Bounded-Variable Fixpoint Queries are PSPACE-complete [⋆]

Stefan Dziembowski
std@mimuw.edu.pl

Warsaw University, Institute of Informatics, Banacha 2, 02-097 Warszawa, Poland, phone: (+48-22) 658-31-65, fax: (+48-22) 658-31-64

Abstract. We study the complexity of the evaluation of bounded-variable fixpoint queries in relational databases. We exhibit a finite database such that the problem of deciding whether a closed fixpoint formula using only 2 individual variables is satisfied in this database is PSPACE-complete. This clarifies the issues raised by Vardi in [Var95]. We study also the complexity of query evaluation for a number of restrictions of fixpoint logic. In particular we exhibit a sublogic for which the upper bound postulated by Vardi holds.

1 Introduction

In [Var95] Vardi studies computational complexity of queries expressed in various logics. There are three notions of the complexity of query evaluation.

1. We can fix a database and evaluate many different queries expressible in a certain logic against this database. In this case we measure the complexity as a function of the length of the expression denoting the query. We call it the *expression complexity* of the logic.
2. We can fix a query and evaluate this query against different databases. In this case we measure the complexity as a function of the size of the database (*data complexity* of the logic).
3. We can evaluate many different queries against many different databases and measure the complexity as a function of the combined size of the database and the expression denoting the query (*combined complexity* of the logic).

Vardi remarks that for many logical languages there is a gap between the data complexity on the one side, and the expression and combined complexities on the other. For example the data complexity of the first order logic (FO) is AC^0 while the expression and combined complexities are complete for PSPACE; the data complexity of the fixpoint logic (FP) is PTIME, while the expression and the combined complexities are EXPTIME-complete.

[⋆] Partially supported by Polish KBN grant No. 8 T11C 002 11. Part of this work was done at BRICS, the Center for **B**asic **R**esearch in **C**omputer **S**cience, Aarhus, Denmark.

The main idea of [Var95] is that for logics with a uniformly bounded number of individual variables this gap narrows. The syntax restriction captures the well-known technique of database programmers of avoiding large intermediate results.

We study the problem of measuring the expression and the combined complexities of the bounded-variable version of the fixpoint first-order logic (FP^k). We show that both of them are PSPACE-complete. In [Var95] Vardi has proposed an NP algorithm for this problem, however the algorithm works only for a subclass of FP^k formulas. Recall that data complexity of FP^k is PTIME.

We also consider various restrictions of the FP^k syntax and study their complexity. At first, we restrict the number of second-order variables. We show that in this case the expression complexity is in ALOGTIME, but the combined complexity is still PSPACE-complete. Since ALOGTIME $\neq$ PSPACE this is, up to our knowledge, the first provable gap between the expression complexity and the combined complexity. We also study the combined and the expression complexities of the prefix version of FP^k (with formulas of the form: $\mathsf{prefix\ of\ fixpoint\ operators}\ .\ \mathsf{first\text{-}order\ formula}\ .\ \mathsf{arguments}$), showing the PSPACE-completeness of both problems. Finally we present a sublogic of FP^k for which the NP $\cap$ co-NP upper bound for the combined complexity holds.

Our results confirm Vardi's idea that bounding the number of variables may lead to narrowing the gap between the data and the combined complexity. Whether this gap is indeed more narrow depends on the hypothesis that PSPACE $\neq$ EXPTIME.

2 Basic Definitions

The definitions presented in this section are based on the definitions from [CH82, Var82]. We change them a bit introducing the notion of database signature, which is similar to the standard notion of signature used in mathematical logic. This technical modification makes the proofs in the paper more readable.

2.1 Databases and Queries

Definition 1. A *database signature* is a pair (S, C, ar), where

– the *set of relational symbols* S and the *set of constants* C are finite disjoint sets of natural numbers, and
– $ar : S \to \mathbb{N}$ is a function giving for each symbol in the set S its *arity*.

Definition 2. A *(relational) database* of the signature $\sigma = (S, C, ar)$ is a tuple

$$\mathcal{B} = \langle |\mathcal{B}|, \sigma, [\![\cdot]\!]_{\mathcal{B}} \rangle$$

where

– the *carrier set* $|\mathcal{B}|$ is a finite set of natural numbers, and
– the *interpretation* $[\![\cdot]\!]_{\mathcal{B}}$ is a function giving for each $c \in C$ an element of $|\mathcal{B}|$ and for each $s \in S$ a relation on $|\mathcal{B}|$ of arity $ar(s)$.

The restriction that the set of symbols and the carrier set are sets of natural numbers is technical (we need it, for example in the Definition 4). Usually we will not respect it and name symbols and carrier set elements in more convenient ways.
The expression $\langle B \rangle$ denotes a database $\mathcal{B}$ with the carrier set B of the signature $(\emptyset, C, [\![\cdot]\!]_{\mathcal{B}})$, such that $C = B$ and for every $c \in C$ we have $[\![c]\!]_{\mathcal{B}} = c$.
Any database can be extended by adding to it a new symbol with its interpretation. Formally we define it in the following way.

Definition 3. An *extension of a database* $\mathcal{B} = \langle |\mathcal{B}|, (S, C, ar), [\![\cdot]\!]_{\mathcal{B}} \rangle$ *with a symbol* $\boldsymbol{T}$ *with an interpretation* U *and arity* a is the database:

$$\mathcal{B}' = \langle |\mathcal{B}|, (S \cup \{\boldsymbol{T}\}, C, ar'), [\![\cdot]\!]_{\mathcal{B}'} \rangle$$

where $[\![\cdot]\!]_{\mathcal{B}'}$ is equal to the function $[\![\cdot]\!]_{\mathcal{B}}$ on the set $S \setminus \{\boldsymbol{T}\}$, and on the argument $\boldsymbol{T}$ it is equal to U, and similarly, functions ar' and ar are equal on $S \setminus \{\boldsymbol{T}\}$ and $ar'(\boldsymbol{T}) = a$. Such an extension will be denoted by $\mathcal{B}[U/\boldsymbol{T} : a]$. When the arity a is clear from the context we will use an abbreviation writing $B[U/\boldsymbol{T}]$ instead of $B[U/\boldsymbol{T} : a]$.

Definition 4. For every database signature σ and every natural number n any function

$$Q : \{\mathcal{B} : \mathcal{B} \text{ is under signature } \sigma\} \to \mathcal{P}(\mathbb{N}^n)$$

such that $Q(\mathcal{B}) \subseteq |\mathcal{B}|^n$ will be called a *database query of a signature* $\sigma \to n$.

Definition 5. A *query language* is a set of expressions $\mathcal{L}$ together with a function Q giving for every $e \in \mathcal{L}$ a query Q_e.

Because in the paper we study complexity issues we make here the formal assumption that, together with a language, we are given a standard way of encoding its elements. We also fix some standard way of encoding databases. We skip here the details.

2.2 Logics as Query Languages

In [Var95] Vardi studied various logical languages (interpreted in relational structures) considered as query languages. Every formula φ (without function symbols) of any such logic $\mathcal{L}$ induces a query defined in the following way. Suppose that the free first-order variables of φ belong to the set $\{x_1, \ldots, x_i\}$, and the free second-order variables (with arity given by the function ar) belong to the set S. Any database $\mathcal{B}$ of the signature (S, C, ar) can be considered as a relational structure. For such a database the expression $e = \langle (x_1, \ldots, x_l)\varphi, (S, ar) \rangle$ defines the query:

$$Q_e(\mathcal{B}) = \{(b_1, \ldots, b_l) \in |\mathcal{B}|^l : \mathcal{B}, [b_1/x_1, \ldots, b_l/x_l] \models_{\mathcal{L}} \varphi\}$$

of arity $(S, C, ar) \to l$. (Where $[b_1/x_1, \ldots, b_l/x_l]$ denotes a valuation v of first-order variables such that for each x_i we have $v(x_i) = b_i$.)

The following abbreviation is useful. We often write $\mathcal{B}, \alpha \models \varphi(\mathbf{t})$ instead of $\mathcal{B}, \alpha[\mathbf{t}/\mathbf{x}] \models \varphi(x)$ (where $\mathbf{x}$ is a tuple of variables, $\mathbf{t}$ is a tuple of the carrier set elements and $\alpha[\mathbf{t}/\mathbf{x}]$ denotes the valuation equal to α outside the variables $\mathbf{x}$ and such that for every x_i we have that $\alpha(x_i) = t_i$).

