On Gödel's Theorems on Lengths of Proofs II: Lower Bounds for Recognizing k Symbol Provability

Samuel R. Buss*

Abstract

This paper discusses a claim made by Gödel in a letter to von Neumann which is closely related to the P versus NP problem. Gödel's claim is that k-symbol provability in first-order logic can not be decided in $o(k)$ time on a deterministic Turing machine. We prove Gödel's claim and also prove a conjecture of S. Cook's that this problem can not be decided in $o(k/\log k)$ time by a nondeterministic Turing machine. In addition, we prove that the k-symbol provability problem is NP-complete, even for provability in propositional logic.

1 Introduction

This paper discusses a letter that Gödel wrote to von Neumann in 1956 in which Gödel raised some questions related to the feasibility of computations and to the $P =?NP$ problem. An excerpt from Gödel's letter is given below, but first we discuss as background the notions of effective versus feasible computability. The notion of *effective* computation is an intuitive (nonformal) concept: a problem is effectively decidable if there is a definite procedure which, in principle, can solve every instance of the problem. Church's Thesis (for which there is strong evidence) states that a problem is effectively computable if and only if it can be recursively computed, or equivalently, if and only if it can be decided by a Turing machine computation. However, effectively computable problems may not be feasibly computable since the computational resources such as time and space needed to solve an effectively computable problem may be enormous. By "feasible" we mean "computable in practice" or "computable in the real world"; or more to the point, a problem is feasibly computable if for any reasonably sized instance of the problem, we are able to solve that instance, albeit perhaps with a large (but not impossibly large) commitment of computing resources.

*Supported in part by NSF grants DMS-8902480 and DMS-9205181.

Like the notion of "effective", the notion of "feasible" is an intuitive concept; so it is natural to look for a feasible analog of Church's Thesis that will give a formal, mathematical model for feasible computation. A big advantage of having a mathematical model for feasible computability is that this allows a mathematical investigation of the power and limits of feasible computation. There is a widespread opinion that polynomial time computability is the correct mathematical model of feasible computation. This widespread belief is based primarily on two facts. First, the class of polynomial time computable functions and predicates is a robust and well-behaved class which has nice closure properties and is invariant for many natural models of computation. Second, and more importantly, it is an empirical observation about actual, real-world algorithms that it is nearly always the case that decision problems which are known to have polynomial time algorithms are also known to be feasibly decidable (see Cook [5] for a discussion of this). However, the case that polynomial time computation corresponds to feasible computation is not as firmly established as Church's Thesis, since an algorithm with runtime bounded by a high degree polynomial may not actually be feasible in any meaningful sense; conversely, an algorithm that takes $n^{\log\log n}$ steps on inputs of length n may well be feasible. Indeed, there are a few researchers who are advocating other mathematical models for feasible computation, such as quasilinear time $n(\log n)^{O(1)}$ on RAM's; and, as we see below, there is some indication that Gödel was considering quadratic time as being the limit of feasible computation.

We use P to denote the class of predicates recognizable in polynomial time. A closely related class is the class NP of problems recognizable in nondeterministic polynomial time; it is however an open problem whether problems in NP are feasibly decidable or are polynomial time decidable. This paper presumes the reader is familiar with the basics of the theory of NP-completeness, say, as contained in the account by Garey-Johnson [7].

Gödel wrote a letter to von Neumann in 1956 in which he made the following statements:

> One can obviously easily construct a Turing machine, which for every formula F in first order predicate logic and every natural number n, allows one to decide if there is a proof of F of length n (length = number of symbols). Let $\psi(F,n)$ be the number of steps the machine requires for this and let $\varphi(n) = \max_F \psi(F,n)$. The question is how fast $\varphi(n)$ grows for an optimal machine. One can show that $\varphi(n) \geq k \cdot n$. If there really were a machine with $\varphi(n) \sim k \cdot n$ (or even $\sim k \cdot n^2$), this would have consequences of the greatest importance [Tragweite]. Namely, it would obviously mean that in spite of the undecidability of the Entscheidungsproblem, the mental work of a mathematician concerning Yes-or-No questions could be completely[2] replaced by a machine. After all, one would

[2] except for the setting up of axioms

> simply have to choose the natural number n so large that when the machine does not deliver a result, it makes no sense to think more about the problem. Now it seems to me, however, to be completely within the realm of possibility that $\varphi(n)$ grows that slowly. Since (1) it seems that $\varphi(n) \geq k \cdot n$ is the only estimation which one can obtain by a generalization of the proof of the undecidability of the Entscheidungsproblem; and (2) after all $\varphi(n) \sim k \cdot n$ (or $\sim k \cdot n^2$) only means that the number of steps as opposed to trial and error can be reduced from N to $\log N$ (or $(\log N)^2$). However, such strong reductions appear in other finite problems, for example in the computation of the quadratic residue symbol using repeated application of the law of reciprocity. It would be interesting to know, for instance, the situation concerning the determination of primality of a number and how strongly in general the number of steps in finite combinatorial problems can be reduced with respect to simple exhaustive search. [3]

To make precise the issues raised in Gödel's letter, we shall think of first-order predicate logic as being formalized in one of the usual Hilbert-style systems which has a finite set of axiom schemata, including equality axioms, and with modus ponens and generalization as the only rules of inference.[4] As usual, the language of first-order logic may contain a variety of function and predicate symbols. It should be noted that we are not using weaker system of first-order logic such as resolution based systems; nor are we allowing the apparently more efficient systems which admit all tautologies as axioms. The problem with having all tautologies as axioms, is that in this case, $P \neq NP$ would imply that it is difficult to recognize valid proofs. For F a first-order formula, we write $\vdash F$ to mean that F is provable, and we write $\stackrel{n}{\vdash} F$ to denote that F has a first-order proof of $\leq n$ symbols. For this paper, we always measure proof length in terms of the number of symbols in the proof; for convenience, the number of symbols does not include symbols in subscripts of variables (so a variable counts as a single symbol).

Gödel then was asking about the difficulty of answering questions of the form "$\stackrel{n}{\vdash} F$?". To state this in computer science terminology, we consider the set

$$A \quad = \quad \{\langle F, 0^n\rangle \colon \stackrel{n}{\vdash} F\}.$$

Here 0^n denotes a string of n 0's and $\langle \cdots \rangle$ denotes a pairing function. For w a string of symbols, we write $|w|$ for the length of w. Gödel's question

[3] We thank the Institute for Advanced Studies, copyright holder of the German original, and Oxford University Press, copyright holder of English translation, for allowing us to include this quotation. The translation is by Peter Clote [4]; the footnote and the underlining are Gödel's.

[4] Modus ponens allows one to infer B from hypotheses A and $A \to B$; generalization allows one to infer $\forall x B$ from B. An example of an axiom schema is $A \to A \vee B$, which allows any formulas to be substituted for A and B. It is well-known that a finite set of axiom schemata, together with modus ponens and generalization, can provide a complete and sound formulation of first-order logic.

is then the question of whether the set A is recognizable on a multitape Turing machine in time $O(n)$ or in time $O(n^2)$; since any $\langle F, 0^n \rangle$ in A is a string of length proportional to n, this is equivalent to asking whether A is in linear or quadratic time.

It is easy to see that there is an effective algorithm for deciding membership in A; this is because, given F and n, one need only check the finitely many possible proofs of $\leq n$ symbols to see if any one is a first-order proof of F. However, this "effective" algorithm is not very feasible since it has exponential runtime, and would be too slow to be useful in practice. Thus one might ask (as Gödel did) whether there is a feasible algorithm for deciding membership in A.

Now it is an interesting and easy observation that the set A is NP-complete. Hence, if A is recognizable in linear or quadratic time, then $P = NP$! To prove that A is NP-complete, we first note that A is in NP since a nondeterministic algorithm for recognizing A consists of guessing a proof of F of $\leq n$ symbols and then verifying in polynomial time that the guessed proof is valid. Second, SAT is many-one reducible to A by the following construction, due to S. Cook. Let $G(p_1, \ldots, p_m)$ be a Boolean formula with the indicated variables. Choose $y, z, x_1, \ldots, x_m$ to be first order variables and define G^* to be the formula obtained from G by replacing each p_i in G by $x_i = y$. Let F be the first-order formula

$$\exists y(\exists z(y \neq z) \rightarrow \exists \vec{x} G^*).$$

It is easy to see that F is valid iff G is satisfiable and, furthermore, that, if G is satisfiable, then F has a proof of $|G|^{O(1)}$ symbols, since a very straightforward proof of F can be given by using one satisfying assignment for G.

A remarkable aspect of Gödel's letter is that it indicates that Gödel was thinking about problems related to the P vs NP problem well before these classes were widely studied.[5] This is not to say that Gödel knew the theory of NP-completeness. To the contrary, one infers from the letter to von Neumann that Gödel was thinking of linear and quadratic time as corresponding to feasible computation; and thus that Gödel had not realized the importance of polynomial time as a mathematical model of feasible computation. Nor is there any indication that Gödel had any formal conception of the class NP or had any knowledge of the NP-completeness of the k-symbol provability problem. However, Gödel was clearly thinking in this general direction, since at the end of quote above, Gödel discusses the fact there are many problems which are known to be solvable in exponential time by 'simple exhaustive search', but for which it is not known whether better algorithms exist. Clearly, Gödel realized that if the set A could be recognized in linear or quadratic time,

[5] See Sipser [13] and Hartmanis [8] for discussions of Gödel's letter and of other early work on feasibility of computation.

then this would substantially simplify the solution of problems solvable by simple exhaustive search.

It is surprising to modern readers that Gödel would say that it is possible (in the original German: "durchaus im Bereich der Möglichkeit liegt") that $P = NP$. However, it is this author's opinion that Gödel did not intend to say it was likely or probable, but merely that he had been unable to disprove it. To the contrary, the natural view would be that a feasible analogue of the undecidability of first-order logic would imply that the k-symbol provability problem is not feasibly decidable. To make clearer the justification of this 'natural view' we can argue as follows:

> PREMISE 1**:** Mathematics (especially the discovery of mathematical truths) is inherently difficult for humans and for Turing machines, even working in concert.
>
> PREMISE 2**:** Some of the inherently difficult aspects of mathematics can be phrased as questions of the form "$\vdash^{n \text{ symbols}} \phi$?" where n is not excessively large.
>
> CONCLUSION**:** The question "$\vdash^{n \text{ symbols}} \phi$?" can not be reliably decided in time $k \cdot n$ or $k \cdot n^2$ with k not excessively large. More generally, "$\vdash^{n \text{ symbols}} \phi$?" can not be feasibly decided.

To the extent that polynomial time corresponds to feasible computation, the CONCLUSION implies that $P \neq NP$. As an aside, we know of no similar argument indicating that the polynomial time hierarchy is proper or even that NP is not closed under complementation.

PREMISE 1 is justified partly by mathematicians' experiences of the difficulty of mathematical research and partly by analogy with the undecidability of validity in first-order logic (the Entscheidungsproblem). Of course, the difficulties of mathematical discovery include both intuitive, nonformal aspects (such as deciding which problems are interesting or reasoning by analogy) and also formal aspects (Gödel's 'Yes-or-No questions'). PREMISE 2 says, in essence, that even the purely formal aspects of mathematics are inherently difficult.

