Complexity of Expanding a Finite Structure and Related Tasks

Antonina Kolokolova, Yongmei Liu, David Mitchell, Eugenia Ternovska

August 7, 2006

Abstract

The authors of [MT05] proposed a declarative constraint programming framework based on classical logic extended with non-monotone inductive definitions. In the framework, a problem instance is a finite structure, and a problem specification is a formula defining the relationship between an instance and its solutions. Thus, problem solving amounts to expanding a finite structure with new relations, to satisfy the formula.

We present here the complexities of model expansion for a number of logics, alongside those of satisfiability and model checking. As the task is equivalent to witnessing the existential quantifiers in ∃SO model checking, the paper is in large part of a survey of this area, together with some new results. In particular, we describe the combined and data complexity of FO(ID), first-order logic extended with inductive definitions [DT04] and the guarded and k-guarded logics of [AvBN98] and [GLS01].

1 Introduction

Fagin's theorem that existential second order logic (∃SO) exactly captures the complexity class NP [Fag74] was the first result that led to the development of descriptive complexity [Imm99], an area studying the relationship between logics and complexity classes. From the practical point of view, descriptive complexity results provide a way for logics to be viewed as "programming languages" for the corresponding classes. This, in turn, suggests taking the idea of logics as programming languages as the basis for practical tools. In particular, in the framework of [MT05], search problems are cast as model expansion (abbreviated MX), which is the task of witnessing the (first block of) existential second order quantifiers in a SO sentence.

Although even in the case of model expansion for unrestricted first-order logic (that is, NP search problems) this approach can be useful thanks to the strength of modern SAT solvers, a natural question is for which logics model expansion is practical. For example, the extension of classical logic with inductive definitions [DT04] allows for a convenient way of representing recursion, including recursion through negation. The use of guarded and k-guarded fragments was motivated by the need for efficient grounding (reduction to SAT).

In this paper, we summarize complexity results for the model expansion problem, and try to fill in the gaps for the logics for which such fragments have not been studied. In particular, we analyze complexity of model expansion for ID-logic of [DT04] and guarded logics. The following table gives an overview of the complexity results presented in this paper. We use $\equiv_C K$ to denote that model checking or model expansion for a logic captures a complexity class K on a class C of finite structures (see Definition 2.1). New results from this paper are marked with *; results without references or the * marks are easy corollaries of the other results.

Logic	Model checking		Model expansion		Satisfiability (finite)
	Combined	**Data**	**Combined**	**Data**	
FO	$PSPACE$-c [Sto74]	$\equiv_{BIT} AC^0$ [BIS90]	$NEXP$-c [Var82, MT05]	$\equiv NP$ [Fag74]	undec [Tra50]
$FO(LFP)$	EXP-c [Var82]	$\equiv_s P$ [Imm82, Var82, Liv82]	$NEXP$-c	$\equiv NP$	undec
$FO(ID)$	EXP-c*	$\equiv_s P^*$	$NEXP$-c*	$\equiv NP$[MT05]	undec
FO^k	P-c [Var95, GO99]	$\in AC^0$	NP-c [Var95]	NP-c, $\not\equiv NP$	$k \geq 3$: undec $k = 2$: $NEXP$-c [GKV97] $k = 1$: EXP-c
GF_k	P-c [GO99, GLS01]	$\in AC^0$	$NEXP$-c* RGF_k:NP-c*	$k \geq 2$: $\equiv NP^*$ $k = 1$: NP-c RGF_k:NP-c, $\not\equiv NP^*$	$k \geq 2$: undec $k = 1$: $2EXP$-c [Grä99]
μGF	$UP \cap co\text{-}UP$ [Jur98]	$\in P$	$NEXP$-c	NP-c	$2EXP$-c [GW99]
$GF_k(ID)$	$\in EXP$	$\in P$	$NEXP$-c	$k \geq 2$: $\equiv NP$	$k \geq 2$: undec

Table 1: Complexity of model checking, model expansion and satisfiability problems for some logics

2 Preliminaries and definitions

In this section we review standard notions of "complexity" of a logic, that is, data and combined complexity for Model Checking, Finite Satisfiability and Model Expansion problems.

