The Logic in Computer Science Column

by

Yuri Gurevich

Microsoft Research
One Microsoft Way, Redmond WA 98052, USA
gurevich@microsoft.com

A Surprising Relationship Between Descriptive Complexity and Proof Complexity

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

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

Moritz Müller
Universität Passau
Fakultät für Informatik und Mathematik
moritz.mueller@uni-passau.de

In [4] Gurevich conjectured that there is no logic that captures PTIME. This is the main open problem of descriptive complexity. A central issue in proof complexity is whether p-optimal proof systems for the set TAUT of tautologies of propositional logic exist. It appears explicitly in [5] and implicitly already in the foundational paper of Cook and Reckow [3].

Around ten years ago Chen and Flum [1] showed that these two problems are tightly related: there is a p-optimal proof system for TAUT if and only if a certain logic considered by Gurevich in [4] captures PTIME. How surprising is this equivalence? It turns out that both statements are equivalent to the membership of a parameterized halting problem for Turing machines in a certain complexity class of parameterized complexity theory [2].

The purpose of this note is to present a short direct proof of (a variant of) the mentioned equivalence. It is intended to be accessible to non-experts. We follow the established style of this column and present the proof in dialogue form.

Professor Gurevich **G** *talks to one of his students* **SG**.

SG: You conjecture that there is no logic capturing PTIME. What is the underlying notion of a logic here?

G: Roughly speaking a *logic* is given by a map L and a binary relation $\models_L$. The map L assigns to every vocabulary τ, i.e., to every finite set of relation symbols, a set $L[\tau]$ of strings, the so-called *τ-sentences of L*. If $\mathcal{A} \models_L \varphi$, then $\mathcal{A}$ is a finite τ-structure and φ a τ-sentence of L. The map and the relation have to satisfy some natural properties, which I will explain if necessary. A sentence $\varphi \in L[\tau]$ *defines* the class $\mathrm{Mod}_L(\varphi)$ of τ-structures $\mathcal{A}$ such that $\mathcal{A} \models_L \varphi$.

SG: What does it mean that a logic captures PTIME?

G: For some vocabulary τ we view all instances of a given *(computational) problem* as τ-structures. We identify the problem with the class of its YES-instances. A logic *captures* PTIME if and only if its sentences define precisely the (classes of YES-instances of) problems in PTIME. Again some additional properties are required, which I will explain if necessary.

SG: Consider the logic L_1 where, for any τ, $L_1[\tau]$ is the set of polynomial time *clocked* algorithms (i.e., each algorithm comes with an explicit polynomial time bound). For such an algorithm $\mathbb{A}$ and any τ-structure declare $\mathcal{A} \models_{L_1} \mathbb{A}$ to mean that $\mathbb{A}$ accepts $\mathcal{A}$. Why does this logic not capture PTIME?

G: Because of an additional property required of a logic: $\mathrm{Mod}_L(\varphi)$ has to be closed under isomorphism for any sentence φ. A polynomial time algorithm might reject (the binary encoding of) some structure but accept (the binary encoding of) an isomorphic one.

SG: OK, then consider the logic L_2 where $L_2[\tau]$ for any τ is the set of polynomial time algorithms $\mathbb{A}$ that are *invariant*: if $\mathcal{A}$ and $\mathcal{B}$ are isomorphic τ-structures, then $\mathbb{A}$ accepts $\mathcal{A}$ if and only if $\mathbb{A}$ accepts $\mathcal{B}$. Define $\models_{L_2}$ as for L_1. Why does this logic not capture PTIME?

G: Because of a further additional property required of a logic: for every τ, the set $L[\tau]$ has to be decidable. But it is undecidable whether a clocked polynomial time algorithm is invariant.

