Proof complexity

Jan Krajíček*

Mathematical Institute†
Academy of Sciences, Prague

Abstract

This note, based on my 4ecm lecture, exposes few basic points of proof complexity in a way accessible to any mathematician.

In many parts of mathematics one finds statements asserting that a finite object with a particular, feasibly verifiable, property does not exist. Such statements include, for example, the unsolvability of an equation, the non-existence of a combinatorial pattern or of an algebraic object, or the non-existence of a computation solving a problem. The qualification "feasibly" will mean throughout the paper "in polynomial time" (shortly, p-time).

A universal statement of this form is a statement that a propositional formula cannot be satisfied by any truth assignment or, equivalently, that the negation of the formula is a tautology. The qualification "universal" means, in particular, that proving statements about the non-existence of particular finite objects can be reduced to proving that particular propositional formulas are tautologies.

The ultimate goal of *proof complexity* is to show that there is no universal propositional proof system allowing for efficient proofs of all tautologies. This is equivalent to showing that the computational complexity class $\mathcal{NP}$ is not closed under the complementation (and it implies the famous conjecture that $\mathcal{P}$ differs from $\mathcal{NP}$).

By the universality propositional proof systems subsume methods from other parts of mathematics used for proving the non-existence statements. Because of this, even the partial results known at present (lower bounds for

*Partially supported by grant # A 101 94 01 of the Academy of Sciences and by project LN00A056 of The Ministry of Education of the Czech Republic.

†Also member of the Institute for Theoretical Computer Science of the Charles University.

some specific proof systems) revealed interesting links of proof complexity to logic, algebra, combinatorics, computational complexity,... .

I shall attempt to explain some basic points of proof complexity in a coherent manner accessible to any mathematician. I will first give few informal examples (Section 1) in order to motivate the main concepts and problems of proof complexity (Section 2). After that I will discuss two particular topics in proof complexity, one classic (Section 3) and one quite recent (Section 4), in some detail.

Informed reader will notice that I do not discuss here at all any lower bounds for particular proof systems although these are generally the most difficult and appreciated results in proof complexity. This is chiefly because of a lack of space (and time in the 4ecm lecture) as one needs a substantial background to appreciate that even results about particular proof systems say something about the original fundamental problems.

1 Examples of proof systems

Let $p_i(x_1, \ldots, x_n) = 0$, $i = 1, \ldots, k$, be a system of polynomial equations in $\mathbf{F}_q$. We want to know if the system is solvable in the field. We do not necessarily want to decide this by some algorithm but rather we want to "prove" that the system is either solvable, or to prove that it is unsolvable.

If the system is solvable then as a "proof" of the solvability will serve any solution $a = (a_1, \ldots, a_n) \in (\mathbf{F}_q)^n$. The soundness of any such proof is verified easily by checking that indeed all equations $p_i(a) = 0$ are satisfied. This can be done in p-time (as long as the polynomials are represented well, e.g. in the so called dense notation as an explicit sum of monomials).

If the system is unsolvable we can take as a "proof" polynomials $q_i \in \mathbf{F}_q[x_1, \ldots, x_n]$, $i = 1, \ldots, k+n$, such that in $\mathbf{F}_q[x_1, \ldots, x_n]$ it holds:

$$\sum_{i \leq k+n} q_i \cdot p_i \ = \ 1$$

where $p_{k+j} := x_j^q - x_j$ for $j = 1, \ldots, n$. This equality can be again checked in p-time. For any $a \in (\mathbf{F}_q)^n$ that would be a solution of the system it would hold that $p_i(a) = 0$ for all $i \leq k+n$. But that is impossible by the equality. Hence our "proofs" are sound: Whenever suitable q_i's exist, the system is unsolvable. In fact, by Hilbert's Nullstellensatz such q_i's always exists as long as the system is unsolvable in the field; hence this "proof system" is also complete.

