a first-order sentence. First-order definability is a desirable property which provides alternative methods to compute the answer sets of the underlying programs [9]. However, not all first-order answer set programs may be reduced to a first-order sentence. In this paper, we study the problem of first-order indefinability of answer set programs. We provide an Ehrenfeucht-Fraïssé game-theoretic characterization for the first-order indefinability of answer set programs on finite structures. Based on this characterization, we propose two concepts named the 0-1 property and unbounded cycles or paths, respectively, from which we develop two sufficient conditions that may be effectively used in proving a program's first-order indefinability. We also show that such sufficient conditions may be further generalized to capture the first-order indefinability for a larger class of programs.

Keywords: Answer set programming, first-order definability, Knowledge representation, Non-classical logic, Non-monotonic reasoning

1 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 (see [6, 19, 20] for a general background and current technologies of ASP and its applications). Recent work on ASP has extended the traditional ASP framework by allowing variables in program rules, which we call first-order ASP, while the semantics of first-order ASP is defined via second-order logic [17, 26]. Under the new semantics, a first-order program may be completely separated from the input database, and hence provides a more succinct representation for a problem. Then with different input databases, the program may present different instances of the same problem. Consequently, unlike the traditional propositional ASP, grounding is no longer a necessary step in the problem solving using first-order ASP.

Comparing to the propositional case, First-order ASP overcomes the restriction of unique name assumption from Herbrand structures, and provides a unified form to represent aggregates [17]. Due to its flexibility in representation and direct connections to classic logics and other nonmonotonic formalisms, in recent years, there is an increasing interest in study first-order ASP from semantics, expressiveness and computations perspectives, e.g., [2, 3, 7, 31, 32].

Nevertheless, computing first-order answer set programs is difficult because of their inherited second-order logic semantics. One related issue is the first-order definability (indefinability) problem. An answer set program with variables is first-order definable on finite structures if the set of its finite answer sets can be captured by a first-order sentence, otherwise it is first-order indefinable on finite structures. Since most of our applications on ASP focus on finite structures, results about the first-order definability on finite structures, both positive and negative, will have important impacts for ASP solver development.

It is observed that the first-order ASP generalizes traditional Datalog, whilee the expressive power and complexity of datalog programs have been well studied [13]. As such, major first-order (in)definability results in Datalog, e.g., [1, 12], may be carried over to first-order ASP. But such results are generally not applicable in proving a program's first-order (in)definability under our context, not only because of the additional consideration of negation as failure under answer set semantics, but also due to the fact that these results are mainly semantic characterizations on datalog programs and queries, which do not help much in proving whether a given program is first-order (in)definable.

We have observed that knowing whether a logic program is first-order definable on finite structures or not can be beneficial for developing an effective first-order ASP solver, e.g., [30]. For instance, for first-order definable logic programs, we may directly ground their corresponding first-order sentences (e.g., [29]) and then use a SAT solver to compute their models, while for those first-order indefinable programs, we may use other approaches to compute their answer sets. Recently, Asuncion, Lin, Zhang Zhou have undertaken comprehensive investigations on this direction. They showed that under finite structures, by introducing auxiliary predicate symbols, every first-order normal logic program Π can be translated into a first-order sentences $OC(\Pi)$ called *ordered completion*, such that the collection of all answer sets of Π is exactly corresponding to the class of models of $OC(\Pi)$ [4]. Based on this result, they further implemented a new ASP solver named `asp2sat` in which a normal logic program Π is firstly translated to its ordered completion $OC(\Pi)$, then

optimization and simplification processes are applied on this first-order theory $OC(\Pi)$. Finally, by grounding the simplified $OC(\Pi)$, a SAT solver is called compute the models of $OC(\Pi)$. Their experimental results demonstrate clear advantages of this approach for computing answer sets for large problem instances. More details are referred to [4].

On the other hand, Chen, Lin, Zhang and Zhou studied the first-order definability problem for normal logic programs under finite structures. They characterized a class of programs named *loop-separable programs* and proved that all loop-separable normal logic programs are first-order definable under finite structures [9]. The class of loop-separable problems is also known as the largest syntactic class of first-order definable programs that we have discovered so far.

In this paper, we focus on the other direction of this problem: how to characterize a class of logic programs *not* first-order definable on finite structures. That is, given an answer set program, we consider whether there does not exist a first-order sentence whose collection of finite models is exactly the set of all finite answer sets of the program. This paper has made the following two major contributions[1]:

1. We provide an Ehrenfeucht-Fraïssé game-theoretic characterization for the first-order indefinability of answer set programs on finite structures. Ehrenfeucht-Fraïssé game-theoretic approach is a powerful tool for proving a property's first-order indefinability (inexpressiveness) in finite model theory. This technique has been also applied to other areas such as Datalog to study the problem of first-order indefinability. We demonstrate how the extended Ehrenfeucht-Fraïssé game-theoretic approach may be useful in proving an answer set program's first-order indefinability.

2. We also propose two concepts named the 0-1 property and unbounded cycles or paths under answer set semantics, respectively. Based on these concepts, we develop two sufficient conditions that may be effectively used in proving a program's first-order indefinability on finite structures under certain circumstances, without being involved in the detail of Ehrenfeucht-Fraïssé game. By observing some embedded limitations of the proposed concepts of the 0-1 property and unbounded cycles or paths, we further provide a generalization of the earlier sufficient conditions which is applicable to a larger class of answer set programs in proving first-order indefinability.

The rest of the paper is organized as follows. Section 2 provides the semantics of first-order answer set programs with extensional databases, and defines the concept of first-order definability on finite structures. Section 3 proves an extended Ehrenfeucht-Fraïssé game theorem for first-order answer set programs. Section 4 develops two sufficient conditions that can be effectively used in proving a program's first-order indefinability. A generalization of these two sufficient conditions is also provided. Section 5 discusses related work and concludes the paper with some remarks.

[1] An extended abstract version of this paper has been published in AAAI-2010 [10].

2 First-order answer set programs

Through out this paper, we will focus on normal logic programs with variables, which we may simply call *first-order asnwer set programs*. In this section, we provide necessary logical concepts and definitions, define the semantics of first-order answer set programs, and introduce the concept of first-order definability for answer set programs on finite structures.

2.1 Logic preliminaries

We consider a second-order language with equality but without function symbols. A *vocabulary* is a finite set that consists of *constant symbols* and *relation symbols* including equality $=$. We denote the sets of constant symbols of a vocabulary τ by $\mathcal{C}(\tau)$ and relation symbols by $\mathcal{R}(\tau)$ respectively. Given a vocabulary, *terms*, *atoms*, (first-order or second-order) *formulas* and *sentences* are defined as usual. An atom is called an *equality* atom if it is of the form $t_1 = t_2$, where t_1 and t_2 are terms, and a *proper* atom otherwise.

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

Given two vocabularies τ_1 and τ_2 where $\tau_2 \subseteq \tau_1$, and a finite structure $\mathcal{A}$ of τ_1, we say that the *restriction of $\mathcal{A}$ on τ_2*, denoted by $\mathcal{A}|\tau_2$, is a structure of τ_2 whose domain is the same as $\mathcal{A}$'s, and for each constant c and relation symbol R in τ_2, $c^{\mathcal{A}}$ and $R^{\mathcal{A}}$ are in $\mathcal{A}|\tau_2$. On the other hand, if we are given a structure $\mathcal{A}'$ of τ_2, a structure $\mathcal{A}$ of τ_1 is an *expansion of $\mathcal{A}'$ to τ_1*, if $\mathcal{A}$ has the same domain of $\mathcal{A}'$ and retains all $c^{\mathcal{A}'}$ and $R^{\mathcal{A}'}$ for all constants c and relation symbols R in τ_2.

Let $\mathcal{A}$ be a structure. We usually write a tuple $(t_1, \cdots, t_n)$ as the form $\overline{t}$, where $\{t_1, \cdots, t_n\}$ is either a set of terms or a set of elements from $\mathsf{Dom}(\mathcal{A})$. If $\overline{a} = (a_1, \cdots, a_s)$ is a tuple of elements from $\mathsf{Dom}(\mathcal{A})$, i.e. $a_i \in \mathsf{Dom}(\mathcal{A})$ $(1 \leq i \leq s)$, then we simply write $\overline{a} \in \mathsf{Dom}(\mathcal{A})^s$. For two tuples $\overline{t} = (t_1, \cdots, t_m)$ and $\overline{t'} = (t'_1, \cdots, t'_n)$ $(m \leq n)$, we may simply write $\overline{t} \subseteq \overline{t'}$ if for each t in $\overline{t}$, there is a t' in $\overline{t'}$ such that $t = t'$, and for all t_i and t_j in $\overline{t}$ where $i \leq j$ and their correspondences t'_h and t'_l in $\overline{t'}$ respectively, we have $h \leq l$.

Consider a structure $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$ and $S \subseteq A$ where $\{c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}\} \subseteq S$. Structure $\mathcal{A} \uparrow S$ is called a *substructure* of $\mathcal{A}$ *generated* from S, if $\mathcal{A} \uparrow S = (S, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}\uparrow S}, \cdots, R_n^{\mathcal{A}\uparrow S})$, where for any tuple $\overline{a}$ from S, $\overline{a} \in R_i^{\mathcal{A}\uparrow S}$ iff $\overline{a} \in R_i^{\mathcal{A}}$ $(1 \leq i \leq n)$.

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

Under a fixed vocabulary τ, we consider two finite structures $\mathcal{A}$ and $\mathcal{B}$, and $m \in \mathbb{N}$. $\mathcal{A}$ and $\mathcal{B}$ are *m-equivalent*, denoted by $\mathcal{A} \equiv_m \mathcal{B}$, if for any first-order sentence φ with $qr(\varphi) \leq m$, $\mathcal{A} \models \varphi$ iff $\mathcal{B} \models \varphi$. $\mathcal{A}$ and $\mathcal{B}$ are called *isomorphic*, denoted as $\mathcal{A} \cong \mathcal{B}$, if there

is a one-to-one and onto mapping h: $\mathsf{Dom}(\mathcal{A}) \rightarrow \mathsf{Dom}(\mathcal{B})$ such that for every constant $c \in \tau$, $h(c^{\mathcal{A}}) = c^{\mathcal{B}}$, and for every relation symbol $R \in \tau$ and every tuple $\overline{a}$ from $\mathsf{Dom}(\mathcal{A})$, $\overline{a} \in R^{\mathcal{A}}$ iff $h(\overline{a}) \in R^{\mathcal{B}}$.

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

2.2 First-order answer set programs with extensional databases

Now we introduce the syntax and semantics of first-order answer set programs, and present necessary definitions and notions relevant to our study in this paper.

