Electronic Colloquium on Computational Complexity, Report No. 8 (2010)

On optimal proof systems and logics for PTIME

Yijia Chen
Department of Computer Science
Shanghai Jiaotong University
yijia.chen@cs.sjtu.edu.cn

Jörg Flum
Mathematisches Institut
Albert-Ludwigs Universität Freiburg
joerg.flum@math.uni-freiburg.de

January 13, 2010

Abstract

We prove that TAUT has a p-optimal proof system if and only if $L_{\leq}$, a logic introduced in [11], is a P-bounded logic for P. Furthermore, using the method developed in [4], we show that TAUT has no *effective* p-optimal proof system under some reasonable complexity-theoretic assumption.

1. Introduction

As the title already indicates, this paper relates two topics which at first glance seem to be unrelated. On the one hand we consider optimal proof systems. A *proof system* in the sense of Cook and Reckhow [5], say for the class TAUT of tautologies of propositional logic, is a polynomial time computable function defined on $\{0,1\}^*$ and with TAUT as range. A proof system is *p-optimal* if it simulates any other proof system in polynomial time appropriately.[1] In their fundamental paper [14] Krajíček and Pudlák derive a series of statements equivalent to the existence of a p-optimal proof system for TAUT and state the conjecture:

Conjecture 1. There is no p-optimal proof system for TAUT.

On the other hand, the question of whether there is a logic capturing polynomial time remains the central open problem in descriptive complexity. There are artificial logics capturing polynomial time, but they do not fulfill a natural requirement to logics in this context:

> There is an algorithm that decides whether $\mathcal{A}$ is a model of φ for all structures $\mathcal{A}$ and sentences φ of the logic and that does this for fixed φ in time polynomial in the size $\|\mathcal{A}\|$ of $\mathcal{A}$. (1)

If this condition is fulfilled for a logic capturing polynomial time, we speak of a P-bounded logic for P. In [11] Gurevich states the conjecture:

Conjecture 2. There is no P-bounded logic for P.

The conjecture is false if one waives the effectivity condition (1). This is shown in [11, Section 7, CLAIM 2]) by considering a logic introduced by Blass and Gurevich and which we denote by $L_{\leq}$. Essentially, for any vocabulary, the sentences of $L_{\leq}$ are the sentences of least fixed-point logic in a vocabulary with an additional binary relation symbol for orderings. In $L_{\leq}$ for a structure $\mathcal{A}$ to be a model of φ it is required that in all structures of cardinality less than or equal to that of $\mathcal{A}$, the validity of φ (as a sentence of least fixed-point logic) does not depend on the chosen ordering.

As $L_{\leq}$ satisfies all other requirements of a P-bounded logic for P, Gurevich implicitly states the conjecture:

Conjecture 2a. $L_{\leq}$ is not a P-bounded logic for P.

The main result of this paper (cf. Theorem 7) tells us that

$$\text{Conjecture 1 is true} \iff \text{Conjecture 2a is true.} \tag{2}$$

[1] All notions will be defined in a precise manner in Section 2.

ISSN 1433-8092

We mentioned that at first glance "p-optimal proof systems for TAUT" and "logics for P" seem to be unrelated topics. However, there are reformulations of Conjecture 1 and Conjecture 2 that are alike. In fact, it is known [17] that TAUT has a p-optimal proof system if and only if there is a (computable) enumeration of all subsets of TAUT that are in P by means of Turing machines that decides them. And it is not hard to see that there is a P-bounded logic for P if and only if there is an enumeration of all polynomial time decidable classes of graphs closed under isomorphisms, again an enumeration in terms of Turing machines that decide these classes. In fact the question for a logic for P was stated in this way by Chandra and Harel [2] in the context of an analysis of the complexity and expressiveness of query languages.

Hence one consequence of (2) (which we only mention in this Introduction) is:

Theorem 1. *If there is an enumeration of all polynomial time decidable subsets of* TAUT*, then there is an enumeration of all polynomial time decidable classes of graphs closed under isomorphisms.*

Using a special feature of the semantics of the logic $L_{\leq}$, one can construct a logic that is an *effectively* P-bounded logic for P, if $L_{\leq}$ is a P-bounded logic for P (cf. Proposition 12). Here this "effectively" means that in (1) we can compute from φ a polynomial bounding the time to decide whether $\mathcal{A}$ is a model of φ. In this way we can strengthen the conclusion of Theorem 1 by requiring that every Turing machine in the enumeration comes with a polynomial time clock. Apparently this is a strengthening, while from any enumeration of the polynomial time decidable subsets of TAUT we obtain one with polynomial time clocks in a trivial manner, namely by systematically adding such clocks.

There is a further consequence of (1) that we do not mention in the main text. Due to our result [3, Theorem 7]) on $L_{\leq}$, we get:

Theorem 2. *The following are equivalent:*

- TAUT *has a p-optimal proof system.*
- *There is an algorithm deciding for every nondeterministic Turing machine $\mathbb{M}$ and every natural number m, given in unary, whether $\mathbb{M}$ accepts the empty input tape in $\leq m$ steps and the algorithm does this for every fixed $\mathbb{M}$ in time polynomial in m.*

In general, the experts tend to believe Conjecture 1, as the existence of an p-optimal proof system for TAUT would have various consequences which seem to be unlikely (see [13, 14]). It is worthwhile to emphasize that Conjecture 1 is equivalent to Conjecture 2a and not to Conjecture 2, as the situation with Conjecture 2 is quite different; no unexpected consequences are known. Moreover, due to results showing that there are logics capturing polynomial time on always larger classes of structures, Grohe [9] "mildly leans towards believing" that there is a P-bounded logic for P.

In [3] we have shown that $L_{\leq}$ is not an effectively P-bounded logic for P under the assumption $\mathrm{NP}[\mathrm{TC}] \not\subseteq \mathrm{P}[\mathrm{TC}^{\log \mathrm{TC}}]$, which means that $\mathrm{NTIME}(h^{O(1)}) \not\subseteq \mathrm{DTIME}(h^{O(\log h)})$ for every time constructible and increasing function h. Under this assumption, we get (see Theorem 15) that TAUT has no effectively p-optimal proof system. Here a proof system P for TAUT is *effectively* p-optimal if from every other proof system for TAUT we can compute a simulation by P.

On the other hand, Krajíček and Pudlák [14] showed, assuming E = NE, that TAUT has a p-optimal proof system. Using our result [3] that under the assumption E = NE the logic $\big(L_{=}$ and hence$\big)$ $L_{\leq}$ is an effectively P-bounded logic for P, we can derive (see Corollary 25) that TAUT has an *effectively* p-optimal proof system if E = NE.

Finally, in Section 5 we extract the main idea underlying the proof of (1) and apply it to other problems.

2. Preliminaries

In this section we recall concepts and results from complexity theory and logic that we will use later and fix some notation.

2.1. Complexity. 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. We denote by P the class of problems Q such that $x \in Q$ is solvable in polynomial time.

