**Basic Research in Computer Science**

BRICS RS-96-41 S. Dziembowski: The Fixpoint Bounded-Variable Queries are PSPACE-Complete

# The Fixpoint Bounded-Variable Queries are PSPACE-Complete

**Stefan Dziembowski**

**BRICS Report Series** **RS-96-41**

**ISSN 0909-0878** **November 1996**

**Copyright © 1996, BRICS, Department of Computer Science
University of Aarhus. All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

**See back inner page for a list of recent publications in the BRICS Report Series. Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

**BRICS publications are in general accessible through World Wide Web and anonymous FTP:**

`http://www.brics.dk/`
`ftp://ftp.brics.dk/pub/BRICS`
**This document in subdirectory** `RS/96/41/`

# The Fixpoint Bounded-Variable Queries are PSPACE-complete *

Stefan Dziembowski
stefand@mimuw.edu.pl

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

**Abstract.** We study complexity of the evaluation of fixpoint bounded-variable queries in relational databases. We exhibit a finite database such that the problem whether a closed fixpoint formula using only 2 individual variables is satisfied in this database is PSPACE-complete. This clarifies the issues raised by Moshe 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 different queries expressible in a 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 different queries against 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 first order logic (FO) is $AC^0$ while the expression and combined complexities are complete in PSPACE, the data complexity of fixpoint logic (FP) is PTIME, while the expression and combined complexities are EXPTIME-complete.

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

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

We study the problem of measuring the expression and combined complexities of the bounded-variable version of the fixpoint first-order logic ($\mathsf{FP}^k$). 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 $\mathsf{FP}^k$ formulas.

We also consider various restrictions of the $\mathsf{FP}^k$ syntax and study their complexity. At first, we fix a 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, first provable gap between expression complexity and combined complexity. We also study the combined and expression complexities of the prefix version of $\mathsf{FP}^k$ (formulas are of the form: prefix of fixpoint operators . first-order formula . arguments), showing the PSPACE-completeness. Finally, we present a sublogic of $\mathsf{FP}^k$ for which the NP $\cap$ co-NP upper bound for combined complexity holds.

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

## 2 Acknowledgments

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.

## 3 Basic Definitions

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

### 3.1 Databases and Queries

**Definition 1.** *Database signature* is a pair $(S, C, ar)$, where

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

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

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

where

- *carrier set* $|\mathcal{B}|$ is a finite subset of natural numbers
- *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 an arity $ar(s)$.

The restriction that the set of symbols and the carrier set are subsets 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 way.
An expression $\langle B \rangle$ will denote a database $\mathcal{B}$, with a carrier set $B$ under a signature $(\emptyset, C, [\![\cdot]\!])$, 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* $Q$ *and an arity* $a$ is a 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 $Q$, similarly, functions $ar'$ and $ar$ are equal on $S \setminus \{\boldsymbol{T}\}$ and $ar'(\boldsymbol{T}) = a$. Such an extension will be denoted by $\mathcal{B}[Q/\boldsymbol{T} : a]$. When an arity $a$ is clear from a context we will use abbreviation, writing $B[Q/\boldsymbol{T}]$ instead of $B[Q/\boldsymbol{T} : a]$.

**Definition 4.** For every database signature $\sigma$ and every natural number $n$, an arbitrary 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 we will study the complexity issues in the paper, we make here a formal assumption, that, together with a language, there is given a standard way of encoding its elements. We fix also some standard way of encoding databases. We skip here the details.