For a given logic L, we consider complexity of three problems.
1. *Model Checking* (MC): given $(\mathcal{A}, \phi)$, where ϕ is a sentence in L and $\mathcal{A}$ is a finite structure for $vocab(\phi)$, does $\mathcal{A} \models \phi$?
2. *Model Expansion* (MX): given $(\mathcal{A}, \phi)$, where ϕ is a sentence in L, $\mathcal{A}$ is a finite σ-structure where $\sigma \subset vocab(\phi)$, is there an expansion $\mathcal{A}'$ of $\mathcal{A}$ to $vocab(\phi)$ such that $\mathcal{A}' \models \phi$?
3. *Finite Model Existence (satisfiability in finite)*: given a sentence ϕ in L, is there a finite $\mathcal{A}$ for $vocab(\phi)$ such that $\mathcal{A} \models \phi$?
The first and the last of these problems have been studied for a long time. We focus our attention on the Model Expansion problem.

Example 2.1. Let $\mathcal{A}$ be a graph $G = (V; E)$, and let ϕ be $\forall x \forall y\ [(Clique(x) \wedge Clique(y)) \supset (x = y \vee E(x, y))]$. Let $\mathcal{B}$ be an expansion of $\mathcal{A}$ to $vocab(\phi)$. Then $\mathcal{B} \models \phi$ iff $Clique^{\mathcal{B}}$ is a set of vertices that forms a clique in $\mathcal{B}$.

For each of the problems (except satisfiability) we consider two notions of complexity (introduced by [Var82]; here we are following [Lib04] presentation). Let $enc()$ denote some standard encoding of structures and formulae by binary strings.

Definition 2.1. Let K be a complexity class and L a logic. Let P be the problem of MC or MX.

- The *data complexity* of P for L is K if for every sentence ϕ of L the language $\{enc(\mathcal{A}) | (\mathcal{A}, \phi) \in P\}$ belongs to K. The *data complexity* of P for L is K-hard if for some sentence ϕ of L the language $\{enc(\mathcal{A}) | (\mathcal{A}, \phi) \in P\}$ is K-hard. The *combined complexity* of L is K (resp. K-hard) if the language $\{(enc(\mathcal{A}), \mathrm{enc}(\phi)) | (\mathcal{A}, \phi) \in P\}$ belongs to K (resp. is K-hard).
- Let C be a class of finite structures. P for L *captures* K on C if the data complexity of P

for L is K and for every property Q of structures from C that can be tested with complexity K there is a sentence ϕ_Q of L such that $\mathcal{A} \models \phi_Q$ iff $\mathcal{A}$ has property Q, for every $\mathcal{A} \in C$.

Note that MX for a logic L is equivalent to MC for $\exists SO(L)$. That is, there exists an expansion of a structure $\mathcal{A}$ that satisfies a formula ϕ iff $\mathcal{A}$ satisfies ϕ preceded by existential quantifiers for all expansion predicates. Clearly, the complexity of MX lies between complexities of MC and satisfiability, since in that case, a part of the input structure is given. E.g. in the case of FO, we avoid undecidability by fixing the universe.

3 Complexity of MX for first-order logic

Complexities of model checking and satisfiability for first-order logic were determined several decades ago. The combined complexity of model checking for FO is $PSPACE$-complete by reduction to QBF [Sto74]. The data complexity of FO is complete for AC^0; moreover, FO captures AC^0 over structures with BIT predicate (or arithmetic structures) [BIS90].

A host of complexity results for MX problems can be obtained from the fact that MX for a logic L is equivalent to MC for $\exists SO(L)$.

Theorem 3.1. *The combined complexity of the MX problem for first-order logic is $NEXP$-complete. Moreover, MX for FO captures NP.*

Proof. The first part is implicit in the proof of expression complexity of $\exists SO$ from [Var82] (a different proof is presented in [MT05].) The second part follows immediately from Fagin's theorem [Fag74]. ☐

This also allows us to capture levels of the polynomial hierarchy: MX for Π_i captures Σ_{i+1}.

Remark 3.2. In some cases, the only information about the model that is given as an instance for the model expansion is the size of the model (i.e., the instance vocabulary σ is empty). In that case, it is reasonable to give the size of a model as a number in binary notation. This leads to an exponential increase in complexity (since the structure itself is exponential in the size of the input).