Note that while we can bound the size of the proofs in the first case by the size of the polynomial system itself (i.e. of the problem instance), no similar bound is apriori possible (in general) for the proofs in the second case. The only bound on the size of some "proofs" is about q^n (this can be exponential in the size of the problem instance). This trivial bound is achieved by an exhaustive search of all potential solutions. When we later use the phrase that there is no apriori bound we mean a bound better than the trivial exponential one.

Our second example is from graph theory. Let G be a graph (finite, unordered). The issue is whether G is 3-colorable or not. Again we are not interested in an algorithm deciding this but in "proof systems" that would allow us to prove that it is or it is not, respectively.

A proof of the affirmative case is simple: Any 3-coloring will do, as its correctness can be checked in p-time. To prove the non-colorability we may use a method devised by Hajós [7]. He showed that there is a finite number of initial graphs, all non-3-colorable, and a couple of elementary rules how to obtain a new graph from two graphs already constructed, such that the class of graphs constructible in this way coincides with the class of non-3-colorable graphs. In particular, a sequence of graphs $H_1, \ldots, H_k$ which are constructed from the initial graphs by Hajós's rules and such that $H_k = G$ is a "proof" of non-3-colorability of G. The soundness of this proof system is obvious from the definition of the rules, the completeness is provided by Hajós's theorem [7].

Similarly as before, while we have a trivial upper bound on the size of proofs in the affirmative case we have no apriori subexponential upper bound for the sequences $H_1, \ldots, H_k$.

The third example I shall discuss is the most important one. Let φ be a propositional formula in some fixed complete language for propositional logic, e.g. the DeMorgan language $0, 1, \neg, \vee, \wedge$. We want to prove that φ is, resp. is not, satisfiable.

The satisfiability can be proved easily: Any satisfying assignment will do as a "proof". For the unsatisfiability we also have an elegant way of proving it: Note that φ is unsatisfiable iff $\neg\varphi$ is a tautology, and hence any proof of $\neg\varphi$ in some propositional calculus will serve well. Again we have no apriori upper bounds on such proofs that would be better than exponential (in the size of φ).

I shall conclude this set of examples by an example of a different nature. Let $f(x, y) = 0$ be a Diophantine equation in $\mathbf{Z}$, and let $B \in \mathbf{N}$ be a parameter. We want to prove, resp. to disprove, that the equation has an integer solution bounded in absolute value by B. The affirmative case is proved by simply producing such a solution. But now - and this is the point of this example - we do not seem to have any elegant way of proving unsolvability other than simply proving it in Mathematics, and then take as a proof its formalization in some theory used for formalizing Mathematics (e.g. in set theory). The soundness of such a proof comes from the soundness of the theory, and of course we have no apriori subexponential upper bound on its size.

Note that the key property that it can be verified in p-time whether something is a proof or not is maintained (this is due to the fact that axioms of set theory are given by a finite number of schemes and so it is p-verifiable if a string is an axiom or not).

2 Basic concepts and problems

We now leave behind classes of particular finite objects (polynomials, graphs, formulas, etc.) and enter the abstract framework usual in theoretical computer science. Finite objects are encoded by binary words, i.e. elements of $\{0,1\}^*$. Decisions problems are identified with the set of those problem instances for which the problem has the affirmative answer. That is, decision problems are languages $L \subseteq \{0,1\}^*$ and the original query is replaced by a query of the form $x \in_? L$. The size of a problem instance x becomes the length $|x|$ of the word x.

Definition 2.1 (Cook-Reckhow[6]) *A* proof system *for $L \subseteq \{0,1\}^*$ is a binary relation $P(x,y)$ satisfying the following three conditions:*

1. *Completeness: $x \in L \to \exists y; P(x,y)$.*
2. *Soundness: $\exists y; P(x,y) \to x \in L$.*
3. *p-verifiability: $P(x,y)$ is a p-time decidable relation.*

The important distinction between the affirmative cases and the negative cases, the existence of apriori short proofs, is formalized by the following definition.

