ON SLICEWISE MONOTONE PARAMETERIZED PROBLEMS AND OPTIMAL PROOF SYSTEMS FOR TAUT

YIJIA CHEN AND JÖRG FLUM

ABSTRACT. For a reasonable sound and complete proof calculus for first-order logic consider the problem to decide, given a sentence φ of first-order logic and a natural number n, whether φ has no proof of length $\leq n$. We show that there is a nondeterministic algorithm accepting this problem which, for fixed φ, has running time bounded by a polynomial in n if and only if there is an optimal proof system for the set TAUT of tautologies of propositional logic. This equivalence is an instance of a general result linking the complexity of so-called slicewise monotone parameterized problems with the existence of an optimal proof system for TAUT.

1. Introduction

In this paper we relate the existence of optimal proof systems for the class TAUT of tautologies of propositional logic with the complexity of slicewise monotone parameterized problems. A *proof system* in the sense of Cook and Reckhow [4], say for the class TAUT, is a polynomial time computable function defined on $\{0,1\}^*$ and with TAUT as range. A proof system P is *optimal* if for any other proof system P' for TAUT there is a polynomial $p \in \mathbb{N}[X]$ such that for every tautology α, if α has a proof of length n in P', then α has a proof of length $\leq p(n)$ in P.[1] In their fundamental paper [9] Krajíček and Pudlák showed that an optimal proof system for TAUT exists if NE $=$ co-NE and they derived a series of statements equivalent to the existence of such an optimal proof system; however they conjectured that there is no optimal proof system for TAUT.

On the other hand, Gödel in a letter to von Neumann of 1956 (see [6]) asked for the complexity of the problem to decide, given a sentence φ of first-order logic and a natural number n, whether φ has a proof of length $\leq n$. In our study [2] of this problem we introduced the parameterized problem

p-GÖDEL	
Instance:	A first-order sentence φ and $n \in \mathbb{N}$ in unary.
Parameter:	$\lvert\varphi\rvert$.
Problem:	Does φ have a proof of length $\leq n$?

[1] All notions will be defined in a precise manner in later sections.

Here we refer to any reasonable sound and complete proof calculus for first-order logic. We do not allow proof calculi, which, for example, admit all first-order instances of propositional tautologies as axioms (as then it would be difficult to recognize correct proofs if P $\neq$ NP).

In a different context, namely when trying to show that a certain logic $L_{\leq}$ for PTIME (introduced in [7]) does not satisfy some effectivity condition, Nash et al. introduced implicitly [12] (and this was done explicitly in [1]) the *parameterized acceptance problem* p-ACC$_{\leq}$ *for nondeterministic Turing machines*:

p-ACC$_{\leq}$	
Instance:	A nondeterministic Turing machine $\mathbb{M}$ and $n \in \mathbb{N}$ in unary.
Parameter:	$\|\mathbb{M}\|$, the size of $\mathbb{M}$.
Problem:	Does $\mathbb{M}$ accept the empty input tape in $\leq n$ steps?

Both problems, p-GÖDEL and p-ACC$_{\leq}$, are *slicewise monotone*, that is, their instances have the form (x, n), where $x \in \{0,1\}^*$ and $n \in \mathbb{N}$ is given in unary,[2] the parameter is $|x|$, and finally for all $x \in \{0,1\}^*$ and $n, n' \in \mathbb{N}$ we have

if (x, n) is a positive instance and $n < n'$, then (x, n') is a positive instance.

A slicewise monotone problem is in the complexity class $\mathrm{XNP}_{\mathrm{uni}}$ if there is a nondeterministic algorithm that accepts it in time $n^{f(|x|)}$ for some function $f\colon \mathbb{N} \to \mathbb{N}$. And co-$\mathrm{XNP}_{\mathrm{uni}}$ contains the complements of problems in $\mathrm{XNP}_{\mathrm{uni}}$. We show:

Theorem 1. TAUT *has an optimal proof system if and only if every slicewise monotone problem in* NP *is in* co-$\mathrm{XNP}_{\mathrm{uni}}$.

There are trivial slicewise monotone problems which are fixed-parameter tractable. However, for the slicewise monotone problems mentioned above we can show:

Theorem 2. TAUT *has an optimal proof system* $\iff$ p-ACC$_{\leq} \in$ co-$\mathrm{XNP}_{\mathrm{uni}}$
$\iff$ p-GÖDEL $\in$ co-$\mathrm{XNP}_{\mathrm{uni}}$.

In [3] we showed that TAUT has a *p-optimal* proof system if and only if a certain logic $L_{\leq}$ is a P-bounded logic for P (=PTIME). The equivalence in the first line of Theorem 2 is the nondeterministic version of this result; in fact, an immediate consequence of it states that TAUT has an optimal proof system if and only if $L_{\leq}$ is an NP-bounded logic for P (a concept that we will introduce in Section 6). It turns out that a slight variant of $L_{\leq}$ is an NP-bounded logic for P (without any assumption).

[2]The requirement that n is given in unary notation ensures that the classical complexity of most slicewise monotone problems we consider is in NP.

The content of the different sections is the following. In Section 2 and Section 3 we recall the concepts and results of parameterized complexity and on optimal proof systems, respectively, we need in Section 4 to derive the equivalence in the first line of Theorem 2. Furthermore, in Section 3 we claim that every problem hard for EEXP under polynomial time reductions has no optimal proof system. In Section 5 we derive some basic properties of slicewise monotone problems, show that p-ACC$_{\leq}$ is of highest parameterized complexity among the slicewise monotone problems with classical complexity in NP, and finally show that all the slicewise monotone problems we consider in a certain sense have the same complexity (see Proposition 14 for the precise statement). This yields Theorem 1 and the remaining equivalence of Theorem 2. As already mentioned, in Section 6 we analyze the relationship of the existence of an optimal proof system for TAUT and the properties of the logic $L_{\leq}$.

2. Some preliminaries

In this section we recall some basic definitions and concepts from parameterized complexity and introduce the concept of slicewise monotone parameterized problem.

We denote the alphabet $\{0,1\}$ by Σ. The length of a string $x \in \Sigma^*$ is denoted by $|x|$. We identify problems with subsets Q of Σ^*. Clearly, as done mostly, we present concrete problems in a verbal, hence uncodified form or by using other alphabets. We denote by P the class of problems Q such that $x \in Q$ is solvable in time polynomial in $|x|$.