Although data complexity of model expansion for full first-order logic is NP-complete, there are fragments of FO for which model expansion is feasible. In particular, the results of [Grä92] translate into the following result.

Definition 3.3. A *universal Horn formula* is a first-order formula consisting of a conjunction of Horn clauses, preceded by universal first-order quantifiers. Here, a clause is Horn if it contains at most one positive occurrence of an expansion predicate. A *universal Krom formula* is defined similarly, except that the restriction is at most two occurrences of expansion predicates per clause.

Theorem 3.4. *The data complexity of the MX problem for universal Horn and Krom formulae is, respectively, P-complete and NL-complete. Moreover, MX for universal Horn and Krom captures P and NL, respectively, over successor structures.*

4 Complexity of MX for guarded fragments of FO

The guarded fragment GF of FO was introduced by Andréka *et al.* [AvBN98], and has recently received considerable attention. Here any existentially quantified subformula ϕ must be conjoined with a guard, i.e., an atomic formula over all free variables of ϕ. Gottlob *et al.* [GLS01] extended GF to the k-guarded fragment GF_k where the conjunction of up to k atoms may act as a guard.

The combined complexity of MC for GF_k is P-complete [GO99, GLS01]. In particular, MC for GF_k can be done in time $O(ln^k)$, where l is the size of the formula, and n is the size of the structure [LL03]. The finite satisfiability problem for GF is $2EXP$-complete [Grä99].

In this section, we discuss complexity of MX for GF_k: we show that the combined complexity of MX for GF_k, $k \geq 1$, is the same as that for FO, and MX for GF_k, $k \geq 2$, captures NP just as MX for FO does. We also identify a fragment of GF_k, which we denote by RGF_k, such that the combined complexity of MX for RGF_k is NP-complete. Although the data complexity of MX for RGF_k is NP-complete, we show that it does not capture NP. As a corollary of our main results, we show that finite satisfiability for GF_k, $k \geq 2$ is undecidable.

Formally, GF_k is defined as follows:

Definition 4.1. The *k-guarded fragment* GF_k of FO is the smallest set of formulas such that
1. GF_k contains atomic formulas;
2. GF_k is closed under Boolean operations;
3. GF_k contains $\exists\bar{x}(G_1 \wedge \ldots \wedge G_m \wedge \phi)$, if the G_i are atomic formulas, $m \leq k$, $\phi \in GF_k$, and the free variables of ϕ appear in the G_i. Here $G_1 \wedge \ldots \wedge G_m$ is called the *guard* of ϕ.

A fragment of GF_k that is of particular interest to MX is RGF_k, which we use to denote sentences from GF_k in which all guards are given by the instance structure (i.e., no expansion predicates appear in guards). Let FO^k denote FO formulas that use at most k distinct variables. Then it is easy to see that any FO^k formula can be rewritten in linear time into an equivalent one in RGF_k, by using atoms of the form $x = x$ as parts of the guards when necessary. For example, the formula $\exists x \exists y[R(x) \wedge E(x, y)]$ can be rewritten into $\exists x \exists y[R(x) \wedge y = y \wedge E(x, y)]$, where R is an instance predicate, and E is an expansion predicate.

Lemma 4.2. *There is a polynomial-time algorithm that, given an arbitrary $\exists SO$ sentence, constructs an equivalent $\exists SO$ sentence whose first-order part is in GF_2.*

Proof. Suppose ϕ is a $\exists SO$ sentence $\exists X_1 \ldots \exists X_m\, \varphi$, where φ is an FO formula. Let l be the size of ϕ, and let k be the width of φ, that is, the maximum number of free variables in any subformula of φ. We introduce k new predicates $U_1, \ldots, U_k$ such that the arity of U_i is i, $1 \leq i \leq k$. Let φ' be the formula obtained from φ by replacing any subformula $\exists\bar{x}\,\psi(\bar{x})$ with $\exists\bar{x}(U_i(\bar{x}) \wedge \psi(\bar{x}))$ and any subformula $\forall\bar{x}\,\psi(\bar{x})$ with $\forall\bar{x}(U_i(\bar{x}) \supset \psi(\bar{x}))$, where i is the length of $\bar{x}$. Let η be the formula

$$\bigwedge_{i=0}^{k-1} \forall x_1 \ldots \forall x_{i+1}(x_1 = x_1 \wedge U_i(x_2 \ldots x_{i+1}) \supset U_{i+1}(x_1 \ldots x_{i+1})).$$

