# Hardness assumptions in the foundations of theoretical computer science

Jan Krajíček[*]

Mathematical Institute[†]
Academy of Sciences, Prague

The original impetus for writing this expository note was an idea of J. Nešetřil to edit a series of articles (in a series edited at the Institute for Theoretical Informatics at the Charles University in Prague) on the state of Theoretical Computer Science (TCS). Additional excuses were a lecture on the P/NP problem I gave during the celebrations of the 50th anniversary of the Mathematical Institute of the Academy of Sciences in Prague (December'02) and a lecture I planned for (but did not deliver) the *Takeuti Symposium* at the Kobe University (December'03). But less specifically, the reason is simply the growing interest of mathematicians in other areas to find out what is all this commotion about P/NP about.

I shall outline four fundamental problems of TCS. First informally, then formally but without excessive details. I shall also sketch three fundamental theorems, and discuss one particular challenge for the future. I shall conclude with a few short, very subjective, remarks on the state and the future of TCS.

The reader will notice that I write here boldly about TCS but later on consider really exclusively only complexity theory. The reason is that that is the part of TCS I know something about. But perhaps it is not such a restriction as the complexity theory is undoubtedly the foundational kernel of TCS.

[*]Partially supported by grant # A 101 99 01 of the Academy of Sciences of the Czech Republic and by project LN00A056 of The Ministry of Education of the Czech Republic. A part of this paper was written while visiting the Mathematical Institute in Oxford, supported by the EPSRC.

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

This note is not supposed to be an account of historic developments (for that the reader may consult [4] or [18]). Neither do I wish to explain basic notions like Turing machine or a boolean circuit. Rather this should be understood as a non-technical explanation of four fundamental problems and one fundamental challenge to an interested outsider having already some basic knowledge.

## 1 Four foundational problems - informally

The mathematical notion of algorithm, defined in the 1930s by A. Turing and A. Church, grew from the study of the strength of formal systems in mathematical logic (D. Hilbert, K. Gödel and others). It may very well be the most important notion mathematics discovered in the last century, and when saying this I realize that there is a serious competition of quite sophisticated constructs in various other fields of mathematics. This lead to the notion of a universal algorithm (universal Turing machine) and to the realization of the fact that there are sets (of strings) that can be enumerated by an algorithm but no algorithm can decide the membership in such a set.

In a formal sense, the NP-completeness and the P/NP problem are just "bounded" versions of these two facts, meaning that we allow algorithms running in polynomial time only. But the actual road to the P and NP has been more subtle.

The notion of a feasible algorithm emerged during 1960s in the work of A. Cobham, J. Edmonds, M. O. Rabin and others. The consensus has been reached that the informal notion of feasibility corresponds to the formal notion of polynomial time. Important properties of this notion is its robustness (independence from the machine model) and the fact that all (deterministic) algorithms intuitively branded as feasible are also polynomial time. The class P is the class of decision problems solvable by a polynomial time algorithm. As usual we encode finite objects by binary strings and identify a decision problem with the set of all strings (a language) for which the problem has the affirmative answer.

Incidentally, the definition of the class NP and the notion of the NP-completeness also grew from mathematical logic, this time from the study of the strength of propositional calculi. A canonical language in the class NP - which I define in a moment - is the problem of satisfiability of propositional formulas (the language is called SAT). S. A. Cook [2] was studying the question whether some propositional calculus can be applied to solve SAT in polynomial time, and he arrived at the P/NP problem.[1] A property SAT has is that even if it may be difficult to decide the membership in SAT, whenever $\varphi \in$ SAT this fact can be witnessed by information encoded in a short string: the satisfying assignment. NP is the class of languages for which the membership of a string can be feasibly witnessed. The P/NP problem asks whether or not the two classes are different. It is generally conjectured that they are indeed different.

We may view the classes P and NP as polynomial time analogs of the classes of recursive and recursively enumerable sets. Its is known that the latter two classes are different. But the diagonal argument used for demonstrating this cannot be used to separate P and NP. However, the notion of a complete recursively enumerable set does have an analogy. It is the notion of an NP-complete problem: A problem is NP-complete if it is in NP and any other NP-problem can be reduced to it by a polynomial time algorithm.

Thinking a bit more about propositional logic leads us to the NP/coNP problem. We may not know how to decide if a formula is satisfiable but can one at least also feasibly witness the fact that it is not? A formula is not satisfiable iff its negation is a tautology. So we are, in effect, asking if it is possible to feasibly witness the fact that a propositional formula is a tautology. A moment' reflection discovers that this is the same as asking if the class NP is closed under the complements, i.e. if NP=coNP, where coNP is the class of complements of the NP-languages. This is the NP/coNP problem. A prevailing view is that the two classes differ.