A complaint that could be made about PREMISE 2 is that it concerns only the decidability of whether n symbol proofs exist, and says nothing about the feasibility of finding a particular formal proof of ϕ and says even less about finding a proof of ϕ which is intelligible to a human mathematician. In other words, PREMISE 2 could be attacked with the argument that perhaps the 'inherently difficult' aspects of mathematics do not reside in the sheer knowledge of existence of proofs, but instead in the human act of understanding mathematical facts and proofs. However, this argument is readily refuted by noting that if the CONCLUSION is false, then the difficulty of discovering and understanding mathematical facts

and proofs would be greatly lessened. To show this, suppose for the sake of concreteness that we have a Turing machine which in time n^2 decides the question " $\vdash^{n \text{ symbols}} \phi$?". If $n = 1,000,000$ then n^2 is 10^{12} and since modern day technology easily allows the construction of computers which simulate 10^8 Turing machine steps per second, we could then decide questions " $\vdash^{n \text{ symbols}} \phi$?" in 10^4 seconds, i.e., less than three hours.[6] Now suppose, that in the course of a mathematical investigation we have already established and understood some results ψ and are interested in establishing whether χ holds. We then ask the questions " $\vdash^{10^6} \psi \to \chi$?" and " $\vdash^{10^6} \psi \to \neg\chi$?" (to two machines, say) and wait 2.8 hours. In 2.8 hours we will either know that χ is true or that χ is false or that χ is neither provable nor refutable in 10^6 symbols from ψ.[7] If one finds that χ is neither provable nor refutable in 10^6 symbols, then one can either allow more time to ask about longer proofs, or, better, one should formulate some simpler questions first and return to the question of χ later. If we learn that χ is true (a similar argument holds if χ is learned to be false), then we are interested in understanding why χ is true. In this case, since we know χ is true, we try to think of possible proofs and formulate lemmas or partial results that may aid in the proof of χ. Using our Turing machine we can determine whether the proposed lemmas or partial results are true, and even if the lemmas shorten the proof of χ. This process can be iterated until complete proofs of the lemmas and a complete proof of χ from the lemmas are found.

In summary, if the CONCLUSION is false, then mathematical research would evolve (or devolve) into a process which alternates between (1) human creativity and intuition making conjectures, and (2) Turing machine computation establishing whether the conjectures are true. Of course, as computers become significantly more powerful in the future, mathematics might evolve in this way anyway; but our point is that the falsity of the CONCLUSION would make this symbiotic human/Turing machine process almost infallible for understanding mathematical truths. But our intuitive feeling is that this would be too good to be true and that mathematics can not become this easy.

The rest of this paper contains two mathematical results. First, in

[6] Here we must assume that there is a reasonably small number bounding the number of states and the alphabet size of the Turing machine; as we must be able to code a description of the Turing machine. Our speed estimates are fairly conservative as it would be possible to build hardware for simulating 10^9 Turing machine steps per second (or more).

[7] One might object to the use of 10^6 here on the grounds that very little mathematics can be formalized in Zermelo-Fraenkel set theory in only 10^6 symbols. However, there is no reason to restrict oneself to set theory; any first-order theory is allowed and one should choose an appropriate language of first order logic in which to express ϕ and χ that will, one hopes, allow a relatively succinct proof or disproof of χ from ψ.

section 2, we prove that the decision problem " $\vdash^n F$?" is NP-complete for *propositional* logic. As we observed above this problem is NP-complete for first-order logic. The proof is significantly more complicated for propositional logic; and it is a somewhat surprising theorem, since the set of provable propositional formulas (i.e., tautologies) is coNP-complete.

Second, in section 3, we prove Gödel's assertion that $\varphi(n) \geq k \cdot n$ for some constant k. To make this formal, we prove:

Theorem 1 *Let A be an alphabet and let $k \geq 1$. Then there is an $\epsilon > 0$, depending only on k and the cardinality of A such that, for any k-tape Turing machine M with alphabet A, if M accepts inputs $\langle \phi, n \rangle$ precisely when ϕ is a first-order formula and $\vdash^n \phi$, then M has runtime $\geq \epsilon \cdot n$ infinitely often. This holds even if the length $|\phi|$ of ϕ is constrained to be $O(\log n)$ and if n is coded in binary notation.*

The extra constraint that ϕ and n are coded by strings of length $O(\log n)$ is necessary to make the theorem non-trivial, since if ϕ is allowed to have length proportional to n or if n were coded in unary, then the Turing machine would require time $\Omega(n)$ just to read its input. Although we shall not prove it here, the theorem is still true even if ϕ is constrained to be much smaller, for example, if $|\phi|$ is $O(\log \log n)$ or $\log^* n$.

Our proof of Theorem 1 is based on a diagonal construction of a self-referential first-order formula Φ_n which asserts of itself that it is not accepted by M in n steps. Thus we conjecture that our proof is similar to the proof envisioned by Gödel. In order to obtain the lower bounds of Theorem 1, it is necessary that if this self-referential formula is not accepted by M, then it has proof of symbol length proportional to n. This leads to the problem of constructing first-order formulas which express the action of n steps of a Turing machine and which have proofs of linear symbol-length (in n). This problem is solved in section 3; improving on earlier approaches which would have yielded first-order formulas with proofs of quadratic or $O(n \log n)$ symbol-length.

In section 4, we extend Theorem 1 to prove a result first conjectured by S. Cook; namely, if the Turing machine M is nondeterministic then it can not have runtime $o(n/\log n)$.

Sections 3 and 4 may be read independently of section 2. In a prequel to this paper, we proved some earlier results of Gödel's on lengths of proofs [3]. However, the present paper can be read completely independently.

The author thanks Steve Cook and Peter Clote for significant, helpful discussions, and the anonymous referee for helpful comments.

2 Propositional k-symbol provability

The k-symbol provability problem is the problem of deciding whether $\vdash^k \phi$, given k and ϕ as inputs; of course, we get distinct k-symbol

provability problems for distinct formal definitions of provability. In this section we prove that the k-symbol provability for a particular formalization of propositional logic is NP-complete.

The system $\mathcal{F}$ of propositional logic that we shall use is a particular kind of Frege proof system. Frege proof systems are propositional proof systems which are axiomatized by a finite set of axiom schemata and by a finite set of schematic rules of inference (see Reckhow [11], Cook-Reckhow[6], Statman [14] and Bonet [1] for more information on Frege systems). It is known that any two Frege proof systems $\mathcal{F}_1$ and $\mathcal{F}_2$ can polynomially simulate each other; which means that there is a polynomial p such that if $\mathcal{F}_1 \stackrel{n}{\vdash} \phi$ then $\mathcal{F}_2 \stackrel{p(n)}{\vdash} \phi$ (with the complication that if $\mathcal{F}_1$ and $\mathcal{F}_2$ have different languages, it may be necessary to translate ϕ into the language of $\mathcal{F}_2$). Furthermore, if $\mathcal{F}_1$ and $\mathcal{F}_2$ have the same language then $\mathcal{F}_1$ and $\mathcal{F}_2$ linearly simulate each other; i.e., there is a constant $c > 0$ such that, for all ϕ, if $\mathcal{F}_1 \stackrel{n}{\vdash}$ then $\mathcal{F}_2 \stackrel{cn}{\vdash} \phi$.

In proving Theorem 2 below, we shall construct a sequence of tautologies $\phi_1, \phi_2, \phi_3, \ldots$ and a sequence of integers $\ell_1, \ell_2, \ell_3, \ldots$ such that each ϕ_i has an $\mathcal{F}$-proof of symbol-length $3 \cdot \ell_i$, but such that it is NP-hard to determine, on input ϕ_i, whether $\mathcal{F} \stackrel{\ell_i}{\vdash} \phi_i$. Thus our proof methods are not strong enough to establish the NP-completeness of the k-symbol provability problem for all Frege proof systems: the problem is that there is only a "linear size gap" between the length ℓ_i of a possible short proof of ϕ_i and the upper bound on the length of a known proof of ϕ_i.

Theorem 2 is somewhat surprising at first, since propositional proofs are methods of recognizing (or witnessing) that a formula is a tautology. And to show that the k-symbol provability problem is NP-complete sounds tantamount to showing that the set of tautologies is NP-complete and thus NP = coNP. But this is not the case: the fact that there is only a "linear size gap" between ℓ_i and the known proof of length $3 \cdot \ell_i$ means that what we are showing is that even for tautologies in a particular special form with known proofs, it can be hard to decide if a shorter proof exists. Thus our proof does not establish any strong new results in complexity theory such as the collapse of the polynomial time hierarchy.

An analogous situation occurred in our earlier work [2] where it was shown that the k-step provability problem for the first-order sequent calculus is undecidable. There we gave a particular integer k and an infinite family of first-order formulas, each of which have proofs of $\leq k+1$ steps, but so that it is undecidable whether a given formula in the family has a proof k lines.

We now define the particular Frege system $\mathcal{F}$ (based on a system in [10]) to which our results apply. The language of $\mathcal{F}$ contains the logical symbols $\neg$, $\vee$, $\wedge$ and $\rightarrow$ and has propositional variables $p_0, p_1, p_2, \ldots$. The formulas of $\mathcal{F}$ are inductively defined by taking every propositional

variable p_i as a formula and, if ϕ and ψ are formulas, letting $(\neg\phi)$, $(\phi\wedge\psi)$, $(\phi\vee\psi)$, and $(\phi\rightarrow\psi)$ be formulas. The *length* $|\phi|$ of a formula ϕ is the number of symbols in the formula. We frequently abbreviate formulas by omitting parentheses when it will not affect readability; the usual precedence conventions apply, with $\rightarrow$ associating from right to left. The axioms of $\mathcal{F}$ are:

$$
\begin{array}{l}
A\rightarrow(B\rightarrow A)\\
(A\rightarrow B)\rightarrow((A\rightarrow(B\rightarrow C))\rightarrow(A\rightarrow C))\\
A\rightarrow(B\rightarrow(A\wedge B))\\
(A\wedge B)\rightarrow A\\
(A\wedge B)\rightarrow B\\
B\rightarrow(A\vee B)\\
A\rightarrow(A\vee B)\\
(A\rightarrow C)\rightarrow((B\rightarrow C)\rightarrow((A\vee B)\rightarrow C))\\
\neg\neg A\rightarrow A\\
(A\rightarrow B)\rightarrow((A\rightarrow\neg B)\rightarrow(\neg A))
\end{array}
$$

where A, B and C may be any formulas. The only rule of inference in $\mathcal{F}$ is modus ponens: from A and $A\rightarrow B$, infer B. The *size* or *length* $|P|$ of a proof P is the sum of the lengths of the formulas in the proof (sometimes this is called the *symbol-length* of P). We write $\mathcal{F}\stackrel{n}{\vdash}\phi$ to denote that ϕ has an $\mathcal{F}$-proof of length $\leq n$.

If ϕ is a formula and n an integer, we write $\langle\phi,0^n\rangle$ to denote a string encoding the pair ϕ and n of length $O(|\phi|+n)$.

Theorem 2 *The decision problem* $KPROV\mathcal{F}=\{\langle\phi,0^n\rangle:\mathcal{F}\stackrel{n}{\vdash}\phi\}$ *is NP-complete.*

This theorem also holds if the lengths of formulas and $\mathcal{F}$-proofs are defined so as to count symbols in subscripts of variables written, say, in binary notation. The same proof construction works, but the analysis and size bounds are slightly different.

Although our proof of Theorem 2 does not apply directly to other Frege systems; it will be obvious that the proof can be modified to hold for many other particular proof systems. In fact, we conjecture that the theorem is true for every Frege proof system.