A *rule* is of the form:

$$a \leftarrow b_1, \cdots, b_k, \mathsf{not}\ c_1, \cdots, \mathsf{not}\ c_l, \tag{1}$$

where a is a proper atom or the falsity $\bot$ (i.e. empty head), and $b_1, \cdots, b_k, c_1, \cdots, c_l$ ($k, l \geq 0$) are atoms. Here a is called the *head*, $\{b_1, \cdots, b_k\}$ the *positive body* and $\{\mathsf{not}\ c_1, \cdots, \mathsf{not}\ c_l\}$ the *negative body* of the rule respectively.

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

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

Now we present the semantics of first-order answer set programs, which is a simplified version of the general stable model semantics [17]. For each rule r of form (1), we use $\widehat{r}$ to denote the sentence

$$\forall \overline{x}(\widehat{Body_r} \supset a),$$

where $\overline{x}$ is the tuple of all variables occurring in r, and $\widehat{Body_r}$ the formula $b_1 \wedge \cdots \wedge b_k \wedge \neg c_1 \wedge \cdots \wedge \neg c_l$. Given a rule r, by $\widehat{\Pi}$, we denote the sentence $\wedge_{r \in \Pi} \widehat{r}$.

Let $\mathcal{P} = \{P_1, \cdots, P_k\}$ and $\mathcal{P}' = \{P'_1, \cdots, P'_k\}$ be two sets of relation symbols where P_i and P'_i are of the same arity. By $\widehat{r}[+\mathcal{P}/\mathcal{P}']$, we mean the formula that is obtained from $\widehat{r}$ by replacing each relation symbol in $\mathcal{P}$ occurring in the head and positive body of r by the corresponding relation symbol in $\mathcal{P}'$. For instance, if r is a rule $R(x) \leftarrow P(x), \mathtt{not}\ Q(x)$, then $\widehat{r}[+\{Q, R\}/\{Q', R'\}] \equiv \forall x((P(x) \wedge \neg Q(x) \supset R'(x))$. We define $\widehat{\Pi}[+\mathcal{P}/\mathcal{P}'] = \wedge_{r \in \Pi} \widehat{r}[+\mathcal{P}/\mathcal{P}']$. Let P and Q be two predicate symbols or variables of the same arity. $P \leq Q$ stands for the formula $\forall \overline{x}(P(\overline{x}) \supset Q(\overline{x}))$. For a given $\mathcal{P} = \{P_1, \cdots, P_k\}$ and

$\mathcal{P}' = \{P'_1, \cdots, P'_k\}$ where all P_i and P'_i have the same arity, $\mathcal{P} \leq \mathcal{P}'$ stands for formula $\bigwedge_{i=1}^{k} P_i \leq P'_i$, and $\mathcal{P} < \mathcal{P}'$ stands for formula $\mathcal{P} \leq \mathcal{P}' \wedge \neg(\mathcal{P}' \leq \mathcal{P})$.

Consider two vocabularies τ_1 and τ_2 where $\tau_2 \subseteq \tau_1$. Let ψ be a first-order or second-order sentence on τ_1 and $\mathcal{A}$ a finite structure of τ_2. We specify $\mathsf{Mod}(\psi)^{\mathcal{A}}_{\tau_1}$ as follows:

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

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

From Definition 1, we can see that the semantics of first-order answer set programs is defined by second-order logic. The semantics shares some similarity of the semantics of circumscription but has a different minimization strategy. In Definition 1, minimization applies on intentional predicates while extensional predicates are viewed as the initial input of the program. In generally, a structure of $\tau(\Pi)$ is an answer set of Π if it is a model of sentence $\widehat{\Pi}$ which minizes the interpretations on all intentional predicates in formula $\widehat{\Pi}[+\mathcal{P}_{int}(\Pi)/\mathcal{P}^*])$.

Also note that Definition 1 is a simplified version of the general stable model semantics, where arbitrary first-order sentences are allowed in a program and any set of predicates in the program may also be specified as intentional [17]. In [9], we have proved that Definition 1 is equivalent to Ferraris et al.'s original first-order answer set program semantics definition [17], when we restrict to normal logic programs and finite structures.

Example 1 We consider a program Π_T consisting of the following rules:

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

where $\tau_{ext}(\Pi_T) = \{E\}$ and $\tau_{int}(\Pi_T) = \{T\}$. Now given a structure of $\mathcal{A} = (A, E^{\mathcal{A}})$ of $\tau_{ext}(\Pi_T)$, where $A = \{a, b, c, d\}$ and $E^{\mathcal{A}} = \{(a, a), (a, b), (b, c), (c, d)\}$, according to Definition 1, it can be shown that the unique answer set of Π_T based on $\mathcal{A}$ is $\mathcal{A}' = (A, E^{\mathcal{A}}, T^{\mathcal{A}'})$, where $T^{\mathcal{A}'} = \{(b, c), (c, d), (b, d)\}$. If we view E as a graph, then T computed by program Π_T is the transitive closure of the induced subgraph of E on the set of vertices that do not have self-loops. □

2.3 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 Π is called* first-order definable *iff there exists a first-order sentence ψ on vocabulary $\tau(\Pi)$ such that for every structure $\mathcal{A}$ of $\tau_{ext}(\Pi)$, $\mathsf{Mod}(\psi)^{\mathcal{A}}_{\tau(\Pi)} = \Im(\Pi, \mathcal{A})$. In this case, we say that ψ* defines *Π.*

Note that in Definition 2, the sentence ψ is independent from any particular extensional input $\mathcal{A}$. Consider program $\Pi = \{P(x) \leftarrow Q(x), \mathsf{not}\ R(x)\}$. According to Definition 2, Π can be defined by the sentence $\forall x(P(x) \equiv (Q(x) \wedge \neg R(x)))$.

Given a class of finite structures $\mathcal{S}$, we say that $\mathcal{S}$ is *first-order definable* if there exists a first-order sentence ψ such that $\mathsf{Mod}(\psi) = \mathcal{S}$ [14]. The following proposition shows an immediate connection between our program first-order definability and a class of structures' first-order definability.

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

Furthermore, the following proposition presents an interesting relationship between the program first-order definability and the first-order definability of the class of all answer sets based on specific extensional databases.

Proposition 2 *If a program Π is first-order definable, then for all finite structure $\mathcal{A}$ of $\tau(\Pi)$, there exists a first-order sentence $\psi^{\mathcal{A}}$ such that $\mathsf{Mod}(\psi^{\mathcal{A}})|\mathsf{Dom}(\mathcal{A}) = \Im(\Pi, \mathcal{A}|\tau_{ext}(\Pi))$*[2]*.*

Proof: Consider an arbitrary finite structure $\mathcal{A}$ of $\tau(\Pi)$. Suppose $\mathsf{Dom}(\mathcal{A}) = A = \{a_1, \cdots, a_n\}$. We specify

$$\Theta = \{\phi \mid \phi \text{ is of the form } R(\overline{x}), x = y \text{ or } x = c \text{ where variables are among } v_1, \cdots, v_n, \text{ and } R \in \tau_{ext}(\Pi)\},$$

and a first-order sentence

$$\varphi_{\mathcal{A}|\tau_{ext}(\Pi)} = \exists v_1 \cdots v_n(\bigwedge\{\phi \mid \phi \in \Theta, \mathcal{A}|\tau_{ext}(\Pi) \models \phi[\overline{a}]\} \wedge \bigwedge\{\neg\phi \mid \phi \in \Theta, \mathcal{A}|\tau_{ext}(\Pi) \models \neg\phi[\overline{a}]\}) \wedge \forall v_{n+1}(v_{n+1} = v_1 \vee \cdots \vee v_{n+1} = v_n)\}.$$

Then we have:

$$\mathsf{Mod}(\varphi_{\mathcal{A}|\tau_{ext}(\Pi)})|\mathsf{Dom}(\mathcal{A}) = \{\mathcal{B} \mid \mathcal{B} \text{ is an expansion of } \mathcal{A}|\tau_{ext}(\Pi) \text{ to } \tau(\Pi)\},$$

that is, the sentence $\varphi_{\mathcal{A}|\tau_{ext}(\Pi)}$ uniquely characterizes the substructure $\mathcal{A}|\tau_{ext}(\Pi)$ in terms of structure expansions. According to the semantics of $\varphi_{\mathcal{A}|\tau_{ext}(\Pi)}$, we define $\psi^{\mathcal{A}} \equiv \Psi \wedge \varphi_{\mathcal{A}|\tau_{ext}(\Pi)}$, where the first-order sentence Ψ defines Π under Definition 2. Then it is easy to see that such $\psi^{\mathcal{A}}$ satisfies the required condition as stated in Proposition 2. □

[2]Note the difference from $\mathsf{Mod}(\phi)^{\mathcal{A}}_{\tau}$ - which is the class of structures that satisfy ϕ and are expansions of $\mathcal{A}$ to τ.

Proposition 2 simply says that if a program Π is first-order definable, then for each given extensional database as the input data for Π, the calss of all answer sets of Π under this input is also first-order definable. But the converse of this result does not hold. This also means that the program first-order definability is a stronger notion than traditional first-order definability with respect to a class of finite structures.

3 Ehrenfeucht-Fraïssé games for first-order answer set programs

In this section we extend the traditional Ehrenfeucht-Fraïssé game-theoretic approach in finite model theory [14] 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 τ-structures $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$ and $\mathcal{B} = (B, c_1^{\mathcal{B}}, \cdots, c_m^{\mathcal{B}}, R_1^{\mathcal{B}}, \cdots, R_n^{\mathcal{B}})$, and $\overline{a} \in A^s$ and $\overline{b} \in B^s$, an *Ehrenfeucht-Fraïssé game*, which is played on $(\mathcal{A}, \overline{a})$ and $(\mathcal{B}, \overline{b})$, is played by two players named *spoiler* and *duplicator*. Each round of the game spoiler starts by picking an element from either A or B, and duplicator responds by picking an element from the opposite domain. 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+i} (or f_{k+i}) to be constant c_i's interpretation in $\mathcal{A}$ (or $\mathcal{B}$ resp.) where $i = 1, \cdots, m$. We say that duplicator *wins* round k ($k \geq 0$) iff the following conditions hold:

1. there is a bijective map h: $\overline{ae} \mapsto \overline{bf}$, where $h(\overline{a}) = \overline{b}$, $h(\overline{e}) = \overline{f}$, $\overline{e} = (e_1, \cdots, e_k, e_{k+1}, \cdots, e_{k+m})$ and $\overline{f} = (f_1, \cdots, f_k, f_{k+1}, \cdots, f_{k+m})$;
2. for any tuple $\overline{t} \subseteq \overline{ae}$, $\overline{t} \in R_i^{\mathcal{A}}$ iff $h(\overline{t}) \in R_i^{\mathcal{B}}$.