The third problem I want to discuss, the P/BPP problem, grew from revisiting the original assumption that feasible algorithms are the polynomial time algorithms. Namely, what about probabilistic algorithms running in polynomial time? BPP is the class of problems decidable by such algorithms where the probability of the correct answer is at least, say, 99%. Clearly BPP contains P and the P/BPP problem asks if this inclusion is proper or not. Unlike the previous two problems where the generally accepted conjecture is that all classes P, NP, and coNP are different, here the consensus is weaker. However, the majority opinion would be that actually P=BPP. The attempts to prove this can be found in the literature under the heading "derandomization" (but there is much more in derandomization than this). I shall review a result supporting this conjecture in Section 4.

[1]This has been independently discovered also by Levin[14]. Other people (notably G. Asser, J. H. Bennett, J. Edmonds, H. Scholz, R. Smullyan) studied classes equivalent to NP, coNP or their intersection but either in a different context without a reference to computational complexity or with informal definitions. It was Cook who defined the notions precisely and formulated the P/NP problem.

It is a high time to mention boolean circuits, introduced by C. Shannon in the late 1940s. The time of an algorithm corresponds to the size of a circuit. Any algorithm can be transformed into a sequence of circuits (with the size bounded by a function polynomial in the time bound) computing the same language. The opposite is not true, however. One can view the circuits as a "non-uniform" algorithm, meaning that the algorithm can actually change with the size of the input; it just needs to obey the same time bound. The problems solvable by polynomial size circuits, i.e. polynomial time non-uniform algorithms, form the class P/poly. Trivially, P/poly contains P. Not trivially but still easily: P/poly also contains BPP.

The last problem I want to mention is, at the first glance, of a bit different character. Briefly: Is cryptography possible? The well developed theory of cryptography based on computability hardness assumptions (see e.g. [6]) does what is intended, i.e. it provides a secure encryption of messages and other cryptographic tasks.

A key requirement for a large part (but not all) of this theory of a secure cryptography is the ability to produce bits that look random to a computationally restricted adversary. Such bits are called pseudo-random and the functions that do produce them are pseudo-random number generators (PRNG). The precise definition of what is needed is this. Let $G : \{0,1\}^* \to \{0,1\}^*$ be a function extending the inputs by one bit and let $G_n : \{0,1\}^n \to \{0,1\}^{n+1}$ be its restriction to $\{0,1\}^n$. The hardness $H(G_n)$ of $G_n$ is the minimal $s$ such that there is a boolean circuit $C$ in $n+1$ variables of size $s$ such that

$$\mid Prob_{x\in\{0,1\}^n}[C(G_n(x))=1] \;-\; Prob_{y\in\{0,1\}^{n+1}}[C(y)=1] \mid \; > \frac{1}{s}$$

The hypothesis needed for the theory of computationally secure cryptography is the following:

- There exists a polynomial time computable function $G$ extending inputs by one bit such that its restriction $G_n : \{0,1\}^n \to \{0,1\}^{n+1}$ to $\{0,1\}^n$ has the hardness $H(G_n) \geq \exp(n^{\Omega(1)})$.

Functions with this property are called strong PRNGs. Here the consensus is still weaker than in the case of the P/BPP problem. But the majority opinion would still be, I suppose, in support of the conjecture.

## 2 Formal statements of the four problems

To explain the problems precisely I shall put them, with the exception of the last one, under one formal umbrella. We shall consider Turing machines $M$ that get as an input a pair $(x, w) \in \{0,1\}^* \times \{0,1\}^*$ and that run in time polynomial in the length $n = |x|$ of $x$. Let $p(n)$ be the polynomial bound. Hence it makes no sense for $w$ to be longer than $p(n)$ as the machine would not have time to even read the whole of $w$.

For $x$ denote by $M_x$ the subset of $\{0,1\}^{\leq p(n)}$ defined by

$$M_x \ := \ \{w \in \{0,1\}^{\leq p(n)} \mid M \text{ accepts the pair } (x, w)\}$$

All classes P, NP, coNP, BPP can be defined in a similar way as the classes of all languages $L$ such that there is a Turing machine $M$ with the constrains as above and such that the membership to the classes is defined by the following respective conditions:

- Class P: $x \in L$ iff $\overline{0} \in M_x$. (The string $\overline{0}$ plays a role of "any fixed string".)
- Class NP: $x \in L$ iff $M_x \neq \emptyset$.
- Class coNP: $x \in L$ iff $M_x = \emptyset$.
- Class BPP:
  1. If $x \in L$ then $|M_x| \geq \frac{99}{100}|\{0,1\}^{\leq p(n)}|$.
  2. If $x \notin L$ then $|M_x| \leq \frac{1}{100}|\{0,1\}^{\leq p(n)}|$.