All deterministic and nondeterministic Turing machines have Σ as their alphabet. If necessary we will not distinguish between a Turing machine and its code, a string in Σ^*. If $\mathbb{M}$ is a Turing machine we denote by $\|\mathbb{M}\|$ the length of its code.

Sometimes statements containing a formulation like "there is a $d \in \mathbb{N}$ such that for all $x \in \Sigma^*$: $\ldots \leq |x|^d$" can be wrong for $x \in \Sigma^*$ with $|x| \leq 1$. We trust the reader's common sense to interpret such statements reasonably.

If $\mathbb{A}$ is any (deterministic or nondeterministic) algorithm and $\mathbb{A}$ accepts x, then we denote by $t_{\mathbb{A}}(x)$ the number of steps of a shortest accepting run of $\mathbb{A}$ on x; if $\mathbb{A}$ does not accept x, then $t_{\mathbb{A}}(x)$ is not defined.

2.1. Parameterized complexity. We view *parameterized problems* as pairs (Q, κ) consisting of a classical problem $Q \subseteq \Sigma^*$ and a *parameterization* $\kappa\colon \Sigma^* \to \mathbb{N}$, which is required to be polynomial time computable. We will present parameterized problems in the form we did it for p-GÖDEL and p-ACC$_{\leq}$ in the Introduction.

A parameterized problem (Q, κ) is *fixed-parameter tractable* (or, in FPT) if $x \in Q$ is solvable by an *fpt-algorithm*, that is, by a deterministic algorithm running in time $f(\kappa(x)) \cdot |x|^{O(1)}$ for some computable $f\colon \mathbb{N} \to \mathbb{N}$.

Let C be a complexity class of classical complexity theory defined in terms of deterministic (nondeterministic) algorithms. A parameterized problem (Q, κ) is in the class $\mathrm{XC}_{\mathrm{uni}}$ if there is a deterministic (nondeterministic) algorithm deciding (accepting) Q and witnessing for every $k \in \mathbb{N}$ that the classical problem

$$(Q, \kappa)_k := \{x \in Q \mid \kappa(x) = k\},$$

the kth *slice* of (Q, κ), is in C. For example, (Q, κ) is in the class $\mathrm{XP}_{\mathrm{uni}}$ if there is a deterministic algorithm $\mathbb{A}$ deciding $x \in Q$ in time $|x|^{f(\kappa(x))}$ for some function $f\colon \mathbb{N} \to \mathbb{N}$. And (Q, κ) is in the class $\mathrm{XNP}_{\mathrm{uni}}$ if there is a nondeterministic algorithm $\mathbb{A}$ accepting Q such that for some function $f\colon \mathbb{N} \to \mathbb{N}$ we have $t_{\mathbb{A}}(x) \leq |x|^{f(\kappa(x))}$ for all $x \in Q$. Finally, a parameterized problem (Q, κ) is in the class co-$\mathrm{XC}_{\mathrm{uni}}$ if its complement $(\Sigma^* \setminus Q, \kappa)$ is in $\mathrm{XC}_{\mathrm{uni}}$.

We have added the subscript "uni" to the names of these classes to emphasize that they are classes of the so-called uniform parameterized complexity theory. If in the definition of $\mathrm{XP}_{\mathrm{uni}}$ and $\mathrm{XNP}_{\mathrm{uni}}$ we require the function f to be computable, then we get the corresponding classes of the strongly uniform theory. For example, FPT is a class of this theory.

A parameterized problem (Q, κ) is *slicewise monotone* if its instances have the form (x, n), where $x \in \Sigma^*$ and $n \in \mathbb{N}$ is given in unary, if $\kappa((x, n)) = |x|$, and finally if the slices are monotone, that is, for all $x \in \Sigma^*$ and $n, n' \in \mathbb{N}$

$$(x, n) \in Q \text{ and } n < n' \text{ imply } (x, n') \in Q.$$

We already remarked that the problems p-GÖDEL and p-$\mathrm{ACC}_{\leq}$ are slicewise monotone.

Clearly, every parameterized problem (Q, κ) with $Q \in \mathrm{NP}$ is in $\mathrm{XNP}_{\mathrm{uni}}$; thus we can replace co-$\mathrm{XNP}_{\mathrm{uni}}$ by $\mathrm{XNP}_{\mathrm{uni}} \cap$ co-$\mathrm{XNP}_{\mathrm{uni}}$ everywhere in Theorem 1 and Theorem 2.

3. Optimal proof systems

Let $Q \subseteq \Sigma^*$ be a problem. A *proof system for Q* is a surjective function $P\colon \Sigma^* \to Q$ computable in polynomial time. Then, if $P(w) = x$, we say that w is a *P-proof* of x. A proof system P for Q is *optimal* if for any other proof system P' for Q there is a polynomial $p \in \mathbb{N}[X]$ such that for every $x \in Q$, if x has a P'-proof of length n, then x has a P-proof of length $\leq p(n)$. Hence, any P'-proof can be translated into a P-proof by a nondeterministic polynomial time algorithm.

The corresponding deterministic concept is the notion of p-optimality. The proof system P for Q is *polynomially optimal* or *p-optimal* if for every proof system P' for Q there is a polynomial time computable $T\colon \Sigma^* \to \Sigma^*$ such that for all $w' \in \Sigma^*$

$$P(T(w')) = P'(w').$$

We list some known results. Part (1) and (2) are immediate from the definitions.

(1) Every p-optimal proof system is optimal.
(2) Every nonempty $Q \in$ PTIME has a p-optimal proof system, every nonempty $Q \in$ NP has an optimal proof system.
(3) ([8]) If Q is nonempty and $Q \leq^p Q'$ (that is, if Q is polynomial time reducible to Q') and Q' has a (p-)optimal proof system, then Q has a (p-)optimal proof system too.
(4) ([10]) Every Q hard for EXP = DTIME$\left(2^{n^{O(1)}}\right)$ under polynomial time reductions has no p-optimal proof system.

It is not known whether there is a problem $Q \notin$ P ($Q \notin$ NP) with a p-optimal (an optimal) proof system. As mentioned in the Introduction, Krajíček and Pudlák [9] conjectured that there is no optimal proof system for the set TAUT of tautologies.