2.3 Complexity

In this paper we study the complexity of queries. We are interested in comparing the complexity of queries expressible in different logical languages. Because queries are functions, we translate our task to the decision problem: given a tuple $\mathbf{t}$, an expression e and a database $\mathcal{B}$, does $\mathbf{t} \in Q_e(\mathcal{B})$ [2] hold ? Generally this problem has 3 parameters: $\mathbf{t}$, e and $\mathcal{B}$. Here we focus on the following two instances:

– the database $\mathcal{B}$ and the language $\mathcal{L}$ are fixed, we measure the complexity of the decision problem of the set:

$$Answer_{\mathcal{L}}(\mathcal{B}, \cdot, \cdot) = \{\langle \mathbf{t}, e\rangle \mid e \in \mathcal{L} \text{ and } \mathbf{t} \in Q_e(\mathcal{B})\}$$

– only the language $\mathcal{L}$ is fixed, we measure the complexity of the decision problem of the set:

$$Answer_{\mathcal{L}}(\cdot, \cdot, \cdot) = \{\langle \mathbf{t}, \mathcal{B}, e\rangle \mid e \in \mathcal{L} \text{ and } \mathbf{t} \in Q_e(\mathcal{B})\}.$$

The complexity of the problems $Answer_{\mathcal{L}}(\mathcal{B}, \cdot, \cdot)$ is called the *expression complexity*; we say that

– *the expression complexity of a language $\mathcal{L}$ belongs to a complexity class $\mathcal{C}$* iff for every $\mathcal{B}$, the $Answer_{\mathcal{L}}(\mathcal{B}, \cdot, \cdot)$ problem is in $\mathcal{C}$, and
– *the expression complexity of a language $\mathcal{L}$ is hard for a complexity class $\mathcal{C}$* iff there exists a database $\mathcal{B}$, such that the $Answer_{\mathcal{L}}(\mathcal{B}, \cdot, \cdot)$ problem is hard for $\mathcal{C}$.

The complexity of $Answer_{\mathcal{L}}(\cdot, \cdot, \cdot)$ is called the combined complexity of the language $\mathcal{L}$. Note that for every $\mathcal{B}$ we have that $Answer_{\mathcal{L}}(\mathcal{B}, \cdot, \cdot)$ is reducible in PTIME to $Answer_{\mathcal{L}}(\cdot, \cdot, \cdot)$, but there need not be any $\mathcal{B}$ such that the converse holds. Therefore $Answer_{\mathcal{L}}(\cdot, \cdot, \cdot)$ is in general harder than $Answer_{\mathcal{L}}(\mathcal{B}, \cdot, \cdot)$. Consequently the expression complexity of $\mathcal{L}$ is not higher than its combined complexity.

3 Fixpoint First-Order Logic

The FP language is an extension of the standard first-order logic with two dual fixpoint operators: μ and ν, denoting the least and the greatest fixpoint, respectively [CH82]. The syntax is extended in the following way. For an arbitrary FP formula φ and any second-order l-ary variable $\boldsymbol{V}$ appearing positively in it (i.e. not

[2] From here on, $\mathbf{t}$ and $\mathcal{B}$ are understood to be of the correct arity.

appearing under negation), the expressions: $(\mu \boldsymbol{V}(x_1,\ldots,x_l).\varphi)(y_1,\ldots,y_l)$ and $(\nu \boldsymbol{V}(x_1,\ldots,x_l).\varphi)(y_1,\ldots,y_l)$ (where $x_1,\ldots x_l$ are distinct first order variables and $y_1,\ldots,y_l$ are first order variables or constants) are formulas. Note that we allow nesting of fixpoint operators.
The set $\mathrm{free}_1(\varphi)$ of free first-order variables in φ is defined as in the standard first-order logic. For the fixpoint formulas we have:

$$\mathrm{free}_1(\theta \boldsymbol{V}(x_1,\ldots,x_l).\varphi)(y_1,\ \ldots,y_l) = (\mathrm{free}_1(\varphi)/\{x_1,\ldots,x_l\}) \cup \{y_1,\cdots,y_l\}.$$

The set $\mathrm{free}_2(\varphi)$ of free second-order variables in φ is also defined in the standard way (second-order variables are bound by μ and ν operators).
In the sequel we interpret FP formulas in databases such that for each free second-order variable in a formula there is a corresponding relation in a database. The semantics of FP should be rather clear, except for the semantics of the fixpoint subformulas. Consider a subformula $(\mu \boldsymbol{V}(x_1,\ldots,x_l).\varphi)(y_1,\ldots,y_l)$ in a database $\mathcal{B}$. We can treat a second-order variable $\boldsymbol{V}$ in the formula φ as a relational symbol and evaluate this formula in a database $\mathcal{B}$ extended with this symbol. In this way, after fixing a valuation α of the free first-order variables in φ we get an operator on l-ary relations, defined in the following way:

$$\widehat{\varphi}(U) = \{(\boldsymbol{z_1},\ldots,\boldsymbol{z_l}) : \mathcal{B}[U/\boldsymbol{V}], \alpha[\boldsymbol{z_1}/y_1,\ldots,\boldsymbol{z_1}/y_l] \models_{\mathsf{FP}} \varphi\}$$

Note that φ may contain free variables other than $y_1,\ldots,y_l$. Because the variable $\boldsymbol{V}$ occurs in the formula φ positively, the operator $\widehat{\varphi}$ is monotone. Thus by the Knaster-Tarski theorem [Tar55] there exists the least fixpoint of this operator, equal to the sum of the following sequence:

$$\emptyset \subseteq \widehat{\varphi}(\emptyset) \subseteq \widehat{\varphi}(\widehat{\varphi}(\emptyset)) \subseteq \cdots \tag{1}$$

Denote this sum by $\widehat{\varphi}^{\infty}$. In this way, for a given valuation α, the expression $\mu \boldsymbol{V}(y_1,\ldots,y_l).\varphi$ in the database $\mathcal{B}$ can be understood as an l-ary predicate. We can now define the semantics of μ-formulas as follows:

$$\mathcal{B}, \alpha \models_{\mathsf{FP}} (\mu \boldsymbol{V}(y_1,\ldots,y_l).\varphi)(x_1,\ldots,x_l) \ \text{ iff } \ (v(x_1),\ldots,v(x_l)) \in \widehat{\varphi}^{\infty}$$

(where v is equal to α on the set of variables and equal to $[\![\cdot]\!]$ on the set of constants). Similarly we define the semantics of $\nu S(\mathbf{x}).\varphi(\mathbf{t})$ as the *greatest* fixpoint of $\widehat{\varphi}$ (in this case we take the intersection of the sequence $|\mathcal{B}|^{|\mathbf{x}|} \supseteq \widehat{\varphi}(|\mathcal{B}|^{|\mathbf{x}|}) \supseteq \widehat{\varphi}(\widehat{\varphi}(|\mathcal{B}|^{|\mathbf{x}|})) \supseteq \cdots$).
If a formula φ has no free variables, we often write $\mathcal{B} \models_{\mathsf{FP}} \varphi$ instead of $\mathcal{B}, \alpha \models_{\mathsf{FP}} \varphi$. The following lemma is useful.

Lemma 6. *Let $\mathcal{B}$ be an arbitrary database, α an arbitrary valuation and $\varphi(\mathbf{x})$ an arbitrary formula. Then the set $\mathcal{V} = \{\mathbf{t} : \mathcal{B}, \alpha \models_{\mathsf{FP}} (\mu \boldsymbol{V}(\mathbf{x}).\varphi(\mathbf{x}))(\mathbf{t})\}$ is equal to*

$$\{\mathbf{t} : \mathcal{B}[\mathcal{V}/\boldsymbol{V}], \alpha \models_{\mathsf{FP}} \varphi(\mathbf{t})\} \tag{2}$$

(where $\mathbf{t}$ denotes a tuple of the carrier set elements, and $\mathbf{x}$ a tuple of the first-order variables, such, that $|\mathbf{t}| = |\mathbf{x}|$).