For a fixed $k \geq 0$, the Ehrenfeucht-Fraïssé 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 $|\overline{a}| = |\overline{b}| = 0$, we also say that the duplicator *wins* the Ehrenfeucht-Fraïssé game of length k on $\mathcal{A}$ and $\mathcal{B}$.

Theorem 1 *[14] The duplicator wins the Ehrenfeucht-Fraïssé game of length k played on $\mathcal{A}$ and $\mathcal{B}$, iff $\mathcal{A} \equiv_k \mathcal{B}$.*

Then we can prove the following theorem to characterize the first-order definability for a given program.

Theorem 2 *Let Π be a program. Π is not first-order definable if and only if for every $k \geq 0$, there are two structures $\mathcal{A}^k$ and $\mathcal{B}^k$ of vocabulary $\tau(\Pi)$ such that*[3]*:*

[3]It is important to note that this theorem is different from the general form of Ehrenfeucht-Fraïssé game theorem [14], where it is required that $\Im(\Pi, \mathcal{A}^k|\tau_{ext}(\Pi))$ and $\Im(\Pi, \mathcal{B}^k|\tau_{ext}(\Pi))$ must be the same class of structures. This is not the case here.

1. $\mathcal{A}^k \in \Im(\Pi, \mathcal{A}^k|\tau_{ext}(\Pi))$, $\mathcal{B}^k \notin \Im(\Pi, \mathcal{B}^k|\tau_{ext}(\Pi))$*; and*

2. *the duplicator wins the Ehrenfeucht-Fraïssé game of length* k *on* $\mathcal{A}^k$ *and* $\mathcal{B}^k$.

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

($\Rightarrow$) 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 $\mathcal{A}^k$ and $\mathcal{B}^k$ of vocabulary $\tau(\Pi)$, $\mathcal{A}^k \in \Im(\Pi, \mathcal{A}^k|\tau_{ext}(\Pi))$ and $\mathcal{A}^k \equiv_k \mathcal{B}^k$ implies $\mathcal{B}^k \in \Im(\Pi, \mathcal{B}^k|\tau_{ext}(\Pi))$.

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

> Statement 2. For some k and all structures $\mathcal{A}^k$ and $\mathcal{B}^k$ of vocabulary $\tau(\Pi)$, $\mathcal{A}^k \in \bigcup_{\mathcal{A}' \in \mathcal{S}(\tau_{ext}(\Pi))} \Im(\Pi, \mathcal{A}')$ and $\mathcal{A}^k \equiv_k \mathcal{B}^k$ implies $\mathcal{B}^k \in \bigcup_{\mathcal{A}' \in \mathcal{S}(\tau_{ext}(\Pi))} \Im(\Pi, \mathcal{A}')$.

Then according to Theorem 2.2.12 in [14], we know that there exists a first-order sentence ψ such that $\mathsf{Mod}(\psi) = \bigcup_{\mathcal{A}' \in \mathcal{S}(\tau_{ext}(\Pi))} \Im(\Pi, \mathcal{A}')$. Finally, from Proposition 1, it is concluded that Π is first-order definable. This completes our proof. □

The program of finding Hamiltonian cycles has been used as a benchmark to test various ASP solvers. As an application of Theorem 2, we will show that this program is not first-order definable.

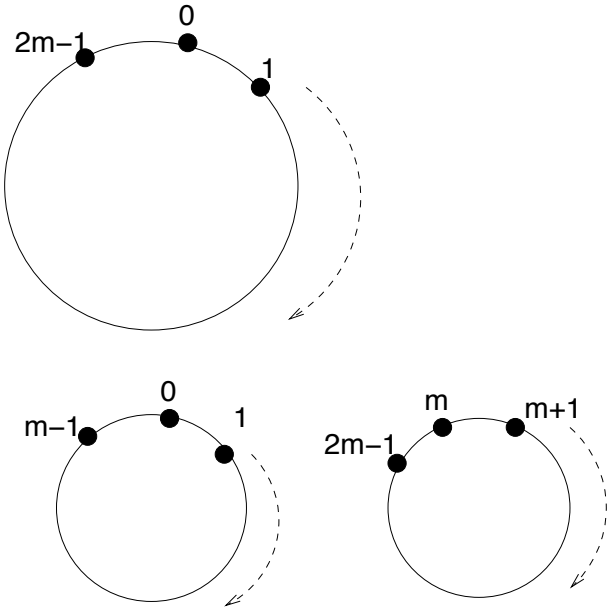
Proposition 3 *The following finding Hamiltonian cycles program* Π_{HC} *is not first-order definable:*

$$
\begin{aligned}
&HC(x,y) \leftarrow E(x,y), \texttt{not}\ OtherRoute(x,y),\\
&OtherRoute(x,y) \leftarrow E(x,y), E(x,z), HC(x,z), y \neq z,\\
&OtherRoute(x,y) \leftarrow E(x,y), E(z,y), HC(z,y), x \neq z,\\
&Reached(y) \leftarrow E(x,y), HC(x,y), Reached(x), \texttt{not}\ InitialVertex(x),\\
&Reached(y) \leftarrow E(x,y), HC(x,y), InitialVertex(x),\\
&\leftarrow \texttt{not}\ Reached(x).
\end{aligned}
$$

Proof: For each $k \geq 0$, we consider two structures $\mathcal{A}^k$ and $\mathcal{B}^k$ of $\tau(\Pi)$, where

$\mathsf{Dom}(\mathcal{A}^k) = A^k = \{0, 1 \cdots, 2m-1\}, m \geq 2^{k+1}$,
$E^{\mathcal{A}^k} = \{(i, i+1) \mid 0 \leq i < (2m-1)\} \cup \{(2m-1, 0)\}$,
$InitialVertex^{\mathcal{A}^k} = \{0\}$,
$HC^{\mathcal{A}^k} = E^{\mathcal{A}^k}$,
$OtherRoute^{\mathcal{A}^k} = \emptyset$,
$Reached^{\mathcal{A}^k} = \{0, 1, \cdots, 2m-1\}$,
$\mathsf{Dom}(\mathcal{B}^k) = \{0, 1, \cdots, 2m-1\}$,
$E^{\mathcal{B}^k} = \{(i, i+1) \mid 0 \leq i < (m-1)\} \cup \{(m-1, 0)\} \cup$
$\{(j, j+1) \mid m \leq j < (2m-1)\} \cup \{(2m-1, m)\}$,
$InitialVertex^{\mathcal{B}^k} = \{0\}$,
$HC^{\mathcal{B}^k} = E^{\mathcal{B}^k}$,
$OtherRoute^{\mathcal{B}^k} = \emptyset$,
$Reached^{\mathcal{B}^k} = \{0, 1, \cdots, 2m-1\}$.

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



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

Then Π_{HC}'s first-order indefinability follows by showing that for all $k \geq 0$, the duplicator wins the Ehrenfeucht-Fraïssé game on $\mathcal{A}^k$ and $\mathcal{B}^k$.

Now we consider the Ehrenfeucht-Fraïssé game of length k played on $\mathcal{A}^k$ and $\mathcal{B}^k$. Without loss of generality, we assume that the game starts with two special points played in each of the graph: $a_{-1} = 0, a_0 = (2m-1)$ from $\mathcal{A}^k$, and their responses $b_{-1} = 0, b_0 = (m-1)$ from $\mathcal{B}^k$ respectively. Intuitively, this means that the two endpoints of the cycle in $\mathcal{A}^k$ have responses of the two endpoints of one cycle in $\mathcal{B}^k$. Then during the game is played, we denote that a point a_i from $\mathcal{A}^k$ has its response b_i from $\mathcal{B}^k$, and *vice versa*. We also define the *distance* between two points in $\mathcal{A}^k$ or $\mathcal{B}^k$ to be the shortest path between them. Note that in $\mathcal{B}^k$, if one point is in one cycle component and the other is in another cycle component, the distance between these two points is infinity.

In order to prove that Π_{HC} is not first-order definable, according to Theorem 2, we only need to show that the duplicator has a winning strategy.

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 $d(a_j, a_l) \leq 2^{k-i}$, then $d(b_j, b_l) = d(a_j, a_l)$,
Condition 2. If $d(a_j, a_l) > 2^{k-i}$, then $d(b_j, b_l) > 2^{k-i}$.

Now we prove these conditions by induction. The case of $i = 0$ immediately follows from our assumption $d(a_{-1}, a_0) = 2m \geq (2 \times 2^{k+1}) > 2^k$, and $d(b_{-1}, b_0) = m \geq 2^{k+1} > 2^k$. 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 $\mathcal{A}^k$.

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, a_{i+1} falls into an interval $a_j < a_{i+1} < a_l$ such that no previously played point is in this interval, where $d(a_j, a_{i+1}) \leq 2^{k-(i+1)}$ and $d(a_{i+1}, a_l) \leq 2^{k-(i+1)}$. That follows $d(a_j, a_l) \leq 2^{k-i}$. Then according to the inductive assumption, $d(b_j, b_l) = d(a_j, a_l) \leq 2^{k-i}$. So we can choose a b_{i+1} such that $d(b_j, b_{i+1}) = d(a_j, a_{i+1})$, $d(b_{i+1}, b_l) = d(a_{i+1}, a_l)$. 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 $\mathcal{B}^k$ whose distance to all previous points is greater than $2^{k-(i+1)}$ in the following way. Suppose a_{i+1} falls into an interval, say $a_j < a_{i+1} < a_l$, where no previous point is in this interval, $d(a_j, a_{i+1}) > 2^{k-(i+1)}$ and $d(a_{i+1}, a_l) > 2^{k-(i+1)}$. This follows $d(a_j, a_l) > 2^{k-i}$. Then from the induction assumption, we have $d(b_j, b_l) > 2^{k-i}$ as well. We show that the duplicator can always choose a point b_{i+1} such that $d(b_j, b_{i+1}) > 2^{k-(i+1)}$ and $d(b_{i+1}, b_l) > 2^{k-(i+1)}$.