Concerning (4) we did not find a corresponding result for optimal proof systems in the literature. We can show:

Proposition 3. *Every Q hard for* EEXP = DTIME$\left(2^{2^{n^{O(1)}}}\right)$ *under polynomial time reductions has no optimal proof system.*

We do not need this result (and will prove it in the full version of the paper). However we state a consequence:

Corollary 4. *There is no optimal proof system for the set of valid sentences of first-order logic.*

3.1. Almost optimal algorithms and enumerations of P-easy subsets. Let $Q \subseteq \Sigma^*$ be a problem. A deterministic (nondeterministic) algorithm $\mathbb{A}$ accepting Q is *almost optimal* or *optimal on positive instances of Q* if for every deterministic (nondeterministic) algorithm $\mathbb{B}$ accepting Q there is a polynomial $p \in \mathbb{N}[X]$ such that for all $x \in Q$

$$t_{\mathbb{A}}(x) \leq p(t_{\mathbb{B}}(x) + |x|).$$

By definition a subset Q' of Q is P-*easy* if $Q' \in$ P. An *enumeration of the* P-*easy subsets of Q by* P-*machines (by* NP-*machines)* is a computable function $M \colon \mathbb{N} \to \Sigma^*$ such that

(i) for every $i \in \mathbb{N}$ the string $M(i)$ is a deterministic (nondeterministic) Turing machine deciding (accepting) a P-easy subset of Q in polynomial time;
(ii) for every P-easy subset Q' of Q there is an $i \in \mathbb{N}$ such that $M(i)$ decides (accepts) Q'.

If in the nondeterministic case instead of (i) we only require

(i′) for every $i \in \mathbb{N}$ the string $M(i)$ is a nondeterministic Turing machine accepting a subset of Q in polynomial time,

we obtain the notion of a *weak enumeration of* P-*easy subsets of Q by* NP-*machines.*

We denote by TAUT the class of all tautologies of propositional logic. We need the following theorem:

Theorem 5. (1) *The following statements are equivalent:*
   (a) TAUT *has a p-optimal proof system.*
   (b) TAUT *has an almost optimal deterministic algorithm.*
   (c) TAUT *has an enumeration of the* P*-easy subsets by* P*-machines.*
(2) *The following statements are equivalent:*
   (a) TAUT *has an optimal proof system.*
   (b) TAUT *has an almost optimal nondeterministic algorithm.*
   (c) TAUT *has a weak enumeration of the* P*-easy subsets by* NP*-machines.*
   (d) TAUT *has an enumeration of the* P*-easy subsets by* NP*-machines.*

The equivalence of (a) and (b) in (1) and (2) is due to [9], the equivalence to (c) to [13]. The equivalence in (2) to (d) will be a by-product of the proof of Theorem 8; the equivalence was already claimed in [13] but its author was so kind to point out to us that he did not realize the difference between (c) and (d): some machines $M(i)$ of a weak enumeration might accept subsets of Q which are not P-easy (but only in NP).

4. Linking slicewise monotone problems and optimal proof systems

The following result yields a uniform bound on the complexity of slicewise monotone problems whose complements have optimal proof systems.

Theorem 6. *Let* (Q, κ) *be a slicewise monotone parameterized problem with decidable* Q*.*
(1) *If* $\Sigma^* \setminus Q$ *has a p-optimal proof system, then* $(Q, \kappa) \in \mathrm{XP_{uni}}$.
(2) *If* $\Sigma^* \setminus Q$ *has an optimal proof system, then* $(Q, \kappa) \in \text{co-XNP}_{\mathrm{uni}}$.

As by (3) on page 5 every nonempty problem in co-NP has a (p-)optimal proof system if TAUT has one, we immediately get:

Corollary 7. *Let* (Q, κ) *be a slicewise monotone parameterized problem with* Q *in* NP*.*
(1) *If* TAUT *has a p-optimal proof system, then* $(Q, \kappa) \in \mathrm{XP_{uni}}$.
(2) *If* TAUT *has an optimal proof system, then* $(Q, \kappa) \in \text{co-XNP}_{\mathrm{uni}}$.

Concerning Theorem 6 (1) we should mention that Monroe [11] has shown that if the complement of (the classical problem underlying) p-$\textsc{Acc}_{\leq}$ has an almost optimal algorithm (which by [9] holds if it has a p-optimal proof system), then p-$\textsc{Acc}_{\leq} \in \mathrm{XP_{uni}}$.

Proof of Theorem 6: We present the proof for (2), the proof for (1) is obtained by the obvious modifications. Let (Q, κ) be slicewise monotone and let $\mathbb{Q}$ be a deterministic algorithm deciding Q. Assume that $\Sigma^* \setminus Q$ has an optimal proof system. It is well-known [9] that then $\Sigma^* \setminus Q$ has an almost optimal nondeterministic algorithm $\mathbb{O}$. We have to show that $(Q, \kappa) \in \text{co-XNP}_{\mathrm{uni}}$.

Let $\mathbb{S}$ be the algorithm that, on $x \in \Sigma^*$, by systematically applying $\mathbb{Q}$ to the inputs $(x, 0), (x, 1), \ldots$ computes

$$n(x) := \text{ the least } n \text{ such that } (x, n) \in Q.$$

If $(x, n) \notin Q$ for all $n \in \mathbb{N}$, then $n(x)$ is not defined and $\mathbb{S}$ does not stop. We show that the following algorithm $\mathbb{A}$ witnesses that $(\Sigma^* \setminus Q, \kappa) \in \mathrm{XNP_{uni}}$.

```
A(x, n)    // x ∈ Σ*, n ∈ ℕ in unary
  1. In parallel simulate S on input x and O on input (x, n)
  2.     if O accepts then accept
  3.     if S stops, then
  4.         if n < n(x) then accept else reject.
```

By our assumptions on $\mathbb{O}$ and $\mathbb{S}$ and the slicewise monotonicity of Q, it should be clear that $\mathbb{A}$ accepts $\Sigma^* \setminus Q$. We have to show that $\mathbb{A}$ does it in the time required by $\mathrm{XNP_{uni}}$. Hence, we have to determine the running time of $\mathbb{A}$ on inputs $(x, n) \notin Q$.