### 3.2 Logics as a query languages

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

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

(where $[b_1/x_1, \ldots, b_l/x_l]$ denotes a valuation of first-order variables, such that, for each $i = 1, \ldots, l$ we have $v(x_i) = t_i$).
The following abbreviation is useful:
For every database $\mathcal{B}$, a valuation $\alpha$, a tuple of variables $\mathbf{x}$, a formula $\varphi(\mathbf{x})$, and a tuple $\mathbf{t}$ of the carrier set elements we will often write $\mathcal{B}, \alpha \models \varphi(\mathbf{t})$ instead of $\mathcal{B}, \alpha[\mathbf{t}/\mathbf{x}] \models \varphi(x)$ (where $\alpha[\mathbf{t}/\mathbf{x}]$ denotes a valuation equal to $\alpha$ outside the variables $\mathbf{x}$ and such that for every $i$ we have that $\alpha(x_i) = t_i$).

### 3.3 Complexity

In this paper we will study the complexity of queries. We are interested in comparing 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 will focus on the following two instances:

- the database $\mathcal{B}$ and the language $\mathcal{L}$ are fixed, we measure the complexity 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 language $\mathcal{L}$ is fixed, we measure the complexity of the set:
$$Answer_{\mathcal{L}}(\cdot, \cdot, \cdot) = \{\langle \mathbf{t}, \mathcal{B}, e\rangle \mid e \in \mathcal{L} \text{ and } \mathbf{t} \in Q_e(\mathcal{B})\}$$

Complexity of the problems $Answer_{\mathcal{L}}(\mathcal{B}, \cdot, \cdot)$ is called *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 $\mathcal{C}$
- *the expression complexity of a language $\mathcal{L}$ is hard in 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 in $\mathcal{C}$

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

The complexity of $Answer_{\mathcal{L}}(\cdot,\cdot,\cdot)$ is called a combined complexity of the language $\mathcal{L}$. Note that for every $\mathcal{B}$, $Answer_{\mathcal{L}}(\mathcal{B},\cdot,\cdot)$ is reducible in PTIME to $Answer_{\mathcal{L}}(\cdot,\cdot,\cdot)$, but need not be any $\mathcal{B}$, such that the converse holds. Therefore $Answer_{\mathcal{L}}(\cdot,\cdot,\cdot)$ is generally harder than $Answer_{\mathcal{L}}(\mathcal{B},\cdot,\cdot)$. Consequently, expression complexity of $\mathcal{L}$ is lower then its combined complexity. The following observation is useful:

**Lemma 6.** *For an arbitrary database $\mathcal{B}$ and an arbitrary logic $\mathcal{L}$ (at least as strong as first-order logic) problems:*

$$Answer_{\mathcal{L}}(\mathcal{B},\cdot,\cdot) \quad \textit{and} \quad Answer_{\mathcal{L}}(\mathcal{B},\cdot) = \{\varphi : \varphi \textit{ is closed and } \mathcal{B} \models_{\mathcal{L}} \varphi\}$$

*are PTIME-equivalent. Similar fact holds for*

$$Answer_{\mathcal{L}}(\cdot,\cdot,\cdot) \textit{ and } \quad Answer_{\mathcal{L}}(\cdot,\cdot) = \{\varphi : \varphi \textit{ is closed and } \mathcal{B} \models_{\mathcal{L}} \varphi\}$$

*Proof.* The reduction from right to left is trivial. Reduction in the opposite direction consists in replacing free variables in $\varphi$ by constants form the tuple $\mathbf{t}$. □

From now on, when considering expression and combined complexities (greater or equal to PTIME) we will restrict ourselves to closed formulas. This restriction is not essential due to Lemma 6.