Definition 2.2 (Cook-Reckhow[6]) *Proof system $P(x,y)$ is* p-bounded *iff there exists $k \geq 1$ such that for all $x,y \in \{0,1\}^*$:*

$$P(x,y) \to \exists z(|z| \leq (|x|+2)^k); P(x,z) \ .$$

That is, the size of proofs in a p-bounded P can be apriori bounded by a polynomial in the size of the problem instance.

Definition 2.3 *$\mathcal{NP}$ is the class of languages L admitting a p-bounded proof system. $co\mathcal{NP}$is the class of complements of $\mathcal{NP}$-languages.*

In our examples, the sets of instances for which the problems has affirmative answer form $\mathcal{NP}$-sets (solvable polynomial systems, 3-colorable graphs, satisfiable formulas, etc.), while the sets of instances with the negative answer form $co\mathcal{NP}$-sets (unsolvable polynomial systems, non-3-colorable graphs, unsatisfiable formulas or tautologies, etc.).

The following problem, the so called *$\mathcal{NP}$ versus $co\mathcal{NP}$ problem*, is the central problem in proof complexity. It formalizes the question whether or not there are efficient ways to prove the negative cases in our, and in many other similar, examples.

Problem 2.4

$$\mathcal{NP} \ =_? \ co\mathcal{NP}$$

That is, does the implication $L \in \mathcal{NP} \rightarrow \{0,1\}^* \setminus L \in \mathcal{NP}$ hold for any L?

It is a prevailing conjecture that $\mathcal{NP} \neq co\mathcal{NP}$. The universality of propositional tautologies mentioned in the introduction means precisely what is stated in the following theorem. The theorem is a consequence of the so called $\mathcal{NP}$-completeness of the set of satisfiable formulas (Cook [4]).

Theorem 2.5 (Cook-Reckhow[6]) *$\mathcal{NP} = co\mathcal{NP}$ iff the set $TAUT$ of propositional tautologies (in DeMorgan language) admits a p-bounded proof system.*

Proof complexity (tacitly propositional proof complexity) studies proof systems for $TAUT$ with the main aim to prove that none of them is p-bounded. It is a many-faceted area where computational complexity theory overlaps with mathematical logic. Topics studied include:

- Connections with first order theories and central issues of mathematical logic (bounded arithmetic, Gödel's theorem, etc.).

- Lower bounds for particular proof systems (resolution, bounded depth Frege systems, effective Nullstellensatz, polynomial calculus, cutting planes, theory of discretely ordered modules, etc.).

- Upper bounds and various simulations between proof systems.
- Connections with computational complexity (boolean complexity, cryptography, derandomization, communication complexity, etc.).

I shall say nothing about the middle two items (with the exception of a couple of brief remarks at the end of Section 3) but I will discuss examples from the first and from the last items in the next two sections.

3 Consistency statements

Let $A(x, y)$ be a p-time relation. Given $a \in \{0,1\}^n$ we want to express that

$$\forall y \in \{0,1\}^m; A(a, y) \ .$$

For example, we can express in this way that a is an unsolvable polynomial system, an unsatisfiable formula, etc. In these cases $m \leq poly(n)$, and we will always assume this bound.

Consider an algorithm executed by a Turing machine running in time $poly(n, m) \leq poly(n)$ and deciding $A(x, y)$. Turing machine is the established mathematical model of a computer but any other faithful model would work just fine. At time $t = 0$ the tape of the machine (i.e. the memory) contains bits of x, y, and additional bits, a finite number of them encoding the state of the machine. Call all these bits $w^0_1, w^0_2, \ldots, w^0_T$. Generally, in time t, the memory holds bits w^t_i, for $i \leq T$. The parameter T here is some universal bound deduced from the time bound of the machine, so $T \leq poly(n)$.

The key observation is that any bit w^{t+1}_i depends only on a finite number of bits w^t_j, some j's. This allows to write down propositional formula $Correct^A_n(x, y, w)$, the conjunction of all local conditions expressing that each bit w^{t+1}_i is correctly computed from w^t_j's. Note that the number of bits w^{t+1}_i is $poly(n)$ and so the size of $Correct^A_n(x, y, w)$ is also $poly(n)$.