Case "$(x, \ell) \notin Q$ for all $\ell \in \mathbb{N}$": In this case $\mathbb{S}$ on input x does not stop. Hence, the running time of $\mathbb{A}$ on input (x, n) is determined by $\mathbb{O}$. The following algorithm $\mathbb{O}_x$ accepts $\Sigma^* \setminus Q$: on input (y, ℓ) the algorithm $\mathbb{O}_x$ checks whether $y = x$. If so, it accepts and otherwise it runs $\mathbb{O}$ on input (y, ℓ) and answers accordingly. Clearly, for all $\ell \in \mathbb{N}$

$$t_{\mathbb{O}_x}((x, \ell)) \leq O(|x|).$$

As $\mathbb{O}$ is almost optimal, we know that there is a constant $d_x \in \mathbb{N}$ (depending on x) such that for all $(y, \ell) \in \Sigma^* \setminus Q$

$$t_{\mathbb{O}}((y, \ell)) \leq \big(|(y, \ell)| + t_{\mathbb{O}_x}((y, \ell))\big)^{d_x}.$$

In particular, we have

$$t_{\mathbb{A}}((x, n)) = O(t_{\mathbb{O}}((x, n))) \leq O\Big(\big(|(x, n)| + O(|x|)\big)^{d_x}\Big) \leq n^{d'_x}$$

for some constant $d'_x \in \mathbb{N}$ (depending on x).

Case "$(x, \ell) \in Q$ for some $\ell \in \mathbb{N}$": Then $\mathbb{S}$ will stop on input x. Thus, in the worst case, $\mathbb{A}$ on input (x, n) has to wait till the simulation of $\mathbb{S}$ on x stops and then $\mathbb{A}$ must check whether the result $n(x)$ of the computation of $\mathbb{S}$ is bigger than n or not and answer according to Line 4. So in the worst case $\mathbb{A}$ takes time $O(t_{\mathbb{S}}(x) + O(n)) \leq n^{O(t_{\mathbb{S}}(x))}$. □

We show the equivalence in the first line of Theorem 2:

Theorem 8. (1) TAUT *has a p-optimal proof system iff* p-$\mathrm{Acc}_{\leq} \in \mathrm{XP_{uni}}$.
(2) TAUT *has an optimal proof system iff* p-$\mathrm{Acc}_{\leq} \in \text{co-}\mathrm{XNP_{uni}}$.

Proof. Again we only prove (2) and by the previous corollary it suffices to show the corresponding implication from "right to left."

So assume that the complement of p-ACC$_{\leq}$ is in XNP$_{\rm uni}$ and let $\mathbb{A}$ be a nondeterministic algorithm witnessing it; in particular, $t_{\mathbb{A}}((\mathbb{M}, n)) \leq n^{f(\|\mathbb{M}\|)}$ for some function f and all $(\mathbb{M}, n) \notin p$-ACC$_{\leq}$. We show that TAUT has an enumeration of the P-easy subsets by NP-machines (and this suffices by Theorem 5).

We fix a deterministic Turing machine $\mathbb{M}_0$ that given a propositional formula α and an assignment checks if this assignment satisfies α in time $|\alpha|^2$.

For a deterministic Turing machine $\mathbb{M}$ let $\mathbb{M}^*$ be the nondeterministic machine that on empty input tape

- first guesses a propositional formula α;
- then checks (by simulating $\mathbb{M}$) whether $\mathbb{M}$ accepts α and rejects if this is not the case;
- finally guesses an assignment and accepts if this assignment does not satisfy α (this is checked by simulating $\mathbb{M}_0$).

A deterministic Turing machine $\mathbb{M}$ is *clocked* if (the code of) $\mathbb{M}$ contains a natural number time$(\mathbb{M})$ such that $n^{\text{time}(\mathbb{M})}$ is a bound for the running time of $\mathbb{M}$ on inputs of length n (in particular, a clocked machine is a polynomial time one).

Finally, for a clocked Turing machine $\mathbb{M}$ let $\mathbb{M}^+$ be the nondeterministic Turing machine that on input α accepts if and only if (i) and (ii) hold:

(i) $\mathbb{M}$ accepts α;
(ii) $(\mathbb{M}^*, |\alpha|^{\text{time}(\mathbb{M})+4}) \notin p$-ACC$_{\leq}$.

The machine $\mathbb{M}^+$ checks (i) by simulating $\mathbb{M}$ and (ii) by simulating $\mathbb{A}$. Hence, if $\mathbb{M}^+$ accepts α, then

$$t_{\mathbb{M}^+}(\alpha) \leq O\left(|\alpha|^{\text{time}(\mathbb{M})} + t_{\mathbb{A}}\big((\mathbb{M}^*, |\alpha|^{\text{time}(\mathbb{M})+4})\big)\right),$$

and as $t_{\mathbb{A}}\big((\mathbb{M}^*, |\alpha|^{\text{time}(\mathbb{M})+4})\big) \leq |\alpha|^{(\text{time}(\mathbb{M})+4)\cdot f(\|\mathbb{M}^*\|)}$, the Turing machine $\mathbb{M}^+$ accepts in time polynomial in $|\alpha|$.

We show that $\mathbb{M}^+$, where $\mathbb{M}$ ranges over all clocked machines, yields an enumeration of all P-easy subsets of TAUT by NP-machines. First let $\mathbb{M}$ be a clocked machine. We prove that $\mathbb{M}^+$ accepts a P-easy subset of TAUT.

$\mathbb{M}^+$ *accepts a subset of* TAUT: If $\mathbb{M}^+$ accepts α, then, by (i), $\mathbb{M}$ accepts α and by (ii), $(\mathbb{M}^*, |\alpha|^{\text{time}(\mathbb{M})+4}) \notin p$-ACC$_{\leq}$. Therefore, by definition of $\mathbb{M}^*$, every assignment satisfies α and hence $\alpha \in$ TAUT.

$\mathbb{M}^+$ *accepts a P-easy set*: If $(\mathbb{M}^*, m) \in p$-ACC$_{\leq}$ for some m, then, by slicewise monotonicity of p-ACC$_{\leq}$, the machine $\mathbb{M}^+$ accepts a finite set and hence a P-easy set. If $(\mathbb{M}^*, m) \notin p$-ACC$_{\leq}$ for all m, then $\mathbb{M}^+$ accepts exactly those α accepted by $\mathbb{M}$; as $\mathbb{M}$ is clocked, this is a set in P.

Now let $Q \subseteq$ TAUT be a P-easy subset of TAUT and let $\mathbb{M}$ be a clocked machine deciding Q. Then $\mathbb{M}^+$ accepts Q. □

5. Slicewise monotone parameterized problems

In this section we observe that p-$\textsc{Acc}_{\leq}$ is a complete problem in the class of slicewise monotone parameterized problems with underlying classical problem in NP. Furthermore, we shall see that in Theorem 8 we can replace the problem p-$\textsc{Acc}_{\leq}$ by other slicewise monotone parameterized problems (among them p-$\textsc{Gödel}$) by showing for them that they are in the class $\mathrm{XP}_{\mathrm{uni}}$ (co-$\mathrm{XNP}_{\mathrm{uni}}$) if and only if p-$\textsc{Acc}_{\leq}$ is.