Theorem 2 is proved by giving a many-one polynomial time (Karp) reduction from $3SAT$ to $KPROV\mathcal{F}$. The many-one reduction is defined as follows. Recall that an instance ϕ of $3SAT$ is a set of clauses, each containing three literals; the set of clauses is interpreted as a conjunction of disjunctions of size 3. We next define a formula ϕ^*, so that the mapping $\phi\mapsto\phi^*$ will be the desired many-one reduction. Let ϕ use variables $p_1,\ldots,p_n$ and contain c clauses: each clause is of the form $\langle\pm p_i,\pm p_j,\pm p_k\rangle$ where $\pm p$ denotes either p or $\neg p$ and where i,j,k are distinct. Let σ_i and τ_i, for $1\leq i\leq n$, be $2n$ distinct tautologies; namely, let σ_i be $(p_i\vee\neg p_i)$ and let τ_i be $(\neg p_i\vee p_i)$. Let $\perp$ be the unsatisfiable

formula $(p_0 \wedge \neg p_0)$ and let $\top$ be the tautology $(p_0 \vee \neg p_0)$. Define σ_i^ℓ to be $(\bot \vee (\bot \vee (\cdots (\bot \vee \sigma_i)) \cdots))$ and define τ_i^ℓ to be $(\bot \vee (\bot \vee (\cdots (\bot \vee \tau_i)) \cdots))$, where there are ℓ many occurrences of $\bot$ in each of the formulas; it is important that the disjunctions in σ_i^ℓ and τ_i^ℓ are associated from right to left. We let σ_i^* and τ_i^* be the formulas σ_i^m and τ_i^m, where m is a constant depending on ϕ which will be picked to make our construction work (in fact, $m = c^2$). If x is a literal, define x^* to be τ_i^* if x is p_i and to be σ_i^* if x is $\neg p_i$. For $\alpha = \langle x_1, x_2, x_3 \rangle$ a clause in ϕ, define α^* to be the formula $x_1^* \vee (x_2^* \vee x_3^*)$.

To finish the definition of ϕ^*, we need the notion of a *balanced* conjunction. We use $\bigwedge\!\!\!\bigwedge$ exclusively to denote balanced conjunctions, and we define $\bigwedge\!\!\!\bigwedge_{i=1}^{1} A_i$ to be just A_1 and, for $n > 1$, define $\bigwedge\!\!\!\bigwedge_{i=1}^{n} A_i$ to be

$$\left(\bigwedge\!\!\!\bigwedge_{i=1}^{\lfloor n/2 \rfloor} A_i \right) \wedge \left(\bigwedge\!\!\!\bigwedge_{i=\lfloor n/2 \rfloor + 1}^{n} A_i \right).$$

Notice that the depth of nesting of $\wedge$'s in $\bigwedge\!\!\!\bigwedge_{i=1}^{n} A_i$ is exactly $\lceil \log_2 n \rceil$.

Let ϕ contain clauses $\alpha_1, \ldots, \alpha_c$ and define α_{c+i} to be $(p_i \vee \neg p_i)$. Then α_{c+i}^* is to be $(\tau_i^* \vee \sigma_i^*)$. We define ϕ^* to be the formula

$$\bigwedge\!\!\!\bigwedge_{i=1}^{c+n} \alpha_i^*.$$

It is obvious that ϕ^* is a tautology, since it is a conjunction of disjunctions of σ_i^*'s and τ_i^*'s, which are tautologies. Since $\mathcal{F}$ is complete, ϕ^* certainly has an $\mathcal{F}$-proof — we shall examine carefully the minimum number of symbols needed in an $\mathcal{F}$-proof of ϕ. (Recall that m is a constant depending on ϕ.)

Lemma 3 *The formulas σ_i^* and τ_i^* have $\mathcal{F}$-proofs of symbol length*

$$\frac{3}{2}(3 + |\bot|)m^2 + O(m) = \frac{27}{2}m^2 + O(m).$$

Proof A proof of τ_i^* starts with a proof of the tautology τ_i^0 (i.e., τ_i) and continues as:

$\vdots$	
τ_i^0	
$(\tau_i^0 \to \tau_i^1)$	Axiom (since τ_i^1 is $\bot \vee \tau_i^0$)
τ_i^1	Modus ponens
$(\tau_i^1 \to \tau_i^2)$	Axiom
τ_i^2	Modus ponens
$\vdots$	
τ_i^m	

Since $|\bot| = |\tau_i^0|$, each τ_i^ℓ has exactly $|\bot| + (3 + |\bot|)\ell$ symbols. The tautology τ_i^0 has a constant size proof (independent of i, since symbols in subscripts are not counted). Thus the above proof of τ_i^m has number of symbols equal to

$$\begin{aligned} O(1) + |\tau_i^m| + \sum_{\ell=0}^{m-1} (3 + 2|\tau_i^\ell| + |\tau_i^{\ell+1}|) & \\ &= \sum_{\ell=0}^{m-1} 3|\tau_i^\ell| \ + \ O(m) \\ &= \sum_{\ell=0}^{m-1} 3(3 + |\bot|)\ell \ + \ O(m) \\ &= 3(3 + |\bot|)\frac{m(m-1)}{2} + O(m) \\ &= \frac{3}{2}(3 + |\bot|)m^2 + O(m). \end{aligned}$$

That proves the lemma for τ_i^*; of course the same construction works for σ_i^*. □

Lemma 4 *(See also Bonet [1]) From the extra hypotheses $A_1, \ldots, A_r$, the balanced conjunction $\bigwedge_{i=1}^{r} A_i$ has an $\mathcal{F}$-proof of length $O((\sum_{i=1}^r |A_i|) \cdot \log r)$.*

Proof The obvious proof of a balanced conjunction consists of proving $\bigwedge_{i=1}^{r} A_i$ from its two conjuncts $\bigwedge_{i=1}^{\lfloor r/2 \rfloor} A_i$ and $\bigwedge_{i=\lfloor r/2 \rfloor + 1}^{r} A_i$ — this part takes $O(\sum_{i=0}^r |A_i|)$ symbols. Each of the two conjuncts is proved in the same fashion, etc. Straightforward calculation of the total number of symbols in the proof yields the lemma. □

The previous two lemmas allow us to give an upper bound on the size of $\mathcal{F}$-proofs of ϕ^*:

Corollary 5 *The formula ϕ^* has an $\mathcal{F}$-proof containing*

$$3(3 + |\bot|)nm^2 + O(mc \log c) = 27nm^2 + O(mc \log c)$$

symbols.

Proof An $\mathcal{F}$-proof of ϕ^* can be given as follows: First prove the $2n$ formulas σ_i^* and τ_i^* with a total of $3(3+|\bot|)nm^2 + O(nm)$ symbols, using the construction of Lemma 3. Then each conjunct α^* of ϕ^* is of the form $\sigma_i^* \vee (\cdots)$ or $\tau_i^* \vee (\cdots)$ and can be proved in $O(m)$ additional symbols; i.e., all of the α^*'s can be derived in $O(m(c+n))$ symbols. From this, by

Lemma 4, ϕ^* can be proved with $O(m(c+n)\log(c+n))$ further symbols, since $|\phi^*|$ is $O(m(c+n))$ and since ϕ has $c+n$ conjuncts. Altogether, this gives a proof of $\phi*$ of

$$3(3+|\bot|)nm^2 + O(nm + m(c+n) + m(c+n)\log(c+n))$$

which proves the corollary since w.l.o.g. $n < 3c$. □

Corollary 5 gives an upper bound on the size of an $\mathcal{F}$-proof needed for ϕ^* obtained from an arbitrary ϕ; the next lemma gives a sharper upper bound under the additional assumption that ϕ is satisfiable.

Lemma 6 *If ϕ is satisfiable, then ϕ^* has an $\mathcal{F}$-proof of length*

$$\frac{3}{2}(3+|\bot|)nm^2 + O(mc\log c) = \frac{27}{2}nm^2 + O(mc\log c).$$

Proof Let ϕ have a satisfying assignment $\nu : \{x_1, \ldots, x_n\} \to \{T, F\}$. A proof of ϕ^* can be given as follows. For each i, such that $\nu(x_i) = T$, prove τ_i^* with $\frac{3}{2}(3+|\bot|)m^2 + O(m)$ symbols. For each i such that $\nu(x_i) = F$, prove σ_i^* with the same bound on the number of symbols. By the definition of ϕ^* and since ν is a satisfying assignment, every conjunct α^* of ϕ^* is a disjunction of size 2 or 3, with at least one of the disjuncts an already proved σ_i^* or τ_i^*. Hence each of the $n+c$ disjuncts can be proved from the σ_i^*'s and τ_i^*'s with $O(m)$ additional symbols (each disjunct has $O(m)$ symbols). Finally, by Lemma 4, ϕ^* can be proved with $O(mc\log c)$ symbols (since $n < 3c$). Altogether this gives the desired proof of size

$$\frac{3}{2}(3+|\bot|)nm^2 + O(mc\log c). \quad \square$$

We now define the constant m to be equal to c^2. The reason for this choice of m is, that for c sufficiently large, m^2 dominates $mc\log c$:

Corollary 7 *For c sufficiently large, if ϕ is satisfiable, then ϕ^* has an $\mathcal{F}$-proof of length less than $\frac{3}{2}(3+|\bot|)(n+1)m^2 = \frac{27}{2}(n+1)m^2$.*

Lemma 6 gives an upper bound on the size of proofs of ϕ^* when ϕ is satisfiable. To complete the proof that $\phi \mapsto \phi^*$ is a reduction of $3SAT$ to $KPROV\mathcal{F}$ we must give a lower bound on the size of proofs of ϕ^* when ϕ is not satisfiable. Obviously this lower bound must be larger that the upper bound of Lemma 6; since this allows us to reduce the question of the satisfiability of ϕ to a question about the size of the shortest $\mathcal{F}$-proof of ϕ^*. Namely, we shall show that ϕ is satisfiable if and only if ϕ^* has an $\mathcal{F}$-proof of less than $\frac{3}{2}(3+|\bot|)(n+1)m^2$ symbols.

Definition Let P be a proof. A formula ψ is said to be *used* in P if and only if ψ is used as a hypothesis of an inference in P. We say that ψ and χ are *cohypotheses* in P if there is a rule of inference which has both ψ and χ as hypotheses.

We shall assume w.l.o.g. that any formula in an $\mathcal{F}$-proof is either the formula being proved or is used in the proof; in the latter case, it must have at least one cohypothesis. Obviously this assumption is true of any minimum size proof.

Definition Let ψ be a formula and consider a particular τ_i^ℓ. We define $\psi[\![\tau_i^\ell]\!]$ to be the formula obtained from ψ by replacing every occurrence of $\tau_i^{\ell+1}$ as a (not necessarily proper) subformula of ψ with the formula $\bot$.

If P is a proof, then $P[\![\tau_i^\ell]\!]$ is the sequence of formulas obtained by replacing every formula ψ in P with $\psi[\![\tau_i^\ell]\!]$. We define $\psi[\![\sigma_i^\ell]\!]$ and $P[\![\sigma_i^\ell]\!]$ similarly. Also, if $\vec{\rho}$ is a vector of τ_i^ℓ's and σ_i^ℓ's then $\psi[\![\vec{\rho}]\!]$ and $P[\![\vec{\rho}]\!]$ are defined in the obvious way: it is assumed that the vector $\vec{\rho}$ contains no duplicate elements.

Note that $P[\![\tau_i^\ell]\!]$ will not, in general, be a valid proof.

For the next lemmas it will be convenient to work with a Frege proof system $\mathcal{F}'$ which is defined to be the system $\mathcal{F}$ plus the additional axiom scheme $A \to A$.

Lemma 8 *Let P be an $\mathcal{F}'$-proof of ϕ^*. Suppose that the formula $\tau_i^\ell \to \tau_i^{\ell+1}$ is not used in P. Then $P[\![\tau_i^\ell]\!]$ is a valid $\mathcal{F}'$-proof of $\phi^*[\![\tau_i^\ell]\!]$.*

Similarly, if $\sigma_i^\ell \to \sigma_i^{\ell+1}$ is not used in P, then $P[\![\sigma_i^\ell]\!]$ is a valid $\mathcal{F}'$-proof of $\phi^*[\![\sigma_i^\ell]\!]$.