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

(2) b_j and b_l are in different cycles of $\mathcal{B}^k$. Then the duplicator places b_{i+1} in one of the cycles $\mathcal{B}^k$. Clearly, for all points b_j in the other cycle of $\mathcal{B}^k$, $d(b_j, b_{i+1}) = \infty > 2^{k-(i+1)}$. Now we show that the duplicator can choose a position in the cycle for b_{i+1} such that for

all points b_l in the cycle, $d(b_{i+1}, b_l) > 2^{k-(i+1)}$ 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_{i+1} in the middle of this interval. This follows that for all points b_l in the cycle, $d(b_l, b_{i+1}) > 2^{k-(i+1)}$ 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 $a_j < a_{i+1} < a_l$, such that $d(a_j, a_{i+1}) \leq 2^{k-(i+1)}$ and $d(a_{i+1}, a_l) > 2^{k-(i+1)}$, and no other point falls into this interval.

Again, there are two cases. (1) Suppose $d(a_j, a_l) > 2^{k-i}$. From induction assumption, $d(b_j, b_l) > 2^{k-i}$. (a) if b_j and b_l are in the same cycle of $\mathcal{B}^k$, 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, $d(b_{i+1}, b_l) = d(b_j, b_l) - d(b_j, b_{i+1}) > 2^{k-(i+1)}$. (b) if b_j and b_l are in different cycles of $\mathcal{B}^k$, 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 $\mathcal{B}^k$. So it also follows $d(b_{i+1}, b_l) > 2^{k-(i+1)}$.

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

Case 2. The spoiler makes his move in $\mathcal{B}^k$. Arguments are similar to Case 1.

Based on the above argument, we further show that for each k, the duplicator wins the game of length k. From Theorem 1, that is, we need to prove $\mathcal{A}^k \equiv_k \mathcal{B}^k$. More specifically, we show that after k rounds, for any a_i, a_j from $\mathcal{A}^k$ and the corresponding b_i, b_j from $\mathcal{B}^k$, the following statements hold:

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

For (1), suppose that after all k rounds have been played, there is an edge between a_i and a_j, that is, $d(a_i, a_j) = 1$. This implies $(a_i, a_j) \in E^{\mathcal{A}^k}$. Then from the above Condition 1, we have $d(b_i, b_j) = 1$. So $(b_i, b_j) \in E^{\mathcal{B}^k}$. Conversely, suppose that there is an edge between b_i, b_j, i.e. $(b_i, b_j) \in E^{\mathcal{B}^k}$. If there is no edge between a_i, a_j, we have $d(a_i, a_j) > 1$. Then from the above condition, it follows $d(b_i, b_j) > 1$. This contradicts with our assumption that there is an edge between b_i and b_j.

For (2), it is clear that $InitialVertex^{\mathcal{A}^k} = InitialVertex^{\mathcal{B}^k} = \{0\}$, and in the game, we have assigned $a_{-1} = 0$ and $b_{-1} = 0$. So (2) holds. Since $HC^{\mathcal{A}^k} = E^{\mathcal{A}^k}$ and $HC^{\mathcal{B}^k} = E^{\mathcal{B}^k}$, and we already know (1) holds, so (3) holds as well. For (4), since $OtherRoute^{\mathcal{A}^k} = OtherRoute^{\mathcal{B}^k} = \emptyset$, (4) holds. Finally, we have $Reached^{\mathcal{A}^k} = Reached^{\mathcal{B}^k} = \mathsf{Dom}(\mathcal{A}^k) = \mathsf{Dom}(\mathcal{B}^k) = \{0, 1, \cdots, 2m-1\}$. So (5) holds. This completes our proof. □

4 Sufficient conditions for first-order indefinability

From the proof of Proposition 3, it is observed that showing a program to be first-order indefinable is rather technical. In particular, during an Ehrenfeucht-Fraïssé 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 Proposition 3 would be hardly applied as a general approach to show indefinability for other programs.

On the other hand, existing results in finite model theory regarding the sufficient conditions to ensure winning strategies in Ehrenfeucht-Fraïssé games, for instance, those results developed in [5], are just too general to apply under our ASP setting. Preferably, we would like to develop some stronger 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 take a closer look at the proof of Proposition 3. We observe that for every integer k, the program Π_{HC} always has an answer set $\mathcal{A}^k$ in which the extensional relations actually form a very large cycle. Then we are able to find another structure $\mathcal{B}^k$ where its corresponding extensional relations form a graph with the same size of $\mathcal{A}^k$'s but $\mathcal{B}^k$ itself is not an answer set of Π, and we can also show $\mathcal{A}^k \equiv_k \mathcal{B}^k$.

We observe that there seem to have two important factors to effectively apply the Ehrenfeucht-Fraïssé game technique: (1) both the given program's intentional and extensional relations have to be considered during the game; and (2) the embedded structural form (e.g. a cycle) of extensional relations also significantly affects the duplicator's winning strategy in the game. Based on these observations, in the following, we will develop useful sufficient conditions for proving a program's first-order indefinability which are easier to be used in various situations.

4.1 Programs with the 0-1 property

Let $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$ be a structure. A relation $R_i^{\mathcal{A}}$ in $\mathcal{A}$ is called *0-relation* if $R_i^{\mathcal{A}} = \emptyset$, it is called *1-relation* if $R_i^{\mathcal{A}} = A^h$, were h is the arity of R_i. In general, a relation $R_i^{\mathcal{A}}$ in $\mathcal{A}$ is called *0-1 relation* if it is either a 0-relation or a 1-relation.

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

Intuitively, if a program has the 0-1 property, then this program must have a 0-1 answer set with arbitrary size, in which the interpretation for each intentional predicate of this program is a either 0-relation or 1-relation. Consequently, for programs with the 0-1 property, their corresponding 0-1 answer sets greatly simplify the underlying programs' output values (intentional predicates). Such property will be useful for proving a program's first-order indefinability, as will be showed next.

Example 2 Let us consider the following 2-coloring program for a graph containing at least one edge Π_{2Color}:

$E(a,b) \leftarrow$,
$Color1(x) \leftarrow Vertex(x), \texttt{not}\ Color2(x)$,
$Color2(x) \leftarrow Vertex(x), \texttt{not}\ Color1(x)$,
$\leftarrow E(x,y), Color1(x), Color1(y)$,
$\leftarrow E(x,y), Color2(x), Color2(y)$.

Now we show that Π_{2Color} does not hold the 0-1 property. Suppose that for each $k \geq 1$, Π has an answer set $\mathcal{A}$ with $|\mathsf{Dom}(\mathcal{A})| \geq \mathsf{k}$, and both intentional relations $Color1^{\mathcal{A}}$ and $Color2^{\mathcal{A}}$ are 0-1 relations in $\mathcal{A}$. From Π, we can see that the interpretations for the two intentional predicates $Color1$ and $Color2$ are always mutually excluded. This means that $Color1^{\mathcal{A}}$ and $Color2^{\mathcal{A}}$ cannot be both 0-relations or 1-relations. So we assume $Color1^{\mathcal{A}}$ is a 0-relation and $Color2^{\mathcal{A}}$ is a 1-relation. This actually implies that $Color1(a) = Color1(b) = 0$ and $Color1(a) = Color1(b) = 1$. But this is not possible because we assume that $\mathcal{A}$ is an answer set of Π where $Color1$ ($Color2$) cannot be interpreted with the same truth value on a and b.

Now we consider program $\Pi_{RChecking}$ which checks whether each vertex in a graph is reachable from the given initial vertex (vertices):

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

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

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

Theorem 3 *(**The 0-1 theorem**) Let Π be a 0-1 program and $\mathcal{A}$ a 0-1 answer set of Π. Π is not first-order definable if for each $k \geq 0$, there exists a structure $\mathcal{B}$ of $\tau(\Pi)$, such that $\mathcal{B}$ is not an answer set of Π, and $\mathcal{A}|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}|_{\tau_{ext}(\Pi)}$, where for each $P \in \tau_{int}(\Pi)$, $P^{\mathcal{B}} = P^{\mathcal{A}}$.*

Proof: According to Theorem 1, we know that the duplicator wins the Ehrenfeucht-Fraïssé game of length k played on $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$. Now we show that the duplicator also wins the Ehrenfeucht-Fraïssé game of length k over structures $\mathcal{A}$ and $\mathcal{B}$. Suppose after k rounds, elements $\overline{a} = (a_1, \cdots, a_k)$ and $\overline{b} = (b_1, \cdots, b_k)$ have been picked up from $\mathsf{Dom}(\mathcal{A})$ and $\mathsf{Dom}(\mathcal{B})$ respectively. Also let $\overline{c}^{\mathcal{A}} = (c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}})$ and $\overline{c}^{\mathcal{B}} = (c_1^{\mathcal{B}}, \cdots, c_m^{\mathcal{B}})$ be the interpretations of all constants of τ in $\mathcal{A}$ and $\mathcal{B}$ respectively. Then from the fact that $\mathcal{A}$ is a 0-1 answer set of Π, and each $P \in \tau_{int(\Pi)}$ is interpreted either as a 0-relation or a 1-relation in $\mathcal{B}$ as it is in $\mathcal{A}$ respectively, we have that for each $\overline{t} \subseteq \overline{ac}^{\mathcal{A}}$, $\overline{t} \in R^{\mathcal{A}}$ iff $\overline{t'} \in R^{\mathcal{B}}$, where $R^{\mathcal{A}}$

and $R^{\mathcal{B}}$ are the relations interpreting P in $\mathcal{A}$ and $\mathcal{B}$ respectively, and $\overline{t'}$ is the tuple of the corresponding elements from $\overline{ac}^{\mathcal{B}}$. This follows that $\mathcal{A} \equiv_k \mathcal{B}$. Finally, from Theorem 2, it concludes that Π is not first-order definable, $\square$

By Theorem 3, if a program has the 0-1 property, then when we prove the program's first-order indefinability, we may only apply the Ehrenfeucht-Fraïssé game over the restricted structures generated by extensional relations, e.g. $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$, instead of the whole structures, which are usually simpler. This is because in general, extensional relations for a program can be arbitrary. Consequently, their corresponding structures are also allowed to be flexible so that an Ehrenfeucht-Fraïssé game is easier to be proposed on such flexible structures, as illustrated by the following table.

0-1 answer set $\mathcal{A}^k$ of Π, a structure $\mathcal{B}^k$ of $\tau(\Pi)$ $(k \geq 0)$	
intentional predicates of Π	extensional predicates of Π
0-1 relations in $\mathcal{A}^k$ and $\mathcal{B}^k$ $\mathcal{A}^k|_{\tau_{int}(\Pi)} \equiv_k \mathcal{B}^k|_{\tau_{int}(\Pi)}$	show $\mathcal{A}^k|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}^k|_{\tau_{ext}(\Pi)}$ using Ehrenfeucht-Frassïé game (Theorem 2)

From the above table, we can see that by using Theorem 3, for each k, we can reduce the proof of $\mathcal{A}^k \equiv_k \mathcal{B}^k$ to the proof of $\mathcal{A}^k|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}^k|_{\tau_{ext}(\Pi)}$, by setting $\mathcal{A}^k$ to be the 0-1 answer set of Π. This will give us a great deal of freedom to consider the input structure $\mathcal{A}^k$ when we construct the underlying Ehrenfeucht-Fraïssé game. Example 3 shows an application of Theorem 3.