Proof. Let $\widehat{\varphi}$ be an operator induced by $\varphi(\mathbf{x})$. We have that $\mathcal{V} = \widehat{\varphi}^{\infty} = \widehat{\varphi}(\widehat{\varphi}^{\infty}) = \widehat{\varphi}(\mathcal{V})$, which is equal to (2). □

4 The Complexity of FP^k

In [Var95] Vardi considers several languages $\mathcal{L}^k$, each one obtained from a certain language $\mathcal{L}$ by restricting the set of individual variables allowed in the formulas to $\{x_1, \ldots, x_k\}$. Note that this does not bound the quantifier depth since variables may be reused. In this way we get FP^k from FP. Below we show that there exists a database $\mathcal{B}$ such that $\mathit{Answer}_{\mathsf{FP}^2}(\mathcal{B}, \cdot, \cdot)$ is PSPACE-hard and consequently, that $\mathit{Answer}_{\mathsf{FP}^2}(\cdot, \cdot, \cdot)$ is PSPACE-hard. This is the best lower bound because for every k the straightforward algorithm for $\mathit{Answer}_{\mathsf{FP}^k}(\cdot, \cdot, \cdot)$ operates in PSPACE and consequently for every $\mathcal{B}$ we have that $\mathit{Answer}_{\mathsf{FP}^k}(\mathcal{B}, \cdot, \cdot)$ is in PSPACE.

Theorem 7. *There exists a database $\mathcal{B}$ such that $\mathit{Answer}_{\mathsf{FP}^2}(\mathcal{B}, \cdot, \cdot)$ is PSPACE-complete.*

Proof. From the previous remarks it is sufficient to show PSPACE-hardness. The proof goes by the reduction of the *Quantified Boolean Formulas* (QBF) problem. Recall that the syntax of QBF is given by the following grammar.

$$\mathtt{F} ::= \mathtt{F} \land \mathtt{F} \mid \mathtt{F} \lor \mathtt{F} \mid \forall x_i.\ \mathtt{F} \mid \exists x_i.\ \mathtt{F} \mid x_i \mid \neg x_i$$

The variables take the boolean values T and F. We can assume that each x_i is quantified only once. The QBF problem is the set $\{\psi : \psi \text{ is closed and } \models_{\mathsf{QBF}} \psi\}$. At first sight, one may think that we can reduce QBF to FP^k by taking a database $\mathcal{B} = \langle\{\mathsf{T}, \mathsf{F}\}\rangle$ and translating a given QBF formula into a first-order formula by translating every x_i to $(x_i = \mathsf{T})$ and $\neg x_i$ to $(x_i = \mathsf{F})$. This attempt is not satisfactory however, because the number of variables cannot be bounded. Therefore we have to use a more sophisticated method and make use of fixpoint operators. Let $\mathcal{B} = \langle\{\mathsf{T}, \mathsf{F}, \mathsf{A}\}\rangle$. A function ξ transforming QBF formulas to FP^2 formulas is defined by structural induction.

$$\begin{aligned}
\xi(\psi_1 \land \psi_2) &= \xi(\psi_1) \land \xi(\psi_2) \\
\xi(\psi_1 \lor \psi_2) &= \xi(\psi_1) \lor \xi(\psi_2) \\
\xi(\forall x_i.\psi) &= \forall y \in \{\mathsf{T}, \mathsf{F}\}.\ \mu \boldsymbol{V_i}(x).\left(\bigvee \begin{matrix} x \neq \mathsf{A} \land x = y \\ x = \mathsf{A} \land \xi(\psi) \end{matrix}\right)(\mathsf{A}) \\
\xi(\exists x_i.\psi) &= \exists y \in \{\mathsf{T}, \mathsf{F}\}.\ \mu \boldsymbol{V_i}(x).\left(\bigvee \begin{matrix} x \neq \mathsf{A} \land x = y \\ x = \mathsf{A} \land \xi(\psi) \end{matrix}\right)(\mathsf{A}) \\
\xi(x_i) &= \boldsymbol{V_i}(\mathsf{T}) \\
\xi(\neg x_i) &= \boldsymbol{V_i}(\mathsf{F})
\end{aligned}$$

Note that in this construction there is a bijection between variables $x_1, \ldots, x_n$ of a QBF formula and $\boldsymbol{V_1}, \ldots, \boldsymbol{V_n}$ of an FP formula. In formulas $\mu\boldsymbol{V_i}\cdots$ there is a free variable y, so the least fixpoint defined by this formula depends on the actual value of y. If it is T then in the first iteration T enters the least fixpoint (F will never enter). This information is further used in the evaluation of subformulas $\boldsymbol{V_i}(\mathsf{T})$ and $\boldsymbol{V_i}(\mathsf{F})$. The proof of the following lemma completes the proof of the theorem (note that for any closed FP formula φ we have that $\langle\langle\rangle, ()\varphi\rangle \in \mathit{Answer}(\mathcal{B}, \cdot, \cdot)$ iff $\mathcal{B} \models_{\mathsf{FP}} \varphi$).

Lemma 8. $\models_{\mathsf{QBF}} \psi$ *iff* $\mathcal{B} \models_{\mathsf{FP}} \xi(\psi)$.

Proof. The proof goes by induction on the structure of the QBF formula. We show the following more general fact. Let $\psi(x_1, \ldots, x_l)$ be an arbitrary QBF formula. Then

$$\begin{array}{l}\text{for every valuation } v \text{ of the variables } x_1, \ldots, x_l \\ v \models_{\mathsf{QBF}} \psi \text{ iff } \mathcal{T}^{\psi}(v) \models_{\mathsf{FP}} \xi(\psi)\end{array} \tag{3}$$

where $\mathcal{T}^{\psi}(v) = \mathcal{B}[\{v(x_1)\}/\boldsymbol{V_1}, \ldots, \{v(x_l)\}/\boldsymbol{V_l}]$ is the database $\mathcal{B}$ extended with the unary symbols $\boldsymbol{V_1}, \ldots, \boldsymbol{V_l}$.

1. The proof of the hypothesis for the literals x_i and $\neg x_i$ is easy. We show it for $\neg x_i$. Let us fix a valuation v. If $v \models_{\mathsf{QBF}} \neg x_i$ then $v(x_i) = \mathsf{F}$. Thus $[\![\boldsymbol{V_i}]\!]_{\mathcal{T}^{\psi}(v)} = \{\mathsf{F}\}$. Therefore $\mathcal{T}^{\psi}(v) \models_{\mathsf{FP}} \xi(\neg x_i)$. On the other hand, if $v \not\models_{\mathsf{QBF}} \neg x_i$ then $v(x_i) = \mathsf{T}$. Thus $[\![\boldsymbol{V_i}]\!]_{\mathcal{T}^{\psi}(v)} = \{\mathsf{T}\}$. Therefore $\mathcal{T}^{\psi}(v) \not\models_{\mathsf{FP}} \xi(\neg x_i)$.
2. To prove the hypothesis for $\wedge$ let us fix a valuation v and QBF formulas ψ_1 and ψ_2. Now:

$$v \models_{\mathsf{QBF}} \psi_1 \wedge \psi_2 \ \text{ iff } \ v \models_{\mathsf{QBF}} \psi_1 \text{ and } v \models_{\mathsf{QBF}} \psi_2 \tag{4}$$

$$\text{(by the induction hypothesis) iff } \left(\text{and } \begin{array}{l} \mathcal{T}^{\psi_1}(v) \models_{\mathsf{FP}} \xi(\psi_1) \\ \mathcal{T}^{\psi_2}(v) \models_{\mathsf{FP}} \xi(\psi_2) \end{array} \right) \tag{5}$$

   This, after extending $\mathcal{T}^{\psi_1}(v)$ and $\mathcal{T}^{\psi_2}(v)$ to $\mathcal{T}^{\psi_1 \wedge \psi_2}(v)$ is equivalent to

$$\mathcal{T}^{\psi_1 \wedge \psi_2}(v) \models_{\mathsf{FP}} \xi(\psi_1) \text{ and } \mathcal{T}^{\psi_2 \wedge \psi_2}(v) \models_{\mathsf{FP}} \xi(\psi_2)$$

   which holds if and only if $\mathcal{T}^{\psi_1 \wedge \psi_2}(v) \models_{\mathsf{FP}} \xi(\psi_1 \wedge \psi_2)$. The proof for $\vee$ is analogous.
3. The case of quantifiers is the most interesting one. Let us consider the universal quantifier (the proof for existential quantifier is similar). Take a QBF formula $\forall x_i.\psi$ and an arbitrary valuation v of its free variables. Suppose (3) holds for ψ. From the definition of ξ we get:

$$\xi(\forall x_i.\psi) = \forall y \in \{\mathsf{T}, \mathsf{F}\}.(\mu \boldsymbol{V_i}(x).\varphi(x, y))(\mathsf{A}) \tag{6}$$

   where

$$\varphi(x, y) = \left(\vee \begin{array}{c} x \neq \mathsf{A} \wedge x = y \\ x = \mathsf{A} \wedge \xi(\psi) \end{array} \right). \tag{7}$$