Proof Consider any $\mathcal{F}'$-axiom ψ not equal to $\tau_i^\ell \to \tau_i^{\ell+1}$. We claim that $\psi[\![\tau_i^\ell]\!]$ must also be an $\mathcal{F}'$-axiom. This claim is proved by checking all the axioms and using the fact that the outermost connective of τ_i^ℓ is $\vee$. The first non-trivial case is for an axiom of the form

$$(\bot \to C) \to ((\tau_i^\ell \to C) \to (\tau_i^{\ell+1} \to C)),$$

which becomes an axiom of the form

$$(\bot \to C) \to ((\tau_i^\ell \to C) \to (\bot \to C)).$$

The second non-trivial case is for the axiom $\bot \to \bot \vee \tau_i^\ell$ which becomes the axiom $\bot \to \bot$. (Since we are using the system $\mathcal{F}'$.) In all other cases, the axioms maintain the same form.

Secondly, since $\tau_i^{\ell+1}$ has outermost connective $\vee$, we have that $(\psi \to \chi)[\![\tau_i^\ell]\!]$ must equal $\psi[\![\tau_i^\ell]\!] \to \chi[\![\tau_i^\ell]\!]$. Hence every modus ponens inference in P remains a modus ponens inference in $P[\![\tau_i^\ell]\!]$. This suffices to prove the lemma. □

Iterating Lemma 8, we obtain:

Corollary 9 *Suppose P is an $\mathcal{F}'$-proof of ϕ^* and suppose $\rho_1, \ldots, \rho_n$ are formulas such that each ρ_i is either $\sigma_i^{\ell_i}$ or $\tau_i^{\ell_i}$ for some $\ell_i < m$. Let ρ_i' be $\sigma_i^{\ell_i+1}$ or $\tau_i^{\ell_i+1}$, respectively. Further suppose that none of the formulas $\rho_i \to \rho_i'$ is used in P. Then $P\llbracket\vec{\rho}\rrbracket$ is a valid $\mathcal{F}'$-proof of $\phi^*\llbracket\vec{\rho}\rrbracket$.*

Lemma 10 *Let P be an $\mathcal{F}'$-proof of ϕ^* and fix $1 \leq i \leq n$. Then either (1) every formula $\tau_i^\ell \to \tau_i^{\ell+1}$, for $0 \leq \ell < m$, appears in P, or (2) every formula $\sigma_i^\ell \to \sigma_i^{\ell+1}$, for $0 \leq \ell < m$, appears in P.*

Proof Suppose, for sake of a contradiction, that neither $\tau_i^\ell \to \tau_i^{\ell+1}$ nor $\sigma_i^{\ell'} \to \sigma_i^{\ell'+1}$ are used in P. Then, by two uses of Lemma 8, $P\llbracket\tau_i^\ell, \sigma_i^{\ell'}\rrbracket$ is an $\mathcal{F}'$-proof of $\phi^*\llbracket\tau_i^\ell, \sigma_i^{\ell'}\rrbracket$. But this is impossible, since $\phi^*\llbracket\tau_i^\ell, \sigma_i^{\ell'}\rrbracket$ is not a tautology as ϕ^* contains a conjunct $\tau_i^m \vee \sigma_i^m$ which, in $\phi^*\llbracket\tau_i^\ell, \sigma_i^{\ell'}\rrbracket$ becomes $\bot^{m-\ell} \vee \bot^{m-\ell'}$ where $\bot^j$ is defined to be a disjunction of j occurrences of $\bot$ (associated from right to left). $\square$

Lemma 11 *Suppose ϕ^* has an $\mathcal{F}$-proof of less than $\frac{3}{2}(3+|\bot|)(n+1)m^2 = \frac{27}{2}(n+1)m^2$ symbols. Then ϕ is satisfiable.*

Proof First, we need a preliminary result. Consider a particular τ_i^* (the same results hold for σ_i^* too). Suppose each formula $\tau_i^\ell \to \tau_i^{\ell+1}$, for $0 \leq \ell < m$, is used in P. Then we claim that the number of symbols occurring in the formulas $\tau_i^\ell \to \tau_i^{\ell+1}$ and their cohypotheses in P is $\geq \frac{3}{2}(3+|\bot|)m^2$. This claim is proved by (1) noting that any cohypothesis of $\tau_i^\ell \to \tau_i^{\ell+1}$ is either τ_i^ℓ or is of the form $(\tau_i^\ell \to \tau_i^{\ell+1}) \to (\cdots)$ and hence has at least $|\tau_i^\ell|$ symbols, and (2) observing that if $\ell \neq \ell'$, then $\tau_i^\ell \to \tau_i^{\ell+1}$ and $\tau_i^{\ell'} \to \tau_i^{\ell'+1}$ can not share cohypotheses, and (3) using the calculations from the proof of Lemma 3.

Now we also have that if A and B are distinct formulas, each of the form $\tau_i^\ell \to \tau_i^{\ell+1}$ or $\sigma_i^\ell \to \sigma_i^{\ell+1}$, then A and B have no cohypotheses in common. It follows, that if P has $< \frac{3}{2}(3+|\bot|)(n+1)m$ symbols, then no more than n of $\tau_1, \ldots, \tau_n, \sigma_1, \ldots, \sigma_n$ have all of their associated formulas $\tau_i^\ell \to \tau_i^{\ell+1}$ or $\sigma_i^\ell \to \sigma_i^{\ell+1}$ (for $0 \leq \ell < m$) used in P. Together with Lemma 10, this implies that there are $\rho_1, \ldots \rho_n$ with each ρ_i of the form $\tau_i^{\ell_i}$ or $\sigma_i^{\ell_i}$ such that, setting ρ_i' to be $\tau_i^{\ell_i+1}$ or $\sigma_i^{\ell_i+1}$ respectively, none of the formulas $\rho_i \to \rho_i'$ appears in P.

Hence, by Lemma 9, $P\llbracket\vec{\rho}\rrbracket$ is an $\mathcal{F}'$-proof of $\phi^*\llbracket\vec{\rho}\rrbracket$; in particular, $\phi^*\llbracket\vec{\rho}\rrbracket$ is a tautology. Define a truth assignment ν so that $\nu(p_i)$ equals T (respectively, F) if ρ_i is $\sigma_i^{\ell_i}$ (respectively, $\tau_i^{\ell_i}$). To prove Lemma 11, it suffices to show that ν satisfies ϕ. However, this is more-or-less immediate from the definitions. A clause α in ϕ has three literals. In ϕ^*, each unnegated literal p_i in α is replaced by τ_i^m and each negated literal $\neg p_i$ is replaced by σ_i^m. In $\phi^*\llbracket\vec{\rho}\rrbracket$, if $\nu(p_i) = T$ then σ_i^m is replaced by an unsatisfiable formula, and if $\nu(p_i) = F$ then τ_i^m is replaced by

an unsatisfiable formula. Now in order for $\phi^*[\![\vec{\rho}]\!]$ to be a tautology, each conjunct in ϕ^* must have at least one disjunct not replaced by an unsatisfiable formula. This is clearly equivalent to each clause in ϕ having at least one literal assigned the value T by ν; i.e., equivalent to ν being a satisfying assignment for ϕ. □

Corollary 7 and Lemma 11 complete the proof of Theorem 2, since we have given a polynomial time construction of a propositional formula ϕ^* from an instance ϕ of $3SAT$ such that, for ϕ having sufficiently many clauses, ϕ is satisfiable if and only if ϕ^* has an $\mathcal{F}$-proof of $< \frac{27}{2}(n+1)m^2$, where n is the number of distinct variables in ϕ and m is the square of the number of clauses in ϕ.

3 Deterministic-time lower bound

In this section we prove Theorem 1 which is our formalization of the claim made by Gödel in the letter to von Neumann. In view of the difficulty (namely, Theorem 2) of the k-provability problem even for propositional logic, one might suspect that one reason for Theorem 1 to hold is that any Turing machine M satisfying the conditions of Theorem 1 could be used to help solve the k-provability problem. However, it is possible to state a strengthened version of Theorem 1 that avoids this connection to the k-provability problem, as follows:

Theorem 12 *Let A be an alphabet and let $k \geq 1$. Then there is an $\epsilon > 0$, depending only on k and the cardinality of A so that the following holds:*

There is no Turing machine with alphabet A and with k tapes such that for all first-order sentences ϕ,

(a) *If $\vdash^{\underline{n \text{ symbols}}} \phi$, then M accepts ϕ in $\leq \epsilon \cdot n$ steps,*

(b) *If $\nvdash \phi$, then M on input ϕ does not accept (but might never halt either).*

In fact, no such M exists even if (a) is weakened to apply only to ϕ such that $|\phi| = O(\log n)$.

Note that a Turing machine M which satisfies (a) and (b) may not help at all with deciding if ϕ's shortest proof has exactly n symbols; this is because if M accepts in ϵm steps this merely signifies the existence of some proof of ϕ, not necessarily of a proof of $\leq m$ symbols.

The proof will use a diagonal construction to create self-referential first-order formulas ϕ which assert that they are not accepted by M in ϵn steps but nonetheless are provable in n symbols (here the constant ϵ is determined below). This will then lead to a contradiction since we will establish that

$$\begin{aligned} \nvdash \phi &\Rightarrow M \text{ does not accept } \phi \\ &\Rightarrow M \text{ does not accept } \phi \text{ in } \leq \epsilon m \text{ steps} \\ &\Rightarrow \vdash^{\overline{m \text{ symbols}}} \phi. \end{aligned}$$

The third implication is the non-trivial one and is the hardest part of the proof. We shall have to show, that given a Turing machine M, there is a first-order formula ψ and a constant c such that ψ is valid if and only if M accepts (on blank input) and such that, if M accepts in n steps, then ψ has a proof of length $\leq c \cdot n$. From the formula ψ, the desired self-referential formulas ϕ will be constructed by a diagonalization process and letting $\epsilon \approx 1/c$.

We now embark on the construction of ψ. Without loss of generality, we shall prove Theorem 12 under the assumption that the alphabet A is $\{0, 1, \$, \flat\}$; by standard techniques, our results hold for other alphabets. The symbol $\flat$ is the blank symbol and the symbol $\$$ will be constrained to be used a special 'block delimiter'. Namely, all k of M's tapes are presumed to be initialized to contain blocks of constant length ℓ, marked by having $\$$'s in every $(\ell+1)$st tape square. Between $\$$'s, the symbols 0, 1 and $\flat$ may be written; initially all blocks have only $\flat$'s written between the $\$$'s. (Later we will also allow an input tape which has the input string broken into blocks of size ℓ.) We say that M is in a $\$$-*configuration* if all of its tape heads are over $\$$'s. M will always begin executing in a $\$$-configuration. The execution of M from one $\$$-configuration to the next $\$$-configuration is called a *block step* of M. M is required to run so that each block step satisfies:

(1) Each block step of M takes between $\ell+1$ and $4(\ell+1)$ ordinary steps,

(2) During a block step, each tape head stays within $\ell+1$ tape squares of the $\$$ where it was at the beginning of the block step.

The idea is that M will operate so that, in a single block step, a given tape head must stay in the two blocks immediately adjacent to its starting position — these adjacent blocks are called the *current blocks*. Tape heads can switch to be centered at different $\$$'s only at block step boundaries. For example, a possible configuration of current blocks of four consecutive block steps can be pictured as in Figure 1 below. In Figure 1, the tape head twice moved right one block and then moved left one block.