5.1. The complexity of slicewise monotone problems. We start with some remarks on the complexity of slicewise monotone problems. In [1, 2] we have shown that p-$\textsc{Acc}_{\leq}$ and p-$\textsc{Gödel}$ are not fixed-parameter tractable if "P $\neq$ NP holds for all time constructible and increasing functions," that is, if $\mathrm{DTIME}(h^{O(1)}) \neq \mathrm{NTIME}(h^{O(1)})$ for all time constructible and increasing functions $h\colon \mathbb{N} \to \mathbb{N}$. However:

Proposition 9. (1) [2] *Let (Q, κ) be slicewise monotone. Then (Q, κ) is non-uniformly fixed-parameter tractable, that is, there is a $c \in \mathbb{N}$, a function $f\colon \mathbb{N} \to \mathbb{N}$, and for every k an algorithm deciding the slice $(Q, \kappa)_k$ in time $f(k) \cdot n^c$.*

(2) *Let (Q, κ) be slicewise monotone with enumerable Q. Then $(Q, \kappa) \in \mathrm{XNP}_{\mathrm{uni}}$.*

Proof. (2) Let $\mathbb{Q}$ be an algorithm enumerating Q. The following algorithm shows that $(Q, \kappa) \in \mathrm{XNP}_{\mathrm{uni}}$: On input (x, n) it guesses $m \in \mathbb{N}$ and a string c. If c is the code of an initial segment of the run of $\mathbb{Q}$ enumerating (x, m), then it accepts if $m \leq n$. □

We remark that there are slicewise monotone problems with underlying classical problem of arbitrarily high complexity that are fixed-parameter tractable. In fact, let $Q_0 \subseteq \Sigma^*$ be decidable. Then the slicewise monotone (Q, κ) with

$$Q := \{(x, n) \mid x \in Q_0,\ n \in \mathbb{N}, \text{ and } |x| \leq n\}$$

(and $\kappa((x, n)) := |x|$) is in FPT.

To compare the complexity of parameterized problems we use the standard notions of reduction that we recall first. Let (Q, κ) and (Q', κ') be parameterized problems. We write $(Q, \kappa) \leq^{\mathrm{fpt}} (Q', \kappa')$ if there is an *fpt-reduction* from (Q, κ) to (Q', κ'), that is, a mapping $R\colon \Sigma^* \to \Sigma^*$ with:

(1) For all $x \in \Sigma^*$ we have $(x \in Q \iff R(x) \in Q')$.
(2) $R(x)$ is computable in time $f(\kappa(x)) \cdot |x|^{O(1)}$ for some computable $f\colon \mathbb{N} \to \mathbb{N}$.
(3) There is a computable function $g\colon \mathbb{N} \to \mathbb{N}$ such that $\kappa'(R(x)) \leq g(\kappa(x))$ for all $x \in \Sigma^*$.

We write $(Q, \kappa) \leq^{\mathrm{xp}} (Q', \kappa')$ if there is an *xp-reduction* from (Q, κ) to (Q', κ'), which is defined as $(Q, \kappa) \leq^{\mathrm{fpt}} (Q', \kappa')$ except that instead of (2) it is only required that $R(x)$ is computable in time $|x|^{f(\kappa(x))}$ for some computable $f\colon \mathbb{N} \to \mathbb{N}$.

These are notions of reductions of the usual (strongly uniform) parameterized complexity theory. We get the corresponding notions $\le^{\rm fpt}_{\rm uni}$ and $\le^{\rm xp}_{\rm uni}$ by allowing the functions f and g to be arbitrary (and not necessarily computable).

We shall use the following simple observation.

Lemma 10. *If* $(Q, \kappa) \le^{\rm xp}_{\rm uni} (Q', \kappa')$ *and* $(Q', \kappa') \in {\rm XP}_{\rm uni}$*, then* $(Q, \kappa) \in {\rm XP}_{\rm uni}$*. The same holds for* ${\rm XNP}_{\rm uni}$ *instead of* ${\rm XP}_{\rm uni}$*.*

We turn again to slicewise monotone problems. Among these problems with underlying classical problem in NP the problem p-ACC$_\le$ is of highest complexity.

Proposition 11. *Let* (Q, κ) *be slicewise monotone and* $Q \in {\rm NP}$*. Then*

$$(Q, \kappa) \le^{\rm fpt} p\text{-}\textsc{Acc}_{\le}.$$

Note that this result together with Theorem 8 (2) yields Theorem 1.

Proof of Proposition 11: Let $\mathbb{M}$ be a nondeterministic Turing machine accepting Q. We may assume that for some $d \in \mathbb{N}$ the machine $\mathbb{M}$ on input (x, n) performs exactly $|(x, n)|^d$ steps. For $x \in \Sigma^*$ let $\mathbb{M}_x$ be the nondeterministic Turing machine that on empty input tape, first writes x on the tape, then guesses a natural number m, and finally simulates the computation of $\mathbb{M}$ on input (x, m). We can assume that there is a polynomial time computable function h such that $\mathbb{M}_x$ makes exactly $h(x, m) \in O\big(|x| + m + |(x, m)|^d\big)$ steps if it chooses the natural number m. Furthermore we can assume that $h(x, m) < h(x, m')$ for $m < m'$.

Then $(x, n) \mapsto \big(\mathbb{M}_x, h(x, n)\big)$ is an fpt-reduction from (Q, κ) to p-ACC$_\le$: Clearly, if $(x, n) \in Q$ then $\big(\mathbb{M}_x, h(x, n)\big) \in p$-ACC$_\le$ by construction of $\mathbb{M}_x$. Conversely, if $\big(\mathbb{M}_x, h(x, n)\big) \in p$-ACC$_\le$, then by the properties of h we see that $\mathbb{M}$ accepts (x, m) for some $m \le n$. Thus, $(x, m) \in Q$ and therefore $(x, n) \in Q$ by slicewise monotonicity. □

Later on we shall use the following related result.