   The formula $(\mu \boldsymbol{V_i}(x).\varphi(x, y))(\mathsf{A})$ has exactly one free variable: y. Now we prove that

$$\begin{array}{l}\text{for every value } \boldsymbol{p} \in \{\mathsf{T}, \mathsf{F}\} \\ \mathcal{T}^{\psi}(v) \models_{\mathsf{FP}} (\mu \boldsymbol{V_i}(x).\varphi(x, \boldsymbol{p}))(\mathsf{A}) \ \text{ iff } \ v[\boldsymbol{p}/x_i] \models_{\mathsf{QBF}} \psi.\end{array} \tag{8}$$

   This fact easily implies the induction hypothesis, because:

$$\begin{array}{l} \mathcal{T}^{\psi}(v) \models_{\mathsf{FP}} \forall y \in \{\mathsf{T}, \mathsf{F}\}.(\mu \boldsymbol{V_i}(x).\varphi(x, y))(\mathsf{A}) \\ \text{iff for every } \boldsymbol{p} \in \{\mathsf{T}, \mathsf{F}\} \text{ we have } \mathcal{T}^{\psi}(v) \models_{\mathsf{FP}} (\mu \boldsymbol{V_i}(x).\varphi(x, \boldsymbol{p}))(\mathsf{A}) \\ \text{iff (from (8)) for every } \boldsymbol{p} \in \{\mathsf{T}, \mathsf{F}\} \text{ we have } v[\boldsymbol{p}/x_i] \models_{\mathsf{QBF}} \psi \\ \text{iff } v \models_{\mathsf{QBF}} \forall x_i.\psi \end{array}$$

Let us now prove (8). Take an arbitrary $\boldsymbol{p} \in \{\mathsf{T}, \mathsf{F}\}$ as a value for y. We have that

$$\mathcal{T}^{\psi}(v) \models_{\mathsf{FP}} (\mu \boldsymbol{V_i}(x).\varphi(x, \boldsymbol{p}))(\mathsf{A}) \tag{9}$$

is equivalent to

$$\mathsf{A} \in \mathcal{V} \tag{10}$$

where $\mathcal{V}$ denotes the least fixpoint:

$$\mathcal{V} = \{\boldsymbol{r} : \mathcal{T}^{\psi}(v) \models_{\mathsf{FP}} (\mu \boldsymbol{V_i}(x).\varphi(x, \boldsymbol{p}))(\boldsymbol{r})\}. \tag{11}$$

Now, applying the Lemma 6 to the formula in (11), we get

$$\mathcal{V} = \{\boldsymbol{r} : \mathcal{T}^{\psi}(v)[\mathcal{V}/\boldsymbol{V_i}] \models_{\mathsf{FP}} \varphi(\boldsymbol{r}, \boldsymbol{p})\} \tag{12}$$

This equation is sufficient for determining the value of $\mathcal{V}$. We are mostly interested in the value of this predicate on A. However, to evaluate it we have to know the value of $\mathcal{V}$ on the set $\{\mathsf{T}, \mathsf{F}\}$. Take an arbitrary $\boldsymbol{q} \in \{\mathsf{T}, \mathsf{F}\}$. By (12) we have that $\boldsymbol{q} \in \mathcal{V}$ if and only if $\mathcal{T}^{\psi}(v)[\mathcal{V}/\boldsymbol{V_i}] \models_{\mathsf{FP}} \varphi(\boldsymbol{q}, \boldsymbol{p})$ which, by the definition of φ (7), is equivalent to

$$\mathcal{T}^{\psi}(v)[\mathcal{V}/\boldsymbol{V_i}] \models_{\mathsf{FP}} \left(\vee \begin{matrix} \boldsymbol{q} \neq \mathsf{A} \wedge \boldsymbol{q} = \boldsymbol{p} \\ \boldsymbol{q} = \mathsf{A} \wedge \xi(\psi) \end{matrix} \right) \tag{13}$$

Because $\boldsymbol{q} \neq \mathsf{A}$ we have that (13) is equivalent to $\mathcal{T}^{\psi}(v)[\mathcal{V}/\boldsymbol{V_i}] \models_{\mathsf{FP}} (\boldsymbol{q} = \boldsymbol{p})$. Thus:

$$\text{every } \boldsymbol{q} \in \{\mathsf{T}, \mathsf{F}\} \text{ belongs to } \mathcal{V} \text{ if and only if } \boldsymbol{q} = \boldsymbol{p}. \tag{14}$$

Let us now evaluate the value of $\mathcal{V}$ on A, (i.e. evaluate (10)). Note that in the formula φ the variable $\boldsymbol{V_i}$ occurs only in the subformulas of the form $\boldsymbol{V_i}(\mathsf{T})$ and $\boldsymbol{V_i}(\mathsf{F})$. Thus, after applying (14) to the equation (12), we get that the set $\mathcal{V}$ is equal to $\{\boldsymbol{r} : \mathcal{T}^{\psi}(v)[\{\boldsymbol{p}\}/\boldsymbol{V_i}] \models_{\mathsf{FP}} \varphi(\boldsymbol{r}, \boldsymbol{p})\}$. By the definition of $\mathcal{T}^{\psi}$ we have $\mathcal{T}^{\psi}(v)[\{\boldsymbol{p}\}/\boldsymbol{V_i}] = \mathcal{T}^{\psi}(v[\boldsymbol{p}/x_i])$, thus by the definition of φ we get that (10) holds if and only if

$$\mathcal{T}^{\psi}(v[\boldsymbol{p}/x_i]) \models_{\mathsf{FP}} \left(\vee \begin{matrix} \mathsf{A} \neq \mathsf{A} \wedge \mathsf{A} = \boldsymbol{p} \\ \mathsf{A} = \mathsf{A} \wedge \xi(\psi) \end{matrix} \right) \tag{15}$$

which is equivalent to

$$\mathcal{T}^{\psi}(v[\boldsymbol{p}/x_i]) \models_{\mathsf{FP}} \xi(\psi) \tag{16}$$

Now by the induction hypothesis (16) is equivalent to $v[\boldsymbol{p}/x_i] \models_{\mathsf{QBF}} \psi$ This observation completes the proof. □

5 Complexity of restrictions of FP^k

In this section we study the complexity of queries over languages obtained from FP^k by various restrictions of the syntax.

5.1 Bounded Number of Second-Order Variables

One can ask whether the bijection between $\boldsymbol{V_i}$'s and x_i's in the proof of the Theorem 7 is essential, i.e. whether it is possible to prove PSPACE-hardness when we restrict also the number of second-order variables. In this section we show that the answer is positive when we ask about the combined complexity, but is negative for the expression complexity.
Let FP^k_n be a sublogic of FP^k, such that the number of first-order variables in its formulas is bounded by k and the number of second-order variables in its formulas is bounded by n.

Combined Complexity. We prove now the following

Theorem 9. *For every $k \geq 3$, $n \geq 2$ the $Answer_{\mathsf{FP}^k_n}(\cdot,\cdot,\cdot)$ problem is PSPACE-complete.*

Proof. The membership in PSPACE is trivial. It remains to prove the PSPACE-hardness of $Answer_{\mathsf{FP}^3_2}$. Let us fix a QBF formula ψ (with variables $x_1,\ldots,x_n$). We show how to construct in polynomial time a pair $\langle$ database $\mathcal{B}_\psi$, and an FP^3_2 formula $\rho(\psi)$ $\rangle$, such that:

$$\models_{\mathsf{QBF}} \psi \quad \text{iff} \quad \mathcal{B}_\psi \models_{\mathsf{FP}} \rho(\psi) \tag{17}$$

In the construction we define a database in such a way that we are able to remember the valuation, representing it by relations in $\mathcal{B}$. Let

$$\mathcal{B}_\psi = \langle\{1,\ldots,n,\mathsf{T},\mathsf{F},\mathsf{A}\}\rangle\ [\{1,\ldots,n\}/\boldsymbol{Var},\ \emptyset/\boldsymbol{V_0}]$$

Below we present a function ρ that transforms the formula ψ to a FP^k_n formula, proceeding top-down. In φ we use binary relation variables $\boldsymbol{V_i}$. Then we show how to reduce the number of $\boldsymbol{V_i}$'s to 2. States in the database $\mathcal{B}_\psi$ (all, except A) are used to remember the valuation in the following way. The subformulas (c.f. definition below) $\mu\boldsymbol{V_{d+1}}\cdots$ represent the valuation of the propositional variables $x_1,\ldots,x_n$ in the sense that $\boldsymbol{V_d}(i,\mathsf{T})$ holds iff the value of x_i is T and $\boldsymbol{V_d}(i,\mathsf{F})$ holds iff the value of x_i is F. Note that the actual value of such subformulas depends on the value of variables that occur free in this formula. In this way, using fix-point formulas we are be able to capture the whole tree of possible valuations. We define a transformation $\rho(\psi) = \xi(\psi, 0)$ using an auxiliary function