The main property of this propositional representation is this:

- $\forall y \in \{0,1\}^m; A(a, y)$ holds true iff the propositional formula

  $$Correct^A_n(a, y, w) \rightarrow w^T_{output}$$

  is a tautology.

Here w^T_{output} is the bit representing the output of the machine, i.e. it is equal to 1, true, iff the machine yields the affirmative answer. We shall denote the implication by symbols:

$$||A||^n(a, y) \ .$$

So this is a propositional formula that has bits of a substituted for bits $x_1, \dots, x_n$, has bits $y_1, \dots, y_m$, and also bits w^t_i that we do not show explicitly in this notation.

With this general way of translating bounded universal statements into propositional formulas we now define a formula expressing *the soundness* of a proof system Q:

$$\forall x, y, z(|x|, |z| \leq |y|); Q(x, y) \rightarrow Sat(x, z) \ .$$

We assume with a loss of generality that $Q(x, y)$ implies $|x| \leq |y|$ (we can always stipulate that a formula x is "a part" of a proof y). $Sat(x, z)$ is the p-time relation "z is a truth assignment satisfying x", $|z| \leq |x| \leq |y|$.

Let $||Ref_Q||^n(x, y, z)$ be the propositional translation of this formula for $|y| = n$. *Assume* that we have a proof σ_n of this tautology in a proof system P. Given any Q-proof π of any tautology τ reason *in* P is follows:

1. $Q(\tau, \pi) \rightarrow Sat(\tau, z)$

   [this is an instance of tautology $||Ref_Q||^n(x, y, z)$ proved by σ_n]

2. $Q(\tau, \pi)$

   [this is a true instance and is proved by evaluating it]

3. $Sat(\tau, z)$

   [from 1. and 2.]

4. $Sat(\tau, z) \equiv \tau(z)$

   [this is a general fact that is verified by induction on the logical complexity of τ]

5. $\tau(z)$

   [from 3. and 4.]

To summarize: Given a P-proof σ_n of $||Ref_Q||^n(x, y, z)$ we can transform any Q-proof π of size n into a P-proof π^* of the same formula, and of size

$$|\pi^*| \leq O(|\sigma_n| + poly(n)) \ .$$

That is, "any" P proving tautologies $||Ref_Q||^n(x, y, z)$ by p-size proofs is "at least as good as" Q. The concept "at least as good as" is called *simulation*: P simulates Q iff P-proofs are at most polynomially longer than Q-proofs. In particular, if Q is p-bounded, so is P. I remark without elaborating on it that many Q do prove their own soundness in p-size, and so the formulas $||Ref_Q||^n(x, y, z)$ are actually the hardest tautologies that Q proves in p-size. See [8, Chpt.9] for details.

The formula expressing the soundness of Q has the form $\forall y; A(x, y)$ (we leave out the bounds on y) such that it is universally true for all x and not just for some instances $x := a$. For such formulas it seems more natural to consider their proofs in a first order theory rather than to perform a redundant translation, for each length of x, to propositional formulas and then to prove all these formulas separately.

A relation between first order theories and proof systems can be indeed developed; the theory is called *bounded arithmetic* (the term comes from a universal form of the theories one can take in this context). I shall not elaborate on it but just informally state its main points. See [8] for details (original references include [5, 15, 1]).

Proof systems P and theories T come in pairs such that:

1. When T proves $\forall x, y; A(x, y)$ then tautologies $||A||^n(x, y)$ have p-size P-proofs.

2. T proves the soundness of P and if T proves the soundness of Q then P simulates Q.

3. A form of converse to 1. (this needs some background in model theory that I shall omit).

One can view the relation between theories and proof systems as being analogous to the relation of algorithms and circuits; it is the uniform and the non-uniform version respectively of the same concept.

Before leaving these issues we remark that it allows to interpret lower bounds for particular proof system P as lower bounds to a whole class of proof systems whose soundness can be proved in P. This is often the case when the proof system is based on utilizing one combinatorial or algebraic fact (as was the case in some of our examples and it is the case in most proof systems studied so far).