All Turing machines have Σ as their alphabet and are deterministic ones if not stated otherwise explicitly. Where necessary we identify Turing machines with (their codes, that is, with) strings over the alphabet Σ. If $\mathbb{M}$ is a Turing machine we denote by $\|\mathbb{M}\|$ the length of its encoding.

By $m^{O(1)}$ we denote the class of polynomially bounded functions from $\mathbb{N}$ to $\mathbb{N}$. Sometimes statements containing a formulation like "there is $d \in \mathbb{N}$ such that for all $x \in \Sigma^*$: $\ldots \leq |x|^d$" can be wrong for $x \in \Sigma^*$ with $|x| = 1$. We trust the reader's common sense to interpret such statements reasonably.

Optimal proof systems, almost optimal algorithms and enumerations of P-easy subsets. Let Q be a problem, $Q \subseteq \Sigma^*$.

A *proof system for* Q is a surjective function $P : \Sigma^* \to Q$ computable in polynomial time. 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 $T : \Sigma^* \to \Sigma^*$ such that for all $w \in \Sigma^*$ we have

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

If $\mathbb{A}$ is any algorithm we denote by $t_{\mathbb{A}}(x)$ the number of steps of the run of $\mathbb{A}$ on input x; if $\mathbb{A}$ on x does not stop, then $t_{\mathbb{A}}(x)$ is not defined.

An algorithm $\mathbb{A}$ deciding Q is *almost optimal* or *optimal on positive instances of* Q if for every algorithm $\mathbb{B}$ deciding Q there is a polynomial $p \in \mathbb{N}[X]$ such that for all $x \in Q$

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

(note that nothing is required of the relationship between $t_{\mathbb{A}}(x)$ and $t_{\mathbb{B}}(x)$ for $x \notin Q$).

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

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

We denote by TAUT the class of tautologies of propositional logic. The following theorem is well-known (cf. [14] for the equivalence of the first two statements and [17] for the equivalence to the third one):

Theorem 3. *The following are equivalent:*

- TAUT *has a p-optimal proof system.*
- TAUT *has an almost optimal algorithm.*
- TAUT *has an enumeration of the* P-*easy subsets.*

2.2. Logic. A *vocabulary* τ is a finite set of relation symbols. Each relation symbol has an *arity*. A *structure* $\mathcal{A}$ of vocabulary τ, or τ-*structure* (or, simply structure), consists of a nonempty set A called the *universe*, and an interpretation $R^{\mathcal{A}} \subseteq A^r$ of each r-ary relation symbol $R \in \tau$. We say that $\mathcal{A}$ is finite, if A is a finite set. *All structures in this paper are assumed to be finite.*

For a structure $\mathcal{A}$ we denote by $\|\mathcal{A}\|$ the size of $\mathcal{A}$, that is, the length of a reasonable encoding of $\mathcal{A}$ as string in $\{0,1\}^*$ (e.g., cf. [7] for details). If necessary, we can assume that the universe of a finite structure is $[m] := \{1, \ldots, m\}$ for some natural number $m \geq 1$, as all the properties of structures we consider are invariant under isomorphisms; in particular, it suffices that from the encoding of $\mathcal{A}$ we can recover $\mathcal{A}$ up to isomorphism. The reader will easily convince himself that we can assume that there is a computable function lgth such that for every vocabulary τ and $m \geq 1$ (we just collect the properties of lgth we will use in this paper):

- $\|\mathcal{A}\| = \text{lgth}(\tau, m)$ for every τ-structure $\mathcal{A}$ with universe of cardinality m (that is, for fixed τ and m, the encoding of each τ-structure with universe of m elements has length equal to $\text{lgth}(\tau, m)$);
- for fixed τ, the function $m \mapsto \text{lgth}(\tau, m)$ is computable in polynomial time if m is given in unary;

– $\text{lgth}(\tau \cup \{R\}, m) = O(\text{lgth}(\tau, m) + m^r)$ for every r-ary relation symbol R not in τ.

We assume familiarity with first-order logic and its extension *least fixed-point logic* LFP (e.g. see [6]). We denote by $\text{LFP}[\tau]$ the set of sentences of vocabulary τ of LFP. By a result due to Immerman [12] and Vardi [18] we know that LFP captures polynomial time on the class of ordered structures.

As we will introduce further semantics for the formulas of least fixed-point logic, we write $\mathcal{A} \models_{\text{LFP}} \varphi$ if the structure $\mathcal{A}$ is a model of the LFP-sentence φ. An algorithm based on the inductive definition of the satisfaction relation for LFP shows (see [19]):

Proposition 4. *The model-checking problem $\mathcal{A} \models_{\text{LFP}} \varphi$ for structures $\mathcal{A}$ and* LFP*-sentences φ can be solved in time*

$$\|\mathcal{A}\|^{O(|\varphi|)}.$$

Logics capturing polynomial time. 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*;

– 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 $(\mathcal{A}, \varphi) \in \models_L$, then, for some τ, we have that $\mathcal{A}$ is a τ-structure and $\varphi \in L[\tau]$; furthermore for each $\varphi \in L[\tau]$ the class of structures $\mathcal{A}$ with $\mathcal{A} \models_L \varphi$ is closed under isomorphisms.

We say that $\mathcal{A}$ is a *model* of φ if $\mathcal{A} \models_L \varphi$ $\big($that is, if $(\mathcal{A}, \varphi) \in \models_L\big)$. We set

$$\text{Mod}_L(\varphi) := \big\{\mathcal{A} \mid \mathcal{A} \models_L \varphi\big\}$$

and say that φ *axiomatizes* the class $\text{Mod}_L(\varphi)$.

We partly take over the following terminology from [16].

Definition 5. 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

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

(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 every fixed φ the algorithm $\mathbb{A}$ runs in time polynomial in $\|\mathcal{A}\|$.

Hence, if L is a P-bounded logic for P, then for every L-sentence φ the algorithm $\mathbb{A}$ witnesses that $\text{Mod}_L(\varphi) \in \text{P}$. However, we do not necessarily know ahead of time the bounding polynomial.

(c) *L is an effectively* P*-bounded logic for* P if L is a P-bounded logic for P and if in addition to the algorithm $\mathbb{A}$ as in (b) there is a computable function that assigns to every L-sentence φ a polynomial $q \in \mathbb{N}[X]$ such that $\mathbb{A}$ decides whether $\mathcal{A} \models_L \varphi$ in $\leq q(\|\mathcal{A}\|)$ steps.

The logic $L_\leq$ and invariant sentences. In this section we introduce the logic $L_\leq$, a variant of least fixed-point logic.

For every vocabulary τ we let $\tau_< := \tau \cup \{<\}$, where $<$ is a binary relation symbol not in τ chosen in some canonical way. We set

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

for every vocabulary τ. Before we define the satisfaction relation for $L_\leq$ we introduce the notion of $\leq m$-invariant.

Definition 6. Let φ be an $L_{\leq}[\tau]$-sentence.