It is easy to see that any model of η interprets U_i as the i-ary universal relation, $1 \leq i \leq k$. Now let ϕ' be the $\exists SO$ sentence $\exists X_1 \ldots \exists X_m \exists U_1 \ldots \exists U_k\ (\varphi' \wedge \eta)$. Clearly, $\varphi' \wedge \eta \in GF_2$, and ϕ' is equivalent to ϕ. Also, both φ' and η are of size $O(l^2)$, and hence ϕ' is of size $O(l^2)$. □

Lemma 4.3. *There exists a polynomial-time algorithm that, given a structure M and an $\exists SO$ sentence ϕ, constructs a structure M' and an $\exists SO$ sentence ϕ_M such that the first-order part of ϕ_M is in GF_1, and $M \models \phi$ iff $M' \models \phi_M$.*

Proof. Suppose M is a structure, and ϕ is an $\exists SO$ sentence. Let n be the size of M, and let l be the size of ϕ. For each domain element a of M, we introduce a new constant symbol c_a. Let M' be the structure that expands M by interpreting c_a as a. Let ϕ' be the $\exists SO$ sentence constructed from ϕ as in the proof of the above lemma. Now let ϕ_M be the sentence obtained from ϕ' by replacing each subformula $\forall x_1 \ldots \forall x_{i+1}(x_1 = x_1 \wedge U_i(x_2 \ldots x_{i+1}) \supset U_{i+1}(x_1 \ldots x_{i+1}))$ with

$$\bigwedge_{a \in dom(M)} \forall x_2 \ldots \forall x_{i+1}(U_i(x_2 \ldots x_{i+1}) \supset U_{i+1}(c_a x_2 \ldots x_{i+1})).$$

Clearly, the first-order part of ϕ_M is in GF_1, $M \models \phi$ iff $M' \models \phi_M$, and the size of ϕ_M is $O(l^2 n)$. □

Theorem 4.4. *(1) The combined complexity of MX for GF_k, $k \geq 1$ is NEXP-complete. (2) MX for GF_k, $k \geq 2$ captures NP. (3) The finite satisfiability problem for GF_k, $k \geq 2$ is undecidable.*

Proof. (1) follows from Lemma 4.3 and that the combined complexity of MX for *FO* is in NEXP. (2) follows from Lemma 4.2 and that MX for *FO* captures NP. (3) By the proof of Lemma 4.2, finite satisfiability for *FO* can be reduced to that for GF_2. □

Lemma 4.5 ([MT05]). *3-SAT can be reduced to MX for a formula $\phi \in RGF_1$.*

Proof. Suppose $\Gamma = \{C_1, \ldots, C_m\}$ is a set of 3-clauses. Let $\mathcal{A}$ be the structure with universe $\{a, \neg a \mid a \in atoms(\Gamma)\}$ such that $\mathcal{A}$ interprets *Clause* as the set of clauses in Γ and interprets *Complements* as the set of complementary literals. Let ϕ be

$$\forall x \forall y \forall z(Clause(x, y, z) \supset True(x) \vee True(y) \vee True(z))$$
$$\wedge \forall x \forall y(Complements(x, y) \supset (True(x) \equiv \neg True(y))).$$

Clearly, $\phi \in RGF_1$, and Γ is satisfiable iff $\mathcal{A}$ can be expanded to a model of ϕ. □

We quote the following result concerning polynomial-time grounding of RGF_k sentences:

Lemma 4.6 ([PLTG06]). *There exists an algorithm that, given a structure $\mathcal{A}$ and a RGF_k sentence ϕ, constructs in $O(l^2 n^k)$ time a propositional formula ψ such that $\mathcal{A}$ can be expanded to a model of ϕ iff ψ is satisfiable, where l is the size of ϕ, and n is the size of $\mathcal{A}$.*

Theorem 4.7. *(1) The combined complexity of MX for RGF_k is NP-complete. (2) The data complexity of MX for GF_1 and RGF_k is NP-complete. (3) MX for RGF_k and hence also FO^k does not capture NP.*