The P/NP problem thus asks if we can avoid the exhaustive search for witnesses; more precisely, if we can decide whether $M_x$ is empty or not without looking at any witness. The NP/coNP problem similarly asks if we can replace an existential quantification over $\{0,1\}^{\leq p(n)}$ by a universal quantification. The P/BPP problem asks again if we can avoid looking at witnesses assuming that there is either very few or very many of them.

The existence of strong PRNGs does not seem to have a natural reformulation in this style (contrary to what is occasionally claimed about the class UP).

## 3 Three important theorems

I shall mention three important results that provide reductions of three of the four problems to hardness assumptions of a similar nature.[2]

The question of a possibility of a similar reduction for the fourth problem NP/coNP is discussed in Section 4.

### 3.1 NP-completeness

A language $L \in$ NP is NP-complete iff any other NP-language $L'$ can be reduced to $L$ by a polynomial time function $f : \{0,1\}^* \to \{0,1\}^*$: For all $x$, $x \in L'$ iff $f(x) \in L$. The intuitive meaning is that any algorithm solving $L$ can be transformed - via $f$ - into an algorithm solving $L'$.

**Theorem 3.1 ([2, 14])** *SAT is NP-complete.*

There are literally thousands of NP-complete problems. This means that the notion is very successful; it works as a threshold of unfeasibility for many problems.

Using the trivial inclusion of P in P/poly we can formulate a hardness reduction: P $\neq$ NP assuming that

- SAT cannot be decided by polynomial size boolean circuits.

### 3.2 Nisan-Wigderson generators

Assume we had a small subset $A \subseteq \{0,1\}^{\leq p(n)}$ such that for any language $L$ in BPP and for any machine $M$ certifying it in the sense of Section 2 it holds that

$$(*) \qquad \left| \frac{|M_x \cap A|}{|A|} - \frac{|M_x|}{|\{0,1\}^{\leq p(n)}|} \right| \leq \frac{1}{4}$$

Using the set $A$ we could "derandomize" machine $M$ as follows. Given $x \in \{0,1\}^*$, we run $M$ on all pairs $(x, w)$, $w \in A$, and then we output the answer that appeared in the majority of cases. If $x \in L$, then $\frac{|M_x|}{|\{0,1\}^{\leq p(n)}|} \geq \frac{99}{100}$ and so we get an accepting answer for the fraction of at least $\frac{|M_x \cap A|}{|A|} \geq \frac{99}{100} - \frac{1}{4} \geq \frac{7}{10}$ elements $w$ of $A$. On the other hand, if $x \notin L$ then we get

[2]As pointed out by S. A. Cook, the conjectures that P $\neq$ NP and NP $\neq$ coNP can be deduced also from *easiness* assumptions. In particular, if exponential time E (resp. nondeterministic exponential time NE) has subexponential (even submaximal) size circuits then P $\neq$ NP (resp. NP $\neq$ coNP).

a positive answer in at most $\frac{1}{100} + \frac{1}{4} \leq \frac{3}{10}$ fraction of all $w \in A$. Hence our simulation correctly decided the membership in $L$. The only issue is how effective it really is. Clearly, this depends on how effectively we can list elements of $A$ and how large $A$ is. What we need is $A$ of size $n^{O(1)}$ listed by an algorithm running also in time $n^{O(1)}$. That is exactly provided by the following theorem.

E is the class of languages computable in exponential time $2^{O(n)}$.

**Theorem 3.2 ([8])** *Assume that there is $\delta > 0$ and a language $L$ in E such that any boolean circuit computing $L$ must have the size at least $2^{\delta n}$.*

*Then for any polynomial time bound $p(n)$ there are $k \geq 1$ and a function*

$$g : \{0,1\}^{k \log(n)} \to \{0,1\}^{\leq p(n)}$$

*computable in time $n^{O(1)}$ such that the set $A := Rng(g)$ satisfies the condition $(*)$ for all machines $M$ running in time $\leq p(n)$.*

Hence we have the following hardness reduction: P = BPP assuming that

- Boolean circuits computing the complete language in the exponential time class E must have the size at least $2^{\Omega(n)}$.

## 3.3 Strong pseudo-random number generators

The hypothesis that there exist strong PRNGs (see Section 2) does not look terribly intuitive and one hopes that it is true simply for utilitarian reasons.