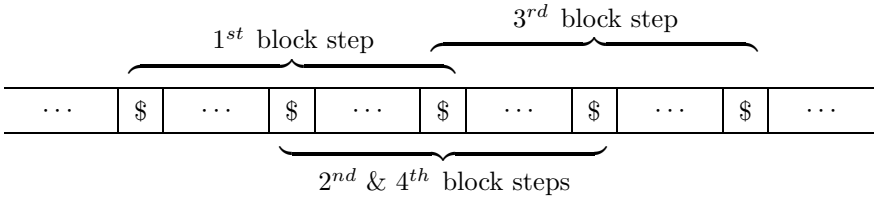

Figure 1

Clearly, for a computation of M beginning and ending in \$-configurations, we have that

$$\begin{aligned}(\text{number of steps}) &\geq (\ell+1)(\text{number of block steps})\\ &\geq \frac{1}{4}(\text{number of steps})\end{aligned}$$

We can assume without loss of generality that M respects block boundaries in the above fashion, since it is easy to see that an arbitrary Turing machine can be modified to respect block boundaries with only a linear change in runtime (see [9] for a different version of block-respecting Turing machines).

Given a fixed Turing machine M which respects block boundaries, with alphabet A and k tapes, we shall construct first-order formulas describing the execution of M. We choose the language of first-order logic to contain ϵ as a constant symbol, to contain four unary functions f_0, f_1, $f_\$$, $f_\flat$ and to contain a lot of relation symbols which are introduced below. The constant symbol ϵ is used to represent the empty string. A string $w \in A^*$ is represented by a term τ_w which is defined by letting τ_ϵ be the constant ϵ and letting $\tau_{w\alpha}$ be $f_\alpha(\tau_w)$, for $\alpha \in A$. For example, the string $\$1\flat\$$ is represented by the term $f_\$(f_\flat(f_1(f_\$(\epsilon))))$.

A block step of an execution of M is indexed by a *time* t, which is an integer ≥ 0 coded by the string in $1\{0,1\}^* \cup \{\epsilon\}$ which is the binary representation of t. The first block step is indexed by the string 1, the second by 10, etc. The time ϵ refers to an imaginary block step occurring just before the execution of the Turing machine. In first-order logic, the term τ_t will index block step t.

At a given block step t of the exection of M, there are k pairs of *current blocks* which are accessible to the tape heads. Each pair of current blocks contains a string of the form $w = \$a\$b\$$ where $a, b \in \{0,1,\flat\}^\ell$; of course, this is represented by the term τ_w. To specify the actions of a block step, we need to know the state of M and the contents of its current blocks at the beginning of the block step. To specify the exit condition of a block step, we need to know the state and current block contents at the end of the block step and we need to know, for each tape head, whether it has ended up at the left, right or center \$. Our language of first-order

logic contains the following relation symbols, for $1 \leq i \leq k$, which describe these facts about block steps:

$\mathit{BeforeContents}_i(t, w)$ - At the beginning of block step t, the i-th tape contains w in its current blocks

$\mathit{Contents}_i(t, w)$ - At the end of block step t, the i-th tape contains w in its current blocks

$\mathit{ExitLeft}_i(t)$ - At the end of block step t, the i-th tape head is at the leftmost \$ of its current blocks

$\mathit{ExitRight}_i(t)$ - At the end of block step t, the i-th tape head is at the rightmost \$ of its current blocks

$\mathit{ExitMiddle}_i(t)$ - At the end of block step t, the i-th tape head is at the middle \$ of its current blocks

$\mathit{BeforeState}(t)$ - In state q at the beginning of block step t

$\mathit{State}(t)$ - In state q at the end of block step t

$\mathit{Left}_i(t_1, t_2)$ - On tape i, the current blocks of block step t_1 overlap and are to the left of those of block step t_2, and, letting $t = \min\{t_1, t_2\}$, the current blocks of block step t were not current at any block step between t_1 and t_2.

The intuitive meanings of these predicates are self-explanatory with the exception of the predicates Left_i. The point of these latter predicates is to keep track of the 'topology' of the Turing machine tapes. An unusual aspect of our encoding of the execution of the Turing machine is that there is no explicit mention of the tape head position at time t. That is to say, tape blocks are indexed only by the times at which they are accessible, and there is no predicate which specifies the tape head position at the beginning and end of each block step. Instead, the Left_i predicates suffice to keep track of the connectedness of the tapes. For example, suppose the first tape head moves according to Figure 1; then the following predicates would be true:

$$\mathit{Left}_1(\tau_1, \tau_{10}) \qquad \mathit{Left}(\tau_{100}, \tau_{11})$$
$$\mathit{Left}_1(\tau_{10}, \tau_{11}) \qquad \mathit{Left}(\tau_1, \tau_{100})$$

Note that the terms $\tau_1, \ldots, \tau_{100}$ refer to the first through fourth block steps. For technical reasons, we require also that $\mathit{Left}_i(t, \tau_\epsilon)$ and $\mathit{Left}_i(\tau_\epsilon, t)$ hold whenever the current blocks of block step t is the leftmost or rightmost (respectively) blocks accessed on the i-tape so far. Thus, the predicates $\mathit{Left}_1(\tau_\epsilon, \tau_\epsilon)$, $\mathit{Left}_1(\tau_1, \tau_\epsilon)$, $\mathit{Left}_1(\tau_\epsilon, \tau_1)$, $\mathit{Left}_1(\tau_{10}, \tau_\epsilon)$ and $\mathit{Left}_1(\tau_{11}, \tau_\epsilon)$ all hold in the example of Figure 1. The idea here is that the block accessed at time 0 (i.e., τ_ϵ) is the generic blank block.

Definition The *execution description* (ED) of an execution of M is defined to be the set of all atomic formulas $\mathit{BeforeContents}_i(\cdots)$, $\mathit{Contents}_i(\cdots)$,

$Left_i(\cdots)$, $BeforeState_q(\cdots)$, $State_q(\cdots)$, $ExitLeft_i(\cdots)$, $ExitMiddle_i(\cdots)$ and $ExitRight_i(\cdots)$ that are true of the execution of M.

Lemma 13 *Given an execution of M of n steps, then its ED contains $O(n + \frac{n}{\ell}\log\frac{n}{\ell})$ symbols.*

Proof Since M executes n steps, it uses $\leq n/(\ell+1)$ block steps. Thus the ED contains $O(n/\ell)$ formulas. Each time t indexing a block step is $\leq n/(\ell+1)$ and is represented by a term of $O(\log(n/\ell))$ symbols. The block size is ℓ and thus a term representing the contents of a block requires $O(\ell)$ symbols. Thus each formula in the ED has $O(\ell + \log(n/\ell))$ symbols. Overall, we have seen that the ED has $O(n/\ell)$ formulas, each having $O(\ell + \log(n/\ell))$ symbols, from whence the lemma follows. □

We henceforth fix the blocking to be $\ell = \delta \cdot \log n$ where n is the number of steps in an execution of M and where δ will be a constant depending only on the cardinality of the alphabet A and the number k of tapes. Then $(n/\ell)\log(n/\ell) = O(n)$ so we have:

Corollary 14 *Under the above conditions, the number of symbols in an ED of M of n steps is $O(n)$.*

Recall that our proof of Theorem 12 will involve constructing a first-order sentence saying that M does not accept in n steps and, if so, the sentence should be provable in $O(n)$ symbols. The proof of the first-order sentence will proceed by proving all the formulas in the ED of n steps of M's execution. The corollary states that this ED has $O(n)$ symbols; we now must establish that all the ED formulas can be *proved* with only $O(n)$ symbols. The natural way to derive the ED formulas is to start with a library of first-order sentences, called *Block Step Formulas*, or BSF's for short, which describe the action of single block step of M. One such family of BSF's is:

$$BSF_1: \qquad \forall t\Big[\bigwedge_{i=1}^{k} BeforeContents_i(t,\tau_{v_i}) \wedge BeforeState_q(t) \rightarrow \bigwedge_{i=1}^{k} Contents_i(t,\tau_{w_i}) \wedge ExitX_i(t) \wedge State_{q'}(t)\Big]$$

for all choices of states q, q', all choices of strings v_i, w_i of length $2\ell+3$ and all choices of X_i being $Left_i$, $Right_i$ or $Middle_i$; such that, if M is in state q having $v_1,\ldots,v_k$ as current block contents at the beginning of a block step, then at the end of the block step, M is in state q', M has $w_1,\ldots,w_k$ as the contents of the same blocks, and M has its tape head positions given by $X_1,\ldots,X_k$.

Further BSF's, regarding movement of the tape heads are:

$$BSF_2 \qquad \forall t_1,t_2,v\Big[ExitMiddle_i(t_1) \wedge Contents_i(t_1,v) \wedge Succ(t_1,t_2)$$

$$\rightarrow \mathit{BeforeContents}_i(t_2, v)\Big]$$

BSF_3 $$\forall t_1, t_2, t_3\Big[\mathit{ExitMiddle}_i(t_1) \wedge \mathit{Succ}(t_1, t_2) \wedge \mathit{Left}_i(t_3, t_1) \rightarrow \mathit{Left}_i(t_3, t_2)\Big]$$

BSF_4 $$\forall t_1, t_2, t_3\Big[\mathit{ExitMiddle}_i(t_1) \wedge \mathit{Succ}(t_1, t_2) \wedge \mathit{Left}_i(t_1, t_3) \rightarrow \mathit{Left}_i(t_2, t_3)\Big]$$

BSF_5 $$\forall t_1, t_2, t_3\Big[\mathit{ExitLeft}_i(t_1) \wedge \mathit{Contents}_i(t_1, \tau_u) \wedge \mathit{Left}_i(t_2, t_1) \wedge \mathit{Less}(t_2, t_1) \wedge \mathit{Contents}_i(t_2, \tau_{u'}) \wedge \mathit{Succ}(t_1, t_3) \rightarrow \mathit{BeforeContents}_i(t_3, \tau_v)\Big]$$

BSF_6 $$\forall t_1, t_2, t_3, t_4\Big[\mathit{ExitLeft}_i(t_1) \wedge \mathit{Left}_i(t_2, t_1) \wedge \mathit{Left}_i(t_3, t_2) \wedge \mathit{Less}(t_2, t_1) \wedge \mathit{Succ}(t_1, t_4) \rightarrow \mathit{Left}_i(t_3, t_4)\Big]$$

BSF_7 $$\forall t_1, t_2\Big[\mathit{ExitLeft}_i(t_1) \wedge \mathit{Succ}(t_1, t_2) \rightarrow \mathit{Left}_i(t_2, t_1)\Big]$$

BSF_8 $$\forall t_1, t_2, t_3\Big[\mathit{ExitRight}_i(t_1) \wedge \mathit{Contents}_i(t_1, \tau_u) \wedge \mathit{Left}_i(t_1, t_2) \wedge \mathit{Less}(t_2, t_1) \wedge \mathit{Contents}_i(t_2, \tau_{u'}) \wedge \mathit{Succ}(t_1, t_3) \mathit{BeforeContents}_i(t_3, \tau_v)\Big]$$

BSF_9 $$\forall t_1, t_2, t_3, t_4\Big[\mathit{ExitRight}_i(t_1) \wedge \mathit{Left}_i(t_1, t_2) \wedge \mathit{Left}_i(t_2, t_3) \wedge \mathit{Less}(t_2, t_1) \wedge \mathit{Succ}(t_1, t_4) \rightarrow \mathit{Left}_i(t_4, t_3)\Big]$$

BSF_{10} $$\forall t_1, t_2\Big[\mathit{ExitRight}_i(t_1) \wedge \mathit{Succ}(t_1, t_2) \rightarrow \mathit{Left}_i(t_1, t_2)\Big]$$

The first three formulas concern tape head (non)movement and tape contents when *ExitMiddle* holds. The second three cases concern the case when *ExitLeft* holds; the final three apply when *ExitRight* holds. In all of the above definitions, i is allowed to range from 1 to k; i.e., each BSF formula above represents a finite set of formulas. The predicates $\mathit{Succ}(t, t')$ and $\mathit{Less}(t, t')$ are to be true exactly when $t + 1 = t'$ and $t < t'$; we shall later give axioms defining these predicates. In the formulas BSF_5, u, u' and v are strings of $2\ell + 3$ symbols of the forms $u = \$x\$y\$$, $u' = \$x'\$y'\$$, and $v = \$x'\$x\$$ with $x, x', y, y' \in \{0, 1, \flat\}^\ell$. Equation BSF_5 represents the fact that if u codes the contents of the current blocks at the end of block step t_1 and if the tape head exits left to be centered at a pair of blocks

such that the last block step, t_2, to be centered there had contents u', then v is the initial contents of block step $t_1 + 1$. Note that there are $k \cdot 3^{4\ell}$ many BSF_5 axioms, since i, u, u' may vary and v is determined by u and u'. Axioms BSF_6 and BSF_7 describe how the *Left* predicate is defined in an *ExitLeft* step. Axioms BSF_8-BSF_{10} describe an block step which exits right; in BSF_8, we must have $u = \$x\$y\$$, $u' = \$x'\$y'\$$, and $v = \$y\$y'\$$.