– For $m \geq 1$ we say that φ is $\leq m$*-invariant* if for all structures $\mathcal{A}$ with $|A| \leq m$ we have

$$(\mathcal{A}, <_1) \models_{\text{LFP}} \varphi \iff (\mathcal{A}, <_2) \models_{\text{LFP}} \varphi.$$

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

– φ is *invariant* if it is $\leq m$-invariant for all $m \geq 1$.

Finally we introduce the semantics for the logic $L_{\leq}$ by

$$\mathcal{A} \models_{L_{\leq}} \varphi \quad \iff \quad \Big(\varphi \text{ is } \leq |A|\text{-invariant and } (\mathcal{A}, <_A) \models_{\text{LFP}} \varphi\Big),$$

where here (and later) $<_A$ denotes some ordering on A, say, the ordering on A given by the encoding of $\mathcal{A}$. As least fixed-point logic captures P on ordered structures it is not hard to see that $L_{\leq}$ is a logic for P.

For later purposes we remark that for every $L_{\leq}[\tau]$-sentence φ and $m \geq 1$ we have

$$\varphi \text{ is } \leq m\text{-invariant} \iff \neg\varphi \text{ is } \leq m\text{-invariant},$$

and thus for every τ-structure $\mathcal{A}$

$$(\varphi, \leq |A|) \in \textsc{Inv} \iff \big(\mathcal{A} \models_{L_{\leq}} \varphi \text{ or } \mathcal{A} \models_{L_{\leq}} \neg\varphi\big). \tag{3}$$

In particular,

$$\varphi \text{ is } \leq m\text{-invariant} \iff \big(\mathcal{A}(\tau, m) \models_{L_{\leq}} \varphi \text{ or } \mathcal{A}(\tau, m) \models_{L_{\leq}} \neg\varphi\big), \tag{4}$$

where $\mathcal{A}_m$ is the τ-structure with universe $\{1, \ldots, m\}$, where every relation symbol in τ is interpreted by the empty relation of the corresponding arity.

3. The main theorem

The main result of this paper reads as follows:

Theorem 7. *The following are equivalent:*

(1) TAUT *has an p-optimal proof system.*

(2) $L_{\leq}$ *is a* P*-bounded logic for* P*.*

In view of Theorem 3 we show one direction of Theorem 7 with the following lemma.

Lemma 8. *If* $L_{\leq}$ *is a* P*-bounded logic for* P*, then there is an enumeration of the* P*-easy subsets of* TAUT*.*

Proof: It is easy to introduce a vocabulary τ such that in polynomial time we can associate with every propositional formula α a τ-structure $\mathcal{A}(\alpha)$ such that

– we can recover α from any $\mathcal{B}$ isomorphic to $\mathcal{A}(\alpha)$ in polynomial time;

– every variable X corresponds to two distinct elements a_X, b_X of $\mathcal{A}(\alpha)$ and there is a unary relation symbol $P \in \tau$ such that $P^{\mathcal{A}(\alpha)} = \{a_X \mid X \text{ variable of } \alpha\}$;

– there is a first-order sentence $\varphi(\textsc{Prop})$ of vocabulary τ axiomatizing the class

$$\big\{\mathcal{B} \mid \mathcal{B} \cong \mathcal{A}(\alpha) \text{ for some } \alpha \in \textsc{Prop}\big\}.$$

Again let $\tau_< := \tau \cup \{<\}$ with a new binary $<$. Note that a $\tau_<$-structure of the form $(\mathcal{A}(\alpha), <)$ yields an assignment of the variables of α, namely the assignment sending a variable X to TRUE if and only if $a_X < b_X$. There is an LFP[$\tau_<$]-formula $\varphi(\text{sat})$ that for every $\alpha \in \text{PROP}$ expresses in $(\mathcal{A}(\alpha), <)$ that the assignment given by $<$ satisfies α.

We introduce the $L_\leq[\tau]$-sentence

$$\varphi_0 := (\varphi(\text{PROP}) \to \varphi(\text{sat})).$$

By the definition of $\models_{L_\leq}$ we see that for every $\alpha \in \text{PROP}$ and every $L_\leq[\tau]$-sentence φ

$$\text{if } \mathcal{A}(\alpha) \models_{L_\leq} (\varphi_0 \wedge \varphi), \text{ then } \alpha \in \text{TAUT}. \tag{5}$$

We claim that the class of models of $(\varphi_0 \wedge \varphi)$, more precisely,

$$Q(\varphi) := \big\{\alpha \in \text{PROP} \mid \mathcal{A}(\alpha) \models_{L_\leq} (\varphi_0 \wedge \varphi)\big\},$$

where φ ranges over all $L_\leq[\tau]$-sentences, yields the desired enumeration of P-easy subsets of TAUT. By (5), we have $Q(\varphi) \subseteq$ TAUT.

For $\varphi \in L_\leq[\tau]$ let the Turing machine $\mathbb{M}_\varphi$, given an input $\alpha \in \text{PROP}$, first construct $\mathcal{A}(\alpha)$ and then check whether $\mathcal{A}(\alpha) \models_{L_\leq} (\varphi_0 \wedge \varphi)$. Clearly, $\mathbb{M}_\varphi$ decides $Q(\varphi)$ and does this in polynomial time, as $L_\leq$ is a P-bounded logic for P.

Conversely, let Q be a P-easy subset of TAUT. If Q is finite, it is easy to see that $Q = Q(\varphi)$ for some $\varphi \in L_\leq[\tau_0]$. Now let Q be infinite. The class

$$\big\{\mathcal{B} \mid \mathcal{B} \cong \mathcal{A}(\alpha) \text{ for some } \alpha \in Q\big\}$$

is in P, and therefore it is axiomatizable by an $L_\leq[\tau]$-sentence φ. As the class contains arbitrarily large structures, the formula φ is invariant. We show that $Q = Q(\varphi)$.

Assume first that $\alpha \in Q(\varphi)$, i.e., $\mathcal{A}(\alpha) \models_{L_\leq} (\varphi_0 \wedge \varphi)$. Then, by invariance of φ, we have $\mathcal{A}(\alpha) \models_{L_\leq} \varphi$, that is, $Q(\varphi) \subseteq Q$.