$\xi(\psi : \mathsf{QBF}$ formulas, $d : \mathbb{N})$ which is defined inductively by the following clauses.

$$
\begin{aligned}
\xi(\psi_1 \wedge \psi_2, d) &= \xi(\psi_1, d) \wedge \xi(\psi_2, d) \\
\xi(\psi_1 \vee \psi_2, d) &= \xi(\psi_1, d) \vee \xi(\psi_2, d) \\
\xi(\forall x_i.\psi, d) &= \forall y \in \{\mathsf{T}, \mathsf{F}\}. \\
&\qquad \mu \boldsymbol{V_{d+1}}(x, z). \left(\vee \begin{array}{ccc} \boldsymbol{Var}(x) & \wedge & \left(\vee \begin{array}{c} x = i \wedge z = y \\ x \neq i \wedge \boldsymbol{V_d}(x, z) \end{array} \right) \\ x = \mathsf{A} & \wedge & \xi(\psi, d+1) \end{array} \right) (\mathsf{A}, 0) \\
\xi(\exists x_i.\psi, d) &= \exists y \in \{\mathsf{T}, \mathsf{F}\}. \\
&\qquad \mu \boldsymbol{V_{d+1}}(x, z). \left(\vee \begin{array}{ccc} \boldsymbol{Var}(x) & \wedge & \left(\vee \begin{array}{c} x = i \wedge z = y \\ x \neq i \wedge \boldsymbol{V_d}(x, z) \end{array} \right) \\ x = \mathsf{A} & \wedge & \xi(\psi, d+1) \end{array} \right) (\mathsf{A}, 0) \\
\xi(x_i, d) &= \boldsymbol{V_d}(i, \mathsf{T}) \\
\xi(\neg x_i, d) &= \boldsymbol{V_d}(i, \mathsf{F})
\end{aligned}
$$

We claim that (17) holds. The proof is similar to the one in Section 4. We skip it, showing only the induction hypothesis:

For every QBF formula ψ with free variables $x_1, \ldots, x_n$
every valuation v of these variables and every $d \in \mathbb{N}$ we have that (18)
$\alpha \models_{\mathsf{QBF}} \psi$ iff $\mathcal{B}_\psi[\{(i, \alpha(x_i)) : i = 1, \ldots, n\}/\boldsymbol{V_d}] \models_{\mathsf{FP}} \xi(\psi, d)$.

Now observe that in each subformula of $\rho(\psi)$ of the form $\mu \boldsymbol{V_{d+1}}(x, z).\varphi$, we have that $\mathrm{free}_2(\varphi) \cap \{\boldsymbol{V_1}, \ldots\} = \{\boldsymbol{V_d}\}$, therefore all other $\boldsymbol{V_i}$'s "are not visible" from φ and can be reused. Formally we do it by gluing all $\boldsymbol{V_{2n}}$'s into one variable and all $\boldsymbol{V_{2n+1}}$'s into another one. □

Expression Complexity. In the above construction the size of the database is not bounded, and can be even linear in the size of the given QBF formula. Below we argue that the linear lower bound is essential by showing the upper bound for the expression complexity. The proof is similar to the proof establishing the complexity of $Answer_{\mathsf{FO}^k}$ in [Var95]. The key observation is that for a fixed database, and a fixed number of the first- and second-order variables, the arity of all relations is fixed too, and thus the number of all definable relations (and relations on relations) is bounded. This gives us PTIME as an easy upper bound. We can improve it however by using a technique from [Lyn77]. Recall that a *parenthesis grammar* is a context-free grammar with two distinguished terminals: "(" and ")" such that each production is of the form $A \to (x)$ with x parenthesis free. Such a grammar generates a *parenthesis language.* In our proof we make use of the the fact from [Bus87] that all parenthesis languages are recognisable in ALOGTIME.

Theorem 10. *The* $Answer_{\mathsf{FP}^k_n}(\mathcal{B}, \cdot, \cdot)$ *problem is in ALOGTIME for every database* $\mathcal{B}$ *every* k *and every* n*.*

Proof sketch. Fix a database $\mathcal{B}$, a number k of the first-order variables and a number n of the second-order variables. We show how to construct a parenthesis grammar G, such that for every formula $\varphi \in \mathsf{FP}_n^k$ and for every tuple $\mathbf{t}$ of the length l

$$\mathcal{B}, \mathbf{t} \models_{\mathsf{FP}} (y_1, \ldots, y_l)\varphi \ \text{ iff } \ (\mathbf{t} \models_{\mathsf{FP}} (y_1, \ldots, y_l)\varphi) \in L(G). \tag{19}$$

Suppose that $x_1, \ldots, x_k$ are first order variables and $\boldsymbol{V_1}, \ldots, \boldsymbol{V_n}$ are second-order variables (each of arity $ar(\boldsymbol{V_i})$). Let $\mathcal{B} = \{|\mathcal{B}|, \sigma, [\![\cdot]\!]\}$ be an arbitrary database. Moreover let Val be the set of all valuations of the variables $x_1, \ldots, x_k$, let B be the set of all extensions of the database $\mathcal{B}$ with symbols $\boldsymbol{V_1}, \ldots, \boldsymbol{V_n}$, and let $\mathcal{P}(B \times Val) = \{T_1, \ldots, T_l\}$. The grammar consists of

- the initial symbol S and a set of nonterminal symbols: $\{T_1, \ldots, T_l\}$
- a set of terminal symbols: all first-order variables, all tuples of them, all second order variables and all symbols in $\mathcal{B}$, $\neg, \wedge, \vee, (,), \models_{\mathsf{FP}}, ., \mu, \nu, \exists$ and $\forall$
- productions:
  For every T_i, T_{i_1}, T_{i_2}, every $(\mathcal{D}, v) \in T_i$ every two variables x and y and every two tuples of variables $\mathbf{x}$ and $\mathbf{y}$ we have:

$$\begin{array}{lll}
S \to (\mathbf{t} \models_{\mathsf{FP}} (y_1, \ldots, y_l)T_i) & \text{where} & \mathbf{t} = ((v(y_1), \ldots, v(y_l))) \\
T_i \to (x = y) & \text{iff} & v(x) = v(y) \\
T_i \to (x \neq y) & \text{iff} & v(x) \neq v(y) \\
T_i \to (\boldsymbol{R_j}(\mathbf{x})) & \text{iff} & T_i = \{(\mathcal{D}, v) : \mathcal{D}, v \models_{\mathsf{FP}} (\boldsymbol{R_j}(\mathbf{x})\} \\
T_i \to (\neg \boldsymbol{R_j}(\mathbf{x})) & \text{iff} & T_i = \{(\mathcal{D}, v) : \mathcal{D}, v \not\models_{\mathsf{FP}} (\boldsymbol{R_j}(\mathbf{x})\} \\
T_i \to (\boldsymbol{V_j}(\mathbf{x})) & \text{iff} & T_i = \{(\mathcal{D}, v) : \mathcal{D}, v \models_{\mathsf{FP}} (\boldsymbol{V_j}(\mathbf{x})\} \\
T_i \to (T_{i_1} \wedge T_{i_2}) & \text{iff} & T_i = T_{i_1} \cap T_{i_2} \\
T_i \to (T_{i_1} \vee T_{i_2}) & \text{iff} & T_i = T_{i_1} \cup T_{i_2} \\
T_i \to (\exists y.T_h) & \text{iff} & T_i = \{(\mathcal{D}, v) : \exists \boldsymbol{p}.\ (\mathcal{D}, v[\boldsymbol{p}/y]) \in T_h\} \\
T_i \to (\forall y.T_h) & \text{iff} & T_i = \{(\mathcal{D}, v) : \forall \boldsymbol{p}.\ (\mathcal{D}, v[\boldsymbol{p}/y]) \in T_h\}
\end{array}$$

  The most interesting is the case of the fixpoint formulas. Note that for every nonterminal symbol T_h and every tuple $\mathbf{x}$ of variables we have the following operator on relations on $|\mathcal{B}|$.

$$\rho_{T_h, \mathbf{x}}(\mathcal{V}) = \{\mathbf{z} : (\mathcal{D}[\mathcal{V}/\boldsymbol{V_j}], v[\mathbf{z}/\mathbf{x}]) \in T_h\}$$

  If the operator $\rho_{T_h, \mathbf{x}}$ is monotone then it has the least fixpoint $\rho^{\infty}_{T_h, \mathbf{x}}$. For arbitrary tuples $\mathbf{y}$ and $\mathbf{z}$ (of the length $ar(\boldsymbol{V_j})$) of first-order variables we set

$$T_i \to ((\mu \boldsymbol{V_j}(\mathbf{x}).T_h)(\mathbf{y})) \text{ iff } T_i = \{(\mathcal{D}, v) : (v(y_1), \ldots, v(y_l)) \in \rho^{\infty}_{T_h, \mathbf{x}}\}.$$

  The production for the greatest fixpoint is defined similarly.