The following theorem shows that the hypothesis about the existence of strong PRNGs is equivalent to another hypothesis that looks intuitively much more plausible.[3]

A one-way function is a polynomial time function that is very hard to invert, even on a small fraction of values. A one-way function is strong if one needs exponential size circuits for inverting the function for a subexponential fraction of all values. Prominent conjectured one-way functions are the multiplication of two natural numbers bigger than one, and the exponentiation in finite fields (the functions are known as factoring and discrete logarithm, the names for the inverses).

[3]Nevertheless, we should watch out. The advent of quantum algorithms brought one big surprise: Both factoring and discrete logarithm are actually feasible for such machines, cf. [17]. In fact, it is possible that classical physics also allows some computations unfeasible on Turing machines, see [20] and the papers cited there. So far the quantum machines had no effect on the hardness of SAT.

**Theorem 3.3 ([7])** *Assume that a strong one-way function exists. Then strong PRNGs exist.*

In particular, we have a reduction of the existence of sufficiently strong PRNGs, and hence of the possibility of a secure cryptography, to the following concrete hardness assumption:

- Factoring is intractable by subexponential size boolean circuits. In particular, subexponential size circuits can factor only an exponentially small fraction of integers that are products of two primes of comparable sizes.

Note that none of the three theorems proves a lower bound. In fact, there are no lower bounds in complexity theory that could be reasonably branded as a step towards proving that P and NP are different (or towards a proof of some of the other three conjectures).

## 4 A particular challenge

We saw in the last section that three out of the four conjectures can be deduced from a hardness assumption of the following general form:

- Every boolean circuit performing a particular, explicitly given, computational task has to be large.

It is an interesting challenge to try to reduce the NP $\neq$ coNP conjecture to a hypothesis of the same form. It may not be possible but I see no a priori reason ruling out such a reduction.

The study of the NP/coNP problem goes under the name (propositional) proof complexity. The reason is that non-deterministic acceptors of the set of propositional tautologies TAUT can be seen as general propositional proof systems. More precisely, a proof system (tacitly propositional) is a polynomial time computable function $P : \{0,1\}^* \rightarrow \{0,1\}^*$ such that $Rng(P) = \text{TAUT}$. Any $x \in P^{-1}(y)$ is called a $P$-proof of $y$. If $Q$ is any of the ordinary propositional calculi we may embed it into this general notion as follows. Define a function $P_Q(x)$ to be $y$ if the string $x$ is a $Q$-proof of the formula $y$, and put $P_Q(x) := 1$ otherwise. Then $Rng(P_Q) = \text{TAUT}$ follows from the soundness and from the completeness of $Q$. The polynomial time computability of $P_Q$ follows from the fact that $Q$ is given by schematic rules for axioms and inferences. See [5] or [9] for details.

In this terminology, NP $\neq$ coNP iff there are no proof system $P$ and a polynomial $p(n)$ such that any $y \in$ TAUT has a $P$-proof of size at most $p(|y|)$. So NP $\neq$ coNP would follow if we could demonstrate super-polynomial lower bounds to the lengths of proofs in all proof systems. Such lower bounds are known for a few weak systems but not even for the ordinary Hilbert-style calculus based on modus ponens and a finite set of axiom schemes (a Frege system in the terminology of [5]).

A recent attempt to reduce the NP $\neq$ coNP conjecture to a hardness assumption can be found in [10, 1, 11, 15, 12, 16][4]. But it is too early to speculate about an eventual outcome of this project. Instead, I shall describe a theorem linking the NP/coNP problem with an eminently logical topic: consistency statements.

Let me briefly recall a terminology (see [9, Chpt.14] for details). A proof system $P$ simulates proof system $Q$ iff there is a polynomial $p(n)$ such that if $Q(x) = y$ then $\exists z, |z| \leq p(|x|) \wedge P(z) = y$. That is, $P$-proofs are at most polynomially longer than $Q$-proofs. A proof system simulating all other proof systems (if it exists) is called an optimal proof system.

A $P$-hard sequence of tautologies is a sequences $\{\tau_n\}_n$ from TAUT such that $|\tau_n| \geq n$ and $\tau_n$'s are computed by a polynomial time function from strings $1^{(n)}$, and such that there is no polynomial bound on the shortest $P$-proofs of the formulas.