Conversely, assume that $\alpha \in Q$. Then $\mathcal{A}(\alpha) \models_{L_\leq} \varphi$. As $\alpha \in$ TAUT, in order to get $\mathcal{A}(\alpha) \models_{L_\leq} (\varphi_0 \wedge \varphi)$ (and hence, $\alpha \in Q(\varphi)$) it suffices to show that $(\varphi_0 \wedge \varphi)$ is $\leq |A(\alpha)|$-invariant. So let $\mathcal{B}$ be a τ-structure with $|B| \leq |A(\alpha)|$. If $\mathcal{B} \not\models_{L_\leq} \varphi$, then, by invariance of φ, we have $(\mathcal{B}, <^B) \not\models_{\text{LFP}} (\varphi_0 \wedge \varphi)$ for all orderings $<^B$ on B; if $\mathcal{B} \models_{L_\leq} \varphi$, then $\mathcal{B} \cong \mathcal{A}(\beta)$ for some $\beta \in Q \subseteq$ TAUT. Hence, $(\mathcal{B}, <^B) \models_{L_{\text{LFP}}} (\varphi_0 \wedge \varphi)$ for all orderings $<^B$ on B. □

Remark 9. In the previous proof we have used the definition of the satisfaction relation $\models_{L_\leq}$ in order to express the universal second-order quantifier in the statement "all assignments satisfy α." Similarly, we can do with every Π^1_1-sentence $\forall R\varphi$, where φ is a first-order formula or (equivalently) LFP-formula and show in this way that there is an enumeration of the P-easy subsets closed under isomorphisms of the class of models of such a sentence, if $L_\leq$ is a P-bounded logic for P. In fact, let k be the arity of R. If a structure $\mathcal{A}$ has n elements, we consider a structure $\mathcal{B}$ with additional unary disjoint relations $U^\mathcal{B}, P_0^\mathcal{B}, P_1^\mathcal{B}$ such that

$$B = U^\mathcal{B} \cup P_0^\mathcal{B} \cup P_1^\mathcal{B}, \quad U^\mathcal{B} = A, \quad |P_0^\mathcal{B}| = n^k \quad |P_1^\mathcal{B}| = n^k$$

and with an ordering $<^\mathcal{B}$.

With the elements in $P_0^\mathcal{B}$ interpreted as 0s and the elements in $P_1^\mathcal{B}$ interpreted as 1s, the first n^k-elements of the ordering in $P_0^\mathcal{B} \cup P_1^\mathcal{B}$ represent a natural number $< 2^{n^k}$ and thus a k-ary relation R on A, which we can compute in polynomial time (polynomial in n); hence we can express R by an LFP-formula. As in this way, by changing the ordering, we have access to all such k-ary relations R on A, we can express the quantifier $\forall R$ using the invariance requirement of $\models_{L_\leq}$. In the terminology of classical model theory this shows that the class of models of such a Π^1_1-sentence is the class of relativized reducts of models of some $L_\leq$-sentence.

For example, let C be the class of all pairs $(\mathcal{G}, \mathcal{H})$ of graphs such that $\mathcal{H}$ is *not* a homomorphic image of $\mathcal{G}$. Of course, a subclass D of C is closed under isomorphisms if

$$\mathcal{G} \cong \mathcal{G}', \ \ \mathcal{H} \cong \mathcal{H}' \ \text{ and } \ (\mathcal{G}, \mathcal{H}) \in D \ \text{ imply } \ (\mathcal{G}', \mathcal{H}') \in D.$$

By the previous observation, we see that there is an enumeration of the P-easy subclasses of C closed under isomorphisms if $L_\leq$ is a P-bounded logic for P.

As the models of such a Π^1_1-sentence corresponds to a problem Q in co-NP, a simple complexity-theoretic argument shows that there is an enumeration of the P-easy subsets of Q provided there is one for the P-easy subsets of TAUT (see also [1]). However, in this way, in the previous example we would not get an enumeration of those P-easy subclasses that are *closed under isomorphisms*.

The remaining direction in Theorem 7 is provided by the following result.

Lemma 10. *If* TAUT *has an almost optimal algorithm, then $L_\leq$ is a* P*-bounded logic for* P.

Proof: We assume that TAUT has an almost optimal algorithm $\mathbb{O}$ and have to show that there exists a function h and an algorithm that decides $\mathcal{B} \models_{L_\leq} \varphi$ in time $\|\mathcal{B}\|^{h(\varphi)}$.

By the definition of $\mathcal{B} \models_{L_\leq} \varphi$ it suffices to show the existence of an algorithm $\mathbb{A}$ that for every $L_\leq$-sentence φ and every $m \in \mathbb{N}$ decides whether φ is $\leq m$-invariant and does this for fixed φ in time $m^{O(1)}$.

We set

$$Q := \Big\{ \Big(\chi, \ell, \text{lgth}(\tau,\ell)^{|\chi|}\Big) \Big| \tau \text{ a vocabulary }, \chi \in \text{LFP}[\tau],\ \ell \geq 1,\ \text{lgth}(\tau,\ell)^{|\chi|} \text{ in unary,} \\ \text{there is a structure } \mathcal{B} \text{ with } \big(|B| \leq \ell \text{ and } \mathcal{B} \models_{\text{LFP}} \chi\big) \Big\}$$

(compare Section 2.2 for the definition of the function lgth). By Proposition 4, $Q \in$ NP. Thus there is a polynomial time reduction $R : Q \leq^p$ SAT. We can assume that from $R(x)$ we can recover x in polynomial time.

Let φ be an $L_\leq[\tau]$-sentence. Then

$$\begin{aligned}
&\varphi \text{ is not } \leq m\text{-invariant} \\
&\quad\iff \text{there is a structure } \mathcal{B} \text{ and orderings } <_1, <_2 \text{ with} \\
&\qquad \big(|B| \leq m, (\mathcal{B}, <_1) \models_{\text{LFP}} \varphi(<_1) \text{ and } (\mathcal{B}, <_2) \models_{\text{LFP}} \neg\varphi(<_2)\big) \\
&\quad\iff \text{there is a structure } \mathcal{B} \text{ and orderings } <_1, <_2 \text{ with} \\
&\qquad \big(|B| \leq m \text{ and } (\mathcal{B}, <_1, <_2) \models_{\text{LFP}} \underbrace{(\varphi(<_1) \wedge \neg\varphi(<_2))}_{\varphi^*}\big) \\
&\quad\iff \Big(\varphi^*, m, \text{lgth}(\tau \cup \{<_1, <_2\}, \ell)^{|\varphi^*|}\Big) \in Q \\
&\quad\iff R\Big(\varphi^*, m, \text{lgth}(\tau \cup \{<_1, <_2\}, \ell)^{|\varphi^*|}\Big) \in \text{SAT}.
\end{aligned}$$

We set

$$\alpha(\varphi, m) := R\Big(\varphi^*, m, \text{lgth}(\tau \cup \{<_1, <_2\}, \ell)^{|\varphi^*|}\Big).$$

Hence

$$\varphi \text{ is } \leq m\text{-invariant} \iff \neg\alpha(\varphi, m) \in \text{TAUT}. \tag{6}$$

It is clear that there is an algorithm that on input (φ, m), where m is given in unary, computes $\alpha(\varphi, m)$ and for fixed φ

$$\text{it computes } \alpha(\varphi, m) \text{ in time } m^{O(1)}, \text{ in particular, } |\alpha(\varphi, m)| \leq m^{O(1)}, \tag{7}$$

as for fixed τ, the function $m \mapsto \text{lgth}(\tau, m)$ is polynomial in m.