Let examine how many symbols there are in all the BSF formulas. Since the time t are all universally quantified, the number of symbols depends on M and the block size ℓ (and thus depends only indirectly on runtime). Since the strings u, u', v all have length $O(\ell)$, each BSF formula has $O(\ell)$ symbols. Since there are at most $3^{2\ell}$ choices for the current block contents v_i of tape i, there are $O(3^{2\ell k})$ many BSF_1 formulas. Similarly, there are $O(k3^{4\ell})$ many BSF_5 and BSF_8 formulas. Letting $k' = \max\{2, k\}$, we have that there are $O(k'3^{2\ell k'})$ many BSF formulas, each of size $O(\ell)$. Recall that $\ell = \delta \cdot \log n$. Taking the constant δ to be $< (2k' \log 3)^{-1}$, we have that the total number of symbols in all the BSF formulas is

$$\begin{aligned} O(3^{2\delta k' \log n} \cdot k'\delta \log n) &= O(n^{2\delta k' \log 3} \cdot k'\delta \log n) \\ &= o(n). \end{aligned}$$

It should be noted that the BSF formulas are readily expressed as universally quantified Horn clauses. This is because they are all of the form $\forall \vec{x}(A \to B)$ where A and B are conjunctions of atomic formulas. (We shall henceforth refer to universally quantified Horn clauses as *Horn sentences*.) Furthermore, the BSF formulas, together with a formula $INIT$ describing the configuration of M at the beginning of block step 1 will be sufficient to imply an entire ED of M. In view of Corollary 14 and the fact that the BSF's are Horn sentences, it is easy to see that starting with the BSF formulas and $INIT$ as hypotheses, all the formulas in an ED of n steps of M can be derived in $O(n)$ symbols.

This makes it tempting to use a formula of the form

$$BSF \wedge INIT \to \exists t State_{Accept}(t) \tag{1}$$

as the formula ψ which says that M accepts (where *Accept* is the accepting state of M). If M does accept in n steps, then it is possible to derive $\exists t State_{Accept}(t)$ in $O(n)$ symbols from the BSF and $INIT$ formulas given as *axioms*. From this fact, the deduction theorem says that the implication (1) must be provable; however, the size of the proof may be $O(n^2)$. This is because the deduction theorem gives only a quadratic upper bound on the size of a proof of B from A in terms of the size of a proof of $A \to B$.[8]

[8] See Bonet [1] for more discussion of this. The question of whether a there is a subquadratic upper bound on proof size for the deduction theorem is stated as an open problem in [4].

Since it doesn't work to use the formula (1) with the BSF formulas as hypotheses, we instead arrange for first-order order proofs of the BSF formulas. For this, we need auxiliary predicates with more Horn sentences as axioms:

- $Succ(t, t')$ - means $t' = t + 1$.

  (S-1) $Succ(\epsilon, f_1(\epsilon))$.
  (S-2) $\forall w(Succ(f_0(w), f_1(w)))$.
  (S-3) $\forall w_1, w_2(Succ(w_1, w_2) \rightarrow Succ(f_1(w_1), f_0(w_2))$.

- $Less(t, t')$ - t is less than t'.

  (L-1) $Less(\epsilon, f_1(\epsilon))$
  (L-2) $\forall w(Less(\epsilon, w) \rightarrow Less(\epsilon, f_a(w)))$, for $a \in \{0, 1\}$.
  (L-3) $\forall w_1, w_2(Less(w_1, w_2) \rightarrow Less(f_a(w_1), f_b(w_2)))$, for $a, b \in \{0, 1\}$.
  (L-4) $\forall w(Less(f_0(w), f_1(w))$.

- $Concat(x, y, z)$ - $z = x^\frown y$, the concatenation of x and y.

  (C-1) $\forall x Concat(x, \epsilon, x)$.
  (C-2) $\forall x, y, z(Concat(x, y, z) \rightarrow Concat(x, f_a(y), f_a(z)))$,
  for $a \in \{0, 1, \flat, \$\}$.

- $EqLen$ - $|x| = |y|$, x and y are equal length.

  (E-1) $EqLen(\epsilon, \epsilon)$.
  (E-2) $\forall w_1, w_2(EqLen(w_1, w_2) \rightarrow EqLen(f_a(w_1), f_b(w_2)))$,
  for $a, b \in \{0, 1, \flat, \$\}$.

- $Reverse(x, y)$ - $x = Rev(y)$, the reversal of y.

  (R-1) $Reverse(\epsilon, \epsilon)$.
  (R-2) $\forall w_1, w_2, w_3(Reverse(w_1, w_2) \wedge Concat(f_a(\epsilon), w_1, w_3) \rightarrow Reverse(w_3, f_a(w_1)))$,

  for $a \in \{0, 1, \flat, \$\}$.

We shall frequently write the function symbols $Rev(y)$ and $x^\frown y$ in first-order formulas; these are to be interpreted as abbreviations for more complicated formulas that use the predicates $Reverse$ and $Concat$. For example, a formula $\forall \vec{z} A(Rev(x))$ abbreviates $\forall y, \vec{z}(Reverse(x, y) \rightarrow A(y))$; this preserves the property of being (equivalent to) a Horn sentence. A formula $\exists \vec{z} A(Rev(x))$ abbreviates $\exists y, \vec{z}(Reverse(x, y) \wedge A(y))$.

Finally, our first-order language has a $(4k + 2)$-place predicate $Yields(\vec{u}, q, \vec{x}, q')$ where $\vec{u}$ and $\vec{x}$ are vectors of $2k$ strings and where q and q' are (names of) states of M. If $\vec{u}$ is the vector $u_1^L, u_1^R, \ldots, u_k^L, u_k^R$,

we write $\mathit{Config}(\vec{u}, q)$ to denote the configuration of M in state q with the strings u_i^L and u_i^R giving contents of the current blocks of M's i-th tape; namely, M's i-th tape has u_i^L written to the left of its tape head and has $Rev(u_i^R)$ written to the right of its tape head, with the tape head reading the last symbol of u_i^R. The intended interpretation of $Yields(\vec{u}, q, \vec{x}, q')$ is that M can reach $\mathit{Config}(\vec{x}, q')$ from $\mathit{Config}(\vec{u}, q)$. It is understood that M's tape heads are not to move past the ends of the tape contents specified by $\vec{u}$; hence it will always be the case that $|u_i^L| + |u_i^R| = |x_i^L| + |x_i^R|$. The axioms for $Yields$ are as follows (Config is not a predicate symbol of our first-order logic):

(Y-1) For each instruction of M, there is an axiom for $Yields$ expressing a step of M according to that instruction. This is best illustrated by an example: an instruction *"if in state q_1, reading a '1', then write '0' and go to state q_2'"* would correspond to the axiom:

$$\forall u_1^L, u_1^R (Yields(u_1^L, u_1^R{}^\frown 1, q_1, u_1^L, u_1^R{}^\frown 0, q_2)).$$

(Y-2) Reflexivity and Transitivity of $Yields$:

$\forall \vec{u}(Yields(\vec{u}, q, \vec{u}, q))$, for q a state.

$\forall \vec{u}, \vec{v}, \vec{w}(Yields(\vec{u}, q, \vec{v}, q') \wedge Yields(\vec{v}, q', \vec{w}, q'') \rightarrow Yields(\vec{u}, q, \vec{w}, q''))$,

for q, q', q'' states.

(Y-3) $Yields$ implies BSF's: (for q, q' states)

$$\begin{aligned}
\forall \vec{u}, \vec{x}, t, t' \Big[& Yields(\vec{u}, q, \vec{x}, q') \wedge \mathit{BeforeState}_q(t) \wedge \\
& \bigwedge_{i=1}^{k} \mathit{BeforeContents}_i(t, u_i^L{}^\frown Rev(u_i^R)) \\
& \wedge \big(x_i^L = \epsilon \vee \mathit{EqLen}(x_i^L{}^\frown \$, x_i^R) \vee x_i^R = \$\big) \\
& \rightarrow \mathit{State}_{q'}(t) \wedge \bigwedge_{i=1}^{k} \mathit{Contents}_i(t, x_i^L{}^\frown Rev(x_i^R)) \\
& \quad \wedge \bigwedge_{i=1}^{k} \Big\{ \big(x_i^L = \epsilon \rightarrow \mathit{ExitLeft}_i(t)\big) \\
& \qquad \wedge \big(x_i^R = \$ \rightarrow \mathit{ExitRight}_i(t)\big) \\
& \qquad \wedge \big(\mathit{EqLen}(x_i^L{}^\frown \$, x_i^R) \rightarrow \mathit{ExitMiddle}_i(t)\big) \Big\} \Big]
\end{aligned}$$

Axioms (Y-1)-(Y-3) allow the formulas BSF_1 to be proved instead of taken as axioms. Similarly the following axioms imply the formulas BSF_5 and BSF_8 concerning the movement of the tapeheads: (for $1 \leq i \leq k$)

$$BSF'_5 \qquad \forall t_1, t_2, t_3, x, y, z, x', y', z', u, u', v$$
$$\Big[u = \$^\frown x^\frown \$^\frown y \$ \wedge u' = \$^\frown x'^\frown \$^\frown y'^\frown \$ \wedge v = \$^\frown x'^\frown \$^\frown x^\frown \$$$
$$\wedge \mathit{ExitLeft}_i(t_1) \wedge \mathit{Contents}_i(t_1, u) \wedge \mathit{Left}_i(t_2, t_1)$$
$$\wedge \mathit{Less}(t_2, t_1) \wedge \mathit{Contents}_i(t_2, u') \wedge \mathit{Succ}(t_1, t_3)$$
$$\rightarrow \mathit{BeforeContents}_i(t_3, v)\Big]$$

$$BSF'_8 \qquad \forall t_1, t_2, t_3, x, y, z, x', y', z', u, u', v$$
$$\Big[u = \$^\frown x^\frown \$^\frown y^\frown \$ \wedge u' = \$^\frown x'^\frown \$^\frown y'^\frown \$ \wedge v = \$^\frown y^\frown \$^\frown y'^\frown \$$$
$$\wedge \mathit{ExitRight}_i(t_1) \wedge \mathit{Contents}_i(t_1, u) \wedge \mathit{Left}_i(t_1, t_2)$$
$$\wedge \mathit{Less}(t_2, t_1) \wedge \mathit{Contents}_i(t_2, u') \wedge \mathit{Succ}(t_1, t_3)$$
$$\rightarrow \mathit{BeforeContents}_i(t_3, v)\Big]$$

We let Hyp denote the conjunction of all the axioms above, with the exception of BSF_1, BSF_5, and BSF_8. It is easy to see that all instances of BSF_1, BSF_5 and BSF_8 are consequences of Hyp; we need to analyze how many symbols are required to prove these from Hyp. Let's begin by analyzing all instances of $EqLen(x, y)$ that are needed in the proof — these consist of all literals $EqLen(\tau_{\$a\$}, \tau_{\$b\$})$ with $|a| = |b| = \ell$ and $a, b \in \{0, 1, \flat\}^*$. There are $3^{2\ell}$ such literals, each can be straightforwardly derived in $O(\ell^2)$ symbols. Since

$$\ell^2 \cdot 3^{2\ell} = \delta^2 \log^2 n \cdot 3^{2\delta \log n} < \left(\frac{\log n}{2k' \log 3}\right)^2 n^{1/k' \log 3} = o(n)$$

they require $o(n)$ symbols to derive. Likewise all necessary instances of $Reverse(x, y)$ can be derived in $o(n)$ symbols.