## 4 Fixpoint First-Order Logic

The FP language is an extension of the standard first-order logic with two dual fixpoint operators: $\mu$ and $\nu$, denoting the least and the greatest fixpoint, respectively [CH82]. The syntax is extended in the following way. For an arbitrary FP formula $\varphi$ and every second-order $l$-ary variable $\boldsymbol{V}$ appearing positively in it (i.e. not 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 $free_1(\varphi)$ of free first-order variables in $\varphi$ is defined as in the standard first-order logic. For the fixpoint formulas we have: $free_1(\theta\boldsymbol{V}(x_1,\ldots,x_l).\varphi) = free_1(\varphi)/\{x_1,\ldots,x_l\}$. The set $free_2(\varphi)$ of free second-order variables in $\varphi$ is also defined in the standard way (second-order variables are bound by $\mu$ and $\nu$ operators).
In the sequel we will interpret FP formulas under databases such that for every free second-order variable in a formula there is a corresponding relation in a database. The semantics of the FP should be rather clear, except the semantics of the fixpoint subformulas. Consider a subformula $(\mu\boldsymbol{V}(y_1,\ldots,y_l).\varphi)(x_1,\ldots,x_l)$ in a database $\mathcal{B}$. We can treat a second-order variable $\boldsymbol{V}$ in a formula $\varphi$ as a relational symbol and evaluate this formula in a database $\mathcal{B}$ extended with this symbol. In this way, after fixing a valuation $\alpha$ of free first-order variables in $\varphi$ we get an operator on $l$-ary relations, defined in the following way:

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

(where $\alpha[\boldsymbol{z}/y]$ denotes a valuation equal to $\alpha$ outside $y$ and equal $\boldsymbol{z}$ on $y$)
Note, that $\varphi$ may contain some free variables other than $y_1, \ldots, y_l$. Because the variable $\boldsymbol{V}$ occurs in the formula $\varphi$ positively, the operator $\widehat{\varphi}_{\mathcal{B}}$ 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:

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

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

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

(where $v$ is equal to $\alpha$ 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 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 $\varphi$ has no free variables, we will often write $\mathcal{B} \models_{\mathsf{FP}} \varphi$ instead of $\mathcal{B}, \alpha \models_{\mathsf{FP}} \varphi$. The following lemma will be useful:

**Lemma 7.** *Let $\mathcal{B}$ be an arbitrary database, $\alpha$ an arbitrary valuation and $\varphi(\mathbf{x})$ an arbitrary formula. Then 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 $\mathcal{V} = \widehat{\varphi}^{\infty} = \widehat{\varphi}(\widehat{\varphi}^{\infty}) = \widehat{\varphi}(\mathcal{V})$, what is equal to (2). □

## 5 The Complexity of $\mathsf{FP}^k$

In [Var95] Vardi considered several languages, each $\mathcal{L}^k$ 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 $\mathsf{FP}^k$ from $\mathsf{FP}$. We will show that there exists a database $\mathcal{B}$ such that $Answer_{\mathsf{FP}^2}(\mathcal{B}, \cdot)$ is PSPACE-hard and, consequently, that $Answer_{\mathsf{FP}^2}(\cdot, \cdot)$ is PSPACE-hard. This is the best lower bound because on the other hand it is known that, for every $k$, $Answer_{\mathsf{FP}^k}(\cdot, \cdot)$ is in PSPACE ([Var95]), and, consequently for every $\mathcal{B}$, $Answer_{\mathsf{FP}^k}(\mathcal{B}, \cdot)$ is in PSPACE.

**Theorem 8.** *There exists a database $\mathcal{B}$, such, that $Answer_{\mathsf{FP}^2}(\mathcal{B}, \cdot)$ is PSPACE-complete*

*Proof.* From the previous remarks it suffices 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:

$$\mathtt{F} ::= \mathtt{F} \wedge \mathtt{F} \mid \mathtt{F} \vee \mathtt{F} \mid \forall x_i.\ \mathtt{F} \mid \exists x_i.\ \mathtt{F} \mid x_i \mid \neg x_i$$

The variables take the boolean values $\mathsf{T}$ and $\mathsf{F}$. We can assume that each $x_i$ is quantified only once. The QBF problem is a set $\{\psi : \psi \text{ is closed and } \models_{\mathsf{QBF}} \psi\}$. At first sight, one may think that we can reduce QBF to $\mathsf{FP}^k$ 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 can not be bounded. Therefore we have to use a bit more sophisticated method and make use of fixpoint operators. Let $\mathcal{B} = \langle\{\mathsf{T}, \mathsf{F}, \mathsf{A}\}\rangle$. A function $\xi$ transforming QBF formulas to $\mathsf{FP}^2$ formulas is defined by structural induction.