SG: A further attempt with the logic L_3. Let $L_3[\tau] := L_1[\tau]$ but now define $\mathcal{A} \models_{L_3} \mathbb{A}$ to mean that $\mathbb{A}$ accepts $\mathcal{A}$ and $\mathbb{A}$ is invariant. For non-invariant $\mathbb{A}$ we have $\mathrm{Mod}_{L_3}(\mathbb{A}) = \emptyset$. Why does this logic not capture PTIME?

G: Because of an extra condition in what it means that a logic L captures PTIME: we require that for each τ there exists a *model-checker*, i.e., an algorithm that, given a τ-structure $\mathcal{A}$ and $\varphi \in L[\tau]$, decides whether $\mathcal{A} \models_L \varphi$. Such an algorithm does not exist for L_3.

SG: My last attempt. Set $L_4[\tau] := L_1[\tau]$ and define $\mathcal{A} \models_{L_4} \mathbb{A}$ to mean that $\mathbb{A}$ accepts $\mathcal{A}$ and $\mathbb{A}$ is n-invariant where n is the size of the universe of $\mathcal{A}$. That $\mathbb{A}$ is

n-invariant means: if $\mathcal{A}$ and $\mathcal{B}$ are isomorphic τ-structures of size at most n, then $\mathbb{A}$ accepts $\mathcal{A}$ if and only if $\mathbb{A}$ accepts $\mathcal{B}$. Why does this logic not capture PTIME?

G: Well, we do not know whether L_4 captures PTIME. More precisely, we do not know whether L_4 satisfies a final condition on a logic L capturing PTIME. This condition requires that for each fixed φ, the model-checker runs in polynomial time when restricted to inputs $(\mathcal{A}, \varphi)$.

SG: Where does this requirement come from?

G: Intuitively, it allows to view a logic capturing PTIME as a high level programming language for PTIME. More precisely, every φ in the logic is viewed as a program, and the model-checker is an interpreter which executes this program on any structure $\mathcal{A}$ in time polynomial in the size of $\mathcal{A}$. In addition, for every PTIME problem we can write such a program φ.

SG: To sum up, a logic capturing PTIME consists of a map L from vocabularies τ to sets $L[\tau]$ of τ-sentences, a relation $\models_L$ between τ-structures and τ-sentences, and a *model-checker*, an algorithm that given a τ-structure $\mathcal{A}$ and $\varphi \in L[\tau]$ decides whether $\mathcal{A} \models_L \varphi$, such that for every τ

- the set $L[\tau]$ is decidable;
- for every $\varphi \in L[\tau]$, the class $\mathrm{Mod}_L(\varphi) = \{\mathcal{A} \mid \mathcal{A} \models_L \varphi\}$ is closed under isomorphism;
- every problem in PTIME, if viewed as a class of τ-structures, equals $\mathrm{Mod}_L(\varphi)$ for some $\varphi \in L[\tau]$;
- for every fixed $\varphi \in L[\tau]$, the runtime of the model-checker on $(\mathcal{A}, \varphi)$ is polynomial in the size of $\mathcal{A}$.

G: Yes, this is how the question whether there exists a logic capturing PTIME is formulated in [4]. Your *naive logic* L_4 satisfies the first three items but I conjecture it does not satisfy the last one.

SG: It would be sufficient to have an algorithm that given $(\mathbb{A}, n)$ decides whether $\mathbb{A}$ is n-invariant in time $p_{\mathbb{A}}(n)$ where $p_{\mathbb{A}}$ is a polynomial that may depend on $\mathbb{A}$.

G: In fact, this is also necessary for L_4 capturing PTIME: let $\neg\mathbb{A}$ behave as $\mathbb{A}$ but flip the answer, i.e., $\neg\mathbb{A}$ accepts if and only if $\mathbb{A}$ rejects. To decide whether $\mathbb{A}$ is n-invariant, take some arbitrary structure $\mathcal{A}$ of size n and use the model-checker to check whether $\mathcal{A} \models_{L_4} \mathbb{A}$ or $\mathcal{A} \models_{L_4} \neg\mathbb{A}$.

Professor Cook **C** *talks to one of his students* **SC**.