Proof. (1) follows from Lemmas 4.6 and 4.5. (2) follows from Lemma 4.5 and that the data complexity of MX for *FO* is in NP. (3): Since SAT can be decided in nondeterministic $O(n^2)$ time, by Lemma 4.6, MX for RGF_k can be decided in nondeterministic $O(n^{2k})$ time. By Cook's *NTIME* hierarchy theorem [Coo73], for any $i > 2k$, there is a problem that can be solved in nondeterministic $O(n^i)$ time but not nondeterministic $O(n^{i-1})$ time. Thus there are infinitely many problems in *NP* that cannot be expressed by MX for RGF_k. □

5 Complexity of ID-logic

One disadvantage of first-order logic as a programming language is its lack of mechanism for recursion and induction. Therefore, a natural way to extend first-order logic is by adding inductive definitions. One such approach, called ID-logic, is presented in [DT04]. ID-logic is an extention of classical logic in which (non-monotone) definitions can appear as atomic formulae.

Definition 5.1. An *inductive definition* Δ is a set of rules of the form $\forall\bar{x}(X(\bar{t}) \leftarrow \phi)$ where X is a predicate symbol (constant or variable) of arity r, $\bar{x}$ is a tuple of object variables, $\bar{t}$ a tuple of object variables of length r, ϕ is an arbitrary first-order formula.

The semantics of the logic is defined by the standard truth recursion of classical logic, augmented with one additional rule saying that a valuation I satifies a definition D if it is the 2-valued well-founded model of this definition, as defined in the context of logic programming.

Example 5.1. Consider formula $\Delta_{even}(E) \wedge \forall x(E(x) \vee O(s(x)))$, where

$$\Delta_{even} \equiv \left\{ \begin{array}{rl} E(x) & \leftarrow x = 0 \\ E(s(s(x))) & \leftarrow E(x) \wedge \neg E(s(x)) \end{array} \right\}.$$

This formula states that every number is either even or odd. Definition Δ_{even} is one of possible definitions of even numbers, which is total on natural numbers, but not on integers.

5.1 Equivalence between $FO(ID)$ and $FO(LFP)$

In this section we show that first-order logic with inductive definitions, $FO(ID)$, is equivalent to first-order logic with least fixed point operator, $FO(LFP)$, just as first-order logic with monotone inductive definitions. This allows us to transfer known complexity results for $FO(LFP)$ to $FO(ID)$ logic. Here we only talk about first-order logic with inductive definitions; therefore, we will use $FO(ID)$ and ID-logic interchangeably.

Lemma 5.2. *ID-logic* $\subseteq FO(LFP)$.

Proof. We start by showing how to evaluate a single definition Δ (which can have multiple defined predicates). If a definition is not total on I_0, we need to ensure that there is no model for the whole theory. Then we can use evaluated definitions to construct a $FO(LFP)$ formula corresponding to the original formula of ID-logic.

A definition Δ for a given initialization of open predicates from I_0 is evaluated as follows.

Replace in Δ all occurrences of X_i by X'_i for new variables X'_i. For example, a rule $\forall\bar{x}(X_i(\bar{t}(\bar{x})) \leftarrow \neg X_j(\bar{t}'(\bar{x})))$ becomes replaced with $\forall\bar{x}(X_i(\bar{t}(\bar{x})) \leftarrow \neg X'_j(\bar{t}'(\bar{x})))$ Let ϕ be a formula encoding Δ after this substitution.

Computing one (double) step of the evaluation (a step corresponding to evaluating ϕ with I and then J giving the values for negated literals) becomes

$$\psi \equiv LFP_{\bar{x},\bar{X}}\phi([LFP_{\bar{x},\bar{X}}\phi]^j/X'_j), \qquad (*)$$

by semantics of ID-logic. Here fixpoints are simultaneous on all X_i and the notation $[LFP_{\bar{x},\bar{X}}\phi]^j/X'_j$ means replacing the occurrences of X'_j in ϕ with the fixpoint of X_j in the simultaneous least fixed point of ϕ over all $\bar{X}$.