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

Note, that in this construction there is a bijection between variables $x_1, \ldots, x_n$ in a QBF formula and $\boldsymbol{V_1}, \ldots, \boldsymbol{V_n}$ in 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 $\mathsf{T}$ then in the first iteration $\mathsf{T}$ enters to the least fixpoint ($\mathsf{F}$ will never enter). This information will be further used in the evaluation of subformulas $\boldsymbol{V_i}(\mathsf{T})$ and $\boldsymbol{V_i}(\mathsf{F})$.

**Lemma 9.** $\models_{\mathsf{QBF}} \psi$ *iff* $\mathcal{B} \models_{\mathsf{FP}} \xi(\psi)$

*Proof:* The proof goes by induction on the structure of a QBF formula. We show the following more general fact:
Let $\psi(x_1, \ldots, x)$ be a QBF formula and $\alpha$ a valuation of the variables $x_1, \ldots, x_l$

$$\alpha \models_{\mathsf{QBF}} \psi \text{ iff } \mathcal{T}^{\psi}(\alpha) \models_{\mathsf{FP}} \xi(\psi) \tag{3}$$

where $\mathcal{T}^{\psi}(v) = \mathcal{B}[v(x_1)/\boldsymbol{V_1}, \ldots, v(x_l)/\boldsymbol{V_l}]$

1. 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 $\psi_1$ and $\psi_2$. Now:

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

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

Which, 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 $\mathcal{T}^{\psi_1 \wedge \psi_2}(v) \models_{\mathsf{FP}} \xi(\psi_1 \wedge \psi_2)$. Similarly we can prove the induction step for $\vee$

3. The case of quantifiers is the most interesting one. Let us consider universal quantifier (the proof for existential quantifier is similar). Take a $\mathsf{QBF}$ formula $\forall x_i.\psi$ and an arbitrary valuation $v$ of its free variables. Suppose (3) holds for $\psi$. From the definition of $\xi$ 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

$$\varphi(x,y) = \begin{pmatrix} \vee & x \neq \mathsf{A} \wedge x = y \\ & x = \mathsf{A} \wedge \xi(\psi) \end{pmatrix} \tag{7}$$

Formula $(\mu \boldsymbol{V_i}(x).\varphi(x,y))(\mathsf{A})$ has exactly one free variable: $y$. We will prove, that

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

This fact is 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, after applying the Lemma 7 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 will be sufficient for evaluating the value of $\mathcal{V}$. We are mostly interested in the value of this predicate on the element $\mathsf{A}$. However, to evaluate it we have to know the value of $\mathcal{V}$ on the set $\{\mathsf{T},\mathsf{F}\}$. Take an arbitrary $\boldsymbol{q} \in \{\mathsf{T},\mathsf{F}\}$. 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 $\varphi$ (7), is equivalent to

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

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 $\mathsf{A}$, (i.e. evaluate (10)). Note, that in the formula $\varphi$ 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 $\varphi$ we get that (10) holds if and only if

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

which is equivalent to

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

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

## 6 Complexity of restrictions of $\mathsf{FP}^{\boldsymbol{k}}$

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

### 6.1 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 8 is essential, i.e. whether it is possible to prove PSPACE-hardness when we restrict also the number of second-order variables to some $n$. In this section we show that the answer is positive when we ask about the combined complexity, but it is negative for the expression complexity.
Let $\mathsf{FP}^k_n$ be a sublogic of $\mathsf{FP}^k$, 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 will prove now the following