SC: Some conjecture that there is no p-optimal proof system for the set TAUT of tautologies of propositional logic. What does this mean, and, in particular, what is the underlying notion of a proof system here?

C: A *proof system* is a polynomial time computable function P from the set of binary strings onto TAUT. A binary string x is a *P-proof* of the tautology $P(x)$. Being *p-optimal* means that for every other proof system P' there is a polynomial time computable function T translating P'-proofs into P-proofs of the same tautologies, i.e., such that $P'(x) = P(T(x))$ for all binary strings x.

SC: Define the proof system P_0 as follows. On input $(P, x, 1^t)$ where P is (an algorithm computing a) a proof system, x is a binary string, and $t \in \mathbb{N}$, simulate P on x for at most t many steps; if the simulation halts, return its output $P(x)$; otherwise, return some fixed tautology, say $(X \vee \neg X)$; also return $(X \vee \neg X)$ on inputs that are not of the required form. This is a map onto TAUT. If P is a proof system and p is a polynomial bound for its running time, then $x \mapsto (P, x, 1^{p(|x|)})$ is a translation as required. Why isn't P_0 a p-optimal proof system?

C: Because it is not decidable whether a given polynomial time algorithm is a proof system.

SC: Well, then we define P_1 as P_0 but now we consider inputs $(\mathbb{A}, x, 1^t)$ where $\mathbb{A}$ is an arbitrary (clocked) polynomial time algorithm. On such an input, P_1 first spends t steps to check whether $\mathbb{A}$ is $|x|$-sound before simulating it and proceeding as P_0. That $\mathbb{A}$ is *n-sound* means: $\mathbb{A}(y)$ is a tautology for all y with $|y| \leqslant n$. If the check fails or P_1 runs out of time, then it outputs $(X \vee \neg X)$.

C: It is unknown whether your *naive proof system* P_1 is p-optimal. Your translation $x \mapsto (\mathbb{A}, x, 1^{p(|x|)})$ needs a polynomial $p(|x|)$ so that the simulation and the $|x|$-soundness check can be done in time $p(|x|)$.

SC: It would be sufficient to have an algorithm that given $(\mathbb{A}, n)$ decides whether $\mathbb{A}$ is n-sound in time $p_{\mathbb{A}}(n)$ for some polynomial $p_{\mathbb{A}}$ that may depend on $\mathbb{A}$.

The two students **SG** *and* **SC** *meet,* **SC** *asks* **SG** *for her interests, and* **SG** *recounts her conversation with* **G**.

SC: After listening to you I believe that the existence of a p-optimal proof system implies that the naive logic L_4 is a logic for PTIME.

SG: Why?

SC: Assume there is a p-optimal proof system P. By a classical result of Levin [6], P has an optimal inverter $\mathbb{I}$. Being an *inverter* means that $\mathbb{I}$, given a tautology, outputs a P-proof of it; on other inputs $\mathbb{I}$ diverges. Being *optimal* means: for every inverter $\mathbb{I}'$ there is a polynomial p' such that $t_{\mathbb{I}}(x) \leqslant p'(t_{\mathbb{I}'}(x) + |x|)$ for every tautology x. Here, $t_{\mathbb{I}}(x)$ and $t_{\mathbb{I}'}(x)$ denote the runtimes of $\mathbb{I}$ and $\mathbb{I}'$ on x.

For $(\mathbb{A}, n)$, where $\mathbb{A}$ is a polynomial time algorithm and $n \geqslant 1$, to decide whether $\mathbb{A}$ is n-invariant is a problem in coNP (the complement is in NP!). By coNP-completeness of TAUT, there is a polynomial time function assigning to $(\mathbb{A}, n)$ a propositional formula $F^n_{\mathbb{A}}$ such that $F^n_{\mathbb{A}}$ is a tautology if and only if $\mathbb{A}$ is n-invariant.