Let $\mathbb{S}$ be the algorithm that on input φ by systematically going through all structures with universe $\{1\}$, all with universe $\{1, 2\}$,... computes

$$m(\varphi) := \text{ the least } m \text{ such that } \varphi \text{ is not } \leq m\text{-invariant}.$$

If φ is invariant, $m(\varphi)$ is not defined and $\mathbb{S}$ does not stop.

We show that the following algorithm $\mathbb{A}$ has the desired properties.

$\mathbb{A}(\varphi, m)$
//φ an $L_{\leq}$-sentence, $m \in \mathbb{N}$ in unary

1. Compute $\alpha(\varphi, m)$.
2. In parallel simulate $\mathbb{S}$ on input φ and $\mathbb{O}$ on input $\neg\alpha(\varphi, m)$.
3. **if** $\mathbb{O}$ stops first, **then** output its answer.
4. **if** $\mathbb{S}$ stops first, **then**
5. **if** $m < m(\varphi)$ **then** accept **else** reject.

By our assumptions on $\mathbb{O}$ and $\mathbb{S}$ and by (6), it should be clear that $\mathbb{A}$ on input (φ, m) decides whether φ is $\leq m$-invariant. We have to show that for fixed φ it does it in time polynomial in m.

Case "φ is invariant": Then for all m we have $\neg\alpha(\varphi, m) \in$ TAUT. Thus the following algorithm $\mathbb{O}_\varphi$ decides TAUT: on input $\beta \in$ PROP the algorithm $\mathbb{O}_\varphi$ checks whether $\beta = \neg\alpha(\varphi, m)$ for some $m \geq 1$. If so, it accepts and otherwise it runs $\mathbb{O}$ on input β and answers accordingly. By (7), we have

$$t_{\mathbb{O}_\varphi}(\neg\alpha(\varphi, m)) \leq m^{O(1)}.$$

As $\mathbb{O}$ is optimal, we know that there is a constant d such that for all $\beta \in$ TAUT

$$t_{\mathbb{O}}(\beta) \leq \big(|\beta| + t_{\mathbb{O}_\varphi}(\beta)\big)^d. \tag{8}$$

In particular, we have

$$t_{\mathbb{O}}(\neg\alpha(\varphi, m)) \leq \big(|\neg\alpha(\varphi, m)| + t_{\mathbb{O}_\varphi}(\neg\alpha(\varphi, m))\big)^d \leq m^{O(1)}.$$

By this inequality and (7), we see that for invariant φ we have $t_{\mathbb{A}}(\varphi, m) \leq m^{O(1)}$.

Case "φ is not invariant": Then $\mathbb{S}$ will stop on input φ. Thus, in the worst case, $\mathbb{A}$ on input (φ, m) has to wait till the simulation of $\mathbb{S}$ on φ stops and then must check whether the result $m(\varphi)$ of the computation of $\mathbb{S}$ is bigger than m or not and answer accordingly. So the algorithm $\mathbb{A}$ at most takes time $m^{O(1)} + O(t_{\mathbb{S}}(\varphi) + m) \leq m^{O(1)}$ (note that we fix φ, so that $t_{\mathbb{S}}(\varphi)$ is a constant). □

Proof of Theorem 7. The claimed equivalence follow from Lemma 8 and Lemma 10 using Theorem 3. □

Corollary 11. *If* TAUT *has an p-optimal proof system, then there is an* effectively P*-bounded logic for* P.

This result follows from Theorem 7 using the following proposition.

Proposition 12. *If $L_{\leq}$ is a* P*-bounded logic for* P*, then there is an* effectively P*-bounded logic for* P.

Proof: In Section 2.2 we have seen that for every $L_{\leq}$-sentence φ and $m \geq 1$ we have that

$$\varphi \text{ is } \leq m\text{-invariant} \iff \big(\mathcal{A}(\tau, m) \models_{L_{\leq}} \varphi \text{ or } \mathcal{A}(\tau, m) \models_{L_{\leq}} \neg\varphi\big),$$

where $\mathcal{A}(\tau, m)$ denotes the "empty structure" of vocabulary τ with universe $\{1, \ldots, m\}$.

Now assume that $L_{\leq}$ is a P-bounded logic for P and let $\mathbb{A}$ be an algorithm deciding $\models_{L_{\leq}}$ and witnessing that $L_{\leq}$ is a P-bounded logic for P. By (4), there is function h assigning to every $L_{\leq}$-sentence φ a polynomial $h(\varphi) \in \mathbb{N}[X]$ such that $\mathbb{A}$ decides whether φ is $\leq m$-invariant in time $h(\varphi)(m)$.

We consider the logic $T(L_{\leq})$, *time-clocked* $L_{\leq}$, defined as follows:

– for every vocabulary τ

$$T(L_{\leq})[\tau] := \big\{(\varphi, p) \mid \varphi \in L_{\leq}[\tau] \text{ and } p \in \mathbb{N}[X]\big\};$$

– $\mathcal{A} \models_{T(L_{\leq})} \varphi$ iff (a) and (b) are fulfilled, where

(a) $\mathbb{A}$ shows via (4) in $\leq p(|A|)$ steps that φ is $\leq |A|$-invariant;

(b) $(\mathcal{A}, <_{\mathcal{A}}) \models_{\text{LFP}} \varphi$ (recall that by $<_A$ we denote the ordering of A given by the encoding of $\mathcal{A}$).

It is not hard to verify that $T(L_{\leq})$ is an effectively P-bounded logic for P. □

Remark 13. In a slightly different way but using the same idea one can define the time-clocked version $T(L)$ for any P-bounded logic L for P. However, in general, $T(L)$ is not an effectively P-bounded logic L for P, as the class of models of a $T(L)$-sentence must not be closed under isomorphisms. In the case of $T(L_{\leq})$ this is guaranteed by the fact that condition (a) in the definition of $\mathcal{A} \models_{T(L_{\leq})} \varphi$ only refers to the cardinality of the universe of $\mathcal{A}$.

4. Effective versions

Let

$$\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$$

mean that $\text{NTIME}(h^{O(1)}) \not\subseteq \text{DTIME}(h^{O(\log h)})$ for every time constructible and increasing function h. In [3] we have shown:

Proposition 14. *Assume that* $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$. *Then* $L_{\leq}$ *is not an effectively* P*-bounded logic for* P.

Moreover, in [4] we have seen that $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$ holds under the assumption that NP contains an E-bi-immune problem. In fact, $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$ seems to be a much weaker assumption.

Are there natural effective versions of the properties on TAUT listed in Theorem 3 equivalent to the statement "$L_{\leq}$ is not an effectively P-bounded logic for P" and which therefore, by Proposition 14, could not hold under the assumption $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$? We did not find them (see Remark 26). However, by analyzing the proof of Proposition 14, we isolate a property of an effective P-bounded logic for P that cannot be fulfilled if $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$. It turns out that this is equivalent to natural effective versions of the properties on TAUT under consideration. Before defining them, we already state the result we aim at.

