Electronic Colloquium on Computational Complexity, Report No. 53 (2021)

# Information in propositional proofs and algorithmic proof search

Jan Krajíček

Faculty of Mathematics and Physics
Charles University*

**Abstract**

We study from the proof complexity perspective the (informal) proof search problem (cf. [16, Secs.1.5 and 21.5]):

- *Is there an optimal way to search for propositional proofs?*

We note that for any fixed proof system there exists a time-optimal proof search algorithm. Using classical proof complexity results about reflection principles we prove that a time-optimal proof search algorithm exists w.r.t. all proof systems iff a p-optimal proof system exists.

To characterize precisely the time proof search algorithms need for individual formulas we introduce a new proof complexity measure based on algorithmic information concepts. In particular, to a proof system $P$ we attach **information-efficiency function** $i_P(\tau)$ assigning to a tautology a natural number, and we show that:

- $i_P(\tau)$ characterizes time any $P$-proof search algorithm has to use on $\tau$ and that for a fixed $P$ there is such an information-optimal algorithm,
- a proof system is information-efficiency optimal iff it is p-optimal,
- for non-automatizable systems $P$ there are formulas $\tau$ with short proofs but having large information measure $i_P(\tau)$.

We isolate and motivate the problem to establish *unconditional* super-logarithmic lower bounds for $i_P(\tau)$ where no super-polynomial size lower bounds are known. We also point out connections of the new measure with some topics in proof complexity other than proof search.

## 1 Introduction

The central notion of proof complexity is that of a **propositional proof system** as defined by Cook and Reckhow [7]: a p-time function

$$P \ : \ \{0,1\}^* \to \{0,1\}^*$$

*Sokolovská 83, Prague, 186 75, The Czech Republic, krajicek@karlin.mff.cuni.cz

ISSN 1433-8092

whose range is exactly the set of propositional tautologies TAUT; for the definiteness we take all tautologies in the DeMorgan language. Any $w \in \{0,1\}^*$ such that $P(w) = \tau$ is called a ***P*-proof of $\tau$**.

The primary concern is the size of proofs (i.e. their bit-length) and, in particular, the existence of short proofs. The efficiency of a proof system $P$ is measured by the growth rate of **the length-of-proof function**:

$$s_P(\tau) := \min\{|w| \mid \ w \text{ is a } P\text{-proof of } \tau \ \} \ .$$

(We are interested in values of these functions on TAUT only and may, for the definiteness, define it to be equal to 0 or $\infty$ outside TAUT.)

**The fundamental problem** of proof complexity theory asks if this function is, for some $P$, bounded by $|\tau|^{O(1)}$, for all $\tau \in$ TAUT. This is equivalent to the question whether the computational complexity class NP is closed under complementation: **NP** $=_?$ **coNP**, cf. Cook and Reckhow [7].

The second principal open problem of proof complexity is **the optimality problem**: Is there a proof system $P$ such that $s_P$ has at most polynomial slow-down over any $s_Q$? If we define a quasi-ordering $P \geq Q$ on the set of all proof systems by

$$s_P(\tau) \leq s_Q(\tau)^{O(1)}$$

then the problem asks if there is a $\geq$-maximal proof system. Such a maximal proof system $P$ would be lengths-of-proofs optimal. The quasi-ordering $\geq$ has a finer version $\geq_p$: $P \geq_p Q$ iff there is a p-time function $f$ (called **p-simulation**) such that for all $w$:

$$P(f(w)) = Q(w) \ .$$

In words: $f$ translates $Q$-proofs into $P$-proofs of the same formulas. The reader can find this basic background in [16, Chpt.1].

While the existence of short proofs of tautologies is the primary concern of proof complexity, the theory also relates quite closely to the complexity of proof search and SAT algorithms. For proof search algorithms this is obvious: the time complexity of an algorithm searching for $P$-proofs is lower-bounded by function $s_P$.

For SAT algorithms (i.e. algorithms finding a satisfying assignment for a propositional formula, if it exists) the relation is indirect. In particular, we can interpret the run of a SAT algorithm $S$ that fails to find a satisfying assignment for $\neg\tau$ as a *proof* that $\tau \in$ TAUT. Hence $S$ can be studied also as a proof system $P_S$:

$$P_S(w) = \tau \ \text{ iff } \ (\textit{w is the failing computation of S on } \neg\tau)$$

and the time complexity of $S$ on unsatisfiable formulas is essentially the same as the length-of-proof function for proof system[1] $P_S$. Hence lower bounds to the latter function imply lower bounds for the time complexity of $S$. In fact, proof complexity lower bounds apply more generally: a lower bound for $s_Q$ implies

[1] Note that algorithm $S$ is, in particular, also an algorithm searching for $P_S$-proofs.

time lower bounds for all SAT algorithms whose soundness is efficiently provable in $Q$, cf. [14].

While $P_S$ is defined more abstractly than usual logical calculi, the proof system is actually often equal (or close) - in the sense of p-simulation - to some standard logical calculi as is, for example, **resolution R**. This then allows to interpret various technical proof complexity results as results about the original algorithm $S$. In this sense proof complexity contributes to the analysis of some classes of SAT algorithms. This facet of proof complexity is surveyed by Buss and Nordström [3].

In this paper we are interested in efficiency of proof search algorithms too. However, rather than analyzing particular algorithms we consider an outstanding informal problem[2]:

- *Is there an optimal way to search for propositional proofs?*

Surely this problem must have occurred to everybody interested in proof search, and there are other natural informal questions one can ask (cf. [17] for examples).

In this paper we investigate what can proof complexity say about the problem in precise mathematical terms. We start with a definition of a proof search algorithm that seems natural (cf. [16, Sec.21.5]).

**Definition 1.1** *A* **proof search algorithm** *is a pair $(A, P)$, where $P$ is a proof system and $A$ is a deterministic algorithm that stops on every input*[3] *and such that $A(\tau)$ is a $P$-proof of $\tau$, for all tautologies $\tau \in TAUT$.*

A key to a formalization of the proof search problem is to define a suitable quasi-ordering on the class of all proof search algorithms. In Section 2 we consider a quasi-ordering by the time complexity and in Section 3 we resort to algorithmic information theory and replace time with information, introducing a new notion of information-efficiency of proof systems. This notion offers a precise characterization of the time any algorithm searching for a proof of a particular formula must use.

In both quasi-orderings (by time or information efficiency) there are optimal proof search algorithms when the proof system is fixed, and these two algorithms are essentially the same. Hence the question whether there is an overall optimal proof search algorithm $(A, P)$ (a maximal element in the respective quasi-ordering) depends only on proof systems $P$ and not on algorithms $A$. We show that in both quasi-orderings the existence of such an optimal system is equivalent to the existence of a p-optimal proof system, and thus the proof search problem reduces to the optimality problem.