To prove all instances of (Y-1), note that there are $3^{2\ell}(2\ell + 3)$ choices for the pair u_1^L and u_1^R. This must be raised to the power k since there are k tapes. Since $\ell = \delta \log n$ with $\delta < 1/2k' \log 3$, these all require $o(n)$ symbols. Similar size bounds apply to all requisite instances of (Y-2), note that in the second (Y-2) axiom, $Config(\vec{w}, q'')$ may w.l.o.g. be obtained from $Config(\vec{v}, q')$ in a single Turing machine step.

To derive all needed instances of (Y-3) and BSF_1, BSF_5, and BSF_8 is also straightforward and takes $o(n)$ symbols, by exactly the same calculation as used earlier to bound the total size of the BSF axioms.

Now we consider all instances of the axioms for $Succ$ and $Less$. Here the number of instances needed depends on the number m of block steps. We need to derive $Succ(\underline{i}, \underline{i+1})$ for all $0 \leq i < m$ — this needs $O(m \log m)$ symbols. Also we need to derive many instances of $Less(\underline{i}, \underline{j})$ with $i < j \leq m$. Any particular $Less(\underline{i}, \underline{j})$ can be derived in $O(\log m)$ symbols. Fortunately, we do not need to to derive all of these. Instead, for each fixed block step j, we need $\leq 2k$ many instances of $Less(i, j)$, since we may need up to two per tape. So overall we need $O(2km \log m)$ many symbols; i.e., $O(m \log m)$ since k is fixed. This is satisfactory since $m \log m$ is $O(n)$.

The final axioms for first-order logic concern the initial configuration of M; namely:

(I-1) $BeginBlock(\$\flat\cdots\flat\$\flat\cdots\flat\$)$, with ℓ $\flat$'s per block.

(I-2) $\forall w(BeginBlock(w) \rightarrow Contents_i(\epsilon, w))$, $1 \leq i \leq k$.

(I-3) $Left_i(\epsilon, \epsilon)$, $1 \leq i \leq k$.

(I-4) $ExitMiddle_i(\epsilon)$, $1 \leq i \leq k$.

(I-5) $State_{Init}(\epsilon)$, where $Init$ is M's initial state.

It should be clear that (I-1) through (I-5) together with Hyp are sufficient to derive the full ED of M, under the assumption that M starts with tapes that are blank except for \$'s delimiting blocks of length ℓ.

We are now ready to construct self-referential formulas that say that they are not accepted by a Turing machine M. For convenience, we will assume that M runs for $\leq m$ block steps, and to make the details of the self-referential process simpler, we assume $m = 2^N$ is a power of two. Thus, the block size is $\ell = \lceil \delta N \rceil$, where δ is now fixed to equal $1/p$ where $p = \lceil 2k' \log 3 \rceil$.

Definition Given a fixed Turing machine M' (to be determined momentarily), ϕ_N is the formula

$$Hyp_N \rightarrow \bigvee_{q \neq Accept} State_q(2^N)$$

where Hyp_N is the conjunction of Hyp and (I-1)-(I-5), using the instructions of M' for the axioms of $Yields$ and using $\ell = \lceil \delta N \rceil$ in (I-1). *Accept* is M''s accepting state.

Suppose a Turing machine M is given (which may be arbitrary but must respect the blocking conventions). Define M' to be the machine such that M' has tapes corresponding to the tapes of M plus additional work tapes and such that when M' starts on null input it:

(1) Determines the block size ℓ, and

(2) Writes the formula ϕ_N, where $N = p \cdot \ell$, on the 'input' tape (this must respect the blocking so ϕ_N is actually written with \$'s interspersed every $\ell + 1$ symbols), and

(3) Then simulates M in a step-by-step fashion, accepting if and when M does.

It is easy to see that ϕ_N depends on N in a very simple fashion, as N is used only in axiom (I-1); thus (1) and (2) can be done in time $O(N)$. Hence, M' on null input with block size N/p does in $n + O(N)$ steps, exactly what M on input ϕ_N does in n steps. Furthermore, (1) and (2)

can be done in a *constant* number of block steps, so M' does in $m + O(1)$ block steps, exactly what $M(\phi_N)$ does in m block steps.

We use M'_ℓ to denote M' with block size equal to ℓ.

The point of the construction of ϕ_N is that we have:

Lemma 15 *Suppose $N = p \cdot \ell$. If M'_ℓ does not accept in $\leq 2^N$ block steps, then ϕ_N is provable in $\leq d \cdot \ell \cdot 2^N$ symbols for some constant d (which depends on M').*

The proof of the lemma hinges on the observation that when M'_ℓ does not accept in $\leq 2^N$ block steps, then M'_ℓ will have run for $\leq n = 4(\ell + 1)2^N$ steps. By the above construction, it takes $O(n)$ symbols to derive all of the ED of M'_ℓ for 2^N block steps, which suffices to prove the lemma. One detail to note, is that in our above construction, we counted the number symbols needed to derive the ED from Hyp_N taken as a hypothesis. Now, we can argue by the deduction theorem that, for each formula ψ in the ED, $Hyp_N \rightarrow \psi$ is provable (without any extralogical hypotheses). Since Hyp_N has $O(\log n)$ symbols and the average formula used in the proof of ED has $\Omega(\log n)$ symbols, the overall size of the whole proof of ϕ_N is still $O(n)$. □

Proof of Theorem 12. Suppose M is a Turing machine and $\epsilon > 0$ so that

(a) If $\vdash^{n \text{ symbols}} \phi$, then M accepts ϕ in $\leq \epsilon \cdot n$ steps,

(b) If $\nvdash \phi$, then M on input ϕ does not accept (but might never halt either).

We must give a lower bound on ϵ (in terms of k and $|A|$). Let $N = p \cdot \ell$ and $m = 2^N$. Then

$$\begin{aligned} M'_\ell \text{ accepts} &\Rightarrow M(\phi_N) \text{ accepts} \\ &\Rightarrow \phi_N \text{ is true (since, by (b), } \vdash \phi_N) \\ &\Rightarrow M'_\ell \text{ does not accept in } \leq 2^N \text{ block steps} \end{aligned}$$

In any event, it follows that M'_ℓ does not accept in $\leq 2^N$ block steps. On the other hand,

$$\begin{aligned} &M'_\ell \text{ does not accept in } \leq 2^N \text{ block steps} \\ &\quad\Rightarrow \phi_N \text{ is provable in } \leq d\ell 2^N \text{ symbols (by Lemma 15)} \\ &\quad\Rightarrow M(\phi_N) \text{ accepts in } \leq \epsilon d\ell 2^N \text{ steps (by (a))} \\ &\quad\Rightarrow M(\phi_N) \text{ accepts in } \leq \epsilon d 2^N \text{ block steps} \\ &\quad\Rightarrow M'_\ell \text{ accepts in } \leq \epsilon d 2^N + O(1) \text{ block steps.} \end{aligned}$$

This is a contradiction if $\epsilon < 1/d$. Thus taking $0 < \epsilon < 1/d$, we have that M fails to correctly accept the formulas ϕ_N, for N sufficiently large.
Q.E.D. Theorem 12

4 Nondeterministic time lower bound

In this section we prove a conjecture of S. Cook (made in a letter to H. Wang), giving a lower bound on the run time of a nondeterministic Turing machine for the k-provability problem. For this section, all formulas and formal proofs are in first-order logic.

Theorem 16 *The set* $\{\langle \phi, n\rangle : \vdash^{n \text{ symbols}} \phi\}$ *is not recognizable in nondeterministic time* $o(n/\log n)$. *This is true even if* $|\phi|$ *is constrained to be* $O(\log n)$ *and* n *is coded in binary.*

Proof Suppose, for sake of a contradiction, that a nondeterministic Turing machine M accepts the set $\{\langle \phi, n\rangle : \vdash^{n} \phi\}$ in time $o(n/\log n)$. By Sieferas-Fisher-Meyer [12] there is a nondeterministic Turing machine M_0, running in time $n2^n$, which recognizes a language in $NTIME(n2^n)$ but not in $NTIME(o(n2^n))$.[9] We shall prove that, for any input x to M_0 of length n, there is a first-order sentence ϕ_x of length $O(n)$ such that

$$\begin{aligned} \phi_x \text{ is valid} &\Leftrightarrow M_0 \text{ accepts in } \leq n2^n \text{ steps} \\ &\Leftrightarrow \vdash^{O(n^2 \cdot 2^n)} \phi_x \end{aligned}$$

Furthermore ϕ_x will be easily constructible from x in deterministic time $O(n)$. This will yield a contradiction since we have that

$$\begin{aligned} &M_0(x) \text{ accepts in } \leq n2^n \text{ steps} \\ &\quad\Leftrightarrow \vdash^{O(n^2 2^n)} \phi_x \\ &\quad\Leftrightarrow M(\phi_x) \text{ accepts in } o(\tfrac{n^2 2^n}{\log(n^2 2^n)}) = o(n2^n) \text{ steps} \end{aligned}$$

By construction, if $M_0(x)$ accepts, then it accepts in $\leq n2^n$ steps. Hence the language accepted by M_0 can be recognized in nondeterministic time $o(n2^n)$; this contradicts the choice of M_0.

To complete the proof of Theorem 16 it suffices to show that the desired formula ϕ_x can be constructed. We shall accomplish this by modifying the constructions of section 3 above. Unlike in the proof of Theorem 12 where all function symbols were unary, it seems necessary to have have nonunary functions in the language of first-order logic to prove Theorem 16. For convenience we shall take the language of first-order logic to have the function and relation symbols of section 3, plus some more relation symbols introduced below and plus a binary function H and $(4k)$-ary functions g_σ for each choice of $\sigma = \langle q, q', X_1, \ldots, X_k\rangle$ with q, q' states of M_0 and each $X_i \in \{Left_i, Mid_i, Right_i\}$. Note that there are $|Q|^2 3^k$ many functions g_σ, where Q is the set of states of M_0.[10]

[9] Professor Cook suggested to us the idea of using this result of Seiferas-Fisher-Meyer.

[10] Of course, obvious techniques would allow us to replace all these non-unary function symbols with a single binary function symbol, with only a linear change in the lengths of proofs.