Let $\mathbb{A}$ be invariant. Then all $F^n_{\mathbb{A}}$ are tautologies. One easily defines a proof system P' that has 1^n as a P'-proof of $F^n_{\mathbb{A}}$. By p-optimality of the proof system P there is a translation T, i.e., $P' = P \circ T$. Define an inverter $\mathbb{I}'$ of P that maps $F^n_{\mathbb{A}}$ to $T(1^n)$ – we can assume that one can recover n from $F^n_{\mathbb{A}}$ in polynomial time. By optimality of the inverter $\mathbb{I}$ of P, also $\mathbb{I}$ on $F^n_{\mathbb{A}}$ needs time $p_{\mathbb{A}}(n)$ for some polynomial $p_{\mathbb{A}}$.

SG: But runtime $p_{\mathbb{A}}(n)$ is ensured only on inputs $(\mathbb{A}, n)$ where $\mathbb{A}$ is invariant. On other inputs ($\mathbb{I}$ and) your algorithm might even diverge.

SC: Right. So run the algorithm in parallel with some brute force procedure that on $(\mathbb{A}, n)$ computes the minimal m such that $\mathbb{A}$ is not m-invariant; for invariant $\mathbb{A}$ this procedure does not halt. Otherwise it halts in some time depending only on $\mathbb{A}$. If it halts, check $n < m$. Then the runtime on inputs $(\mathbb{A}, n)$ with non-invariant $\mathbb{A}$ is also bounded as desired.

SG: I'm impressed. Now we know: if p-optimal proof systems exist, then my naive logic captures PTIME.

SG *asks* **SC** *for her interests, and* **SC** *recounts her conversation with* **C**.

SG: After listening to you I believe that if my naive logic captures PTIME, then your naive proof system is p-optimal.

SC: Why?

SG: So far we used formulations like "an algorithm accepts a structure." But what does it mean that an abstract structure is an input to an algorithm? Now we have to be more precise. We use binary codes of structures. For this purpose we assume that we deal with *standard structures*, the universe of a standard structure is the set $[n]$ $(:= \{1, 2, \ldots, n\})$ for some $n \geqslant 1$. Of course, every abstract structure is isomorphic to a standard structure. Then, once we have fixed an ordering on the relations of a vocabulary τ, every standard τ-structure corresponds to a unique binary string. To apply the algorithm to the structure means that this string is the input to the algorithm.

So let's come back to our problem. Let $\tau := \{<, One, Zero\}$ with binary $<$ and unary *One* and *Zero*. For a binary string $x = x_1 \ldots x_{|x|}$ and a natural number $m > |x|$ let the τ-structure $\mathcal{A}(x, m)$ have universe $[2m]$, interpret $<$ by the natural order on $[2m]$, *One* by $\{i \mid x_i = 1\}$, and *Zero* by $\{i \mid x_i = 0\}$. Given a (standard) τ-structure $\mathcal{B}$, one can check in polynomial time whether $\mathcal{B}$ is isomorphic to some $\mathcal{A}(x, m)$, and in the positive case compute the unique such pair (x, m). Such a $\mathcal{B}$ *codes* an assignment $\alpha_{\mathcal{B}}$ to m propositional variables: assign true or false to the j-th variable depending on whether $2j - 1$ is smaller than $2j$ in the order $<^{\mathcal{B}}$, the interpretation of $<$ in $\mathcal{B}$. Clearly, every assignment to m variables is coded by some $\mathcal{B} \cong \mathcal{A}(x, m)$.

SC: What does this help to reduce my soundness problem to your invariance problem?

SG: Let $\mathbb{A}$ be a polynomial time algorithm and without loss of generality assume that on any input it always outputs a propositional formula. Furthermore let $q_{\mathbb{A}}$ be a strictly increasing polynomial such that $q_{\mathbb{A}}(n)$ is an upper bound for the number of variables of $\mathbb{A}(x)$ for all x with $|x| \leqslant n$.