Time a proof search algorithm needs to use is traditionally lower bounded by the minimum size of any proof of the formula in question. In Section 4 we

[2] Which we included as a third basic problem of proof complexity under the name **the proof search problem** in [16, Secs.1.5 and 21.5]

[3] See the second paragraph of Sec.2 for this condition.

compare size (measure) with information (measure) and we note that for non-automatizable proof systems the information measure is more precise for proving time lower bounds than proof size is: there are formulas having short proofs but having large information measure (i.e. while the proofs are short to find them requires long time). Note that it is known that essentially all complete proof systems are non-automatizable under various plausible computational complexity hypotheses. In Section 5 we motivate and isolate the problem to establish *unconditional* lower bounds for $i_P(\tau)$ where no lower bounds are known for $s_P(\tau)$. We conclude the paper with remarks on further connections of the information measure to proof complexity in Section 6 and some general comments in Section 7.

The reader can find basic proof complexity background in [16, Chpt.1]. We use only classic facts and we always point to a place in [16] where they can be found. From algorithmic information theory we use only the original ideas and notions from Kolmogorov [10, 11] modified to a time-bounded version of Levin [22]. Standard notions from computational complexity (as are classes P, NP, one-way permutations or pseudo-random generators) can be found in any textbook.

## 2 Time optimality

The first thing that comes to mind is perhaps to compare two proof search algorithms by the time they use. This is analogous to the fact that in the optimality problem we compare two proof systems by the growth rate of their lengths-of-proofs functions, i.e. by the non-deterministic time. For a deterministic algorithm $A$ that stops on all inputs we denote by $time_A(w)$ the time $A$ needs to stop on input $w$.

We shall assume that proof search algorithms stop on every input, not just on inputs from TAUT. Namely, if $A$ is an algorithm that stops on TAUT but maybe not everywhere outside TAUT, define new algorithm $A'$ that in even steps computes as $A$, and stops if $A$ does, and in odd steps performs an exhaustive search for a falsifying assignment and stops if it finds one before $A$ stopped. The time complexity of $A$ and $A'$ on inputs from TAUT are proportional and $A'$ stops everywhere.

Note that in the following definition the two proof search algorithms do not necessarily use the same proof system.

**Definition 2.1** *For two proof search algorithms $(A, P)$ and $(B, Q)$ define*

$$(A, P) \geq_t (B, Q)$$

*iff $(A, P)$ has at most polynomial slow-down over $(B, Q)$:*

$$time_A(\tau) \ \leq \ time_B(\tau)^{O(1)} \ .$$

*for all $\tau \in TAUT$.*

**Lemma 2.2** *For any fixed proof system $P$ there is algorithm $A$ such that $(A, P)$ is $\geq_t$-maximal among all $(B, P)$, i.e. it is* **time-optimal** *among all $(B, P)$.*

**Proof :**

This is proved analogously as the existence of a universal NP search algorithm (cf. Levin [21]): given input $\tau$, A tries for $i = 1, 2, \ldots$ lexicographically first $i$ algorithms for $i$ steps until it finds a $P$-proof of $\tau$.

We may assume w.l.o.g. that the size of the $i$-th algorithm is $O(\log i)$ and that $A$ simulates its $t$ steps in time $O((t + \log i)^2)$. Hence for a fixed $B$ that is $j$-th in the ordering then

$$time_A(\tau) \leq O(time_B(\tau)^2) \ .$$

**q.e.d.**

**Notation:** *Let $(A_P, P)$ denote the proof search algorithm described in the above proof. Hence $(A_P, P)$ is time-optimal among all $(B, P)$.*

The optimality problem (both its versions for $\geq$ and $\geq_p$) relates to a number of questions in a surprisingly varied areas and there are quite a few relevant statements known cf. [16, Chpt.21]). We shall recall just one statement that we will use in the second proof of Theorem 2.4.

**Theorem 2.3 (K. and Pudlák [18, Thm.2.1])**

*A p-optimal proof system exists iff there is a deterministic algorithm $M$ computing the characteristic function $\chi_{TAUT}$ of $TAUT$ such that for any other deterministic algorithm $M'$ computing $\chi_{TAUT}$ it holds that:*

$$time_M(\tau) \leq time_{M'}(\tau)^{O(1)} \ , \quad \textit{for all } \tau \in TAUT \ .$$

Now we shall relate the existence of p-optimal proof systems and time-optimal proof search algorithms. We give two proofs as they illustrate different facets of the statement.

**Theorem 2.4** *Let $P$ be any proof system containing resolution $R$ and having the property that for some $c \geq 1$, for every $\tau$ and every $\tau'$ obtained from $\tau$ by substituting constants for some atoms it holds $s_P(\tau') \leq s_P(\tau)^c$.*

*Then $P$ is p-optimal iff $(A_P, P)$ is time-optimal among all proof search algorithms $(B, Q)$.*

*In particular, a p-optimal proof system exists iff a time-optimal proof search algorithm (i.e. $\geq_t$-maximal) exists.*

**First proof :**

The only-if-direction is obvious, using Lemma 2.2. For the non-trivial if-direction of the theorem we use the fact that for any $Q$ there is a *p-time construable sequence* of tautologies

$$\langle Ref_Q \rangle_n \ , \ n \geq 1$$

such that $n \leq |\langle Ref_Q \rangle_n|$ and if $P$-proofs of these formulas are p-time computable then $P$ p-simulates $Q$. These formulas formalize the soundness of $Q$ and their exact definition is not important here. Their relation to (p-)simulations is a classic fact of proof complexity going back to Cook [5]; see [16, Secs.19.2 or 21.1] for this background.

Define a proof system $Q'$ in which $1^{(n)}$ is a proof of $\langle Ref_Q \rangle_n$ and any other $w$ is interpreted as a resolution proof. Further take algorithm $B$ which upon receiving $\tau$ first looks whether $\tau = \langle Ref_Q \rangle_n$ for some $n$ (a priori $\leq |\tau|$) in which case it produces $1^{(n)}$, and otherwise it uses some fixed resolution searching algorithm to find a proof.

Because $(A_P, P)$ is supposed to be time optimal, $A(\langle Ref_Q \rangle_n)$ has to compute in p-time a $P$-proof of $\langle Ref_Q \rangle_n$. But by the stated properties of these formulas $P \geq_p Q$ follows.