To simplify the presentation assume, using the fact that simultaneous LFP is equivalent to LFP, that a variable X encodes all variables X_i. Then, the simultaneous LFPs from ψ become just LFPs.

Let Y be a variable encoding the fixpoint of X after the double step $(*)$. This variable is used to initialize X_i' before the next double step. Since after each step ψ the variable Y contains the partial truth assignments on structure I after i^{th} (double) step of the evaluation procedure, Y is monotone. Therefore, there exists a fixpoint of Y defined by ψ, and it is the least fixed point. Therefore, the formula

$$\Psi_\Delta(\bar{u}) \equiv [LFP_{\bar{y},Y}\psi(Y)]\bar{u}$$

computes the values of the defined predicates in ϕ whenever the fixpoint exists. This is also true when Y is treated as a list of predicates $X_1 \dots X_k$ being defined in Δ, in which case LFP in Ψ_Δ is a simultaneous fixed point.

It is possible, though, that the value computed using the upper bound estimation (the innermost LFP of the double step $(*)$) is different from the outer LFP in the double step. If this is the case, then the following formula is false:

$$CONS_\Delta \equiv \forall \bar{z}([LFP_{\bar{y},Y}\psi(Y)]\bar{z} \leftrightarrow LFP_{\bar{x},\bar{X}}\psi(\bar{x}, LFP_{\bar{y},Y}\psi(Y)/X_i'))[\bar{z}]$$

Suppose now that the theory of ID-logic is defined by a formula with multiple definitions. Let ϕ' be a first-order formula with occurrences of definitions $\Delta_1 \dots \Delta_m$ for some m. To simplify the presentation, view each definition as defining one predicate P_i. If the fixpoint of Δ_i exists, then $\forall \bar{x} P_i(\bar{x}) \leftrightarrow \Psi_{\Delta_i}(\bar{x})$, so occurrences of P_i in ϕ' can be treated as occurrences of Ψ_{Δ_i}. From the point of view of evaluation, it is more efficient to compute $P_i \equiv \Psi_{\Delta_i}$ and then refer just to P_i.

Finally, ϕ' is converted to a formula

$$\Phi \equiv \bigwedge_{i=1}^{m} CONS_{\Delta_i} \wedge \phi'((\forall \bar{x}(P_i(\bar{x}) \leftrightarrow \Psi_{\Delta_i}(\bar{x})))/\Delta_i)$$

That is, Φ is a conjunction of two parts: the conjunction of consistency formulas ensures that all definitions were total, and ϕ' remains the same except all definitions are replaced by the $FO(LFP)$ formulas computing them.

The resulting formula is in $FO(LFP)$, which completes this direction of the proof. □

Example 5.2. Recall the formula from Example 5.1 stating that every number is either even or odd. The following describes a construction of an equivalent $FO(LFP)$ formula.

A formula corresponding to Δ_{even} becomes, after replacing $\neg E$ with $\neg E'$,

$$\{(\phi_E(x, E, E') \equiv (\exists y(x = y \wedge y = 0)) \vee (\exists y(x = s(s(y)) \wedge E(y) \wedge \neg E'(s(y)))))\}.$$

Define $\psi_E(z, E') \equiv [LFP_{x,E}\phi_E(x, E, LFP_{E,x}\phi_E(x, E, E')))]z$. This computes one iteration of the stable operator ST^2_Δ.

Now, $\Psi_\Delta \equiv LFP_{z,E'}\psi_E(z, E')$. Consistency is checked by $\forall u \Psi_\Delta(u) \leftrightarrow [LFP_{x,E}\psi_E(x, E, \Psi_\Delta)]u$. Now, the final formula becomes

$$(\forall u \Psi_\Delta(u) \leftrightarrow [LFP_{x,E}\psi_E(x, E, \Psi_\Delta)]u) \wedge (\forall x(P(x) \leftrightarrow \Psi_\Delta(x)) \wedge (P(x) \vee O(s(X))).$$

Here, the first conjunct checks that the definition "makes sense", otherwise the formula does not have a model, the second part is a syntactic sugar defining a particular variable $P(x)$ to represent the defined E, and the last part uses P outside of the definition Δ_E.