Theorem 15. *Assume that* $\text{NP}[\text{TC}] \not\subseteq \text{P}[\text{TC}^{\log \text{TC}}]$. *Then:*

(1) TAUT *has no effectively p-optimal proof system.*

(2) TAUT *has no effectively almost optimal algorithm.*

(3) There is no effective enumeration of the P*-easy subsets of* TAUT.

Definition 16. (1) We define the binary relation INV between $L_{\leq}$-sentences and natural numbers $m \geq 1$ by

$$(\varphi, m) \in \text{INV} \iff \varphi \text{ is a } \leq m\text{-invariant sentence.}$$

(2) We say that INV is *effectively decidable on invariant sentences* (more precisely, we should say *decidable and effectively decidable on invariant sentences*) if there is an algorithm $\mathbb{A}$ deciding INV and a computable function f such that for all *invariant* φ and $m \geq 1$

$$t_{\mathbb{A}}(\varphi, m) \leq m^{f(\varphi)}.$$

The following is immediate by Proposition 14 and (4):

Lemma 17. *If* $L_{\leq}$ *is an effectively* P*-bounded logic for* P*, then* INV *is decidable effectively on invariant sentences.*

Definition 18. Let $Q \subseteq \Sigma^*$. We say that Q has an *effective enumeration of* P*-easy subsets*, if it has an enumeration $M : \mathbb{N} \to \Sigma^*$ of P-easy subsets of Q such that there are functions $I : \Sigma^* \times \mathbb{N}[X] \to \mathbb{N}$ and $b : \Sigma^* \times \mathbb{N}[X] \to \mathbb{N}[X]$ such that for every Turing machine $\mathbb{M}$ and polynomial $p \in \mathbb{N}[X]$,

> if the Turing machine $\mathbb{M}$ recognizes a subset $Q' \subseteq Q$ with time bound $p \in \mathbb{N}[X]$, then the machine $M(I(\mathbb{M}, p))$ recognizes Q' with time bound $b(\mathbb{M}, p)$.

We turn to the effective version of Lemma 8.

Lemma 19. *If* INV *is effectively decidable on invariant sentences, then there is an effective enumeration of the* P*-easy subsets of* TAUT.

Sketch of Proof. We use the terminology of Lemma 8, in particular, let τ and φ_0 be as there. We show that the enumeration of P-easy subsets of TAUT given there has the desired effectivity property. By the proof of the Immerman-Vardi Theorem (stating that least fixed-point logic captures P on ordered structures) we know that we can assign to every Turing machine $\mathbb{M}$ that recognizes a subset Q of PROP with time bound $p \in \mathbb{N}[X]$ an LFP$[\tau_<]$-sentence $\varphi(\mathbb{M}, p)$ such that

$$\begin{aligned} Q &= \big\{\alpha \in \textsc{Prop} \;\big|\; (\mathcal{A}(\alpha), <) \models_{\text{LFP}} \varphi(\mathbb{M}, p) \text{ for some ordering on } A(\alpha)\big\} \\ &= \big\{\alpha \in \textsc{Prop} \;\big|\; (\mathcal{A}(\alpha), <) \models_{\text{LFP}} \varphi(\mathbb{M}, p) \text{ for all orderings on } A(\alpha)\big\}. \end{aligned}$$

Hence, the sentence $\varphi(\mathbb{M}, p)$ is invariant. As we assume that INV is effectively decidable on invariant sentences, we therefore can compute from $(\mathbb{M}, p)$ a polynomial bounding the time needed to recognize Q. □

Definition 20. Let $Q \subseteq \Sigma^*$ and $\mathbb{A}$ an algorithm deciding Q. The algorithm $\mathbb{A}$ is *effectively almost optimal* if there is a computable function $b : \Sigma^* \to \mathbb{N}[x]$ such that for every algorithm $\mathbb{B}$ deciding Q we have for every $x \in \Sigma^*$ we have

$$t_{\mathbb{A}}(x) \leq b(\mathbb{B})\big(t_{\mathbb{B}}(x) + |x|\big). \tag{9}$$

We now come to the effective version of Lemma 10.

Lemma 21. *If* TAUT *has an effectively almost optimal algorithm, then* INV *is effectively decidable on invariant sentences.*

Sketch of Proof. The result is obtained by analyzing the proof of Lemma 10. Of course, now we assume that $\mathbb{O}$ is an effectively almost optimal algorithm for TAUT. It should be clear that we can associate with every $L_\leq$-sentence φ a polynomial p such that we can replace (7) by

$$\text{there is an algorithm that on input } (\varphi, m) \text{ computes } \alpha(\varphi, m) \text{ in time } p(m).$$

Furthermore, from φ we can compute in polynomial time the algorithm $\mathbb{O}_\varphi$. By the effectivity property of $\mathbb{O}$, we also can compute the constant d satisfying (8) from φ. This shows that the algorithm $\mathbb{A}$ in the proof has the required effectivity property. □

Finally we define:

Definition 22. Let $Q \subseteq \Sigma^*$. A proof system P for Q is *effectively p-optimal* if there are two computable functions $S : \Sigma^* \times \mathbb{N}[X] \to \Sigma^*$ and $b : \Sigma^* \times \mathbb{N}[X] \to \mathbb{N}[X]$ such that for every proof system P' for Q with time bound $p \in \mathbb{N}[x]$ and every $w \in \Sigma^*$, we have

$$P'(w) = P(S(P', p)(w)),$$

where $S(P', p)$ is (the code of) a Turing machine with time bound $b(P', p)$ and $S(P', p)(w)$ denotes the output of $S(P', p)$ on input w.

Theorem 23. *The following are equivalent:*

(1) TAUT *has an effectively p-optimal proof system.*

(2) TAUT *has an effectively almost optimal algorithm.*

(3) TAUT *has an effective enumeration of the P-easy subsets.*

(4) INV *is effectively decidable on invariant sentences.*

Proof: We already know that (2) implies (4) and (4) implies (3). The equivalence of (1), (2), and (3) are shown by refining known proofs of the equivalence of the non-effective versions of (1)–(3). We shall give the proof in the full version of the paper. □

Lemma 24. *If* $\mathrm{NP}[\mathrm{TC}] \not\subseteq \mathrm{P}[\mathrm{TC}^{\log \mathrm{TC}}]$*, then* INV *is not effectively decidable on invariant sentences.*

Proof: In [3] we have shown the following fact (already more or less explicit in [16]). There is an algorithm that assigns to every nondeterministic Turing machine $\mathbb{M}$ a natural number $m_0(\mathbb{M}) \geq 1$ and an $L_{\leq}$-sentence $\varphi_{\mathbb{M}}$ such that for every $m \geq m_0(\mathbb{M})$

$$(\varphi_{\mathbb{M}}, \leq m) \in \textsc{Inv} \iff \mathbb{M} \text{ does not accept the empty input tape in} \leq m \text{ steps.} \tag{10}$$

By contradiction, we assume that INV is effectively decidable on invariant sentences. Then, by (10), there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm $\mathbb{A}$ deciding for every nondeterministic Turing machine $\mathbb{M}$ and every $m \geq 1$ (not only for $m \geq m_0(\mathbb{M})$) whether $\mathbb{M}$ does not accept the empty input in $\leq m$ steps and doing this in time

$$m^{f(\|\mathbb{M}\|)} \tag{11}$$

for machines $\mathbb{M}$ with the property that all runs started with the empty input are infinite. We may assume that f is increasing.