Property 1. can be used to prove upper bounds on the size of P-proofs (it is often much easier to see that T proves a universal formula than to construct short P-proofs of its individual instances). Property 2. can be used for proving simulations between proof systems. In fact, all more involved upper bounds or simulations have been first prove via bounded arithmetic.

4 Proof complexity generators

The first issue one has to deal with when trying to prove lower bounds for a proof system P, or a class of proof systems, is to formulate tautologies that could be hard for the proof system. To produce sensible candidate tautologies appears actually surprisingly difficult. The point is that the tautology should be "hard" but should have also a clear intuitive meaning and structure in order that we are able to devise a lower bound proof. I use the verb "devise" because it can be showed that *any* lower bound proof amounts to constructing a model (in a precise logical meaning) for the negation of the tautology.

We have, at present, three categories of such candidate hard tautologies:

1. Formulas $||Ref_Q||^n(x, y, z)$ for "strong" Q.
2. Combinatorial principles such as the pigeonhole principle.
3. Complexity/logic motivated candidates.

The first type of formulas is difficult to use for lower bounds because the adjective "strong" is difficult to substantiate. The combinatorial formulas work well but only for weaker systems.

I shall discuss in this section an example of candidates of the third type, the so called τ-formulas. I shall give enough details to informally explain main ideas but I make no attempt for an exhaustive or for the most general exposition (the interested reader may start with [12]). The rout to the τ-formulas went via feasible interpolation [9, 16] and provability of the dual weak pigeonhole principle in bounded arithmetic [10] (at least in my case). They have been defined in [10] and independently in [2], and their theory is being developed [11, 18, 12, 19, 13]. A more detailed discussion of background than I offer below can be found in the introductions to [12] and [19].

Consider a p-time map g with restrictions

$$g_n \; : \; \{0,1\}^n \to \{0,1\}^m$$

with $n < m \leq poly(n)$. We assume that $|g(x)|$ depends only on $|x|$. Hence $m := m(n)$ depends only on n and we will assume for simplicity of the notation that $m(n) \neq m(n')$ if $n \neq n'$. As $m > n$, we have $\{0,1\}^m \setminus Rng(g) \neq \emptyset$. Let $b \in \{0,1\}^m \setminus Rng(g)$. Formula $\tau_b(g)$ expresses that b is outside of the range of g, "$g(x_1, \ldots, x_n) \neq b$", i.e. it is

$$||g(x) \neq y||^n(x, y/b) \ .$$

Note that its size is $poly(n, m) \leq poly(n)$.

We will say that g *hard* for P iff

- For every $k \geq 1$ and sufficiently large n, for no $b \in \{0,1\}^m \setminus Rng(g)$ has formula $\tau_b(g)$ a P-proof of size less than $m(n)^k$.

For a fixed $k \geq 1$ define $Easy_k \subseteq \{0,1\}^*$ consisting of those y such that

$$\exists z(|z| \leq |y|^k); P(\tau_y(g), z) \ .$$

The hardness of g means that all these sets $Easy_k$ are finite. So what we want is a map g with parameters as above such that $Rng(g)$ intersects all infinite $\mathcal{NP}$-sets.

This is akin to the definition of pseudo-random number generators from cryptography (more precisely to that of hitting set generators): These are maps that intersect all $\mathcal{P}/poly$-sets of non-negligible density. The relaxation from infinite sets to sets of positive density would not be such a problem for us: It would merely mean that perhaps not all τ-formulas are hard as we have required, but the fraction of b's yielding easy instances would be negligible.

What is difficult is the requirement that g should intersect $\mathcal{NP}$-sets rather than $\mathcal{P}/poly$-sets. Even the existence of pseudorandom generators is proved only under other assumptions (like the existence of one-way functions) and for the version with $\mathcal{NP}$-sets we have nothing similar. There is a construction of a weaker type of generators, the so called Nisan-Wigderson [17] generators, sufficient in derandomization. Their existence can be proved (even w.r.t. $\mathcal{NP}$-sets) from a plausible hypothesis in boolean complexity. But the problem here is that the parameters achieved are insufficient for our purposes (the time complexity of g, and hence the size of $\tau_b(g)$'s, grows with the constant $k \geq 1$). [19] conjectures that original parameters suffice, while different parameters were proposed in [12]. Detailed discussions of this can be found, from somewhat different perspectives, in these two papers.