Lemma 5.3. *$FO(LFP) \subseteq$ID-logic*

Proof. By [EF95] theorem 9.4.2, every $FO(LFP)$ formula is equivalent to one of the form $\forall u[LFP_{\bar{z},Z}\psi]\tilde{u}$, where $\psi \in \Delta_2$. This can be written as an ID-logic formula $Z(\bar{z}) \leftarrow \psi$. □

Therefore, the following theorem holds:

Theorem 5.4. *The complexity of model checking for ID-logic and FO(LFP) coincide over finite structures.*

Corollary 5.5. *Combined complexity of the model checking for $FO(ID)$ is complete for EXP. Expression complexity for $FO(ID)$ is complete for P, moreover, $FO(ID)$ captures P over structures with successor.*

5.2 Complexity of MX for $FO(ID)$

Intuitively, adding polynomial-time computable predicates to an NP predicate should not add any extra power. That allows us to suggest that both combined and data complexity of $FO(ID)$ (or, equivalently, $FO(LFP)$) coincides with the corresponding complexity for the MX of FO without inductive definitions or fixed-point computations.

Theorem 5.6. *Combined complexity of MX for $FO(ID)$ is $NEXP$-complete. Data complexity for MX of $FO(ID)$ is NP-complete, and NP is captured by existential second-order with inductive definitions $\exists SO(ID)$.*

Proof. We know from Theorem 3.1 that data complexity of MX problem is hard for NP and combined complexity for $NEXP$. Therefore, it is sufficient to show that MX problem can be solved within these classes.

The evaluation algorithm proceeds as follows. Use non-determinism to guess the expansion predicates. Now the problem is reduced to evaluating $FO(ID)$ formula on an expanded structure. This can be done in polynomial time of the size of the structure when formula is fixed (by [Imm82, Var82, Liv82]) and in exponential time when the formula is a part of the input by [Var82]. In the second case, the size of the expansion predicates can be exponential in the size of the structure (since their arity is not constant), but in $NEXP$ we can guess exponential-size certificates. □

5.2.1 Fragments of FO(ID) with polytime MX

Recall that MX for universal Horn formulae was P-complete. We would like to add inductive definitions to such formulae so that the complexity of the resulting logic is still in P. The following example shows that allowing unrestricted use of expansion predicates in the inductive definitions makes it possible to encode NP-complete problems

Example 5.3. The classical example of 3-colourability is representable as a formula with three expansion predicates R, B, G, encoding colours:

$$\forall v, w(R(v) \vee B(v) \vee G(v)) \wedge \bigwedge_{Q \in R,G,B} (\neg Q(v) \vee \neg Q(w) \vee \neg E(v,w)).$$

The only part of this formula which is not Horn is the first disjunction. It can be replaced by the inductive definition with a rule $X(i) \leftarrow Q(i)$ for every colour Q. Now, the first disjunction is equivalent to $\forall v X(v)$. Note that the definition of ID-logic requires that such X were minimal, therefore, this does not introduce spurious positives.

However, if we disallow any occurrences of the expansion predicates in inductive definitions, P-completeness is preserved.

Lemma 5.7. *Adding inductive definitions to universal Horn formulae defined on page 3 preserves data complexity of MX problem to be P-complete, when expansion predicates do not occur in inductive definitions.*

Proof. By theorem 3.4, data complexity of MX problem for universal Horn formulae is P-complete. Therefore, a polytime algorithm for MX of universal Horn formulae can first evaluate all inductive definitions, and then run Grädel's algorithm for evaluating existential second-order Horn formulae replacing all defined predicates by their computed values. □

We can also add expansion predicates in a restricted fashion. First, all expansion predicates occurring in definitions have to be defined (i.e, occur in a head of a rule of some definition). Second, such predicates cannot be defined in terms of each other unless they are in the same definition. Third, the definitions can only occur as conjunction to the rest of the formula. Intuitively, in this case, if expansion predicates in the body of a definition are either given values already, or are being defined in that definition, then the definition can be evaluated. The intuition here is similar to the intuition of RGF_k.