**Second proof :**

We now give a second, alternative proof for the last statement of the theorem, using Theorem 2.3. For a proof search algorithm $(A, P)$ define algorithm $M_{(A,P)}$ computing $\chi_{TAUT}$ as follows: On input $\tau$ it computes $A(\tau)$ and checks that $P(A(\tau)) = \tau$. If so, it outputs 1, otherwise it outputs 0.

On the other hand, if $M$ computes $\chi_{TAUT}$ define proof system $P_M$ by

$$P_M(w) = \tau \ \text{ iff } \ (\textit{w is the computation of } M \textit{ on } \tau \textit{ and it outputs } 1)$$

and algorithm $A_M$: On input $\tau$ output the computation of $M$ and $\tau$.

It is easy to verify that:

- if $(A, P)$ is a time-optimal proof search algorithm then $M_{(A,P)}$ is a deterministic algorithm computing $\chi_{TAUT}$ having the time-optimality property from Theorem 2.3, and
- if $M$ is a deterministic algorithm computing $\chi_{TAUT}$ having the time-optimality property from Theorem 2.3 then $(A_M, P_M)$ is time-optimal proof search algorithm.

Theorem 2.3 then implies the statement.

**q.e.d.**

If $P \geq_p Q$ then in any reasonable quasi-ordering of proof search algorithms $(A_p, P)$ will be at least as strong as $(A_Q, Q)$. For the opposite direction (the if-direction) of the theorem we utilized the fact that $(A_P, P)$ is required to find in p-time proofs of simple sequences of formulas as are $\langle Ref_Q \rangle_n$.

A simple sequence of formulas appears also the following situation. Take any proof search algorithm $(A, \mathrm{R})$ searching for resolution proofs. Take a sequence of tautologies that are computed by a p-time function from $1^{(n)}$ and that are hard for R but easy for **Extended resolution ER** and, moreover, their ER-proofs can be computed from $1^{(n)}$ in p-time by some function $f$. Examples of such formulas are formulas $PHP_n$ formalizing the pigeonhole principle, cf. Haken [9] and Cook and Reckhow [7] (or see [16]). Now define a proof search algorithm $(B, \mathrm{ER})$ that on input $\tau$ computes as follows:

1. $B$ checks if $\tau = PHP_n$ for some $n \geq 1$ (this is p-time because it needs to consider only $n \leq |\tau|$).
2. If yes, i.e. $\tau = PHP_n$, then $B$ outputs $f(1^{(n)})$.
3. Otherwise $B$ outputs $A(\tau)$.

Then $(B, ER) >_t (A, R)$ but intuitively it does not seem quite right to claim that $(B, ER)$ is a better algorithm than $(A, R)$; $B$ does not do anything extra except that it remembers one type of simple formulas. One would like to

(*) compare $A$ and $B$ on inputs $\tau$ where *they actually do something non-trivial.*

In [16, Sec.21.5] we proposed a definition of a quasi-ordering of proof search algorithms by time as is $\geq_t$ but measured only on TAUT from which we are allowed to take out, in particular, a p-time construable sequence of tautologies. Subsequently in [17] a stronger variant of that (avoiding all such sequences) was proposed. This could, in principle, allow for the situation that there is an optimal proof search algorithm without having a p-optimal proof system, and thus separate the two questions. However, the resulting quasi-orderings are unintuitive and it is not clear whether they actually help to avoid the if-direction of Theorem 2.4.

A more fundamental issue, related also to (*) above, is that the decision not to count (or not count) $\tau = PHP_n$ when comparing two proof search algorithms is not based only on the individual tautology $\tau$ but depends on the fact that it is one of an infinite series of tautologies defined in a particular uniform way.

These considerations are, of course, quite informal but lead us to notions discussed in the next section.

# 3 Information optimality

We shall assume that every $e \in \{0,1\}^*$ is also a code of a unique Turing machine and we shall consider a universal Turing machine $U$ with three inputs $e, u, 1^{(t)}$ that simulates machine $e$ on input $u$ for at most $t$ steps, stops with the same output if $e$ stops in $\leq t$ steps, and otherwise outputs 0. We shall assume that $U$ runs in polynomial time.

Using this set-up recall **the time-bounded Kolmogorov complexity** of a string $w \in \{0,1\}^*$ as defined by Levin [21]:

$$Kt(w|u) \ := \ \min\{(|e| + \log t) \mid U(e, u, 1^{(t)}) = w\}$$

and

$$Kt(w) \ := Kt(w|0) \ .$$

Intuitively, smaller $Kt(w)$ is simpler $w$ is, in the sense that it can be compressed to a shorter string without loosing information.

Note that we have trivial estimates to $Kt(w|u)$ and $Kt(w)$ in terms of the size $|w|$:

$$\log(|w|) \leq Kt(w|u) \leq Kt(w) \leq |w| + \log(|w|) + O(1) \ . \tag{1}$$

The left inequality holds as need need time $|w|$ to write $w$, the middle one is trivial and the right inequality follows from considering a machine that has $w$ hardwired into its program.

We would like to have inequality $Kt(w) \leq Kt(w|u) + Kt(u)$ that is intuitively justified by composing machine $e_1$ computing $u$ with machine $e_2$ computing $w$ from $u$. However, as pointed out in Kolmogorov [11], the code of the composed machine (and, in general, of the pair $(e_1, e_2)$) does not have length $|e_1| + |e_2|$ but rather can be defined of length $|e_1| + |e_2| + O(\log(|e_1|) + \log(|e_2|))$. Hence we get a slightly worse inequality:

$$Kt(w) \leq Kt(w|u) + Kt(u) + O(\log(Kt(w|u)) + \log(Kt(u))) \tag{2}$$

and similarly

$$Kt(w|u) \leq Kt(w|v) + Kt(v|u) + O(\log(Kt(w|v)) + \log(Kt(v|u))) \ . \tag{3}$$

Now we use $Kt$ to define a new measure of complexity of proofs.

**Definition 3.1** *Let $P$ be a proof system. For any $\tau \in TAUT$ define*

$$i_P(\tau) \ := \ \min\{Kt(w|\tau) \mid P(w) = \tau\} \ .$$

*We shall call $i_P$* **the information efficiency function**.

The function measures the minimal amount of information any $P$-proof of $\tau$ has to contain, *knowing what $\tau$ is.* The next statement shows that stronger proof system do not require much more information.

**Lemma 3.2** *For any $P, Q$, $P \geq_p Q$ implies $i_P(\tau) \leq O(i_Q(\tau))$.*

**Proof :**

Let $f$ be a p-simulation of $Q$ by $P$. Take $w$ that is a $Q$-proof of $\tau$ with $Kt(w|\tau) = i_Q(\tau)$.