Example 3 (Example 2 continued). We show that $\Pi_{RChecking}$ is not first-order definable. In Example 2, we have showed that $\Pi_{RChecking}$ satisfies the 0-1 property. Now for each $k \geq 0$, we specify two structures $\mathcal{A}^k$ and $\mathcal{B}^k$ of $\tau(\Pi_{RChecking})$, where

$\mathsf{Dom}(\mathcal{A}^k) = \{0, 1, \cdots, 2m-1\}$, where $m \geq 2^{k+1}$,
$InitialVertex^{\mathcal{A}^k} = \{0\}$,
$E^{\mathcal{A}^k} = \{(i, i+1) \mid 0 \leq i < 2m-1\}$,
$Reachable^{\mathcal{A}^k} = \{i \mid 0 \leq i \leq 2m-1\}$,
$\mathsf{Dom}(\mathcal{B}^k) = \mathsf{Dom}(\mathcal{A}^k)$,
$InitialVertex^{\mathcal{B}^k} = \{0\}$,
$E^{\mathcal{B}^k} = \{(i, i+1) \mid 0 \leq i < m-1\} \cup \{(j, j+1) \mid m \leq j < 2m-1\} \cup \{(2m-1, m)\}$,
$Reachable^{\mathcal{B}^k} = Reachable^{\mathcal{A}^k}$.

Note that $\mathcal{A}^k|_{\tau_{ext}(\Pi_{RChecking})}$ can be viewed as an acyclic path with length of $2m$, and $\mathcal{B}^k|_{\tau_{ext}(\Pi_{RChecking})}$ can be viewed as two detached components: one is an acyclic path with length m and the other is a cycle of length m.

It is obvious that $\mathcal{A}^k$ is a 0-1 answer set of $\Pi_{RChecking}$, where $\mathcal{B}^k$ is not an answer set of $\Pi_{RChecking}$ because $Reachable$ is interpreted as a 1-relation in $\mathcal{B}^k$ but each j $(m \leq j < 2m)$ is not reachable from the initial vertex 0 in $\mathcal{B}^k$. Then using a similar method of the proof for Proposition 3, we can show that $\mathcal{A}^k|_{\tau_{ext}(\Pi_{RChecking})} \equiv_k \mathcal{B}^k|_{\tau_{ext}(\Pi_{RChecking})}$. According to Theorem 3, it concludes that $\Pi_{RChecking}$ is not first-order definable. $\square$

4.2 Programs with 0-1 unbounded cycles or paths

Theorem 3 can be effective in proving a 0-1 program Π's first-order indefinability if the proof of $\mathcal{A}^k|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}^k|_{\tau_{ext}(\Pi)}$ is already clear through the Ehrenfeucht-Fraïssé game approach. Nevertheless, as has been revealed in finite model theory, directly using the Ehrenfeucht-Fraïssé game approach is technically challenging for general cases [5]. Furthermore, in our first-order indefinability proofs for programs Π_{HC} and $\Pi_{RChecking}$, both programs happen to only have one binary extensional predicates, so that we can use graph representations to specify the game, which makes our proofs easier.

Although Theorems 2 and 3 do not rely on graph representations of structures, when a program involves more than one binary extensional predicates or extensional predicates with arity greater than 2, it does not seem to be obvious to use our method demonstrated in the proof of Proposition 2 to show a program's first-order indefinability.

In this subsection, we will develop another sufficient condition by which we can effectively prove a program's first-order indefinability under certain conditions.

To begin with, we first introduce a useful notion. Let $\mathcal{A}$ be a structure, the *Gaifman graph* of $\mathcal{A}$ [14] is an undirected graph $G(\mathcal{A}) = (A, Edge^{\mathcal{A}})$, where $\mathsf{Dom}(\mathcal{A}) = A$, and $Edge^{\mathcal{A}}$ is defined as follows:

$$Edge^{\mathcal{A}} = \{(a,b) \mid a \neq b \text{ and there are a relation } R^{\mathcal{A}} \text{ in } \mathcal{A} \text{ and } \overline{c} \text{ in } A \text{ such that } \overline{c} \in R^{\mathcal{A}} \text{ and } a \text{ and } b \text{ are among } \overline{c}\}.$$

We say that $\mathcal{A}$ has a *cycle* (or an *acyclic path*[4]) if $G(\mathcal{A})$ contains a connected component that is a cycle (or a path, resp.).

Definition 4 *(***Programs with 0-1 unbounded cycles or paths***) A program* Π *has* unbounded cycles *(or* paths*) if for all* k *and* m *where* $m > k$*, there is a* Π*'s answer set* $\mathcal{A}$ *such that* $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ *contains a cycle (path, resp.) with the length of* m*. A program* Π *has 0-1* unbounded cycles *(or* paths*) if* Π *is a 0-1 program, and for every* $k > 0$*, there is a* Π*'s 0-1 answer set* $\mathcal{A}$ *such that* $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ *contains a cycle (path, resp.) with the length of* m*. In this case,* $\mathcal{A}$ *is called a 0-1* cyclic *(*linear*, resp.) answer set of* Π*.*

Informally, we say that program Π has unbounded cycles (or paths), it simply means that Π has an answer set $\mathcal{A}$ while the Gaifman graph generated from the extensional database of this answer set contains a cycle (path), *and* the length of this cycle (path) cannot be bounded by any integer. We say that Π has 0-1 unbounded cycles (or paths), if the answer set $\mathcal{A}$ mentioned earlier is a 0-1 answer set of Π.

Example 4 Consider the following simple program Π_{PQ}:

$P(z) \leftarrow E(x,y),$
$Q(x) \leftarrow \texttt{not } P(x),$

[4]We will simply call it a path.

It is easy to observe that Π_{PQ} satisfies the 0-1 property and hence it is a 0-1 program. In fact, whenever $E^{\mathcal{A}}$ is not empty in the answer set $\mathcal{A}$, the two intentional predicates P and Q are interpreted as a 1-relation and a 0-relation in $\mathcal{A}$ respectively, otherwise, P and Q are interpreted as a 0-relation and a 1-relation respectively.

We can also see that for each $k > 0$, there exist at least two different types of 0-1 answer sets $\mathcal{A}$ and $\mathcal{B}$ of Π_{PQ}, where (1) $|\mathsf{Dom}(\mathcal{A})| > k$ and $G(\mathcal{A}|_{\{E\}})$ is a cycle with the length of $|\mathsf{Dom}(\mathcal{A})|$, and (2) $|\mathsf{Dom}(\mathcal{B})| > (k+1)$ and $G(\mathcal{B}|_{\{E\}})$ is a path of length $|\mathsf{Dom}(\mathcal{B})| - 1$. □ □

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-Fraïssé game.

Theorem 4 *(***The 0-1 unbounded cycles or paths theorem***) A program* Π *is not first-order definable if (1)* Π *has 0-1 unbounded cycles or paths, and (2) for each* Π*'s 0-1 cyclic or linear answer set* $\mathcal{A}$*,* $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ *contains only* one *cycle or path, while all other connected components of* $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ *are neither cycles nor paths*.

The following Example 5 and Proposition 4 show an application of using Theorem 4 to prove a program's first-order indefinability.

Example 5 Consider program $\Pi_{TCovered}$ as follows:

r_1: $T(x,y) \leftarrow E(x,y), \texttt{not}\ E(x,x), \texttt{not}\ E(y,y)$,
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: $\leftarrow D(x,y), \texttt{not}\ E(x,y)$,
r_6: $\leftarrow \texttt{not}\ Covered(x)$.

The first two rules are the same as the program in Example 1, which compute the transitive closure of the induced subgraph of E on the set of vertices that do not have self-loops. Extensional predicate D specifies a subset of edges of E (i.e. rule r_5), while the intentional predicate $Covered$ represents the set of vertices covered by edges of D (i.e. rules r_3 and r_4). Finally, having the last two constraints r_5 and r_6, it is implied that eventually all vertices should be covered.

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. □

Proposition 4 *Program* $\Pi_{TCovered}$ *in Example 5 is not first-order definable*.

Proof: We prove this result by using Theorem 4. For any given $k > 0$, we consider structure $\mathcal{A}^k$ as follows:

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

It is easy to verify that $\mathcal{A}^k$ 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 $\mathcal{A}^k$. So $\Pi_{TCovered}$ has the 0-1 property. Furthermore, $G(\mathcal{A}^k|_{\{E,D\}})$ 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 $\mathcal{B}$ of $\Pi_{TCovered}$, $G(\mathcal{B}^k|_{\{E,D\}})$ must be of the same form of $G(\mathcal{A}^k|_{\{E,D\}})$ as specified above. So both conditions (1) and (2) in Theorem 4 hold for $\Pi_{TCovered}$. This concludes that $\Pi_{TCovered}$ is not first-order definable. □

Using a similar way demonstrated in the proof of Proposition 4, we have the following result, which may be viewed as another way to show the first-order indefinability of transitivity.

Proposition 5 *The following Π_{TC} is not first-order definable:*

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

Proof: For any $k > 0$, we consider structure $\mathcal{A}^k$ as follows:

$\mathsf{Dom}(\mathcal{A}^k) = \{0, 1, \cdots, m\}$, where $m > k$,
$E^{\mathcal{A}^k} = \{(i, i+1) \mid 0 \leq i < m\} \cup \{(m, 0)\}$,
$T^{\mathcal{A}^k} = \{(i, j) \mid 0 \leq i, j \leq m\}$.

Clearly A^k is a 0-1 answer set of Π_{TC}. So the 0-1 property holds for Π_{TC}. Also, $G(\mathcal{A}^k|_{\{E\}})$ is a cycle of length m. As for any $k > 0$, such A^k exists, this means that Π_{TC} has unbounded cycles. Furthermore, for each 0-1 cyclic answer set $\mathcal{B}$ of Π_{TC}, $\mathcal{B}$ must be of the same form of A^k, so $G(\mathcal{B}|_{\{E\}})$ only contains exactly one cycle. By Theorem 4, we conclude that Π_{TC} is not first-order definable. □

Using the same way as described in the proof of Proposition 5, we can easily show that program Π_T from Example 1 is also not first-order definable:

$T(x, y) \leftarrow E(x, y), \texttt{not}\ E(x, x), \texttt{not}\ E(y, y)$,
$T(x, y) \leftarrow T(x, z), T(z, y)$.