Definition 5.8. Let $\{\bar{X}_1, \ldots, \bar{X}_k\}$ be all expansion predicates occurring in a first-order formula ϕ. Then ϕ is in *RFO(ID)* if (1) for each $\bar{X}_i$ there is a definition Δ_i defining all predicates in $\bar{X}_i$, and Δ_i is conjuncted with the rest of the formula. (2) The only expansion predicates allowed in the body of Δ_i are among $\bar{X}_1, \ldots, \bar{X}_{i-1}$; the body of Δ_1 contains no expansion predicates.
More generally, ϕ is in RuHorn(ID) if there are also expansion predicates $\bar{P}$ which do not occur in the definitions and with all definitions removed, ϕ is universal Horn with respect to $\bar{P}$

Theorem 5.9. *MX problem for RFO(ID) is P-complete.*

Corollary 5.10. *MX problem for RuHorn(ID) is P-complete.*

6 Conclusion and open problems

In this paper, we give a survey of complexity results related to the model expansion framework. Model expansion is a very new approach. Many problems are still unsolved, both theoretical and practical. From the theoretical point of view, it would be interesting to extend complexity results to richer vocabularies (e.g., dealing with function symbols, arithmetic, etc), as well as looking at the model expansion versions of other commonly used logic. Also, we know that $GF(ID)$ (guarded logic with inductive definitions) coincides with μGF on total structures; however, the question is still open whether they coincide everywhere, like $FO(ID)$ and $FO(LFP)$. There, the problem lies in a different treatment of inductive definitions that are not total.

References

[AvBN98] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *J. Phil. Logic*, 49(3):217–274, 1998.

[BIS90] D. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274 – 306, 1990.

[Coo73] S. A. Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, 1973.

[DT04] Marc Denecker and Eugenia Ternovska. A logic of non-monotone inductive definitions. *ACM transactions on computational logic*, V(N):1–50, 2004.

[EF95] H.-D. Ebbinghaus and J. Flum. *Finite model theory.* Springer Verlag, 1995.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of computation, SIAM-AMC proceedings*, 7:43–73, 1974.

[GKV97] E. Grädel, Ph. G. Kolaitis, and M. Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3:53–69, 1997.

[GLS01] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 195–206, 2001.

[GO99] E. Grädel and M. Otto. On logics with two variables. *Theor. Comput. Sci.*, 224:73–113, 1999.

[Grä92] E. Grädel. Capturing Complexity Classes by Fragments of Second Order Logic. *Theoretical Computer Science*, 101:35–57, 1992.

[Grä99] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.

[GW99] Erich Grädel and Igor Walukiewicz. Guarded fixed point logic. In *LICS'99*, pages 45–55, 1999.

[Imm82] N. Immerman. Relational queries computable in polytime. In *STOC'82*, pages 147 –152, 1982.

[Imm99] N. Immerman. *Descriptive complexity.* Springer Verlag, New York, 1999.

[Jur98] M. Jurdzinski. Deciding the winner in parity games is in UP∩co-UP. *Information Processing Letters*, 69:119–124, 1998.

[Lib04] L. Libkin. *Elements of Finite Model Theory.* Springer Verlag, 2004.

[Liv82] A.B. Livchak. Languages for polynomial-time queries. In *Computer-based modeling and optimization of heat-power and electrochemical objects*, page 41, 1982.

[LL03] Y. Liu and H. J. Levesque. A tractability result for reasoning with incomplete first-order knowledge bases. In *Proc. of the 18th Int. Joint Conf. on Artif. Intell. (IJCAI)*, pages 83–88, 2003.

[MT05] David Mitchell and Eugenia Ternovska. A framework for representing and solving NP search problems. In *Proc. of the 20th National Conf. on Artif. Intell. (AAAI)*, pages 430–435, 2005.

[PLTG06] M. Patterson, Y. Liu, E. Ternovska, and A. Gupta. Grounding for model expansion in k-guarded formulas, 2006. Short presentation at LICS'06.

[Sto74] L. Stockmeyer. *The Complexity of Decision Problems in Automata Theory.* PhD thesis, MIT, 1974.

[Tra50] B. Trahtenbrot. The impossibility of an algorithm for the decision problem for finite domains. *Doklady Academii Nauk SSSR*, 70:569–572, 1950. In Russian.

[Var82] Moshe Y. Vardi. The complexity of relational query language. In *STOC'82*, 1982.

[Var95] Moshe Y. Vardi. On the complexity of bounded-variable queries. In *Proc. of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 266–276, 1995.