Proposition 12. *Let* (Q, κ) *be slicewise monotone and assume that there is a nondeterministic algorithm* $\mathbb{A}$ *accepting* Q *such that* $t_{\mathbb{A}}(x, n) \le n^{f(|x|)}$ *for some time constructible* f *and all* $(x, n) \in Q$*. Then*

$$(Q, \kappa) \le^{\rm xp} p\text{-}\textsc{Acc}_{\le}.$$

Proof. Let (Q', κ') be the problem

Instance:	$x \in \Sigma^*$ and $m \in \mathbb{N}$ in unary.
Parameter:	$\vert x\vert$.
Problem:	Is there an $n \in \mathbb{N}$ such that $n^{f(\vert x\vert)} \le m$ and $(x, n) \in Q$?

By the previous proposition we get our claim once we have shown:

(1) (Q', κ') is slicewise monotone and $Q' \in {\rm NP}$.
(2) $(Q, \kappa) \le^{\rm xp} (Q', \kappa')$

To see (1) let $\mathbb{A}$ be as stated above and let $\mathbb{T}$ an algorithm witnessing the time constructibility of f; that is, $\mathbb{T}$ on input $k \in \mathbb{N}$ computes $f(k)$ in exactly $f(k)$ steps. An algorithm $\mathbb{B}$ witnessing that $Q' \in \mathrm{NP}$ runs as follows on input (x, m):

– $\mathbb{B}$ guesses $n \in \mathbb{N}$;
– if $n = 1$, the algorithm $\mathbb{B}$ rejects in case $m = 0$;
  if $n \geq 2$, the algorithm $\mathbb{B}$ simulates m steps of the computation of $\mathbb{T}$ on input $|x|$; if thereby $\mathbb{T}$ does not stop, $\mathbb{B}$ rejects; otherwise, the simulation yields $f(|x|)$ and $\mathbb{B}$ checks whether $n^{f(|x|)} > m$ (this can be detected in time $O(m)$); in the positive case $\mathbb{B}$ rejects;
– finally $\mathbb{B}$ simulates the computation of $\mathbb{A}$ on (x, n) and answers accordingly.

(2) Note that the mapping $(x, n) \mapsto \big(x, n^{f(|x|)}\big)$ is an xp-reduction. $\square$

5.2. Slicewise monotone problems related to logic. In the next section we will use some further slicewise monotone problems related to first-order logic and least fixed-point logic that we introduce now.

We assume familiarity with *first-order logic* FO and its extension *least fixed-point logic* LFP (e.g, see [5]). We denote by $\mathrm{FO}[\tau]$ and $\mathrm{LFP}[\tau]$ the set of sentences of vocabulary τ of FO and of LFP, respectively. In this paper all vocabularies are finite sets of relational symbols.

If the structure $\mathcal{A}$ is a model of the LFP-sentence φ we write $\mathcal{A} \models \varphi$. We only consider structures $\mathcal{A}$ with finite universe A. The size $\|\mathcal{A}\|$ of the structure $\mathcal{A}$ is the length of a reasonable encoding of $\mathcal{A}$ as string in Σ^*. An algorithm based on the inductive definition of the satisfaction relation for LFP shows (see [14]):

Proposition 13. *The model-checking problem $\mathcal{A} \models \varphi$ for structures $\mathcal{A}$ and* LFP-*sentences φ can be solved in time $\|\mathcal{A}\|^{O(|\varphi|)}$.*

Let $L = \mathrm{FO}$ or $L = \mathrm{LFP}$. First we introduce the parameterized problem

p-L-MODEL	
Instance:	An L-sentence φ and $n \in \mathbb{N}$ in unary.
Parameter:	$\lvert\varphi\rvert$.
Problem:	Is there a structure $\mathcal{A}$ with $\mathcal{A} \models \varphi$ and $\lvert A\rvert \leq n$?

Here, $|A|$ denotes the size of the universe A of $\mathcal{A}$. For every vocabulary τ we let $\tau_< := \tau \cup \{<\}$, where $<$ is a binary relation symbol not in τ. For $m \geq 1$ we say that an $L[\tau_<]$-sentence φ is $\leq m$*-invariant* if for all τ-structures $\mathcal{A}$ with $|A| \leq m$ we have

$$(\mathcal{A}, <_1) \models \varphi \iff (\mathcal{A}, <_2) \models \varphi$$

for all orderings $<_1$ and $<_2$ on A.

Finally we introduce the slicewise monotone parameterized problem

p-L-NOT-INV	
Instance:	A vocabulary τ, an $L[\tau_<]$-sentence φ and $m \geq 1$ in unary.
Parameter:	$\lvert\varphi\rvert$.
Problem:	Is φ not $\leq m$-invariant?

5.3. Membership in $\mathrm{XP}_{\mathrm{uni}}$ and co-$\mathrm{XNP}_{\mathrm{uni}}$. Concerning membership in the classes $\mathrm{XP}_{\mathrm{uni}}$ and co-$\mathrm{XNP}_{\mathrm{uni}}$ all the slicewise monotone problems we have introduced behave in the same way:

Proposition 14. *Consider the parameterized problems*

p-GÖDEL, p-FO-MODEL, p-LFP-MODEL, p-FO-NOT-INV,
p-LFP-NOT-INV, *and* p-ACC$_{\leq}$.

If one of the problems is in $\mathrm{XP}_{\mathrm{uni}}$, *then all are; if one of the problems is in* co-$\mathrm{XNP}_{\mathrm{uni}}$, *then all are.*

By Theorem 8 this result yields Theorem 2. We prove it with Lemmas 15–18.

Lemma 15. ([2]) p-GÖDEL $\leq^{\mathrm{fpt}}$ p-FO-MODEL.

Lemma 16. *Let* $L = \mathrm{FO}$ *or* $L = \mathrm{LFP}$. *Then* p-L-MODEL $\leq^{\mathrm{fpt}}$ p-L-NOT-INV.