For any nondeterministic Turing machine $\mathbb{M}$ and every $x \in \Sigma^*$ we let $\mathbb{M}_x$ be the nondeterministic Turing machine that, started with empty input tape, first writes x on some tape and then simulates $\mathbb{M}$ started with x. Clearly we can define $\mathbb{M}_x$ such that $\|\mathbb{M}_x\| = O(\|\mathbb{M}\| + |x| \cdot \log\, |x|)$. We choose $e \in \mathbb{N}$ such that

$$\|\mathbb{M}_x\| \leq e \cdot (\|\mathbb{M}\| + |x|^2). \tag{12}$$

Now we choose an increasing and time constructible function $h : \mathbb{N} \to \mathbb{N}$ such that $f(2e \cdot n^2) \leq h(n)$ for all $n \in \mathbb{N}$. Then, for all $x, y \in \mathbb{N}$

$$f(e \cdot (x + y^2)) \leq h(x) + h(y). \tag{13}$$

We define $g : \mathbb{N} \to \mathbb{N}$ by $g(n) := 2^{h(n)}$; clearly g is time constructible and increasing, too. We show that $\mathrm{NTIME}(g^{O(1)}) \subseteq \mathrm{DTIME}(g^{O(\log g)})$, which yields the desired contradiction.

Let $Q \subseteq \Sigma^*$ be in $\mathrm{NTIME}(g^{O(1)})$. We choose a nondeterministic Turing machine $\mathbb{M}$ and constants $c, d \in \mathbb{N}$ such that the machine $\mathbb{M}$ decides whether $x \in Q$ in time $c \cdot (g|x|)^d$. As g is time constructible, we can assume that every run of $\mathbb{M}$ on every instance x either accepts x in time $c \cdot g(|x|)^d$ steps or is infinite.

For $x \in \{0,1\}^*$ we have (recall the definition of $\mathbb{M}_x$)

$$\begin{aligned} x \in Q &\iff \mathbb{M}_x \text{ accepts the empty input tape in at most } |x| + c \cdot g(|x|)^d \text{ steps} \\ &\iff \mathbb{M}_x \text{ accepts the empty input tape in at most } 2c \cdot g(|x|)^d \text{ steps} \\ &\iff \mathbb{A} \text{ accepts } \big(\mathbb{M}_x, 2c \cdot g(|x|)^d\big). \end{aligned}$$

and

$$x \notin Q \iff \text{all runs of } \mathbb{M}_x \text{ started with empty input tape are infinite.}$$

Hence, for $x \notin Q$, the running time of $\mathbb{A}$ on input $(\mathbb{M}_x, 2c \cdot g(|x|)^d)$, by (11), (12), and (13), is bounded by

$$\big(2c \cdot g(|x|)^d\big)^{f(e \cdot (\|\mathbb{M}\| + |x|^2))} \leq O\Big(g(|x|)^{d \cdot \left(h(\|\mathbb{M}\|) + h(|x|)\right)}\Big) \leq O\Big(g(|x|)^{d \cdot \left(h(\|\mathbb{M}\|) + \log\, g(|x|)\right)}\Big).$$

The desired algorithm $\mathbb{B}$ witnessing that $Q \in \mathrm{DTIME}(g^{O(\log g)})$ runs as follows. On input x it simulates $(2c \cdot (g(|x|))^{d \cdot f(e \cdot (\|\mathbb{M}\| + |x|^2))}$ steps of $\mathbb{A}$. If $\mathbb{A}$ stops in that time, then $\mathbb{B}$ answers accordingly, otherwise $\mathbb{B}$ accepts x. □

Proof of Theorem 15. Immediate by Theorem 23 and Lemma 24. □

In [3] we have shown that if E = NE, then (the logic $L_=$ and hence) $L_{\leq}$ are effectively P-bounded logics for P. By Theorem 23 and Lemma 17 we obtain the following "effective versions" of a result due to Krajíček and Pudlák.

Corollary 25. *Assume* E = NE*. Then:*

- TAUT *has an effectively optimal proof system.*
- TAUT *has an effectively almost optimal algorithm.*
- *There is an effective enumeration of the P-easy subsets of* TAUT.

Remark 26. More or less in the same way as we showed Lemma 24 one can derive:

> If NP[TC] $\not\subseteq$ P[TC$^{\log \text{TC}}$], then INV is decidable and effectively decidable on *noninvariant* sentences.

Clearly, $L_{\leq}$ is an effectively P-bounded logic for P if and only if INV is decidable, effectively decidable on noninvariant sentences, and effectively decidable on invariant sentences.

As already mentioned we do not know any *natural* effectivity property of optimal proof systems equivalent to "'$L_{\leq}$ is an effectively P-bounded logic for P" and by the previous observation thus equivalent to "INV is decidable and effectively decidable on noninvariant sentences."

Of course, there are equivalent versions, but there are not in the spirit of the properties we consider here, e.g.:

> The following are equivalent:
>
> (a) $L_{\leq}$ is an effectively P-bounded logic for P.
>
> (b) There is an algorithm $\mathbb{B}$ and a computable g such that for every Turing machine $\mathbb{M}$ and constant c, if for every $i \in \mathbb{N}$ the machine $\mathbb{M}$ on input i in unary computes $\alpha_i \in$ PROP in time i^c and
>
> $$\big(\alpha_{i+1} \in \text{TAUT implies } \alpha_i \in \text{TAUT}\big),$$
>
> then the algorithm $\mathbb{B}$ applied to $\mathbb{M}, c, i$ decides whether $\alpha_i \in$ TAUT in time $i^{g(\mathbb{M},c)}$.

5. Slicewise monotone parameterized problems

Among others we have shown in Section 3 that $L_{\leq}$ is a P-bounded logic for P if TAUT has an almost optimal algorithm. Extracting the idea underlying that proof we obtain the following general result that we already state, even though we haven't defined all the notions appearing in it so far.

Theorem 27. *Let (Q, κ) be a slicewise monotone parameterized problem with decidable Q. If $\Sigma^* \setminus Q$ has an almost optimal algorithm, then $(Q, \kappa) \in \text{XP}_{\text{uni}}$.*

As every problem in co-NP has an almost optimal algorithm if TAUT has one, we immediately get:

Corollary 28. *Let (Q, κ) be a slicewise monotone parameterized problem with Q in* NP*. If* TAUT *has an almost optimal algorithm, then $(Q, \kappa) \in \text{XP}_{\text{uni}}$.*

We view *parameterized problems* as pairs (Q, κ) consisting of a problem $Q \subseteq \Sigma^*$ and a *parameterization* $\kappa : \Sigma^* \to \mathbb{N}$, which is required to be computable in polynomial time. We exemplify our way of presenting parameterized problems by introducing the *parameterized invariance problem* p-INV, a parameterized version of the problem INV,

p-INV	
Instance:	An $L_{\leq}$-sentence φ and $m \geq 1$ in unary.
Parameter:	$\lvert\varphi\rvert$.
Problem:	Is $\varphi \leq m$-invariant?