Instead discussing the technicalities more we shall consider another type of results, showing that in a particular sense there is "the hardest" g. For that we need to recall the notion of *a boolean circuit*. It is an object similar to a propositional formula except that it is represented very economically: Any subformula (or subcircuit) is written just once, even if it is needed in several occurrences. It is easy to see that a circuit of size s can be encoded by $O(s\log(s))$ bits.

Let C be a circuit with k inputs and of size at most $2^{k/3}$. By the above it is encoded by a string of $O(2^{k/3}k/3) < 2^{k/2}$ bits. Denote by $\mathbf{tt}(C)$ the *truth table* of the boolean function on $\{0,1\}^k$ computed by C; it is an element of $\{0,1\}^{2^k}$. The *truth-table function*

$$\mathbf{tt} \; : \; C \in \{0,1\}^{2^{k/2}} \;\rightarrow\; \mathbf{tt}(C) \in \{0,1\}^{2^k}$$

has the parameters $n = 2^{k/2}$ and $m = 2^k$ we want from g, and it is p-time computable.

The truth-table function will be the hardest one, but first we need to modify a bit the definitions of hardness. I shall discuss this only informally.

An intuitive drawback in the definition of hardness is that although it may be hard to prove that any particular b is outside of the range of g one still cannot "consistently think" in P that g is onto. This is because it may be that some disjunction of the form

$$g(x) \neq b \vee g(x') \neq b' \vee \ldots$$

has a short P-proof, or bit more generally a disjunction of the form

$$g(x) \neq b \vee g(x') \neq b'(x) \vee \ldots$$

where $b'(x)$ is a circuit computing b' from x, etc. Define informally that g is *very hard* for P if "no such" disjunction has p-size P-proof, for $n >> 0$. The technical terms used here are pseudosurjectivity or iterability (cf. [12]).

Theorem 4.1 (Krajíček[12]) *If there is any g very hard for P containing resolution then the truth table function* $\mathbf{tt}$ *is very hard for P too.*

At least for the case of P being resolution we can prove the hypothesis of the theorem.

Theorem 4.2 (Razborov[19]) *There is a g that is very hard for resolution.*

The theorems imply the following corollary.

Corollary 4.3 *The truth table function* $\mathbf{tt}$ *is very hard for resolution. In particular, it is also hard for resolution.*

Formula $\tau_b(\mathbf{tt})$ expresses that b is a truth table of a function with large ($> 2^{k/3}$) circuit complexity (the size $2^{k/3}$ has been chosen for our discussion but can be almost anything). Hence these formulas are indeed hard if it is hard to prove circuit lower bound for *any* boolean function. This puts us in a bit peculiar situation: Our program succeeds, i.e. the τ-formulas are very hard even for strong proof systems, if it is hard to prove circuit lower bounds, i.e. it is hard to carry other programs in complexity theory that reduce various conjectures (like $\mathcal{P} \neq \mathcal{NP}$ or universal derandomization of probabilistic computations) to circuit lower bounds.

5 A broader perspective

Let me conclude with a look at proof complexity problems from a distance. Around 1900 mathematicians were worried about questions as:

- Is the consistency of Mathematics provable?
- Is predicate calculus (i.e. what is true in all structures) algorithmically decidable?

The first issue led to Gödel's theorem and Gentzen's proof theory analysis, while the second led to the work of Turing, Church, Kleene and others (a formal definition of the notion of algorithm, undecidable problems).

We can view the problems of complexity theory as quantitative versions of the the above questions. In particular:

- Is the consistency of Mathematics w.r.t. proofs of size n provable in size *comparable* to n?
- Is it *feasibly* decidable what is true in all structures of size n?