**Theorem 10.** *For every $k \geq 3$, $n \geq 2$ the $Answer_{\mathsf{FP}^k_n}(\cdot,\cdot)$ problem is PSPACE-hard*

*Proof:* The membership in PSPACE is trivial. It remains to prove the PSPACE-hardness of $Answer_{\mathsf{FP}^3_2}$. Let us fix a $\mathsf{QBF}$ formula $\psi$ (with variables $x_1, \ldots, x_n$). We will show how to construct in a polynomial time a pair $\langle$ database $\mathcal{B}_\psi$, and an $\mathsf{FP}^3_2$ 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 our new construction we define a database in such a way that we will be 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 $\rho$ that transforms the formula $\psi$ to a $\mathsf{FP}^k_n$ formula, proceeding top-down. In $\varphi$ we use binary relation variables $\boldsymbol{V_i}$. Then we will show how to reduce the number of $\boldsymbol{V_i}$'s to 2. States in the database $\mathcal{B}_\psi$ (all, except $\mathsf{A}$) are used to remember the valuation in the following way:
The subformulas (c.f. definition below) $\mu\boldsymbol{V_{d+1}}\cdots$ are intended to represent 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 $\mathsf{T}$ and $\boldsymbol{V_d}(i,\mathsf{F})$ holds iff the value of $x_i$ is $\mathsf{F}$. Note, that the actual value of such subformulas will depend on the value of variables that occur free in this formula. In this way, using fix-point formulas we shall be able to capture the whole tree of possible valuations. We define a transformation $\rho(\psi) = \xi(\psi,0)$ using auxiliary function $\xi(\psi : \mathsf{QBF}\text{ formulas }\ d : \mathbb{N})$ which is defined inductively by the following clauses:

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

We claim, that (17) holds. The proof is similar to the one in Section 5. We will skip it here, showing only the induction hypothesis:

$$\begin{array}{l}\text{for every } \mathsf{QBF} \text{ formula } \psi \text{ with free variables } x_1,\ldots,x_n\\ \text{every valuation } v \text{ of these variables and every } d\in\mathbb{N}\\ \text{we have that } \ \alpha \models_{\mathsf{QBF}} \psi \ \text{ iff } \ \{(i,\alpha(x_i)) : i=1,\ldots,n\} \models_{\mathsf{FP}} \xi(\psi,d)\end{array} \qquad (18)$$

Now observe that in each subformula of $\rho(\psi)$ of the form $\mu\boldsymbol{V_{d+1}}(x,z).\varphi$, we have that $free_2(\varphi) \cap \{\boldsymbol{V_1},\ldots\} = \{\boldsymbol{V_d}\}$, therefore all other $\boldsymbol{V_i}$'s "are not visible" from $\varphi$ 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. □

**Expression Complexity** In the above construction the size of the database was not bounded, and could be even linear in the size of a given $\mathsf{QBF}$ formula. Below we argue that it 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 from $A \to (x)$ 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 recognizable in ALOGTIME.

**Theorem 11.** *The* $Answer_{\mathsf{FP}^k_n}(\mathcal{B})$ *problem is in ALOGTIME for every database* $\mathcal{B}$ *every* $k$ *and every* $n$.

*Proof sketch:* Note, that we consider here complexity for which it is not known, whether it contains PTIME. Thus we can not use Lemma 6 here. In the proof therefore we do not forget about the tuple $\mathbf{t}$ and, for each database $\mathcal{B}$, consider the $Answer(\mathcal{B}, \cdot, \cdot)$ problem. We will show how, for a given database $\mathcal{B}$, a number $k$ of the first-order variables and a number $n$ of the second-order variables, to construct a parenthesis grammar $G$, such that every formula $\varphi \in \mathsf{FP}^k_n$ and every tuple $\mathbf{t}$ of the length $l$

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

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 a set of all valuations of the variables $x_1, \ldots, x_k$, let $B$ be a set of all extensions of a 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