Proposition 6 *The following program $\Pi_{TC'}$ is not first-order definable:*

$T'(x,y,z) \leftarrow E(x,y)$,
$T'(x,z,y) \leftarrow T'(x,y,z), E(z,y)$,
$T'(x,y,z') \leftarrow T'(x,y,z)$,
$T(x,y) \leftarrow T'(x,y,z)$.

Proof: Program $\Pi_{TC'}$ is a variation of program Π_{TC}, where both $T(x,y)$ and $T'(x,y,z)$ are intentional predicates. For any $k > 0$, we consider structure $\mathcal{A}^k$ as follows:

$\mathsf{Dom}(\mathcal{A}^k) = \{0, 1, \cdots, m\}$, where $m > k$,
$E^{\mathcal{A}^k} = \{(i, i+1) \mid 0 \leq i < m\} \cup \{(m, 0)\}$,
$T^{\mathcal{A}^k} = \{(i,j) \mid 0 \leq i, j \leq m\}$,
$T'^{\mathcal{A}^k} = \{(i,j,l) \mid 0 \leq i, j, l \leq m\}$.

It can be verified that A^k is an answer set of $\Pi_{TC'}$. Furthermore, the two intentional relations $T^{\mathcal{A}^k}$ and $T'^{\mathcal{A}^k}$ in $\mathcal{A}^k$ are 1-relations. Hence $\mathcal{A}^k$ is also a 0-1 answer set of $\Pi_{TC'}$. That is, $\Pi_{TC'}$ satisfies the 0-1 property.

It is also clear that $G(\mathcal{A}^k|_{\{E\}})$ is a single cycle of length m. As for all $k > 0$, such A^k exists, this means that $\Pi_{TC'}$ has unbounded cycles. Furthermore, for each 0-1 cyclic answer set $\mathcal{B}$ of $\Pi_{TC'}$, $\mathcal{B}$ must be of the same form of A^k, so $G(\mathcal{B}|_{\{E\}})$ only contains exactly one cycle. By Theorem 4, we conclude that $\Pi_{TC'}$ is not first-order definable. □

Example 6 Consider program Π_{PQ} in Example 4 once again. As showed earlier, Π_{PQ} satisfies the 0-1 property and has unbounded cycles (and paths too). So Π_{PQ} satisfies Condition 1 of Theorem 4. However, Π_{PQ} does not satisfy Condition 2 of Theorem 4. For each $k \geq 1$, we can find a 0-1 answer set $\mathcal{A}$ of Π_{PQ}, where

$\mathsf{Dom}(\mathcal{A}) = \{0, 1, \cdots, 4k-1\}$,
$E^{\mathcal{A}} = \{(i, i+1) \mid 0 \leq i < (2k-1)\} \cup \{(2k-1, 0)\} \cup$
$\quad \{(j, j+1) \mid 2k \leq j < (4k-1)\} \cup \{(4k-1, 2k)\}$,
$P^{\mathcal{A}} = \mathsf{Dom}(\mathcal{A}) = \{0, 1, \cdots, 4k-1\}$, and
$Q^{\mathcal{A}} = \emptyset$.

Note that $G(\mathcal{A}|_{\{E\}})$ consists of two detached cycles with each of length $2k$. Hence Theorem 4 is not applicable to program Π_{PQ}. In fact, Π_{PQ} is first-order definable via formula

$$\forall xyz(P(z) \equiv E(x,y)) \wedge \forall x(Q(x) \equiv \neg P(x)).$$

□

Example 7 We consider program Π_Q from [1] 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 the 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 [14]. Taking a glance at this program, we may think that Π_Q is not first-order definable as well. Let us examine whether this is the case.

It is easy to observe that Π_Q has the 0-1 property, because any structure $\mathcal{A}$ of the following form is a 0-1 answer set of Π_Q:

$$\begin{array}{l}\mathsf{Dom}(\mathcal{A}) = \{0, 1, \cdots, m\}, \text{where } m > 0,\\ E^{\mathcal{A}} = \{(i, i+1) \mid 0 \leq i < m\} \cup \{(m, 0)\},\\ Q^{\mathcal{A}} = \{(i, j) \mid 0 \leq i, j \leq m\}.\end{array}$$

Furthermore, it is clear that $G(\mathcal{A}|_{\{E\}})$ contains one cycle. Since there is no bound for m, it means that Π_Q has 0-1 unbounded cycles.

Nevertheless, it is crucial to note that there exists some Π_Q's 0-1 answer set whose Gaifman graph does not just contain one cycle. For each m, we can construct a structure $\mathcal{B}$ as follows:

$$\begin{array}{l}\mathsf{Dom}(\mathcal{B}) = \{0, 1, \cdots, 2m-1\}, \text{where } m > 0,\\ E^{\mathcal{B}} = \{(i, i+1) \mid 0 \leq i < m-1\} \cup \{(m-1, 0)\} \cup\\ \qquad \{(j, j+1) \mid m \leq i < 2m-1\} \cup \{(2m-1, m)\},\\ Q^{\mathcal{B}} = \{(i, j) \mid 0 \leq i, j \leq 2m-1\}.\end{array}$$

Clearly, $\mathcal{B}$ is also a 0-1 answer set of Π_Q, but $G(\mathcal{B}|_{\{E\}})$ contains two disjoint cycles. Hence we cannot apply Theorem 4 to program Π_Q to show its first-order indefinability. In fact, Ajtai and Gurevich showed that program Π_Q can be defined via the following sentence [1]:

$$\begin{array}{l}\forall x x' y y' z(\\ (E(x,y) \supset Q(x,y)) \wedge ((Q(x,z) \wedge Q(z,y)) \supset Q(x,y)) \wedge\\ ((Q(x,x') \wedge Q(x',x') \wedge Q(y',y') \wedge Q(y',y)) \supset Q(x,y)) \wedge\\ ((Q(x,y) \wedge \neg E(x,y)) \supset \exists u(E(x,u) \wedge Q(u,y)) \wedge\\ ((Q(x,y) \wedge \neg E(x,y)) \supset \exists v(E(v,y) \wedge Q(x,v)))).\end{array}$$

□

From Examples 6 and 7, we can see that Theorem 4 may also be used as a method to reveal clues why a program could be first-order definable, in the sense that the program has a 0-1 answer set that contains more than one detached cycles or paths though these cycles or paths may be unbounded.

4.3 Proof of Theorem 4

The basic idea of proving Theorem 4 may be outlined as follows. Under the conditions of Theorem 4, for each $k > 0$ and each Π's 0-1 cyclic or linear answer set $\mathcal{A}^k$ whose domain

size is greater than k, the corresponding Gaifman graph on $\mathcal{A}^k$'s extensional relations will contain a cycle (path, resp.). Under such condition, we can always construct a structure $\mathcal{B}^k$ of $\tau(\Pi)$ in such a way that $\mathcal{B}^k$ is not an answer set of Π, where each intentional predicate in $\tau_{int}(\Pi)$ is interpreted as a 0-1 relation in $\mathcal{B}^k$ as in $\mathcal{A}^k$, and $G(\mathcal{B}^k|_{\tau_{ext}(\Pi)})$ contains two smaller cycles (or one smaller cycle one path, resp.). Then we will prove $A^k|_{\tau_{ext}(\Pi)} \equiv_k B^k|_{\tau_{ext}(\Pi)}$. Finally, by Theorem 3, it concludes that $A^k \equiv_k B^k$, and hence Π is not first-order definable. This is summarized as in the following table.

0-1 cyclic or linear answer set $\mathcal{A}^k$ of Π, a structure $\mathcal{B}^k$ of $\tau(\Pi)$ ($k \geq 0$)	
intentional predicates of Π	extensional predicates of Π
0-1 relations in $\mathcal{A}^k$ and $\mathcal{B}^k$ $\mathcal{A}^k|_{\tau_{int}(\Pi)} \equiv_k \mathcal{B}^k|_{\tau_{int}(\Pi)}$	Prove $\mathcal{A}^k|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}^k|_{\tau_{ext}(\Pi)}$ if: (1) $G(\mathcal{A}^k|_{\tau_{ext}(\Pi)})$ contains one big cycle, and $G(\mathcal{B}^k|_{\tau_{ext}(\Pi)})$ contains two small cycles, or (2) $G(\mathcal{A}^k|_{\tau_{ext}(\Pi)})$ contains a long path, and $G(\mathcal{B}^k|_{\tau_{ext}(\Pi)})$ contains one small cycle and one short path.
$\mathcal{A}^k \equiv_k \mathcal{B}^k$ (By Theorem 3)	

So the key issue here is to prove that under the condition and the construction of $\mathcal{B}^k$ described above, $A^k|_{\tau_{ext}(\Pi)} \equiv_k B^k|_{\tau_{ext}(\Pi)}$. For this purpose, we will need a result in finite model theory [15].

We first present necessary notions and concepts. Consider a structure $\mathcal{A} = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}})$. Let $G(\mathcal{A}) = (A, Edge^{\mathcal{A}})$ be the Gaifman graph of $\mathcal{A}$ and a an element of A. The *neighborhood* $N(a, d)$ of a of radius d is recursively defined as follows:

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

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 point a is the isomorphism type of $\mathcal{A} \uparrow N(a, d)$. That is, if $\mathcal{B}$ is a structure of the same vocabulary of $\mathcal{A}$ and b is an element of $\mathsf{Dom}(\mathcal{B})$, then a and b have the same d-type iff $\mathcal{A} \uparrow N(a, d) \cong \mathcal{B} \uparrow N(b, d)$ under an isomorphism mapping a to b. $\mathcal{A}$ and $\mathcal{B}$ are *d-equivalent* if for every d-type ι, they have the same number of points with d-type ι.

Now the following result will be used in our proof of Theorem 4.

Theorem 5 *[15] For every $k > 0$ and for every $d \geq 3^{k-1}$, if $\mathcal{A}$ and $\mathcal{B}$ are d-equivalent, then $\mathcal{A} \equiv_k \mathcal{B}$.*

The following two lemmas will be important in our proof of Theorem 4.

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

Proof: Note that since Π has unbounded cycles, according to Definition 4, for any positive integer k, Π has an answer set $\mathcal{A}$ where $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ contains a cycle with the length greater than k.