We shall also talk about ordinary first order theories (like Peano arithmetic or set theory). We shall automatically assume that a language of any theory contains the language of arithmetic and that the axioms of the theory form a polynomial time decidable set (this is true about all theories axiomatised by axiom schemes). Further we shall assume that the theories contain some non-trivial part of Peano arithmetic (which I won't specify) and that they are consistent. Let $Con_T(x)$ be a formula[5] in the language of arithmetic expressing that there is no $T$-proof of contradiction of length at most $x$. Using the dyadic numeral $\tilde{n}$ for $n$, $Con_T(\tilde{n})$ is a sentence of length $O(\log n)$ speaking about $T$-proofs of length up to $n$, i.e. coded up to $2^n$.

More details on these definitions can be found in [13] or [9, Chpt.14]. The following theorem is the only theorem in proof complexity giving some non-trivial universal information (as opposed to lower bounds for specific proof systems).

**Theorem 4.1 ([13])** *The following three statements are equivalent:*

[4]Keyword: $\tau$-formulas. An overview can be found in the introduction to [12].

[5]This formula really depends on the axiomatisation and not just on the theory, but I shall omit this here.

1. *There is an optimal proof system.*
2. *There exists a theory*[6] *$S$ such that for any theory $T$ there exists a polynomial $p(n)$ such that all formulas $Con_T(\tilde{n})$ have $S$-proofs of size bounded above by $p(n)$.*
3. *There is a proof system $P$ that does not allow a sequence of $P$-hard tautologies.*

*Further, a failure of any of the statements implies that NP $\neq$ coNP.*

Hence, in particular, it is enough to find a hardness assumption refuting any of the three statements.

## 5 Concluding remarks

A debate occasionally starts whether complexity theory is a "failure" (no real progress on the fundamental problems) or an "extreme success" (as some colleagues boost, counting boldly in 40 years of developments in mathematical logic). I think that it is neither; that it is simply "extremely interesting". The interest comes from facing fundamental scientific problems, not from any potential technological applications (these always come second, as history of physics - electricity and quantum mechanics, in particular - amply show). As S. Smale [19] nicely puts it: The P/NP problem is a gift (of TCS) to mathematicians.

As long as we manage to pass to our students the excitement from the problems and we explain to them that such a gift is rare in science and that they are privileged to study in these exciting times, so long the future of the field will be, I believe, fine. The problems are apparently difficult but a determined graduate student can still learn all relevant developments during the course of his or her PhD. studies. We do not know what the qualification "difficult" really means. Continuum Hypothesis has been difficult but required at the end (after the development of a couple of basic constructions in mathematical logic during the first half of the 20th century) only one really new idea. Fermat's Last Theorem has been also difficult and required several hundred years of a development of a whole part of mathematics.

**Acknowledgements:** I thank S. A. Cook, R. Impagliazzo, P. Pudlák and J. Sgall for comments on the draft of this note.

[6]With the provisions as described earlier.

## References

[1] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson, Pseudorandom generators in propositional proof complexity, Ext. abstract in: *Proc. of the $41^{st}$ Annual Symp. on Foundation of Computer Science*, (2000), pp.43-53.

[2] 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.

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

[4] —— , The P versus NP problem, at: `http://www.claymath.org/prizeproblems/pvsnp.htm`

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

[6] O. Goldreich, *Foundations of Cryptography*, Vol.1, Cambridge University Press, (2001).

[7] J. Hastad, R. Impagliazzo, L. Levin, and M. Luby, A Pseudorandom Generator from any one-way function, *SIAM Journal on Computing*, **Vol 28**, (1999), pp 1364–1396.

[8] R. Impagliazzo, and A. Wigderson, P = BPP unless E has subexponential circuits: derandomizing the XOR lemma, in: *Proc. of the $29^{th}$ Annual ACM Symposium on Theory of Computing*, (1997), pp. 220-229.

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

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

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

[12] ——, Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds, *J. of Symbolic Logic*, to app. ( preprint November 2002).

[13] 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

[14] L. Levin, Universal Search Problems (in Russian), *Problems of Information Transmission*, **9(3)**, (1973), pp.265-266.

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

[16] ——, Pseudorandom generators hard for $k$-DNF resolution and polynomial calculus resolution, preprint, December'02.

[17] P. W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in: *Proc. 35nd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, (1994), pp.124-134.

[18] M. Sipser, The history and status of the P versus NP question, in: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, (1992), pp.603-618. ACM Press.

[19] S. Smale, Mathematical Problems for the Next Century, *Math. Intelligencer*, **20(2)**, (1998), pp.7-15.

[20] A. C.-C. Yao, Classical Physics and the Church-Turing Thesis, ECCC: `http://www.eccc.uni-trier.de/eccc-local/Lists/TR-2002.html` TR02-062.

**Mailing address:**

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