We skip here the formal proof of (19). $\square$

5.2 Prefix Form

Define $\overline{\mathsf{FP}}^k$ formulas as FP^k formulas of the form

prefix of fix-point operators . first-order formula . arguments

(where arguments can be variables or constants). In the $\overline{\mathsf{FP}}^k$ formulas we write $\mu\boldsymbol{V}(\mathbf{x})\leftarrow(\mathbf{y}).\varphi$ instead of $(\mu\boldsymbol{V}(\mathbf{x}).\varphi)(\mathbf{y})$ for the sake of readability. In this section we show that the expression and the combined complexities of $\overline{\mathsf{FP}}^k$ are PSPACE-complete. Namely we show the following.

Theorem 11. *There exists a database $\mathcal{B}$, such that the Answer$_{\overline{\mathsf{FP}}^k}(\mathcal{B},\cdot,\cdot)$ problem is PSPACE-complete.*

Proof sketch. It is sufficient to show PSPACE-hardness. Let

$$\mathcal{B} = \langle\{0, 1, \mathsf{Set}, \mathsf{T}, \mathsf{F}\}\rangle\ [\{1\}/\boldsymbol{W_{p_{-1}}}].$$

For a given QBF formula $\phi = \forall p_1 \exists p_2 \ldots \forall p_{n-1} \exists p_n.\varphi$ (where φ is a sentence which variables are among $p_1, \ldots, p_n$) we show how to construct in PTIME an $\overline{\mathsf{FP}}^k$ formula χ such that

$$\models_{\mathsf{QBF}} \phi \quad \text{iff} \quad \mathcal{B} \models_{\mathsf{FP}} \chi \tag{20}$$

This will complete the proof since the QBF problem remains PSPACE-complete, even when we restrict ourselves to formulas in the prefix form. The formula χ is complex. To define it we first define a function ξ by the following clauses.

$$\begin{array}{rclrcl} \xi(\varphi_1 \wedge \varphi_2) &=& \xi(\varphi_1) \wedge \xi(\varphi_2) & \xi(\neg p_i) &=& \boldsymbol{W_{p_i}}(\mathsf{F}) \\ \xi(\varphi_1 \vee \varphi_2) &=& \xi(\varphi_1) \vee \xi(\varphi_2) & \xi(p_i) &=& \boldsymbol{W_{p_i}}(\mathsf{T}) \end{array}$$

Then using it we define a formula $\mathcal{FORM}^i$ (Table 1). Finally we set $\chi = \mathcal{FORM}^0(\mathsf{Set}, \mathsf{T}, 0)$. Note that in subformulas of the form

$$\mu\boldsymbol{W_{p_i}}(x)\leftarrow(\boldsymbol{x}).(\mathcal{FORM}^{i+1}(x, \boldsymbol{y}, \boldsymbol{z})) \tag{21}$$

there are two parameters: $\boldsymbol{y}$ and $\boldsymbol{z}$. Thus the value of the fixpoints defined by them depends on the value of $\boldsymbol{y}$ and $\boldsymbol{z}$. Moreover it can be shown, by applying $2*(n-1)+1$ times Lemma 6 to (21), that every $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \{\mathsf{T}, \mathsf{F}, 0, 1\}$ satisfy (21) iff $\boldsymbol{x} = \boldsymbol{y}$ or $\boldsymbol{x} = \boldsymbol{z}$. In this way we are able remember valuations of QBF variables. We define $\mathcal{C}(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_i)$ to be an extension of the database $\mathcal{B}$ with symbols $\boldsymbol{W_{p_0}}, \ldots, \boldsymbol{W_{p_n}}$. The interpretation of these symbols is given in the following table (here we assume that i is even).

$\boldsymbol{W_{p_0}}$	$\boldsymbol{W_{p_1}}$	$\boldsymbol{W_{p_2}}$	$\boldsymbol{W_{p_3}}$	$\cdots\cdots$	$\boldsymbol{W_{p_{i-2}}}$	$\boldsymbol{W_{p_{i-1}}}$	$\boldsymbol{W_{p_i}}$	$\boldsymbol{W_{p_{i+1}}}$	$\cdots\cdots$	$\boldsymbol{W_{p_{n-1}}}$	$\boldsymbol{W_{p_n}}$
$0, \mathsf{T}$	$1, \boldsymbol{p}_1$	$0, \boldsymbol{p}_2$	$1, \boldsymbol{p}_3$		$0, \boldsymbol{p}_{i-2}$	$1, \boldsymbol{p}_{i-1}$	$0, \boldsymbol{p}_i$	$0, \boldsymbol{p}_i$		$0, \boldsymbol{p}_i$	$0, \boldsymbol{p}_i$

(in case i odd we have $\boldsymbol{W_{p_{i-2}}} = \{1, \boldsymbol{p}_{i-2}\}$, $\boldsymbol{W_{p_{i-1}}} = \{0, \boldsymbol{p}_{i-1}\}$ and $\boldsymbol{W_{p_i}} = \cdots = \boldsymbol{W_{p_n}} = \{1, \boldsymbol{p}_i\}$). In the proof we show by induction on i that for every odd $i = 1, \ldots, n$ we have that

$$\mathcal{B} \models_{\mathsf{FP}} \chi \quad \text{iff} \quad \forall \boldsymbol{p}_1.\exists \boldsymbol{p}_2.\ldots\exists \boldsymbol{p}_{i-1}.\forall \boldsymbol{p}_i.\mathcal{C}(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_i) \models_{\mathsf{FP}} \mathcal{FORM}^{i+1}(\mathsf{Set}, \boldsymbol{p}_i, 0)$$