Using (3) we can estimate $i_P(\tau) \leq Kt(f(w)|\tau)$ from above by the sum $Kt(f(w)|w) + Kt(w|\tau) = Kt(f(w)|w) + i_Q(\tau)$ plus some log-small terms. But $Kt(f(w)|w) \leq O(\log|w|) + O(1)$ which is also bounded by $O(i_Q(\tau))$ by (1).

**q.e.d.**

The next two statements relate the information measure fairly precisely to time in proof search.

**Lemma 3.3** *Let $(A, P)$ be any proof search algorithm. Then for all $\tau \in TAUT$:*

$$i_P(\tau) \leq Kt(A(\tau)|\tau) \leq |A| + \log(time_A(\tau)) \ .$$

*In particular, $time_A(\tau) \geq \Omega(2^{i_P(\tau)})$.*

**Proof :**

The first inequality is obvious, the second follows from the definition as $A(\tau) = U(A, \tau, 1^{(t)})$, where $t = time_A(\tau)$.

**q.e.d.**

This statement is complemented by the next one essentially saying that *easy proofs are easy to find*[4].

**Lemma 3.4 (i-automatizability)**

*For every proof system $P$ there is an algorithm $B$ such that for all $\tau \in TAUT$:*

$$Kt(B(\tau)|\tau) = i_P(\tau)$$

*and*

$$time_B(\tau) \leq 2^{O(i_P(\tau))} \ .$$

**Proof :**

For $i = 1, 2, \ldots$ algorithm $B$ (using the universal machine $U$) does the following:

- *In the lexico-graphic order tries all pairs $(e, t)$ such that $|e| + \log t = i$ and checks whether $U(e, \tau, 1^{(t)})$ is a $P$-proof of $\tau$. If so, it outputs the proof and $B$ stops.*

There are $\leq (i + 1)2^i$ such pairs $(e, t)$ to consider, computing $U(e, \tau, 1^{(t)})$ takes time $poly(|e|, t) \leq 2^{O(i)}$ and checking whether $P(U(e, \tau, 1^{(t)})) = \tau$ takes time $poly(|U(e, \tau, 1^{(t)})|) \leq 2^{O(i)}$. The procedure takes for one $i$ overall time $2^{O(i)}$ and because $B$ succeeds in the round for $i = i_P(\tau)$, the overall time $B$ takes is $\leq 2^{O(i_P(\tau))}$.

**q.e.d.**

**Notation:** *Let us denote the algorithm described in the proof by $B_P$.*

In fact, the argument in the proof of Lemma 3.4 is another version of the universal search as the next statement shows.

[4] In this sense it establishes automatizability of all proof systems w.r.t. information efficiency as oppose to the original automatizability relating to lengths-of-proofs, cf. Sec. 4 or [16, Sec.17.3].

**Corollary 3.5** *Let $P$ be any proof system and let $A_P$ and $B_P$ be the two algorithms defined earlier. Then*

$$(A_P, P) \geq_t (B_P, P) \geq_t (A_P, P) \ .$$

**Proof :**

The first inequality follows from Lemma 2.2, and the second from Lemmas 3.3 and 3.4.

**q.e.d.**

Because of the algorithm $B_P$ achieves the optimal information efficiency it seems natural to define a quasi ordering of proof systems based on comparing their information-efficiency functions.

**Definition 3.6** *For two proof systems $P$ and $Q$ define:*

$$P \geq_i Q \quad \textit{iff} \quad i_P(\tau) \leq O(i_Q(\tau))$$

*for all $\tau \in TAUT$.*

This is a quasi ordering of proof systems that is, by Lemma 3.2, coarser that $\geq_p$ but, presumably, different than both $\geq_p$ and $\geq$. But as far as optimality goes it does not allow for a new notion.

**Theorem 3.7** *Let $P$ be any proof system containing resolution $R$ and having the property that for some $c \geq 1$, for every $\tau$ and every $\tau'$ obtained from $\tau$ by substituting constants for some atoms it holds $s_P(\tau') \leq s_P(\tau)^c$.*

*Then $P$ is information-optimal (i.e. $\geq_i$-maximal) iff it is p-optimal.*

**Proof :**

Let $P$ be a p-optimal proof system and let $Q$ be any proof system. Assume $f$ is a p-simulation of $Q$ by $P$.

Let $\tau \in TAUT$ and assume $Kt(w|\tau) = i_Q(\tau)$ for some $Q$-proof $w$ of $\tau$. Then $f(w)$ is a $P$-proof of $\tau$ and $Kt(f(w)|w) \leq O(1) + O(\log |w|)$: the $O(1)$ is for the machine computing $f$ and the computation runs in time polynomial in $|w|$. But $|w| \leq 2^{i_Q(\tau)}$, so $Kt(f(w)|w) \leq O(i_Q(\tau))$ and $Kt(f(w)|\tau) \leq O(i_Q(\tau))$ follows by (3). Hence

$$i_P(\tau) \leq O(i_Q(\tau)) \ , \quad \text{all} \quad \tau \in TAUT \ .$$

For the only-if-direction assume that $P$ is an information-optimal proof system and $Q$ is an arbitrary proof system. Take the sequence $\langle Ref_Q \rangle_n$, $n \geq 1$, as in the first proof of Theorem 2.4, and interpret strings $1^{(n)}$ as proofs of these formulas in some proof system $Q'$. We see that

$$i_{Q'}(\langle Ref_Q \rangle_n) \leq O(\log n) \ .$$

By the information optimality of $P$ also

$$i_P(\langle Ref_Q \rangle_n) \leq O(\log n)$$

which, by Lemma 3.4, means that the algorithm $B_P$ finds $P$-proofs of formulas $\langle Ref_Q \rangle_n$ in time $n^{O(1)}$. This implies, as in the first proof of Theorem 2.4, that $P \geq_p Q$.

**q.e.d.**

Theorem 3.7 implies that the information measure approach does not lead to a separation of the proof search problem from the optimality problem either.

# 4 Information vs. size

A natural question is whether the information-efficiency function may give, at least in principle, better time lower bounds for proof search algorithms than the length-of-proof function. By Lemma 3.3 information gives super-polynomially better time lower bound than size if $i_P(\tau)$ cannot be in general bounded above by $O(\log s_P(\tau))$.

Recall a notion introduced by Bonet, Pitassi and Raz [2]: a proof system $P$ is **automatizable** iff there is a proof search algorithm $(A, P)$ such that for all $\tau \in$ TAUT:

$$time_A(\tau) \leq s_P(\tau)^{O(1)} \ .$$

Considering that there are no known non-trivial complete automatizable proof systems this author saw as the only use of the notion that it gives a nice meaning to the failure of feasible interpolation, cf. [16, Sec.17.3]. But now it is exactly what we need to characterize the separation of size from information, using Lemma 3.3.