- 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 $y$, every two tuples of variables $\mathbf{x}$ and $\mathbf{y}$ we have:
  $S \to (\mathbf{t} \models_{\mathsf{FP}} (y_1, \ldots, y_l)T_i)$ where $\mathbf{t} = ((v(y_1), \ldots, v(y_l)))$
  $T_i \to (x = y)$ iff $v(x) = v(y)$
  $T_i \to (x \neq y)$ iff $v(x) \neq v(y)$
  $T_i \to (\boldsymbol{R_j}(\mathbf{x}))$ iff $T_i = \{(\mathcal{D}, v) : \mathcal{D}, v \models_{\mathsf{FP}} (\boldsymbol{R_j}(\mathbf{x})\}$
  $T_i \to (\neg\boldsymbol{R_j}(\mathbf{x}))$ iff $T_i = \{(\mathcal{D}, v) : \mathcal{D}, v \not\models_{\mathsf{FP}} (\boldsymbol{R_j}(\mathbf{x})\}$
  $T_i \to (\boldsymbol{V_j}(\mathbf{x}))$ iff $T_i = \{(\mathcal{D}, v) : \mathcal{D}, v \models_{\mathsf{FP}} (\boldsymbol{V_j}(\mathbf{x})\}$
  $T_i \to (T_{i_1} \wedge T_{i_2})$ iff $T_i = T_{i_1} \cap T_{i_2}$
  $T_i \to (T_{i_1} \vee T_{i_2})$ iff $T_i = T_{i_1} \cup T_{i_2}$
  $T_i \to (\exists y.T_h)$ iff $T_i = \{(\mathcal{D}, v) : \exists \boldsymbol{p}.\ (\mathcal{D}, v[\boldsymbol{p}/y]) \in T_h\}$
  $T_i \to (\forall y.T_h)$ iff $T_i = \{(\mathcal{D}, v) : \forall \boldsymbol{p}.\ (\mathcal{D}, v[\boldsymbol{p}/y]) \in T_h\}$
  The most interesting is the case of the fixpoint formulas. Note, that for every $T_h$ induces a following operator on relations on $|\mathcal{B}|$:

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

For every $T_i$ and every $T_h$, such that the operator $\rho_{T_h}(\mathbf{x})$ is monotone, there exists a least fixpoint $\rho^{\infty}_{T_h(\mathbf{x})}$ of the $\rho_{T_h(\mathbf{x})}$ operator. For an arbitrary tuples $\mathbf{y}$ and $\mathbf{z}$ (of the length $ar(\boldsymbol{V_j})$) of first-order variables we say that

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

A production for the greatest fixpoint is defined similarly.

We skip here the formal proof of (19) □

### 6.2 Prefix Form

Define $\overline{\mathsf{FP}}^k$ formulas as an $\mathsf{FP}^k$ 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 will write $\mu \boldsymbol{V}(\mathbf{x}) \leftarrow (\mathbf{y}).\varphi$ instead of $(\mu \boldsymbol{V}(\mathbf{x}).\varphi)(\mathbf{y})$. This makes them more readable. It this section we will show, that the expression and combined complexities of $\widehat{\mathsf{FP}}^k$ are PSPACE-complete. Namely we show the following

**Theorem 12.** *There exists a database $\mathcal{B}$, such that the Answer$_{\widehat{\mathsf{FP}}^k}$ problem is PSPACE-complete.*

*Proof sketch* 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 $\varphi$ is a sentence which variables are among $p_1, \ldots, p_n$) we will show how to construct in PTIME an $\widehat{\mathsf{FP}}^k$ formula $\chi$, 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 $\chi$ is complex. To define it we first define a function $\xi$ by the following clauses:

$$\begin{array}{rcl@{\qquad}rcl} \xi(\varphi_1 \wedge \varphi_2) & = & \xi(\varphi_1) \wedge \xi(\varphi_2) & \xi(\neg p_n) & = & \boldsymbol{W_{p_n}}(\mathsf{F}) \\ \xi(\varphi_1 \vee \varphi_2) & = & \xi(\varphi_1) \vee \xi(\varphi_2) & \xi(p_n) & = & \boldsymbol{W_{p_n}}(\mathsf{T}) \end{array}$$

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

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

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 7 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 $\boldsymbol{W_{p_1}}, \ldots, \boldsymbol{W_{p_n}}$.