$$
\begin{array}{rl}
\mathcal{FORM}^i(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = & \mu \boldsymbol{V}_{\boldsymbol{p_i}}\ (x,y,z)\leftarrow(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}).\ \mu \boldsymbol{W}_{\boldsymbol{p_i}}\ (x)\leftarrow(x). \\
& \vdots \\
& \mu \boldsymbol{V}_{\boldsymbol{p_n}}\ (x,y,z)\leftarrow(x,y,z).\ \mu \boldsymbol{W}_{\boldsymbol{p_n}}\ (x)\leftarrow(x). \\
& \\
& \quad (x = \mathsf{Set}\ \wedge\ \boldsymbol{W}_{\boldsymbol{p_{-1}}}(1) \wedge \boldsymbol{W}_{\boldsymbol{p_0}}(0) \wedge \boldsymbol{W}_{\boldsymbol{p_1}}(0) \\
& \qquad\qquad \wedge\ ((\boldsymbol{V}_{\boldsymbol{p_1}}(\mathsf{Set},\mathsf{T},1))\quad \wedge\ (\boldsymbol{V}_{\boldsymbol{p_1}}(\mathsf{Set},\mathsf{F},1)))) \\
& \vee\ (x = \mathsf{Set}\ \wedge\ \boldsymbol{W}_{\boldsymbol{p_0}}(0) \wedge \boldsymbol{W}_{\boldsymbol{p_1}}(1) \wedge \boldsymbol{W}_{\boldsymbol{p_2}}(1) \\
& \qquad\qquad \wedge\ ((\boldsymbol{V}_{\boldsymbol{p_2}}(\mathsf{Set},\mathsf{T},0))\quad \vee\ (\boldsymbol{V}_{\boldsymbol{p_2}}(\mathsf{Set},\mathsf{F},0)))) \\
& \vdots \\
& \vee\ (x = \mathsf{Set}\ \wedge\ \boldsymbol{W}_{\boldsymbol{p_l}}(0) \wedge \boldsymbol{W}_{\boldsymbol{p_{l+1}}}(1) \wedge \boldsymbol{W}_{\boldsymbol{p_{l+2}}}(1) \\
& \qquad\qquad \wedge\ ((\boldsymbol{V}_{\boldsymbol{p_{l+2}}}(\mathsf{Set},\mathsf{T},0))\ \vee\ (\boldsymbol{V}_{\boldsymbol{p_{l+2}}}(\mathsf{Set},\mathsf{F},0)))) \\
& \vee\ (x = \mathsf{Set}\ \wedge\ \boldsymbol{W}_{\boldsymbol{p_{l+1}}}(1) \wedge \boldsymbol{W}_{\boldsymbol{p_{l+2}}}(0) \wedge \boldsymbol{W}_{\boldsymbol{p_{l+3}}}(0) \\
& \qquad\qquad \wedge\ ((\boldsymbol{V}_{\boldsymbol{p_{l+3}}}(\mathsf{Set},\mathsf{T},1))\ \wedge\ (\boldsymbol{V}_{\boldsymbol{p_{l+3}}}(\mathsf{Set},\mathsf{F},1)))) \\
& \vdots \\
& \vee\ (x = \mathsf{Set}\ \wedge\ \boldsymbol{W}_{\boldsymbol{p_{n-3}}}(0) \wedge \boldsymbol{W}_{\boldsymbol{p_{n-2}}}(1) \wedge \boldsymbol{W}_{\boldsymbol{p_{n-1}}}(1) \\
& \qquad\qquad \wedge\ ((\boldsymbol{V}_{\boldsymbol{p_{n-1}}}(\mathsf{Set},\mathsf{T},0))\ \vee\ (\boldsymbol{V}_{\boldsymbol{p_{n-1}}}(\mathsf{Set},\mathsf{F},0)))) \\
& \vee\ (x = \mathsf{Set}\ \wedge\ \boldsymbol{W}_{\boldsymbol{p_{n-2}}}(1) \wedge \boldsymbol{W}_{\boldsymbol{p_{n-1}}}(0) \wedge \boldsymbol{W}_{\boldsymbol{p_n}}(0) \\
& \qquad\qquad \wedge\ ((\boldsymbol{V}_{\boldsymbol{p_n}}(\mathsf{Set},\mathsf{T},1))\quad \wedge\ (\boldsymbol{V}_{\boldsymbol{p_n}}(\mathsf{Set},\mathsf{F},1)))) \\
& \\
& \vee\ (x = \mathsf{Set}\ \wedge\ \boldsymbol{W}_{\boldsymbol{p_{n-1}}}(0) \wedge \boldsymbol{W}_{\boldsymbol{p_n}}(1) \\
& \qquad\qquad \wedge\ \xi(\varphi)) \\
& \\
& \vee\ (x = y) \\
& \vee\ (x = z)
\end{array}
$$

Table 1. The definition of $\mathcal{FORM}^i$ (we assume that the number l occurring in the formula is even)

(for i even, the quantifier prefix is: $\forall \boldsymbol{p}_1.\exists \boldsymbol{p}_2.\ldots\forall \boldsymbol{p}_{i-1}.\exists \boldsymbol{p}_i.$). Thus for $i = n$ we get that $\mathcal{B} \models_{\mathsf{FP}} \chi$ holds iff

$$\forall \boldsymbol{p}_1.\exists \boldsymbol{p}_2.\ldots\forall \boldsymbol{p}_{n-1}.\exists \boldsymbol{p}_n.\mathcal{C}(\boldsymbol{p}_1,\ldots,\boldsymbol{p}_n) \models_{\mathsf{FP}} (\mathcal{FORM}^{n+1}(\mathsf{Set},\boldsymbol{p}_n,0)) \tag{22}$$

which is equivalent to

$$\begin{array}{l}\forall \boldsymbol{p}_1.\exists \boldsymbol{p}_2.\ldots\forall \boldsymbol{p}_{n-1}.\exists \boldsymbol{p}_n. \\ \mathcal{B}\ [\{\mathsf{T}\}/\boldsymbol{W}_{\boldsymbol{p_0}}, \{\boldsymbol{p}_1\}/\boldsymbol{W}_{\boldsymbol{p_1}},\ldots,\{\boldsymbol{p}_n\}/\boldsymbol{W}_{\boldsymbol{p_n}}] \models_{\mathsf{FP}} \xi(\varphi)\end{array} \tag{23}$$

Now from the definition of ξ it can be easily seen that (23) holds iff ϕ holds (recall that p_0 does not occur in $\xi(\varphi)$). □

6 When Does the NP ∩ co-NP Bound Hold?

In this section we show a syntax restriction of FP^k, such that the combined complexity of the restricted subset is in NP ∩ co-NP.

Definition 12. $\widehat{\mathsf{FP}}^k$ formulas are the FP^k formulas satisfying the following condition.
• For every subformula $(\theta V(x_1, \ldots, x_n).\varphi)(y_1, \ldots, y_n)$ we require the set of free variables of φ to be contained in $\{x_1, \ldots, x_n\}$.

In the sequel we will use some results on the modal μ-calculus. We recall here that the modal μ-calculus, as introduced by Kozen [Koz83], is a modal logic with two dual fixpoint operators μ and ν. Its formulas are evaluated in the structures of the form:

$$\mathcal{M} = \langle S, Act, Prop, Q : Act \to (S \times S) \to bool,\ \rho : Prop \to S \to bool\rangle$$

where $Act = \{a_1, \ldots, a_n\}$ is the set of *actions*,
$Prop = \{p_1, \ldots, p_m\}$ is the set of *propositional constants*,
Q is a function assigning binary relations on S to actions in Act, and
ρ is a function assigning the subset of S to every constant in $Prop$.
The syntax of the modal μ-calculus is given by the following grammar.

$$\mathtt{F} ::= \mathtt{F} \wedge \mathtt{F} \mid \mathtt{F} \vee \mathtt{F} \mid \mu x_i.\ \mathtt{F} \mid \nu x_i.\ \mathtt{F} \mid \langle a\rangle\ \mathtt{F} \mid [a]\ \mathtt{F} \mid x_i \mid p_i \mid \neg p_i$$

The formal definition of the modal μ-calculus semantics can be found in [Koz83]. We write $\mathcal{M}, s \models_\mu \psi$ to mean that ψ is satisfied in a state s of $\mathcal{M}$ (i.e. $s \in [\![\psi]\!]_{\mathcal{M}}$). In the following two lemmas (13 and 14) we show that, for a fixed $k \geq 2$, $\widehat{\mathsf{FP}}^k$ is in some sense PTIME-equivalent to the modal μ-calculus.

Lemma 13. *There exists a polynomial-time transformation that for a given database $\mathcal{B}$ produces a model $\mathcal{M}_{\mathcal{B}}$ and for a given $\widehat{\mathsf{FP}}^k$ formula ψ produces a modal μ-calculus formula $\tau(\psi)$ such that for every tuple $t \in |\mathcal{B}|^k$ we have*

$$t \in Q_{(x_1,\ldots,x_k)\psi}(\mathcal{B}) \quad \textit{iff} \quad \mathcal{M}_{\mathcal{B}}, t \models_\mu \tau(\psi). \tag{24}$$

Proof sketch.

- Every database $\mathcal{B}$ with the carrier set B and the relational symbols $R_1, \ldots, R_n$ is translated to a model $\mathcal{M}_{\mathcal{B}} = \langle S, Act, Prop, Q, \rho\rangle$
  where $S = B^k$
  $Prop = \{Rel_1, \ldots, Rel_n, \mathtt{equality}\}$
  $Act = \{\mathtt{change}_1, \ldots, \mathtt{change}_k\} \cup \{1, \ldots, k\}^k$
  (remember that $k \geq 2$ is fixed)

  $\rho(Rel_i) = [\![R_i]\!]_{\mathcal{B}}$ for $i = 1, \ldots, n$
  (w.o.l.g we assume that all relations are k-ary)
  $\rho(\mathtt{equality}) = \{(x_1, \ldots, x_k) : x_1 = x_2\}$

  $Q(\mathtt{change}_i) = \{\langle (x_1, \ldots, x_k), (x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_k)\rangle :$
  $x_1, \ldots, x_k, z \in S\}$
  $Q((i_1, \ldots, i_k)) = \{\langle (x_1, \ldots, x_k), (x_{i_1}, \ldots, x_{i_k})\rangle : x_1, \ldots, x_k \in S\}$