**Theorem 4.1** *A proof system $P$ is non-automatizable iff there is an infinite set $X$ of tautologies $\tau$ of unbounded size such that*

$$i_P(\tau) \geq \omega(\log s_P(\tau)) \tag{4}$$

*on $X$.*

To illustrate what type of formulas witness the separation of size from information we shall paraphrase the construction from [19]; there it was done for $P = \text{ER}$ and $h := \text{RSA}$.

Let $h : \{0,1\}^* \to \{0,1\}^*$ be p-time permutation of each $\{0,1\}^n$, i.e. it is a length-preserving and injective function, and let $h_n$ be the restriction of $h$ to $\{0,1\}^n$. For any $b \in \{0,1\}^n$ define formula

$$\mu_b \ := \ [h_n(x) = b \to B(x) = B(h^{(-1)}(b))]$$

where $B(x)$ is a hard-bit of permutation $h$; the statement $h_n(x) = b$ is expressed by a p-size circuit (if $P$ allows them), or using auxiliari variables whose values are uniquely determined by values of $x_1, \ldots, x_n$. Note that $|\mu_b| \leq n^{O(1)}$.

**Lemma 4.2** *Assume that $P$ proves by p-size proofs that $h_n$ are injective, i.e. it proves tautologies expressing*

$$h_n(x) = h_n(y) \to \bigwedge_{i \le n} x_i \equiv y_i \ .$$

*Assume that $h$ is a one-way permutation and $B$ is its hard bit predicate.*

*Then there are $P$-proofs $\pi_b$ of formulas $\mu_b$ such that:*

1. $|\pi_b| \le n^{O(1)}$*, i.e.* $s_P(\mu_b) \ \le \ n^{O(1)}$*,*
2. *for a random $b \in \{0,1\}^n$, with a probability going to 1 as $n \to \infty$, it holds that*

$$i_P(\mu_b) \ge \omega(\log n) \ . \tag{5}$$

*If $h$ is secure even against algorithms running in time $2^{n^\epsilon}$, for some $\epsilon > 0$, then the right-hand term in (5) can be improved to $n^{\Omega(1)}$.*

**Proof :**

Define the wanted $P$-proof $\pi_b$ as follows. Pick $a \in \{0,1\}^n$ such that $h(a) = b$ and prove in $P$, using the injectivity of $h_n$, that

$$h_n(x) = b \to x = a \ .$$

Using this derive $\mu_b$ from $B(a) = B(a)$, and finally prove the true sentence $B(a) = B(a)$ by evaluating it. This proves the first statement.

The second statement follows from the hypothesis that $h$ is one-way: we can try the algorithm $B_P$ from Section 3 on formulas

$$[h_n(x) = b \to B(x) = c]$$

for $c = 0, 1$ and compute in this way the hard bit in p-time. But that is impossible if $B$ is indeed a hard bit of $h$.

**q.e.d.**

Related alternative formulas can be defined as follows. Let $\varphi_n(x)$, $n \ge 1$ and $x = (x_1, \dots, x_n)$, be a sequence of formulas that have p-size $|\varphi_n| \le n^{O(1)}$ but that do not have p-size $P$-proofs:

$$s_P(\varphi_n) \ge n^{\omega(1)} \ , \ n \ge 1 \ .$$

For some proof systems we have such formulas unconditionally, for those which are not p-optimal we can take formulas $\langle Ref_Q \rangle_n$ used earlier, for some $Q >_p P$.

For any $b \in \{0,1\}^n$ define formula

$$\eta_b(x) \ := \ [h_n(x) = b \to \varphi_n(x)] \ .$$

Note that $|\eta_n| \le n^{O(1)}$. Analogously with the proof of the lemma, the formulas have p-size $P$-proofs $\pi_b$ and these particular proofs satisfy $Kt(\pi_b | \eta_b) \ge \omega(\log n)$. It would be interesting if for some $P$ it would hold that any short proof of $\eta_b$ must contain some non-trivial information about $h^{(-1)}(b)$.

# 5 Information alone

A separation of size from information in the sense of (4) implies that no p-time algorithm finds, given $\tau$ and $s_P(\tau)$ in unary, a p-time recognizable (by $P$) object (a $P$-proof), and hence it implies that P $\neq$ NP. In fact, a number of proof systems are known to be non-automatizable assuming various conjectures from complexity theory (cf. [16, Sec.17.3]). We mention just resolution R and its non-automatizability proved under the weakest possible hypothesis that P $\neq$ NP by Atserias and Müller [1]; references for earlier work and other examples can be found there or in [16, Sec.17.3].

Proofs of non-automatizability depend on a p-time reduction of some hard set $Y$ (NP-complete in [1] or hard bit of RSA in [19] or similar, cf. [16, Sec.17.3]) to a set of formulas with p-size proofs that maps the complement $\{0,1\}^* \setminus Y$ to formulas with only long (or none) proofs. These arguments do not yield lower bounds for $i_P(\tau)$ for individual formulas but only speak about the asymptotic behavior of an automatizing algorithm.

We are interested in the question whether one can establish a lower bound for $i_P(\tau)$ by considering formulas individually, not as members of an infinite set or sequence. This would be in a way analogous to lengths-of-proofs lower bounds (e.g. for $PHP_n$ in R in [9]) which work with individual formulas.

A super-polynomial lower bound for $s_P(\tau)$ is used primarily for three purposes:

1. *It implies that no $Q \leq P$ is p-bounded, an instance of NP $\neq$ coNP, and if true for all $P$ then indeed NP $\neq$ coNP follows.*

2. *It implies super-polynomial time lower bounds for a class of SAT algorithms $S$ that are simulated by $P$: $P \geq P_S$ ($P_S$ from the Introduction. Currently known lengths-of-proofs lower bounds imply time lower bounds for large classes of SAT algorithms.*

3. *It implies independence results from a first-order theory attached to $P$ and, in particular, that P $\neq$ NP is consistent with the theory (see [16, Sec.86].*

But having a super-logarithmic lower bound for $i_P(\tau)$ is just as good. Items 2. and 3. hold literally: in the former this is by Lemma 3.3 and for the latter this holds because propositional translations of first-order proofs are performed by p-time algorithms (cf. [16, Part 2]). In item 1 one has to compromise on weakening NP $\neq$ coNP to P $\neq$ NP, a small price to pay.