If "comparable" means polynomially bounded and "feasibly" means in p-time, then the first problem is exactly $\mathcal{NP} =_? co\mathcal{NP}$ while the second one is $\mathcal{P} =_? \mathcal{NP}$.

The links with logic run deep. For example, there is a quantitative version of Gödel's theorem that, if true, would imply that no proof system can simulate all other proof systems, and hence $\mathcal{NP} \neq co\mathcal{NP}$ and $\mathcal{P} \neq \mathcal{NP}$. See [15]; there is an exposition in [8] too or a brief and non-technical one in [14].

References

[1] M. Ajtai, The complexity of the pigeonhole principle, in: *Proc. IEEE 29^{th} Annual Symp. on Foundation of Computer Science*, (1988), pp. 346-355.

[2] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson, Pseudorandom generators in propositional proof complexity, *Electronic Colloquium on Computational Complexity*, Rep. No.**23**, (2000). Ext. abstract in: *Proc. of the 41^{st} Annual Symp. on Foundation of Computer Science*, (2000), pp.43-53.

[3] S. R. Buss, *Bounded Arithmetic.* Naples, Bibliopolis, (1986).

[4] S. A.Cook, The complexity of theorem proving procedures, in: *Proc. 3^{rd} Annual ACM Symp. on Theory of Computing*, (1971), pp. 151-158. ACM Press.

[5] S. A.Cook, Feasibly constructive proofs and the propositional calculus, in: *Proc. 7^{th} Annual ACM Symp. on Theory of Computing*, (1975), pp. 83-97. ACM Press.

[6] S. A. Cook and A. R. Reckhow, The relative efficiency of propositional proof systems, *J. Symbolic Logic*,**44(1)**, (1979), pp.36-50.

[7] G. Hajós, Uber eine Konstruktion nicht n-farbbarer Graphen, Wiss. Z. Martin-Luther-Univ., Halle-Wittenberg, *Math. Natur. Reihe*, **10**, (1961), pp.116-117.

[8] J. Krajíček, *Bounded arithmetic, propositional logic, and complexity theory*, Encyclopedia of Mathematics and Its Applications, Vol. **60**, Cambridge University Press, (1995).

[9] J. Krajíček, Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic, *J. Symbolic Logic*, **62(2)**, (1997), pp. 457-486.

[10] J. Krajíček, On the weak pigeonhole principle, *Fundamenta Mathematicae*, Vol.**170(1-3)**, (2001), pp.123-140.

[11] J. Krajíček, Tautologies from pseudo-random generators, *Bulletin of Symbolic Logic*, **7(2)**, (2001), pp.197-212.

[12] J. Krajíček, Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds, *Journal of Symbolic Logic*, **69(1)**, pp.265-286, (2004).

[13] J. Krajíček, Diagonalization in proof complexity, *Fundamenta Mathematicae*, to app.

[14] J. Krajíček, Hardness assumptions in the foundations of theoretical computer science, *Archive for Mathematical Logic*, to app.

[15] J. Krajíček, and P. Pudlák, Propositional proof systems, the consistency of first order theories and the complexity of computations, *J. Symbolic Logic*, **54(3)**, (1989), pp.1063-1079

[16] J. Krajíček and P. Pudlák, Some consequences of cryptographical conjectures for S^1_2 and EF", *Information and Computation*, Vol. **140 (1)**, (January 10, 1998), pp.82-94.

[17] N. Nisan, and A. Wigderson, Hardness vs. randomness, *J. Comput. System Sci.*, Vol.**49**, (1994), pp.149–167.

[18] A. A. Razborov, Resolution lower bounds for perfect matching principles, in: *Proc. of the 17th IEEE Conf. on Computational Complexity*, (2002), pp.29-38.

[19] A. A. Razborov, Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution, preprint, (May'03).

Mailing address:

Mathematical Institute
Academy of Sciences
Žitná 25, Prague 1, CZ - 115 67
The Czech Republic
`krajicek@math.cas.cz`