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

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

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 $\boldsymbol{W_{p_i}} = \cdots = \boldsymbol{W_{p_n}} = \{1, \boldsymbol{p}_i\}$). In the proof we show that for every odd $i = 1, \ldots, n$ and every $\boldsymbol{b} \in \{\mathsf{T}, \mathsf{F}\}$ we have that

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

(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

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

what is equivalent to

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

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

### 6.3 When Does the NP ∩ co-NP Bound Hold?

In this section we will show a syntax restriction of $\mathsf{FP}^k$, the combined complexity of which is in NP ∩ co-NP.

**Definition 13.** $\widehat{\mathsf{FP}}_k$ formulas are the $\mathsf{FP}_k$ 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 in $\varphi$ to be contained in $\{x_1,\ldots,x_n\}$.

In the sequel we will use some results on the modal $\mu$-calculus. We recall here that the modal $\mu$-calculus, as introduced by Kozen [Koz83], is a modal logic with two dual fixpoint operators $\mu$ and $\nu$. 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 $Act$
$\rho$ is a function assigning the subset of $S$ to every constant in $Prop$
The syntax of the modal $\mu$-calculus is given by the following grammar:

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

Formal definition of the modal $\mu$-calculus semantics can be found in [Koz83]. We write $\mathcal{M} \models \psi$ to mean that $\psi$ is satisfied in every state of $\mathcal{M}$ (i.e. $s \in [\![\psi]\!]_{\mathcal{M}}$).

We will show that, for a fixed $k \geq 2$, $\widehat{\mathsf{FP}}^k$ is PTIME-equivalent to the modal $\mu$-calculus in the following sense:

**Lemma 14.** *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 $\varphi$ produces a modal $\mu$-calculus formula $\psi_\varphi$, such that $\mathcal{B} \models \varphi$ iff $\mathcal{M}_{\mathcal{B}} \models \psi_\varphi$.*

**Lemma 15.** *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 $\mu$-calculus formula $\psi$ produces an $\widehat{\mathsf{FP}}^2$ formula $\varphi_\psi$, such that $\mathcal{B}_{\mathcal{M}} \models \varphi_\psi$ iff $\mathcal{M} \models \psi$.*

The proof Lemma 15 is easy and we will skip it here. Since model-checking for the modal $\mu$-calculus is in NP ∩ co-NP [EJS93], Lemma 14 gives us:

**Theorem 16.** *The Answer$_{\widehat{\mathsf{FP}}^k}(\cdot,\cdot)$ problem is in NP ∩ co-NP*

By Lemma 15 all lower bounds for the modal $\mu$-calculus apply also to the $Answer_{\widehat{\mathsf{FP}}^k}$. Thus [ZSS94] we get:

**Theorem 17.** *The $Answer_{\widehat{\mathsf{FP}}^k}(\cdot,\cdot)$ problem is PTIME-hard*

*Proof sketch of Lemma 14:*

- every database $\mathcal{B}$ with a carrier set $B$ and relational symbols $R_1,\ldots,R_n$ is translated to a model $\mathcal{M}_{\mathcal{B}} = \langle S, Act, Prop, Q, \rho\rangle$