This motivates the following problem that seems to us to be quite fundamental.

**Problem 5.1** *Establish* unconditional *super-logarithmic lower bound*

$$i_P(\tau) \geq \omega(\log|\tau|)$$

*for a set $X \subseteq TAUT$ of unbounded size, for a proof system $P$ for which no super-polynomial lowers bounds for $s_P(\tau)$ are known.*

As a step towards solving the problem it would be interesting to have such unconditional lower bounds at least for $P$ for which super-polynomial lower bounds for $s_P$ are known, but not for formulas from $X$.

Note the emphasis on the requirement that the lower bound is unconditional. Allowing some unproven computational complexity hypotheses the problem becomes easy. For example, if it were that $i_P(\tau) \leq O(\log |\tau|)$ for all $\tau$ then the algorithm $B_P$ form Section 3 runs in p-time and hence P = NP. Or you may take any pseudo-random number generator $g : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ and for $b \in \{0,1\}^{n+1}$ take a formula[5] $\tau_b$ expressing that $b \notin Rng(g)$. Then $i_P(\tau_b)$ cannot be bounded by $O(\log |\tau_b|)$ as otherwise $B_P$ would break the generator in p-time.

In what follows we shall discuss the existence of formulas $\tau$ whose length we shall denote $m$. The formulas will not be a priori members of some infinite series but are considered *individually.* This means that questions and statements about them do depend just on them and not on asymptotic properties of some ambient sequence. But we still wish to use the handy $O$-, $\Omega$- and $\omega$- notations and in doing so we imagine what happens in each particular construction or statement as $m \rightarrow \infty$.

For the sake of the following discussion let us call a size $m$ formula **simple** if $Kt(\tau) = O(\log m)$ and **complex** otherwise, and we apply similar qualifications to its proofs $\pi$ but still relative to parameter $m$ (i.e. not relative to $|\pi|$).

For example, for the truth-table proof system TT, any tautology $\tau$ in $m^{\Omega(1)}$ variables, simple or complex, will have only a complex truth-table proof $\pi$: its size is exponential in $m^{\Omega(1)}$ and (1) implies that $Kt(\pi) \geq i_{TT}(\tau) \geq m^{\Omega(1)}$ as well.

To solve Problem 5.1 we want a class $X \subseteq$ TAUT of formulas $\tau$, $|\tau| = m \rightarrow \infty$, such that

$$i_P(\tau) \geq \omega(\log m) \ .$$

The following lemma formulas two simple conditions on $X$, one necessary and one sufficient.

**Lemma 5.2** *Let $X \subseteq TAUT$ be a set of formulas of unbounded size.*

1. *(a necessary condition)*

   *For $X$ to solve Problem 5.1 it is necessary that all $P$-proofs $\pi$ of $\tau$ are complex:*

   $$Kt(\pi) \geq \omega(\log m) \ .$$

2. *(a sufficient condition)*

   *If $X$ satisfies item 1 then a sufficient condition for it to solve the problem is that all $\tau$ are simple:*

   $$Kt(\tau) \leq O(\log m) \ .$$

[5] See Subsection 6.2.

**Proof :**

For item 1 note that by (1) we have $i_P(\tau) \leq Kt(\pi|\tau) \leq Kt(\pi)$. For item 2 we have by (2):

$$Kt(\pi) \leq Kt(\pi|\tau) + Kt(\tau) + O(\log(Kt(\pi|\tau)) + \log(Kt(\tau))) \ .$$

By (1) we may estimate the last term by $O(\log m)$, and by the hypothesis $Kt(\tau) \leq O(\log m)$ as well. hence we can write the whole inequality as

$$Kt(\pi) - Kt(\pi|\tau) \leq O(\log m) + O(\log Kt(\pi|\tau)) \ . \tag{6}$$

Now distinguish two cases. Either $\pi \leq m^{O(1)}$ or $\pi \geq m^{\omega(1)}$. In the latter case we are finished as $i_P(\tau)$ is lower bounded by $\log s_P(\tau)$. In the former case we can estimate the last term in (6) by $O(\log m)$ and hence get

$$Kt(\pi) - Kt(\pi|\tau) \leq O(\log m) \ . \tag{7}$$

This implies what we need because, by item 1, $Kt(\pi) \geq \omega(\log m)$.

**q.e.d.**

Note that condition 1 in the lemma is not sufficient. To see this take $\tau$ of the form $\rho \vee \neg\rho$, where $\rho$ is random a hence of high $Kt$-complexity. But $\tau$ is a proof of itself in a suitable Frege system (or even in R if $\rho$ is just a clause and $\neg\rho$ is the set of singleton clauses consisting of negations of literals in $\rho$) and $Kt(\tau|\tau) = \log(|\tau|) + O(1)$ is small.

When $\tau$ are complex then the necessary condition holds automatically: given a $P$-proof $\pi$ of $\tau$, either $|\pi| \geq m^{\omega(1)}$ and hence $\omega(\log m)$ lower bounds $Kt(\pi|\tau)$ by (1), or $|\pi| \leq m^{O(1)}$. In the latter case, because $P(\pi) = \tau$ and using (2):

$$Kt(\tau) \leq Kt(\tau|\pi) + Kt(\pi) + O(\log Kt(\tau|\pi) + \log Kt(\pi))$$

which yields

$$\omega(\log m) \leq O(1) + O(\log|\pi|) + Kt(\pi) + O(\log(O(1) + O(\log|\pi|) + \log Kt(\pi))$$

and estimating $\log|\pi| \leq O(\log m)$ we derive:

$$\omega(\log m) \leq Kt(\pi) \ .$$

On the other hand, the computation in the proof of item 2 does not yield anything for complex formulas. But the quantity being estimated from above in (7) still makes sense and if (7) holds for an $X$ (satisfying item 1) then $X$ solves the problem.

In fact, this quantity has been isolated already by Kolmogorov [10, 11]; following him define (the $Kt$-version of) **information that $u$ conveys about** $w$ as

$$It(u : w) \ := \ Kt(w) - Kt(w|u) \ .$$

Hence what we want is $\tau$ having only complex proofs such that for any proof $\pi$ it holds that:

$$It(\tau : \pi) \text{ is small.}$$

In words: $\tau$ knows very little about its proofs.

Many formulas that appear in various contexts of proof complexity (as formulas $PHP_n$ or $\langle Ref_Q \rangle_n$ we encountered earlier), occur as members in a uniformly constructed sequence $\{\tau_n\}_n$. The sequence is often p-time construable from $1^{(n)}$ or, in fact, have even stricter levels of uniformity (cf. [16, Sec.19.1]). When such formulas have short proofs $\pi_n$ in some proof system $P$ it is often the case that the proofs are also uniformly constructed from $1^{(n)}$. But that forces $I(\tau_n : \pi_n)$ to be high: a p-time algorithm extracts from $\tau$ the parameter $n$ and from it constructs $\pi_n$.