Case 1. We first consider the case that $G(\mathcal{A}_{\tau_{ext}(\Pi)})$ exactly contains one such cycle without any other component. For each $d > 0$, we consider such an answer set $\mathcal{A}$ as follows:

$$G(\mathcal{A}|_{\tau_{ext}(\Pi)}) = (A, Edge^{\mathcal{A}}), \text{where } A = \mathsf{Dom}(\mathcal{A}) = \{a_1, \cdots, a_{4d}\},$$
$$Edge^{\mathcal{A}} = \{(a_i, a_{i+1}) \mid a_i, a_{i+1} \in A, 1 \leq i < 4d\} \cup \{(a_{4d}, a_1)\}.$$

Clearly, $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ is a cycle with $4d$ vertices [5]. Now we can construct another structure $\mathcal{B}$ of $\tau(\Pi)$ in such a way:

$$G(\mathcal{B}|_{\tau_{ext}(\Pi)}) = (B, Edge^{\mathcal{B}}), \text{where } B = \mathsf{Dom}(\mathcal{B}) = \{b_1, \cdots, b_{4d}\},$$
$$Edge^{\mathcal{A}} = \{(b_i, b_{i+1}) \mid 1 \leq i < 2d)\} \cup \{(b_{2d}, b_1)\} \cup$$
$$\{(b_j, b_{j+1}) \mid (2d+1) \leq j < 4d)\} \cup \{(b_{4d}, b_{(2d+1)}\}.$$

Note that $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ contains two disjoint cycles and each with a length $2d$. Relations in $\mathcal{B}$ for intentional predicates of Π can be arbitrary, and for extensional predicates will become clear shortly.

By viewing $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ and $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ as two structures, we observe that each d-type in $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ or in $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ is a path with $2d-1$ vertices. Furthermore, each of $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ and $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ has exactly $4d$ points of this d-type. So from the previous definition, we know that $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ and $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ are d-equivalent.

Now we show that by carefully specifying relations in $\mathcal{B}|_{\tau_{ext}(\Pi)}$, we also have that $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ are d-equivalent. For any given d-type: $G(\mathcal{A}|_{\tau_{ext}(\Pi)}) \uparrow N(a,d) \cong G(\mathcal{B}|_{\tau_{ext}(\Pi)}) \uparrow N(b,d)$, we consider any tuple $\overline{t}$ from $N(a,d)$ and any intentional relation $R^{\mathcal{A}}$ from $\mathcal{A}|_{\tau_{ext}(\Pi)}$, we specify $\overline{t'} \in R^{\mathcal{B}}$ in $\mathcal{B}|_{\tau_{ext}(\Pi)}$ iff $\overline{t} \in R^{\mathcal{A}}$, where $\overline{t'}$ is the tuple from $N(b,d)$ corresponding to $\overline{t}$ under the isomorphism of $G(\mathcal{A}|_{\tau_{ext}(\Pi)}) \uparrow N(a,d) \cong G(\mathcal{B}|_{\tau_{ext}(\Pi)}) \uparrow N(b,d)$. This follows $\mathcal{A}|_{\tau_{ext}(\Pi)} \uparrow N(a,d) \cong \mathcal{B}|_{\tau_{ext}(\Pi)} \uparrow N(b,d)$.

Since $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ share the same domain, so as for $\mathcal{B}|_{\tau_{ext}(\Pi)}$ and $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$, it concludes that each of $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ also has exactly $4d$ points of this d-type. That is, $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ are d-equivalent as well.

Case 2. Now we consider the case that $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ not only contains one cycle, but also contains other connected components. In this case, structure $\mathcal{B}$ is constructed in the following way: (1) $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ will contain two cycles which correspond to the cycle in $G(\mathcal{A}|_{tau_{ext}(\Pi)})$, as showed in Case 1, and (2) for each connected component in $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$, $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ will contain exactly the same component as well. Hence, it is easy to see that the only difference between $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ is the cycle parts. Based on Case 1, it follows that $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ are d-equivalent for any $d > 0$. □

[5] Since there is no bound on Π's cycles, such answer set $\mathcal{A}$ always exists for any d according to Definition 4.

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

Proof: The proof will be almost the same as the proof of Lemma 1, except: for each $d > 0$, (1) we consider an answer set $\mathcal{A}$ of Π such that $G(\mathcal{A}|_{tau_{ext}(\Pi)})$ is a path of length $4d$, and (2) we construct another structure $\mathcal{B}$ such that $G(\mathcal{B}|_{tau_{ext}(\Pi)})$ contains one path of length $2d$ and one cycle of length $2d$. Then in a similar way, we can show that $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ are d-equivalent. □

Proof of Theorem 4:
Since Π is a 0-1 program and has 0-1 unbounded cycles or paths, from Lemmas 1 and 2, we know that for any $k > 0$, Π has a 0-1 cyclic or linear answer set $\mathcal{A}$, and we can always find a structure $\mathcal{B}$ of $\tau(\Pi)$ such that $\mathcal{A}|_{\tau_{ext}(\Pi)}$ and $\mathcal{B}|_{\tau_{ext}(\Pi)}$ are d-equivalent for each $d > 0$, where whenever $G(\mathcal{A}|_{\tau_{ext}(\Pi)})$ contains one cycle (path), $G(\mathcal{B}|_{\tau_{ext}(\Pi)})$ contains two cycles (one cycle and one path, resp.). Since this result holds for any $k > 0$ and $d > 0$, from Theorem 5, by setting $d \geq 3^{k-1}$, we then have $\mathcal{A}|_{\tau_{ext}(\Pi)} \equiv_k \mathcal{B}|_{\tau_{ext}(\Pi)}$.

Now by setting every intentional relation of $\mathcal{B}$ to be either 0-relation or 1-relation accordingly as in $\mathcal{A}$, it is concluded that $\mathcal{B}$ cannot be an answer set of Π due to condition (2) of Theorem 4. So by Theorem 3, Π is not first-order definable. □

4.4 A generalization result

In subsection 4.2, we have showed that Theorem 4 can be used in proving first-order indefinability for some programs, where the 0-1 property and unbounded cycles (paths) are two essential features for these programs. It is observed that in order to prove a given program Π's first-order indefinability, the 0-1 property is applied on Π's intentional predicates, and the requirement of having unbounded cycles (paths) is placed on Π's extensional predicates. Nevertheless, Theorem 4 will become inapplicable if a program's answer set does not comply with the 0-1 property on its intentional relations, or the Gaifman graph formed from the extensional relations of the answer set does not contain a single cycle or path. For instance, the program of finding Hamiltonian cycles Π_{HC} illustrated in Proposition 3 does not satisfy the conditions of Theorem 4.

Such observation on Theorem 4 motivates us to generalize the 0-1 property and the definition of unbounded cycles (paths) for a given program, so that we may develop a new sufficient condition to prove the first-order indefinability for a larger class of programs. In particular, we will propose a generalized 0-1 property that applies to an arbitrary set of predicates in a program, instead of only applying to intentional predicates. Accordingly, the generalized unbounded cycles (paths) conditions for a given program will not restrict to extensional predicates.

Definition 5 *(***Separable 0-1 programs***) Program* Π *is* 0-1 separable *if there exists some* $\tau_1 \subseteq \tau(\Pi)$*, such that (1) for each* $k \geq 1$*,* Π *has an answer set* $\mathcal{A}$ *with* $|\mathsf{Dom}(\mathcal{A})| \geq k$*, and (2) for all predicates in* τ_1*, their corresponding relations in* $\mathcal{A}$ *are 0-1 relations. In this case,* Π *is called a* $(0\text{-}1)_{\tau_1}$ program*, and* $\mathcal{A}$ *a* $(0\text{-}1)_{\tau_1}$ answer set.

It is easy to see that Definition 5 generalizes the previous 0-1 property defined in Definition 3, where we only considered intentional relations in Π's answer sets.

Given program Π and $\tau' \subseteq \tau(\Pi)$, we say that Π has τ'*-unbounded cycles (paths)* if for every $k > 0$, there is an answer set $\mathcal{A}$ of Π such that $G(\mathcal{A}|\tau')$ contains a cycle (path, resp.) with length greater than k.

Definition 6 *(***Separable 0-1 unbounded cycles or paths***) A program* Π *has* separable 0-1 unbounded cycles (paths) *if (1)* Π *is a* $(0\text{-}1)_{\tau_1(\Pi)}$ *program for some* $\tau_1 \subseteq \tau(\Pi)$*, and (2) for each* $k > 0$*, there is a* Π*'s* $(0\text{-}1)_{\tau_1}$ *answer set* $\mathcal{A}$ *such that* $G(\mathcal{A}|(\tau(\Pi) \setminus \tau_1))$ *contains a cycle (path, resp.) with length greater than* k*. In this case,* $\mathcal{A}$ *is called* Π*'s* $(0\text{-}1)_{\tau_1}$ cyclic (linear, resp.) answer set.

Now we are ready to present the following separation theorem, which is a generalization of Theorem 4.

Theorem 6 *(***Generalization result***) A program* Π *is not first-order definable if (1)* Π *has separable 0-1 unbounded cycles or paths based on some* $\tau_1 \subseteq \tau(\Pi)$*, and (2) for each* Π*'s* $(0\text{-}1)_{\tau_1}$ *cyclic (or linear, resp.) answer set,* $G(\mathcal{A}|(\tau(\Pi) \setminus \tau_1))$ *contains one cycle (or path, resp.), while all other connected components of* $G(\mathcal{A}|(\tau(\Pi) \setminus \tau_1))$ *are neither cycles nor paths.*

Proof: We extend Lemmas 1 and 2 to the following result, then Theorem 6 will directly follow from this result in the same way as in Proof of Theorem 4.

> Result. *For some* $\tau_1 \subseteq \tau(\Pi)$*, if* Π *has* $(\tau(\Pi) \setminus \tau_1)$*-unbounded cycles (or paths), then there exists a structure* $\mathcal{B}$ *of* $\tau(\Pi)$*, such that (1)* $G(\mathcal{B}|(\tau(\Pi) \setminus \tau_1))$ *contains two disjoint cycles (or disjoint one cycle and one path, resp.), and (2) for each* $d > 0$*,* $\mathcal{A}|(\tau(\Pi) \setminus \tau_1)$ *and* $\mathcal{B}|(\tau(\Pi) \setminus \tau_1)$ *are* d*-equivalent.*

This result can be proved in a similar way as the proofs of Lemmas 1 and 2, so we omit here. □

Example 8 Finding Hamiltonian cycles program Π_{HC} revisited. Proposition 3 shows that Π_{HC} is not first-order definable:

$$
\begin{aligned}
&HC(x,y) \leftarrow E(x,y), \texttt{not}\ OtherRoute(x,y),\\
&OtherRoute(x,y) \leftarrow E(x,y), E(x,z), HC(x,z), y \neq z,\\
&OtherRoute(x,y) \leftarrow E(x,y), E(z,y), HC(z,y), x \neq z,\\
&Reached(y) \leftarrow E(x,y), HC(x,y) Reached(x), \texttt{not}\ InitialVertex(x),\\
&Reached(y) \leftarrow E(x,y), HC(x,y), InitialVertex(x),\\
&\leftarrow \texttt{not}\ Reached(x).
\end{aligned}
$$

By taking a closer look at program Π_{HC}, it is observed that Π_{HC} has two extensional predicates $E(x, y)$ and $InitialVertex(x)$, and three intentional predicates $HC(x, y)$, $OtherRoute(x, y)$ and $Reached(x)$. Then it is easy to check that for each answer set $\mathcal{A}$ of Π_{HC}, if all three intentional relations $HC^{\mathcal{A}}$, $OtherRoute^{\mathcal{A}}$ and $Reached^{\mathcal{A}}$ are 0-1 relations, then $G(\mathcal{A}|_{\{E,InitialVertex\}})$ cannot be a single cycle or path. This implies that previous Theorem 4 is not applicable to program Π_{HC}.