$$\begin{aligned}
\text{where} \quad S &= B^k\\
Prop &= \{Rel_1,\ldots,Rel_n,\mathtt{equality}\}\\
Act &= \{\mathtt{change}_1,\ldots,\mathtt{change}_k\}\cup\{1,\ldots,k\}^k \quad \text{(remember that } k\geq 2 \text{ is fixed)}\\
\rho(Rel_i) &= [\![R_i]\!]_{\mathcal{B}} \text{ for } i=1,\ldots,n \text{ (w.o.l.g we assume that all relations are k-ary)}\\
\rho(\mathtt{equality}) &= \{(x_1,\ldots,x_k) : x_1 = x_2\}\\
Q(\mathtt{change}_i) &= \{\langle (x_1,\ldots,x_k),(x_1,\ldots,x_{i-1},z,x_{i+1},\ldots,x_k)\rangle :\\
&\qquad x_1,\ldots,x_k,z\in S\}\\
Q((i_1,\ldots,i_k)) &= \{\langle (x_1,\ldots,x_k),(x_{i_1},\ldots,x_{i_k})\rangle : x_1,\ldots,x_k\in S\}
\end{aligned}$$

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

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

Now it is easy to prove by structural induction that $\mathcal{B},\alpha\models_{\mathsf{FP}}\psi$ iff $\alpha\in[\![\tau(\psi)]\!]_{\mathcal{M}_{\mathcal{B}}}$, what implies the correctness of the construction.
It is worth to note, that Lemmas 14 and 15 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 $\mu$-calculus*. The best known lower bound for it is PTIME [DJN96].

We also want to emphasize, that Vardi's algorithm for evaluating $\mathsf{FP}^k$ queries from [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 $\mu$-calculus in NP $\cap$ co-NP.

## 7 Conclusion

Below we summarize the results from this paper.

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

We recall, that $\mathsf{FP}^k$ denotes the fixpoint first-order logic with bounded number of first-order variables, $\overline{\mathsf{FP}}^k$ denotes the prefix version of $\mathsf{FP}^k$, sublogic $\mathsf{FP}^k_n$ denotes $\mathsf{FP}^k$ with bounded number of second-order variables and $\widehat{\mathsf{FP}}^k$ denotes the sublogic of $\mathsf{FP}^k$ defined in the Definition 13.

## 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 $\mu$-calculus model checking. unpublished manuscript, 1996.

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

[Koz83] D. Kozen. Results on the propositional $\mu$-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

## Recent Publications in the BRICS Report Series

**RS-96-41** Stefan Dziembowski. *The Fixpoint Bounded-Variable Queries are PSPACE-Complete*. November 1996. 16 pp. Presented at the *10th Annual International Conference of the European Association for Computer Science Logic*, CSL '96.

**RS-96-40** Gerth Stølting Brodal, Shiva Chaudhuri, and Jaikumar Radhakrishnan. *The Randomized Complexity of Maintaining the Minimum*. November 1996. 20 pp. To appear in a special issue of *Nordic Journal of Computing* devoted to the proceedings of SWAT '96. Appears in Karlson and Lingas, editors, *Algorithm Theory: 5th Scandinavian Workshop*, SWAT '96 Proceedings, LNCS 1097, 1996, pages 4–15.

**RS-96-39** Hans Hüttel and Sandeep Shukla. *On the Complexity of Deciding Behavioural Equivalences and Preorders – A Survey*. October 1996. 36 pp.

**RS-96-38** Hans Hüttel and Josva Kleist. *Objects as Mobile Processes*. October 1996. 23 pp.

**RS-96-37** Gerth Stølting Brodal and Chris Okasaki. *Optimal Purely Functional Priority Queues*. October 1996. 27 pp. To appear in *Journal of Functional Programming*, 6(6), December 1996.

**RS-96-36** Luca Aceto, Willem Jan Fokkink, and Anna Ingólfsdóttir. *On a Question of A. Salomaa: The Equational Theory of Regular Expressions over a Singleton Alphabet is not Finitely Based*. October 1996. 16 pp.

**RS-96-35** Gian Luca Cattani and Glynn Winskel. *Presheaf Models for Concurrency*. October 1996. 16 pp. Presented at the *10th Annual International Conference of the European Association for Computer Science Logic*, CSL '96.

**RS-96-34** John Hatcliff and Olivier Danvy. *A Computational Formalization for Partial Evaluation (Extended Version)*. October 1996. To appear in *Mathematical Structures in Computer Science*.