Proof. Let φ be a sentence of vocabulary τ We set $\tau' := \tau \cup \{P\}$ with a new unary relation symbol P and consider the sentence of vocabulary $\tau'_<$

$$\psi(\varphi) := \varphi \wedge \text{“}P \text{ holds for the first element of } <.\text{”}$$

Clearly, for every $n \geq 2$

$$(\varphi, n) \in p\text{-FO-MODEL} \iff \big(\psi(\varphi), n\big) \in p\text{-FO-NOT-INV}$$

and the same equivalence holds for p-LFP-MODEL and p-LFP-NOT-INV. Thus $(\varphi, n) \mapsto \big(\psi(\varphi), n\big)$ is the desired reduction in both cases. □

Lemma 17. p-LFP-NOT-INV $\leq^{\mathrm{xp}}$ p-ACC$_{\leq}$

Proof. Consider the algorithm $\mathbb{A}$ that on input (φ, m), where φ is an LFP-sentence and $m \geq 1$, guesses a structure $\mathcal{A}$ and two orderings $<_1$ and $<_2$ and accepts if $|A| \leq m$, $(\mathcal{A}, <_1) \models \varphi$, and $(\mathcal{A}, <_2) \models \neg\varphi$. Then, by Proposition 13, the algorithm $\mathbb{A}$ witnesses that p-LFP-NOT-INV satisfies the assumptions on (Q, κ) in Proposition 12. This yields the claim. □

Lemma 18. (1) *If* p-GÖDEL $\in \mathrm{XP}_{\mathrm{uni}}$, *then* p-ACC$_{\leq} \in \mathrm{XP}_{\mathrm{uni}}$.
(2) *If* p-GÖDEL $\in$ co-$\mathrm{XNP}_{\mathrm{uni}}$, *then* p-ACC$_{\leq} \in$ co-$\mathrm{XNP}_{\mathrm{uni}}$.

Proof. We give the proof of (2). By standard means we showed in [2, Lemma 7] that there exists a $d \in \mathbb{N}$ and a polynomial time algorithm that assigns to every nondeterministic Turing machine $\mathbb{M}$ a first-order sentence $\varphi_{\mathbb{M}}$ such that for $n \in \mathbb{N}$

(1) $$(\mathbb{M}, n) \in p\text{-ACC}_{\leq} \implies (\varphi_{\mathbb{M}}, n^d) \in p\text{-GÖDEL}.$$

Moreover,

(2) $\varphi_{\mathbb{M}}$ has a proof $\implies$ $\mathbb{M}$ accepts the empty input tape.

Now assume that $\mathbb{A}$ is an algorithm that witnesses that the complement of p-Gödel is in XNP_{uni}. We may assume that every run of $\mathbb{A}$ either accepts its input or is infinitely long. Let $d \in \mathbb{N}$ be as above. We present an algorithm $\mathbb{B}$ showing that the complement of p-$\text{Acc}_{\leq}$ is in XNP_{uni}. On input $(\mathbb{M}, n)$ the algorithm $\mathbb{B}$ first computes $\varphi_{\mathbb{M}}$ and then runs two algorithms in parallel:

- a brute force algorithm that on input $\mathbb{M}$ searches for the least $n_{\mathbb{M}}$ such that $\mathbb{M}$ on empty input tape has an accepting run of length $n_{\mathbb{M}}$;
- the algorithm $\mathbb{A}$ on input $(\varphi_{\mathbb{M}}, n^d)$.

If the brute force algorithm halts first and outputs $n_{\mathbb{M}}$, then $\mathbb{B}$ checks whether $n_{\mathbb{M}} \leq n$ and answers accordingly.

Assume now that $\mathbb{A}$ halts first. Then $\mathbb{A}$ accepts $(\varphi_{\mathbb{M}}, n^d)$ and $\big((\varphi_{\mathbb{M}}, n^d) \notin$ p-Gödel and hence $(\mathbb{M}, n) \notin p$-$\text{Acc}_{\leq}$ by (1) and therefore$\big)$ $\mathbb{B}$ accepts.

The algorithm $\mathbb{B}$ accepts the complement of p-$\text{Acc}_{\leq}$; note that if no run of $\mathbb{A}$ accepts $(\varphi_{\mathbb{M}}, n^d)$, then $(\varphi_{\mathbb{M}}, n^d) \in p$-Gödel and therefore $\mathbb{M}$ accepts the empty input tape by (2), so that in this case the computation of the brute force algorithm eventually will stop.

It remains to see that $\mathbb{B}$ accepts the complement of p-$\text{Acc}_{\leq}$ in the time required by XNP_{uni}. We consider two cases.

$\mathbb{M}$ *halts on empty input tape*: Then an upper bound for the running time is given by the time that the brute force algorithm needs to compute $n_{\mathbb{M}}$ (and the time for the check whether $n_{\mathbb{M}} \leq n$); hence we have an upper bound of the form $n^{c_{\mathbb{M}}}$.

$\mathbb{M}$ *does not halt on empty input tape*: Then, by (2), we have $(\varphi_{\mathbb{M}}, n^d) \notin p$-Gödel; hence an upper bound is given by the running time of $\mathbb{A}$ on input $(\varphi_{\mathbb{M}}, n^d)$. □

It should be clear that Lemmas 15–18 together with Lemma 10 yield a proof of Proposition 14.

6. Optimal algorithms and the logic $L_{\leq}$

In this section we interpret Theorem 2 in terms of the expressive power of a certain logic.

For our purposes a *logic* L consists

- for every vocabulary τ of a set $L[\tau]$ of strings, the set of *L-sentences of vocabulary* τ and of an algorithm that for every vocabulary τ and every string ξ decides whether $\xi \in L[\tau]$ (in particular, $L[\tau]$ is decidable for every τ);
- of a *satisfaction relation* $\models_L$; if $(\mathcal{A}, \varphi) \in \models_L$, written $\mathcal{A} \models_L \varphi$, then $\mathcal{A}$ is a τ-structure and $\varphi \in L[\tau]$ for some vocabulary τ; furthermore for each

$\varphi \in L[\tau]$ the class $\mathrm{Mod}_L(\varphi) := \{\mathcal{A} \mid \mathcal{A} \models_L \varphi\}$ of *models of* φ is closed under isomorphisms.

Definition 19. Let L be a logic.

(a) *L is a logic for* P if for all vocabularies τ and all classes C (of encodings) of τ-structures closed under isomorphisms we have

$$C \in \mathrm{P} \iff C = \mathrm{Mod}_L(\varphi) \text{ for some } \varphi \in L[\tau].$$

(b) *L is a* P*-bounded logic for* P if (a) holds and if there is an algorithm $\mathbb{A}$ deciding $\models_L$ (that is, for every structure $\mathcal{A}$ and L-sentence φ the algorithm $\mathbb{A}$ decides whether $\mathcal{A} \models_L \varphi$) and if moreover, for fixed φ the algorithm $\mathbb{A}$ runs in time polynomial in $\|\mathcal{A}\|$.

The relationship of these concepts with topics of this paper is already exemplified by the following simple observation.