Now for each $k > 0$, we consider a structure $\mathcal{A}^k$ as follows:

$\mathsf{Dom}(\mathcal{A}^k) = \{0, 1 \cdots, 2m-1\}$ where $m > k$,
$E^{\mathcal{A}^k} = \{(i, i+1) \mid 0 \leq i < (2m-1)\} \cup \{(2m-1, 0)\}$,
$InitialVertex^{\mathcal{A}^k} = \{0\}$,
$HC^{\mathcal{A}^k} = E^{\mathcal{A}^k}$,
$OtherRoute^{\mathcal{A}^k} = \emptyset$,
$Reached^{\mathcal{A}^k} = \{0, 1, \cdots, 2m-1\}$.

Clearly $\mathcal{A}^k$ is an answer set of Π_{HC}.

Now we set $\tau_1 = \{Reached, OtherRoute\}$ and $\tau_2(\Pi_{HC}) = \tau(\Pi_{HC}) \setminus \tau_1 = \{E, HC, InitialVertex\}$. Then it is observed that $Reached^{\mathcal{A}^k}$ is a 1-relation in $\mathcal{A}^k$ and $OtherRoute^{\mathcal{A}^k}$ is a 0-relation in $\mathcal{A}^k$. So Π_{HC} is 0-1 separable. Furthermore, we can see that
$G(\mathcal{A}^k|_{\{E,HC,InitialVertex\}})$ only contains a single cycle of length $2m$. Since for each $k > 0$, we can always construct such $\mathcal{A}^k$ with $\mathsf{Dom}(\mathcal{A}^k) > 2k$, this means that Π_{HC} has separable 0-1 unbounded cycles.

On the other hand, it is not difficult to observe that for any answer set $\mathcal{B}$ of Π_{HC} which is $(0\text{-}1)_{\{Reached,OtherRoute\}}$ cyclic, $G(\mathcal{B}|\{E, HC, InitialVertex\})$ has the same form of $G(\mathcal{A}^k|\{E, HC, InitialVertex\})$, i.e. contains exact one single cycle. So from Theorem 6, we know that Π_{HC} is first-order indefinable. □

5 Related work and conclusions

The expressive power of Datalog has been extensively studied in the literature, where the first-order definability of datalog query is one of the central topics. In Datalog, it has been shown that on arbitrary structures, a datalog program is bounded iff the corresponding datalog queries are first-order definable iff the datalog program is equivalent to a recursion-free datalog program. On finite structures, this is no longer true [1, 12]. These results, however, do not provide many hints about how to prove a datalog query's first-order (in)definability.

In finite model theory, Ehrenfeucht-Fraïssé game approach is the primary tool for proving first-order indefinability result [14]. But as it has been well recognized [16], using this approach for specific cases is technically challenging. One way to deal with such difficulty is to develop stronger sufficient conditions to ensure the winning strategy for the duplicator during an Ehrenfeucht-Fraïssé game. Although the existing results in finite model theory,

for instance, those summarized in [16], are just too general to apply to our problems under ASP setting, this idea indeed motivated our work presented in this paper.

The first result of extending Ehrenfeucht-Fraïssé game approach to Datalog was due to Cosmadakis' work [12]. Our Theorem 2 may be viewed as an analogy of Theorem 2.6 in [12] for ASP. Note that both results have looser conditions for the classes of structures than the original Ehrenfeucht-Fraïssé game theorem [14]. Since these two results only provide the corresponding Ehrenfeucht-Fraïssé game-theoretic characterizations on the first-order indefinability for ASP and Datalog respectively, as in finite model theory, they are quite hard to be used in practice. What makes our work significantly distinct from Cosmadakis' is that we have further developed stronger sufficient conditions that completely avoid the construction of Ehrenfeucht-Fraïssé game in the first-order indefinability proof, and can be quite effective for certain circumstances.

We believe that the work presented in this paper establishes an important basis towards a formal study of the expressiveness of first-order answer set programs. Several related issues are left for our future work. One continuing work is to develop more useful sufficient conditions to cover larger classes of first-order indefinable programs. In our current work, we mainly focused on the featured programs complying to the 0-1 property with unbounded cycles or paths. Some ideas of developing these conditions were motivated from relevant results and techniques in finite model theory, e.g. Gaifman graph theorem (Theorem 5 described in section 4.2) [21].

Another important topic is to study answer set programs' first-order (in)definability under specific structures. In this paper, we do not restrict our finite structures to any specific forms. In many applications, nevertheless, we may only deal with specific types of finite structures, such as finite graphs, strings, or ordered structures [22, 28]. Under such situations, new proof techniques should be developed to prove an answer set program's first-order indefinability.

References

[1] M. Ajtai and Y. Gurevich, Datalog vs. first order logic. In *Proceedings of the 30th Annual Symposium of Foundations of Computer Science*, pp 142-147, 1989.

[2] V. Asuncion, Y. Zhang and H. Zhang, Logic programs with ordered disjunction: First-order semantics and expressiveness. In *Proceedings of KR-2014*, 2014.

[3] V. Asuncion, Y. Chen, Y. Zhang and Y. Zhou, Ordered completion for logic programs with aggregates. *Artificial Intelligence*, 2015 (to appear).

[4] V. Asuncion, F. Lin, Y. Zhang and Y. Zhou, Ordered completion for first-order logic programs on finite structures. *Artificial Intelligence* 177-179 (2012) 1-24.

[5] S. Arora and R. Fagin, On winning strategies in Ehrenfeucht-Fraïssé games. *Theoretical Computer Science* 174 (1997) 97-121.

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

[7] M. Bartholomew and J. Lee, System ASPMT2SMT: Computing ASPMT theories by SMT solvers. In *Proceedings of JELIA 2014*, 529-542, 2014.

[8] S. Chaudhuri and M.Y. Vardi, On the equivalence of recursive and nonrecursive Datalog programs. *PODS*, pp 55-66, 1992.

[9] Y. Chen, F. Lin, Y. Zhang and Y. Zhou, Loop-separable programs and their first-order definability. *Artificial Intelligence* 175 (3-4) (2011) 890-913.

[10] Y. Chen, Y. Zhang and Y. Zhou, First-order indefinability of answer set programs on finite structures. In *proceedings of AAAI-2010*, pp 285-290, 2010.

[11] K.L. Clark, Negation as failure. In *Logics and Databases*, pp 293-322. Plenum Press, 1978.

[12] S.S Cosmadakis, On the first-order expressibility of recursive queries. In *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on PODS*, pp311-323, 1989.

[13] E. Dantsin and et al, Complexity and expressive power of logic programming. *ACM Computing Surveys* 33 (2001) 374-425.

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

[15] R. Fagin, L. Stockmeyer and M.Y. Vardi, On monadic NP vs. monadic co-NP. *Information and Computation* 120 (1995) 78-92.

[16] R. Fagin, Easier ways to win logical games. *DIMACS Series in Discrete Mathematics and Computer Science, American Mathematical Society* 13 (1997) 1-32.

[17] P. Ferraris, J. Lee and V. Lifschitz, Stable models and circumscription. *Artificial Intelligence* 175(1) (2011) 236-263.

[18] H. Gaifman, H. Mairson, Y. Sagiv and M.Y. Vardi, Undecidable optimization problems for database logic programs. *Journal of ACM* 40 (1993) 683-713.

[19] M. Gebser and R. Kaminski and B. Kaufmann and T. Schaub, *Answer Set Solving in Practice*, Morgan and Claypool Publishers, 2012.

[20] M. Gelfond and Y. Kahl, *Knowledge Representation, Reasoning, and the Design of Intelligent Agents*, Cambridge University Press, 2014.

[21] E. Grädel, P.G. Kolaitis, L. Libkin, M. Max, J. Spencer, M.Y. Vardi, Y. Venema and S. Weinstein, *Finite Model Theory and Its Applications*. Springer 2007.

[22] L. Hella and P.G. Kolaitis, How to define a linear order on finite models. *Annals of Pure and Applied Logic* 87 (1997) 241-267.

[23] P.G. Kolaitis, Implicit definability on finite structures and unambiguous computations (preliminary report). LICS 1990: 168-180, 1990.

[24] F. Lin and Y. Zhou, From answer set logic programming to circumscription via logic of GK. *Artificial Intelligence* 175(1) (2011) 264-277.

[25] F. Lin and Y. Zhao, ASSAT: computing answer sets of a logic program by SAT solvers. *Artificial Intelligence* 157 (2004) 115-137.

[26] F. Lin and Y. Zhou, From answer set logic programming to circumscription via logic of GK. *Artificial Intelligence* 2010 (to appear).

[27] Smodels: http://www.tcs.hut.fi/Software/smodels/

[28] A. Stolboushkin, Axiomatizable classes of finite models and definability of linear order. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science (LICS'92)*, pp 64-70. IEEE Press, 1992.

[29] J. Wittocx, M. Marien and M. Denecker, Grounding with bounds. In *Proceedings of AAAI-2008*, pp 527-577. 2008.

[30] H. Zhang, Y. Zhang, M. Ying and Y. Zhou, Translating theories into logic programs. In *Proceedings of IJCAI-2011*, pp 1126-1131, 2011.

[31] H. Zhang and Y. Zhang, First-order expressibility and boundedness for disjunctive logic programs. In *Proceedings of IJCAI-2013*, pp 1198-1204, 2013.

[32] Y. Zhang and Y. Zhou, On the progression semantics and boundedness of answer set programs. In *Proceedings of KR-2010*, pp 518-527, 2010.