Define $\mathbb{A}^*$ to check, given a standard τ-structure $\mathcal{B}$, whether it is isomorphic to some $\mathcal{A}(x, q_{\mathbb{A}}(|x|) + 1)$. If not $\mathbb{A}^*$ rejects $\mathcal{B}$; otherwise, it computes $\mathbb{A}(x)$ which we have assumed to be a propositional formula. Let X be the "first" variable not occurring in $\mathbb{A}(x)$. Then the formula $(\mathbb{A}(x) \vee X)$ is satisfiable and ($(\mathbb{A}(x) \vee X)$ is a tautology if and only if $\mathbb{A}(x)$ is a tautology). The algorithm $\mathbb{A}^*$ accepts $\mathcal{B}$ if $\alpha_{\mathcal{B}}$ satisfies $(\mathbb{A}(x) \vee X)$ and otherwise rejects $\mathcal{B}$.

Then $\mathbb{A}$ is n-sound if and only if $\mathbb{A}^*$ is $2(q_{\mathbb{A}}(n) + 1)$-invariant.

SC: Why?

SG: If $\mathbb{A}$ is not n-sound, some $\mathbb{A}(x)$ with $|x| \leqslant n$ is not a tautology. Then $(\mathbb{A}(x) \vee X)$ is not a tautology but satisfiable. Choose a satisfying and a falsifying assignment for $(\mathbb{A}(x) \vee X)$. There are two structures $\mathcal{B}_1$ and $\mathcal{B}_2$ isomorphic to $\mathcal{A}(x, q_{\mathbb{A}}(|x|) + 1)$ such that $\alpha_{\mathcal{B}_1}$ and $\alpha_{\mathcal{B}_2}$ are these assignments. Then $\mathbb{A}^*$ accepts $\mathcal{B}_1$ but rejects $\mathcal{B}_2$. Thus $\mathbb{A}^*$ is not $2(q_{\mathbb{A}}(|x|) + 1)$-invariant and hence not $2(q_{\mathbb{A}}(n) + 1)$-invariant.

Now assume that $\mathbb{A}$ is n-sound. Assume $\mathbb{A}^*$ accepts a size $\leqslant 2(q_{\mathbb{A}}(n) + 1)$ structure $\mathcal{B}$. Then $\mathcal{B} \cong \mathcal{A}(x, q_{\mathbb{A}}(|x|)+1)$ for some $|x| \leqslant n$. Conversely, every such $\mathcal{B}$ has size at most $2(q_{\mathbb{A}}(n) + 1)$ and is accepted by $\mathbb{A}^*$ if $\alpha_{\mathcal{B}}$ satisfies $(\mathbb{A}(x) \vee X)$. By n-soundness of $\mathbb{A}$, the formula $(\mathbb{A}(x) \vee X)$ is a tautology and in particular, satisfied by $\alpha_{\mathcal{B}}$.

SC: To sum up, we proved:

Theorem. *Statements (2), (3) and (4) are equivalent and they imply (1).*

(1) There is a logic capturing PTIME.

(2) The naive logic captures PTIME.

(3) The naive proof system is p-optimal.

(4) There is a p-optimal proof system.

The students ask around whether (1) is equivalent to some natural weakening of (4) but nobody seems to know anything.

Acknowledgement We thank Albert Atserias and anonymous reviewers for comments on an earlier version of this text.

References

[1] Yijia Chen and Jörg Flum. From almost optimal algorithms to logics for complexity classes via listings and a halting problem. *Journal of the ACM*, 59(4):17:1–17:34, 2012.

[2] Yijia Chen and Jörg Flum. A parameterized halting problem. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 364–397. Springer, 2012.

[3] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.

[4] Yuri Gurevich. Logic and the challenge of computer science. In Egon Börger, editor, *Current Trends in Theoretical Computer Science*, pages 1–57. Computer Science Press, 1988.

[5] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *Journal of Symbolic Logic*, 54(3):1063–1079, 1989.

[6] Leonid A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.