Proposition 20. *Let L be a logic for* P *and define p-$\models_L$ by*

p-$\models_L$	
Instance:	A structure $\mathcal{A}$ and an L-sentence φ.
Parameter:	$\lvert\varphi\rvert$.
Problem:	Is $\mathcal{A} \models_L \varphi$

Then L is a P*-bounded logic for* P *if and only if p-$\models_L \in \mathrm{XP}_{\mathrm{uni}}$.*

This relationship suggests the following definition.

Definition 21. *L is an* NP*-bounded logic for* P if it is a logic for P and p-$\models_L \in \mathrm{XNP}_{\mathrm{uni}}$.

We introduce the logic $L_{\leq}$, a variant of LFP.[3] For every vocabulary τ we set

$$L_{\leq}[\tau] = \mathrm{LFP}[\tau_<]$$

(recall that $\tau_< := \tau \cup \{<\}$, with a new binary $<$) and define the semantics by

$$\mathcal{A} \models_{L_\leq} \varphi \iff \Big(\varphi \text{ is } \leq |A|\text{-invariant and } (\mathcal{A}, <) \models_{\mathrm{LFP}} \varphi \text{ for some ordering } < \text{ on } A\Big).$$

Hence, by the previous proposition and the definition of $\models_{L_\leq}$, we get:

Proposition 22. (1) *The following statements are equivalent:*

- $L_\leq$ *is a* P*-bounded logic for* P.
- p-$\models_{L_\leq} \in \mathrm{XP}_{\mathrm{uni}}$.
- p-LFP-Not-Inv $\in \mathrm{XP}_{\mathrm{uni}}$.

[3]In this section, if the structure $\mathcal{B}$ is a model of an LFP-sentence φ we write $\mathcal{A} \models_{\mathrm{LFP}} \varphi$ instead of $\mathcal{A} \models \varphi$.

(2) *The following statements are equivalent:*
   – $L_\leq$ *is an* NP*-bounded logic for* P.
   – $p\text{-}\models_{L_\leq} \in \mathrm{XNP}_{\mathrm{uni}}$.
   – p-LFP-Not-Inv $\in$ co-$\mathrm{XNP}_{\mathrm{uni}}$.

By Theorem 2 and Proposition 14 we get:

Theorem 23. TAUT *has an optimal proof system if and only if* $L_\leq$ *is an* NP*-bounded logic for* P.

Hence, if TAUT has an optimal proof system, then there is an NP-enumeration of P-easy classes of graphs closed under isomorphisms. We do not define the concept of NP-enumeration explicitly, however the enumeration obtained by applying the algorithm in $\mathrm{XNP}_{\mathrm{uni}}$ for $p\text{-}\models_{L_\leq}$ to the classes $\mathrm{Mod}_{L_\leq}(\varphi(\textsc{graph}) \wedge \psi)$, where $\varphi(\textsc{graph})$ axiomatizes the class of graphs and ψ ranges over all sentences of $L_\leq$ in the language of graphs, is such an NP-enumeration. Note that even without the assumption that TAUT has an optimal proof system we know that there is such an NP-enumeration of P-easy classes of graphs closed under isomorphisms, as the following variant $L_\leq(\mathrm{not})$ of $L_\leq$ is an NP-bounded logic for P. The logic $L_\leq(\mathrm{not})$ has the same syntax as $L_\leq$ and the semantics is given by the following clause:

$$\mathcal{A} \models_{L_\leq(\mathrm{not})} \varphi \iff \quad \text{not } \mathcal{A} \models_{L_\leq} \varphi.$$

As the class P is closed under complements, $L_\leq(\mathrm{not})$ is a logic for P. And $L_\leq(\mathrm{not})$ is an NP-bounded logic for P, as p-LFP-Not-Inv $\in \mathrm{XNP}_{\mathrm{uni}}$.

Acknowledgement. This research has been partially supported by the National Nature Science Foundation of China (60970011), the Sino-German Center for Research Promotion (GZ400), and the John Templeton Foundation through Project # 13152, the Infinity Project at the Centre de Recerca Matemàtica.

References

[1] Y. Chen and J. Flum. A logic for PTIME and a parameterized halting problem. In *Proceedings of the 24th IEEE Symposium on Logic in Computer Science (LICS'09)*, pages 397–406, 2009.

[2] Y. Chen and J. Flum. On the complexity of Gödel's proof predicate. *The Journal of Symbolic Logic*, 75(1): 239–254, 2010.

[3] Y. Chen and J. Flum. On p-optimal proof systems and logics for PTIME. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP'10, Track B)*, Lecture Notes in Computer Science 6199, pages 321–332, 2010.

[4] S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44:36–50, 1979.

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

[6] K. Gödel. *Collected Works*, vol. VI, 372–376, Clarendon Press, 2003.

[7] Y. Gurevich. Logic and the challenge of computer science. In *Current Trends in Theoretical Computer Science*, Computer Science Press, 1–57, 1988.

[8] J. Köbler and J. Messner. Complete problems for promise classes by optimal proof systems for test sets. In *Proceedings of the 13th IEEE Conference on Computational Complexity (CCC' 98)*, 132–140, 1998.
[9] J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54:1063–1088, 1989.
[10] J. Messner. On optimal algorithms and optimal proof systems. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science (STACS'99)*, Lecture Notes in Computer Science 1563:361–372, 1999.
[11] H. Monroe. Speedup for natural problems. *Electronic Colloquium on Computational Complexity*, Report TR09-056, 2009.
[12] A. Nash, J. Remmel, and V. Vianu. PTIME queries revisited. In *Proceedings of the 10th International Conference on Database Theory (ICDT'05)*, T. Eiter and L. Libkin (eds.), Lecture Notes in Computer Science 3363:274–288, 2005.
[13] Z. Sadowski. On an optimal propositional proof system and the structure of easy subsets. *Theoretical Computer Science*, 288(1):181–193, 2002.
[14] M.Y. Vardi. On the complexity of bounded-variable queries. In *Proceedings of the 14th ACM Symposium on Principles of Database Systems (PODS'95)*, pages 266–276, 1995.

Yijia Chen
Department of Computer Science
Shanghai Jiaotong University
China
E-mail address: yijia.chen@cs.sjtu.edu.cn

Jörg Flum
Mathematisches Institut
Albert-Ludwigs Universität Freiburg
Germany
E-mail address: joerg.flum@math.uni-freiburg.de