– The function τ transforming $\widehat{\mathsf{FP}}^k$ formulas into formulas of modal μ-calculus is given below. Recall that we consider here only formulas satisfying the condition $\bullet$. Thus if the arity of a relation variable is less than k we can always extend it. Therefore we can assume that all relation variables are of arity k. The variables in a given modal μ-calculus formula are $z_1 \dots z_n$ and they correspond to the relational variables $\boldsymbol{V_1} \dots \boldsymbol{V_n}$ in the resulting $\widehat{\mathsf{FP}}^k$ formula.

$$\begin{aligned}
\tau(\psi_1 \wedge \psi_2) &= \tau(\psi_1) \wedge \tau(\psi_2)\\
\tau(\psi_1 \vee \psi_2) &= \tau(\psi_1) \vee \tau(\psi_2)\\
\tau(\forall x_i . \psi) &= [\mathtt{change}_i]\tau(\psi)\\
\tau(\exists x_i . \psi) &= \langle\mathtt{change}_i\rangle\tau(\psi)\\
\tau((\mu V_i(x_1,\dots,x_k).\psi)(x_{i_1},\dots,x_{i_k})) &= \langle(i_1)\dots(i_k)\rangle(\mu z_i.\tau(\psi))\\
\tau((\nu V_i(x_1,\dots,x_k).\psi)(x_{i_1},\dots,x_{i_k})) &= \langle(i_1)\dots(i_k)\rangle(\nu z_i.\tau(\psi))\\
\tau(x_i = x_j) &= [(i,j,1,\dots,1)]\mathtt{equality}\\
\tau(R_j(x_{i_1},\dots,x_{i_k})) &= [(i_1,\dots,i_k)](Rel_j)\\
\tau(\boldsymbol{V_j}(x_{i_1},\dots,x_{i_k})) &= [(i_1,\dots,i_k)](z_j)
\end{aligned}$$

Now it is easy to prove by structural induction that $\mathcal{B},[t_1/x_1,\dots,t_k/x_k] \models_{\mathsf{FP}} \psi$ iff $\mathbf{t} \in [\![\tau(\psi)]\!]_{\mathcal{M}_\mathcal{B}}$, which implies the correctness of the construction. $\square$

Lemma 14. *There exists a polynomial-time transformation that for a given model $\mathcal{M}$ produces a database $\mathcal{B}_\mathcal{M}$ and for a given modal μ-calculus formula φ produces an $\widehat{\mathsf{FP}}^2$ formula $\xi(\varphi)$ such that for every element s of model $\mathcal{M}$*

$$\mathcal{M}, s \models_\mu \varphi \quad \textit{iff} \quad s \in Q_{(x)\xi(\varphi)}(\mathcal{B}_\mathcal{M}). \tag{25}$$

Proof sketch.

– Every model $\mathcal{M} = \langle S, Act, Prop, Q, \rho\rangle$ is translated to a database $\mathcal{B}_\mathcal{M} = \langle S\rangle[\rho(p_1)/P_i,\dots,\rho(p_n)/P_n][Q(a_1)/A_1,\dots Q(a_m)/A_m]$

– We define the function ξ transforming formulas as follows.

$$\begin{aligned}
\xi(\varphi_1 \wedge \varphi_2, x) &= \xi(\varphi_1, x) \wedge \xi(\varphi_2, x)\\
\xi(\varphi_1 \vee \varphi_2, x) &= \xi(\varphi_1, x) \vee \xi(\varphi_2, x)\\
\xi(\mu x_i.\varphi, x) &= (\mu \boldsymbol{V_i}(y).\xi(\varphi, y))(x)\\
\xi(\nu x_i.\varphi, x) &= (\nu \boldsymbol{V_i}(y).\xi(\varphi, y))(x)\\
\xi(\langle a_i\rangle\varphi, x) &= \exists y.(A_i(x,y) \wedge \xi(\varphi, y))\\
\xi([a_i]\varphi, x) &= \forall y.(A_i(x,y) \Rightarrow \xi(\varphi, y))\\
\xi(x_i, x) &= \boldsymbol{V_i}(x)\\
\xi(p_i, x) &= P_i(x)\\
\xi(\neg p_i, x) &= \neg P_i(x)
\end{aligned}$$

Now we claim that $\mathcal{M}, s \models_\mu \varphi$ iff $\mathcal{B}_\mathcal{M}, [s/x] \models_{\mathsf{FP}} \xi(\varphi)$. The proof goes by structural induction and again we skip it. At first sight one may think that the number of first-order variables cannot be fixed because in the clauses for μ, ν and modalities new variables are introduced. It can be done however, observing that two variables x and y used alternatively are sufficient. It is also easy to see that the condition $\bullet$ is satisfied.

□

Since model-checking for the modal μ-calculus is in NP ∩ co-NP [EJS93], Lemma 13 gives us

Theorem 15. *The $Answer_{\widehat{\mathsf{FP}}^k}(\cdot,\cdot,\cdot)$ problem is in NP ∩ co-NP.*

By Lemma 14 all lower bounds for the complexity of the μ-calculus model checking apply also to $Answer_{\widehat{\mathsf{FP}}^k}(\cdot,\cdot,\cdot)$. Thus by [ZSS94] we get:

Theorem 16. *The $Answer_{\widehat{\mathsf{FP}}^k}(\cdot,\cdot,\cdot)$ problem is PTIME-hard.*

It is worth to note that Lemmas 13 and 14 give us translations between models and formulas independently. Thus the expression complexity of $\widehat{\mathsf{FP}}^k$ is PTIME-equivalent to the *expression complexity of the modal μ-calculus*. The best known lower bound for it is PTIME [DJN96].

We also want to emphasise that Vardi's algorithm for evaluating FP^k queries given in [Var95] works properly for $\widehat{\mathsf{FP}}^k$. Thus, as announced by Vardi, his algorithm still can be viewed as a new proof of the membership of the model-checking problem for the modal μ-calculus in NP ∩ co-NP.

7 Conclusion

Below we summarise the results of this paper.

Language	expression complexity	combined complexity
FP^k	PSPACE-complete	PSPACE-complete
$\overline{\mathsf{FP}}^k$	PSPACE-complete	PSPACE-complete
FP^k_n	ALOGTIME	PSPACE-complete
$\widehat{\mathsf{FP}}^k$	NP ∩ co-NP	NP ∩ co-NP

We recall that FP^k denotes the fixpoint first-order logic with bounded number of first-order variables, $\overline{\mathsf{FP}}^k$ denotes the prefix version of FP^k, FP^k_n denotes the sublogic of FP^k with bounded number of second-order variables and $\widehat{\mathsf{FP}}^k$ denotes the sublogic of FP^k defined in Definition 12.

Acknowledgements. I wish to thank several people for the help with my work. First of all, I wish to express my deep gratitude to Damian Niwiński who spent a lot of time patiently explaining me numerous details and discussing my ideas. Without his interest, support and encouragement this paper would never come into existence. I would also like to thank Igor Walukiewicz for his valuable comments on my ideas and for patience and interest he showed when reading proofs. I also wish to thank Grzegorz Grudziński, Marcin Jurdziński and all other Warsaw University Applied Logic Group people. I am also grateful to the anonymous referee for his helpful suggestions.

References

[Bus87] S.R. Buss. The boolean formula value problem is in ALOGTIME. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (New York City, May 25–27, 1987)*, pages 123–131, New York, 1987. ACM, ACM Press.

[CH82] A. Chandra and D. Harel. Structure and complexity of relational queries. *J. Comput. Syst. Sci.*, 25(1):99–128, August 1982.

[DJN96] S. Dziembowski, M. Jurdziński, and D. Niwiński. On the expression complexity of the modal μ-calculus model checking. unpublished manuscript, 1996.

[EJS93] E. A. Emerson, C. S. Jutla, and A. Sistla. On model-checking for fragments of μ calculus. In *CAV'93, volume 679 of LNCS*, pages 385–396, 1993.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27(3):333–354, 1983.

[Lyn77] N. Lynch. Log space recognition and translation of parenthesis languages. *J. ACM*, 24:583–590, 1977.

[Tar55] A. Tarski. A lattice theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Var82] M.Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th Ann. ACM Symposium on Theory of Computing (San Francisco, CA)*, pages 137–146, New York, 1982. ACM, ACM Press.

[Var95] M. Y. Vardi. On the complexity of bounded-variable queries. In *Proceedings of the 14th ACM Symposium on Principles of Database Systems*, pages 266–276, 1995.

[ZSS94] Shipei Zhang, Oleg Sokolsky, and Scott A. Smolka. On the parallel complexity of model checking in the modal mu-calculus. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 154–163, Paris, France, 4–7 July 1994. IEEE Computer Society Press.

This article was processed using the LaTeX macro package with LLNCS style