A parameterized problem (Q, κ) is in the class XP if $x \in Q$ is solvable in time $|x|^{f(\kappa(x))}$ (more precisely, in time $O(|x|^{f(\kappa(x))})$) for some computable $f : \mathbb{N} \to \mathbb{N}$. Besides this class of the usual (strongly uniform) parameterized complexity theory we need its *uniform* version XP_{uni}: $(Q, \kappa) \in \text{FPT}_{\text{uni}}$ if there is an algorithm solving $x \in Q$ in time $|x|^{f(\kappa(x))}$ for some *arbitrary* $f : \mathbb{N} \to \mathbb{N}$.

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

Proposition 29. *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:	$\|\varphi\|$.
Problem:	Is $\mathcal{A} \models_L \varphi$

Then

$$\begin{aligned} L \text{ is an effectively P-bounded logic for P} &\iff p\text{-}\models_L \in \mathrm{XP} \\ L \text{ is a P-bounded logic for P} &\iff p\text{-}\models_L \in \mathrm{XP}_{\mathrm{uni}} \end{aligned}$$

and for $L := L_\leq$ we even have

$$\begin{aligned} L_\leq \text{ is an effectively P-bounded logic for P} &\iff p\text{-}\models_{L_\leq} \in \mathrm{XP} \\ &\iff p\text{-INV} \in \mathrm{XP} \\ L_\leq \text{ is a P-bounded logic for P} &\iff p\text{-}\models_{L_\leq} \in \mathrm{XP}_{\mathrm{uni}} \\ &\iff p\text{-INV} \in \mathrm{XP}_{\mathrm{uni}}. \end{aligned}$$

A parameterized problem (Q, κ) is *slicewise monotone* if its instances have the form (x, n), where $x \in \{0,1\}^*$ and $n \in \mathbb{N}$ is given in unary, if $\kappa(x, n) = |x|$, and finally if for all $x \in \{0,1\}^*$ and $n, n' \in \mathbb{N}$ we have

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

The "complement" of p-INV, more precisely the problem

p-NOT-INV	
Instance:	An $L_\leq$-sentence φ and $m \geq 1$ in unary.
Parameter:	$\|\varphi\|$.
Problem:	Is φ not $\leq m$-invariant?

is slicewise monotone. Clearly, $\big(p\text{-NOT-INV} \in \mathrm{XP} \iff p\text{-INV} \in \mathrm{XP}\big)$ and the classical problem underlying p-NOT-INV is in NP.

Further slicewise monotone problems with underlying classical problem in NP are:

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

p-FO-FINITE-GRAPH	
Instance:	A first-order sentence φ of vocabulary $\{E\}$ with binary E and $n \in \mathbb{N}$ in unary.
Parameter:	$\|\varphi\|$.
Problem:	Is there a graph $\mathcal{G}$ with $\mathcal{G} \models \varphi$ and $\|G\| \leq n$?

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

Proof of Theorem 27. Let (Q, κ) be slicewise monotone and $\mathbb{Q}$ be an algorithm deciding Q. Assume that $\Sigma^* \setminus Q$ has an almost optimal algorithm $\mathbb{O}$. We have to show that $(Q, \kappa) \in \text{XP}_{\text{uni}}$.

Let $\mathbb{S}$ be the algorithm that, on $x \in \Sigma^*$, by systematically applying $\mathbb{Q}$ to the inputs $(x, 0)$, $(x, 1)$,... 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 $(Q, \kappa) \in \text{XP}_{\text{uni}}$.

$\mathbb{A}(x, n)$
//$x \in \Sigma^*$, $n \in \mathbb{N}$ in unary

1. In parallel simulate $\mathbb{S}$ on input x and $\mathbb{O}$ on input (x, n).
2. **if** $\mathbb{O}$ stops first, **then**
3. **if** $\mathbb{O}$ accepts **then** reject **else** accept.
4. **if** $\mathbb{S}$ stops first, **then**
5. **if** $n < n(x)$ **then** reject **else** accept.

By our assumptions on $\mathbb{O}$ and $\mathbb{S}$ and the slicewise monotonicity of Q, it should be clear that $\mathbb{A}$ decides Q. We have to show that it does it in the time required by XP_{uni}.

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 $\mathbb{O}_x$ decides $\Sigma^* \setminus Q$: 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

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

As $\mathbb{O}$ is 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

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

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 must check whether the result $n(x)$ of the computation of $\mathbb{S}$ is bigger than n or not and answer according to Line 5. So in the worst case the algorithm $\mathbb{A}$ takes time $O(t_{\mathbb{S}}(x) + O(n)) \leq n^{O(t_{\mathbb{S}}(x))}$. □

By the previous remarks, we get:

Corollary 30. *If there is an almost optimal algorithm for* TAUT*, then* p-GÖDEL*,* p-FO-FINITE-GRAPH *and* p-$\text{ACC}_{\leq}$ *are in* XP_{uni}.

References

[1] O. Beyersdorff and Z. Sadowski. Characterizing the existence of optimal proof systems and complete sets for promise classes. In *Proceedings of the 4th Computer Science Symposium in Russia (CSR)*, Lecture Notes in Computer Science 5675, 47–58, 2009, 2008.

[2] A.K. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25, 99–128, 1982.

[3] 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.

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

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

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

[7] J. Flum and M. Grohe. *Parameterized Complexity Theory*, Springer, 2006.

[8] M. Grohe. The quest for a logic capturing PTIME. In *Proceedings of the 22nd IEEE Symposium on Logic in Computer Science (LICS'08)*, pages 267–271, 2008.

[9] M. Grohe. Fixed-point definability and polynomial time. In *Proceedings of the 23rd International Workshop on Computer Science Logic (CSL'09)*, Lecture Notes in Computer Science 5771, pages 20–23, 2009.

[10] Y. Gurevich. Toward logic tailored for computatinal complexity. In *Computation and Proof Theory*, Lecture Notes in Math. 1104, 175–216, 1984.

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

[12] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[13] J. Köbler, J. Messner, and J. Torán. Optimal proof systems imply compllete sets for promise classes. *Information and Computation*, 184:71–92, 2003.

[14] 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.

[15] E. Mayordomo. Almost every set in exponential time is P-bi-immune. *Theoretical Computer Science*, 136(2): 487-506, 1994.

[16] 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.

[17] Z. Sadowski. On an optimal propositionl proof system and the structure of easy subsets. *Theoretical Computer Science*, 288(1):181–193, 2002.

[18] M.Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC'82)*, pages 137–146, 1982.

[19] 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.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de