Hence if we wanted to use for $X$ some uniform formulas they ought to be expected to have only long $P$-proofs (but we may not be able to prove that). Leaving the reflection principles aside, two examples that come to mind are

- $AC^0[p]$-Frege systems and the $PHP_n$ formulas, cf. [16, Sec.10.1 and Problem 15.6.1].
- $AC^0$-Frege systems and the $WPHP_n$ formulas, cf. [16, Problem 15.3.2].

Lower bounds for $AC^0$-Frege systems are known but not for formulas $WPHP_n$ expressing a form of the weak PHP.

For stronger systems no uniform candidates for hard formulas are known which would be supported by some results. Possibly hard formulas are those described in Subsection 6.2 and the are expected to be all complex in the sense of $Kt$ complexity.

# 6 Proof complexity remarks

In this section we remark on several topics in proof complexity that seem to be related to the information measure. It may be worthwhile to explore if there are some deeper connections. The section aims primarily at proof complexity readers but we give references to relevant places in [16] to aid non-specialists.

## 6.1 Random formulas

Müller and Tzameret [23] proved that random 3CNFs with $\Omega(n^{1.4})$ clauses do have (with the probability going to 1) polynomial size refutations in a $TC^0$-Frege system. Their argument is based on formalizing in the proof system (via bounded arithmetic) the soundness of the unsatisfiability witnesses proved to exists with a high probability by Feige, Kim and Ofek [8].

Such a formula $\tau$ has bit size $m = O(n^{1.4} \log n)$ (and, by virtue of being random, it has Kt-complexity $\Omega(m)$). Feige, Kim and Ofek [8] proved that their witness (i.e. also the p-size $TC^0$-Frege proof $\pi$ from [23]) can be found in time $2^{O(n^{0.2}\log n)}$ which is exponential in $m^{\Omega(1)}$. That is, we know that

$$i_{TC^0-F}(\tau) \leq m^{\Omega(1)} \ . \tag{8}$$

This leaves open the possibility that this inequality cannot be significantly improved. In that case the formulas would be witness for Theorem 4.1 for $TC^0$-Frege systems demonstrating even exponential gap.

## 6.2 Proof complexity generators

A fairly brief exposition of the theory of proof complexity generators can be found in [16, Secs.19.4 and 19.6] or in older [15, Chpts.29 and 30]. The theory investigates, in particular, functions $g$ extending $n$ bit strings to $m$-bit strings, $m = m(n) > n$, computable in p-time, and such that formulas $\tau(g)_b$, for $b \in \{0,1\}^m \setminus Rng(g)$, ought to be hard to prove in a given proof system. In particular, function $g$ is defined to be **hard for** $P$ iff for any $c \geq 1$ only finitely many formulas $\tau(g)_b$ have a $P$-proof of size $\leq |\tau(g)_b|^c$.

Function $g$ can be thought of as a decompression algorithm and for $w \in Rng(g)$ we have $Kt(w) \leq n + O(\log n) + O(1)$ which is $<< m$ if, for example, $2n < m$. Note that for $w \in \{0,1\}^m$, the condition $Kt(w) \geq m/2$ implies that $w \notin Rng(g)$. The property $Kt(w) \geq m/2$ cannot be expressed by a p-size tautology as the time involved in the computation of the universal machine may be exponential in $m$. But for a fixed p-time $t(n)$ we can consider complexity $K^t$ by restricting the decompression to a universal Turing machine $U^t$ on inputs $e, u$ (i.e.. no time input) simulating $e$ on $u$ for time $t$. By padding (or restricting) all outputs in some canonical way to size $m = m(n)$ exactly, and taking for the domain $n'$-bit strings with, say, $n' := n + \log n$ (the term $\log n$ swallowing the description of a machine), we can think of $U^t$ as of a generator as well.

By the virtue of constructions of universal $U^t$ (for time $t$ machines) it is straightforward to show in theory PV that $Rng(g) \subseteq Rng(U^t)$ for any generator $g$ as above running in time $\leq t(n)$. Hence (the propositional translations of) this fact are shortly provable in ER, cf. [16, Chpt.12]). It follows that for any $P \geq ER$, if some $\tau$-formulas resulting from $U^t$ have short proofs so do some formulas resulting from $g$. That is, if there is any $g$ computable in time $t$ and hard for $P$ then $U^t$ must be hard as well. Putting it differently, proving tautologies expressing $K^t(w) > m/2$ must be hard for $P$ [6].

## 6.3 Implicit proof systems

Implicit proof systems, introduced in [12], operate with proofs $\pi$ computed bit-by-bit by a circuit (but that is not all). Proof $\pi$ may have size exponential in

[6] It is tempting to look for analogies of these tautologies with formulas occurring in Chaitin's [4] incompleteness theorem. But the interpretation of the role of information in incompleteness phenomenon is overwhelmed with mathematically unsupported or outright incorrect interpretations - see van Lambalgen [20] for analysis of some - and we stay away from any informal discussion of this topic.

comparison with the size of the defining circuit. Hence its $Kt$-complexity may be close to the lower bound $\log |\pi|$ from (1).

For two proof systems $P, Q$ the **implicit proof system** $[P, Q]$ considers a proof of a tautology $\tau$ to be a pair $(\alpha, \beta)$, where $\beta$ is a circuit whose truth-table is a $Q$-proof of $\tau$ and $\alpha$ is a $P$-proof (of the propositional statement formalizing) that $\beta$ indeed computes a $Q$-proof. Note that using circuits $\beta$ alone would not constitute a Cook-Reckhow proof system. For the formal definition see [12] or [16, Sec.7.3].

Implicit proof systems get incredibly strong very fast. For example, **implicit resolution** $iR := [R, R]$ p-simulates ER and iER p-simulates quantified propositional system $G$, cf. [16, Sec.7.3].

## 6.4 Proof systems with advice

Recall from Cook and K. [6] that a **functional**[7] **proof system with** $k(n)$ **bits of advice** is a $P : \{0,1\}^* \to \{0,1\}^*$ whose range is exactly TAUT and such that $P$ is computable in polynomial time using $k(n)$ bits of advice on inputs (i.e. proofs) of length $n$. Cook and K. [6, Thm.6.6] proved that there exists a proof system with 1 bit of advice that p-simulates all classical Cook-Reckhow's proof systems. This suggests[8] to use the only-if direction of Theorem 2.4 and to conclude that there is a *proof search algorithm with advice* $(A, P)$ which is $\geq_t$-better than all ordinary proof search algorithms of Definition 1.1. Here $P$ is the proof system with 1 bit of advice from [6, Thm.6.6] and $A$ is a *non-uniform* p-time algorithm, i.e. it uses p-size advice.

Too see this note that the proof of the only-if direction in Theorem 2.4 appeals to Lemma 2.2 that there is a time-optimal algorithm for any fixed proof system: in the universal search construction we need to check many - but only polynomially many - potential proofs of different lengths and each length requires a different bit of advice. The advice the algorithm $A$ will use collects all these bits together.

## 6.5 Diagonalization

Diagonalization in proof complexity was used in [13] (or see [16, Sec.21.4]) to prove that at least one of the following three statements is true:

1. There is a function $f : \{0,1\}^* \to \{0,1\}$ computable in time $2^{O(n)}$ that has circuit complexity $2^{\Omega(n)}$.

2. NP $\neq$ coNP.

3. There is no $p$-optimal propositional proof system.

[7] Classical proof systems can be formulated either as functional - as we did at the beginning of the paper - or as relational and these two formulations are essentially equivalent from proof complexity point of view. This is no longer true for systems with advice, cf. [6].

[8] To us it was suggested by Igor C. Oliveira.

A key part of that is a way, assuming that item 1 fails, how to compress possibly very long proofs and to represent them by small circuits. Using instead the $Kt$ measure may possibly allow for a stronger result.

# 7 Concluding remarks

Results in Sections 2 and 3 show that the optimality of proof search algorithms reduces to p-optimality of proof systems in both quasi-orderings based on time or information, respectively. This leaves some room for a totally different definition of a quasi-ordering of proof search algorithms that is coarser than those studied here and in which there could be an optimal algorithm without implying also the existence of a p-optimal proof system. On the other hand, the ordering by time of Section 2 is perhaps so rudimentary that it is the finest one among all sensible quasi-orderings; hence the opposite implication ought to hold always. However, it is our view that - from the point of view of proof complexity - the situation is clarified and the proof search problem is simply the p-optimality problem.

This does not quite dispel the doubts about the $\geq_t$ ordering discussed at the end of Section 2. The quasi-orderings considered here are theoretical models of a comparison of proof search algorithms and have shortcomings in modeling actual comparison of practical algorithms that are, we think, quite analogous to shortcomings of p-time algorithms as a theoretical model of practical feasible algorithms. The comparison in of real life algorithms is also more purpose specific and classifying all purposes that arise in practice may not be theoretically well-defined.

However, measure $i_P(\tau)$ may still have some uses for comparing two proof systems from the practical proof search point of view. For example, it can be used to kill all uniform formulas when testing algorithms (cf. the discussion at the end of Section 2) by accepting as test formulas only those satisfying, say, $Kt(\tau) \geq (\log |\tau|)^2$. Also, the information-efficiency functions for $P$, $Q$ such that $P >_p Q$ could lead to a suitable distance function measuring how much better $P$ than $Q$ is, by counting how much more information $Q$-proofs require than $P$-proofs do.

**Acknowledgments:**

I thank Igor C. Oliveira (Warwick U.) for a discussion about a draft of the paper.

# References

[1] A. Atserias and M. Müller, Automating Resolution is NP-Hard, *J. of the ACM*, **67(5)**, Article No. 31, September 2020.

[2] M. L. Bonet, T. Pitassi, and R. Raz, On Interpolation and Automatization for Frege Proof Systems, *SIAM J. of Computing*, **29(6)**, (2000), pp.1939-1967.

[3] S. R. Buss and J. Nordström, Proof Complexity and SAT Solving, in: Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (eds), *Handbook of Satisfiability*, 2nd edition, Chapter 7, (2021), pp.233-350. IOS Press.

[4] G. J. Chaitin, Information-theoretic limitations of formal systems, *J. Assoc. Comput.Mach.*, **21**, (1974), pp.403424.

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

[6] S. A. Cook, and J. Krajíček, Consequences of the Provability of $NP \subseteq P/poly$, *J. of Symbolic Logic*, **72(4)**, (2007), pp. 1353-1371.

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

[8] U. Feige, J. H. Kim and E. Ofek, Witnesses for nonsatisfiability of dense random 3CNF formulas, in: *Proc. of the IEEE 47th Annual Symposium on Foundations of Computer Science (FOCS)*, (2006), pp.497-508.

[9] A. Haken, The intractability of resolution, *Theoretical Computer Science*, **39**, (1985), pp.297-308.

[10] A. N. Kolmogorov, Three approaches to the quantitative definition of information. Problems Inform. Transmission, 1(1):17, 1965.

[11] A. N. Kolmogorov, Logical basis for information theory and probability theory, *IEEE Trans. on Information Theory*, **14(5)**, (1968), pp.662-664.

[12] J. Krajíček, Implicit proofs, *J. of Symbolic Logic*, **69(2)**, (2004), pp.387-397.

[13] J. Krajíček, Diagonalization in proof complexity, *Fundamenta Mathematicae*, **182**, (2004), pp.181-192.

[14] J. Krajíček, A note on SAT algorithms and proof complexity, *Information Processing Letters*, **112**, (2012), pp. 490-493.

[15] J. Krajíček, *Forcing with random variables and proof complexity*, London Mathematical Society Lecture Note Series, No. **382**, Cambridge University Press, (2011).

[16] J. Krajíček, *Proof complexity*, Encyclopedia of Mathematics and Its Applications, Vol. **170**, Cambridge University Press, (2019).

[17] J. Krajíček, Proof search problem (ext.abstract), in: *Mathematical Logic: Proof Theory, Constructive Mathematics* (9.-13.11.2020), Oberwolfach Reports (OWR), Report No. 34/2020, (2020), pp.45-46.

`https://www.karlin.mff.cuni.cz/~krajicek/mfo2020.pdf`

[18] 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.

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

[20] M. van Lambalgen, Algorithmic information theory, *J. Symbolic Logic*, **54**, (1989), pp.13891400.

[21] L. A. Levin, Universal sequential search problems, *Problems of Information Transmission*, **9**, (1973), pp.265266.

[22] L. A. Levin, Randomness conservation inequalities; information and independence in mathematical theories, *Information and Control*, **61**, (1984), pp.1537.

[23] S. Müller and I. Tzameret, Short Propositional Refutations for Dense Random 3CNF Formulas, *Annals of Pure and Applied Logic*, **165(12)**, (2014), pp.1864-1918.