The essential idea of the construction of ϕ_x is that ϕ_x should say that there is a w coding an accepting computation of $M_0(x)$. The problem is how how to make w encode the computation of M_0. The first idea would be to let w consist of $n2^n$ many symbols $w = w_1 \cdots w_{n2^n}$ where the symbol w_i would encode the nondeterministic choice made by $M_0(x)$ in its i-th step, and let ϕ_x be a formula of the form $\exists w_1 \cdots \exists w_{n2^n}(\cdots)$. However, by Theorem 7 of [3], a proof of such a ϕ_x would necessarily require $\Omega(n^2 2^{2n})$ symbols. Intuitively, this is because the $n2^n$ existential quantifiers of ϕ_x have to be handled one at a time; so each of the formulas $\exists w_i \cdots \exists w_{2^n}(\cdots)$ must appear in the proof. These formulas have total combined size $> n^2 2^{2n-1}$. A second idea would be to 'block' the symbols and let w consist of $O(2^n)$ separately quantified variables each of which is to be instantiated by a term encoding $O(n)$ steps of $M_0(x)$. The same reasoning shows that now $\Omega(n2^{2n})$ symbols would be needed in any proof.

Since these two ideas don't work, one is lead to the idea of letting w be a single variable which will be instantiated by a single term encoding the entire execution of M_0. An idea that does not work is to use only unary functions (as in section 3) and if α is a string of symbols coding the $n2^n$ steps of an accepting computation of M_0, proving $\phi_x = \exists w(\cdots w \cdots)$ from $(\cdots \tau_\alpha \cdots)$. The reason that this doesn't work is that, by reasoning similar to the proof of Theorem 7 of [3], some constant fraction of the subterms of τ_α must occur as terms in any proof of $(\cdots \tau_\alpha \cdots)$. And again this is $\Omega(n^2 2^{2n})$ symbols. Instead, the idea that does work to prove Theorem 16 is use a binary function H to allow the encoding of $O(n2^n)$ symbols describing a computation of M_0 with a term of $O(n2^n)$ symbols and with depth $O(n)$. Specifically, we shall do the following:

Definition First-order logic will contain a three place relation symbol ED. The intuitive meaning of $ED(w, t_1, t_2)$ is that w is a term encoding a partial computation of M beginning with block step $t_1 + 1$ and ending with block step t_2. This is accomplished by:
(1) Interpreting

$$ED(g_\sigma(\vec{u}, \vec{v}, \vec{t}', \vec{t}''), t-1, t)$$

where $\sigma = \langle q, q', X_1, \ldots, X_k \rangle$, to mean that at the beginning of block step t, M_0 is in state q with $u_1, \ldots, u_k$ the current block contents and, at the end of block step t, M_0 is in state q' with $v_1, \ldots, v_k$ now the contents of those blocks and with the i-th tape tape exiting to the left, middle or right according to $X_i \in \{Left_i, Mid_i, Right_i\}$. The k-vectors $t' = \langle t'_1, \ldots, t'_k \rangle$ and $t'' = \langle t''_1, \ldots, t''_k \rangle$ indicate that $Left_i(t'_i, t)$ and $Left_i(t, t''_i)$ hold.
And, (2) otherwise, interpreting

$$ED(H(w_1, w_2), t_1, t_2)$$

to mean that $ED(w_1, t_1, \frac{t_1+t_2}{2})$ and $ED(w_2, \frac{t_1+t_2}{2}, t_2)$ both hold.

We shall use this latter interpretation of ED only when t_1 and t_2 are of the forms $r2^s$ and $(r+2)2^s$ and thus $(t_1+t_2)/2$ is $(r+1)2^s$. Accordingly,

our first-order logic will include a predicate symbol $Split(t_1, t_2, t_3)$ with the intended meaning that t_1, t_2, t_3 are of the form $r2^s, (r+1)2^s, (r+2)2^s$. The defining axioms for $Split$ are:

(Sp-1) $Split(\epsilon, 1, 10)$

(Sp-2) $\forall t_2, t_3(Split(\epsilon, t_2, t_3) \rightarrow Split(\epsilon, t_2{}^\frown 0, t_3{}^\frown 0))$

(Sp-3) $\forall t_1, t_2, t_3(Split(t_1, t_2, t_3) \rightarrow Split(t_1{}^\frown 0, t_2{}^\frown 0, t_3{}^\frown 0))$

(Sp-4) $\forall t_1, t_2(Succ(t_1, t_2) \rightarrow Split(t_1{}^\frown 0, t_1{}^\frown 1, t_1{}^\frown 0))$

It is easy to see that these four Horn sentences are sufficient to derive any instance of $Split$. Our first-logic also has a binary predicate $LessEq(t_1, t_2)$ meaning that $t_1 \leq t_2$. For technical reasons, we must first define a predicate $NLZ(t)$ which means that t codes a binary string with no leading zeros; its axioms are:

(NLZ-1) $NLZ(\epsilon)$

(NLZ-2) $\forall t(NLZ(t) \rightarrow NLZ(t^\frown 1))$

(NLZ-3) $\forall t(NLZ(t) \wedge Less(\epsilon, t) \rightarrow NLZ(t^\frown 0))$

$LessEq$'s first two defining axioms are:

(LE-1) $\forall t(NLZ(t) \rightarrow LessEq(t, t))$

(LE-2) $\forall t_1, t_2(NLZ(t_1) \wedge NLZ(t_2) \wedge Less(t_1, t_2) \rightarrow LessEq(t_1, t_2))$

In addition we need some (definitely non-Horn) axioms which allow us to derive the "closed world assumption" for $LessEq$:

(LE-3) $\forall t_0, t_1, t_2, t_3\Big(LessEq(t_1, t_0) \wedge LessEq(t_0, t_3) \wedge Split(t_1, t_2, t_3)$
$\rightarrow LessEq(t_2, t_0) \vee LessEq(t_0, t_2)\Big)$

(LE-4) $\forall t_0, t_1, t_2\Big(Succ(t_1, t_2) \wedge LessEq(t_1, t_0) \wedge LessEq(t_0, t_2)$
$\rightarrow t_0 = t_1 \vee t_0 = t_2\Big)$

(LE-5) $\forall t_1, t_2(LessEq(t_1, t_2) \rightarrow NLZ(t_1) \wedge NLZ(t_2))$

Axiom (LE-4) is the reason $LessEq$ is defined with NLZ to apply only to numbers t in their unique binary representation without leading zeros. The reason we want (LE-3) thru (LE-5) as axioms is so that the following lemma holds:

Lemma 17 *Let $\phi(x)$ be a formula, let $\underline{0}, \ldots, \underline{2^n}$ be the $2^n + 1$ terms coding the integers 0 through 2^n satisfying NLZ. Then, from the axioms (LE-1)-(LE-5), there is a first-order proof of*

$$\bigwedge_{i=0}^{2^n} \phi(\underline{i}) \;\rightarrow\; \forall t(LessEq(t, \underline{2^n}) \rightarrow \phi(t))$$

consisting of $O(2^n)$ lines and of $O(n \cdot 2^n \cdot |\phi|)$ symbols.

Proof Recall that big conjunctions are always balanced. The first-order proof proceeds by a 'divide and conquer' approach, proving all instances of

$$\bigwedge_{i=r2^s}^{(r+a)2^s} \phi(\underline{i}) \;\rightarrow\; \forall t(LessEq(\underline{r2^s}, t) \wedge LessEq(t, \underline{(r+1)2^s}) \rightarrow \phi(t))$$

for all integers $r, s \geq 0$ such that $(r+1)2^s \leq 2^n$. First, all of these with $s = 0$ are proved with the aid of (LE-4). Then, iteratively on s, all these formulas with $s = s' + 1$ are proved from the formulas with $s = s'$ with the aid of (LE-3). The bounds on the number of lines and symbols are easily verified by straightforward calculation. □

We are now ready to give Horn sentences defining the ED predicate:

(ED-1) $\forall t_1, t_2, t_3, w_1, w_2 \Big[ED(H(w_1, w_2), t_1, t_3) \wedge Split(t_1, t_2, t_3)$
$\rightarrow ED(w_1, t_1, t_2) \wedge ED(w_2, t_2, t_3) \Big]$

(ED-2) $\forall \vec{u}, \vec{v}, t_1, t_2, t_3, t_4, \vec{t}', \vec{t}'' \Big[ED(g_\sigma(\vec{u}, \vec{v}, \vec{t}', \vec{t}''), t_1, t_2) \wedge Succ(t_1, t_2)$
$\rightarrow BeforeState_q(t_2) \wedge State_{q'}(t_2) \wedge$
$\bigwedge_{i=1}^{k} \Big\{ BeforeContents_i(u_i) \wedge Contents_i(v_i) \wedge ExitX_i(t_2) \wedge$
$\wedge (LessEq(t'_i, t_i) \wedge LessEq(t_i, t''_i) \;\rightarrow\; Left_i(t'_i, t_i) \wedge$
$Left_i(t_i, t''_i)) \Big\} \Big]$
where $\sigma = \langle q, q', X_1, \ldots, X_k \rangle$ with $X_i \in \{Left_i, Mid_i, Right_i\}$.

For x an input to M_0 of length $|x| = n$, we shall form a formula ϕ_x which states that M_0 does not accept x in $\leq 2^n$ block steps. We can assume w.l.o.g. that M_0 respects block boundaries and block size equal to $\ell = \delta \cdot \log(n2^n)$ with $\delta = 1/p$ with p the integer chosen as in section 3. Strictly speaking, to apply the machinery of section 3, M_0 is supposed to start with all of its tapes blank. So we shall modify M_0 to start with blank tapes (except for block delimiters) and let M_0 operate nondeterministically to guess an input string in its first p block steps. Now instead of asking whether M_0 accepts x, we ask if M_0 has an accepting computation that begins by guessing x in its first p block steps.

The formula ϕ_x will be

$$\exists w[Hyp'_x \wedge ED(w, \underline{0}, \underline{2^n}) \wedge InputIs(x) \wedge State_{Accept}(\underline{2^n}) \wedge$$
$$\forall t(LessEq(\underline{0}, t) \wedge LessEq(t, \underline{2^n}) \rightarrow OK(t))]$$

Here Hyp'_x is the conjunction of the axioms (Sp-1) through (ED-2) and of the axioms (L-1) through (I-5) of section 3, with the exception of axiom (Y-3). $InputIs(x)$ is a formula saying that M_0 nondeterministically guesses x as its 'input'; this is just a conjunction of statements about

$Content_1(t,\cdots)$ for $1 \le t \le p$ and of $ExitRight_1(t)$ for $1 \le t < p$ (since we may assume that x is written from left-to-right on the first tape).

Finally $OK(t)$ is defined as follows: $OK1(t)$ says that before block step t, the current block contents are correctly set and $OK2(t)$ says that a valid computation of M has occurred during block step t.

$$LeftS(u,v,w) \Leftrightarrow$$
$$\exists a,b,c(u = \$^\frown a^\frown \$^\frown b\$ \wedge v = \$^\frown c^\frown \$^\frown d\$ \wedge w = \$^\frown c^\frown \$^\frown a\$)$$

$$RightS(u,v,w) \Leftrightarrow$$
$$\exists a,b,c(u = \$^\frown a^\frown \$^\frown b\$ \wedge v = \$^\frown c^\frown \$^\frown d\$ \wedge w = \$^\frown b^\frown \$^\frown d\$)$$

$$\begin{aligned}
OK1(t_1) \Leftrightarrow\ & \exists \vec{u},\vec{z},\vec{z}',\vec{z}'',\vec{t}_2,\vec{t}',\vec{t}''\Big(Succ(t_2,t_1) \wedge \\
& \bigvee_q \big(State_q(t_2) \wedge BeforeState_q(t_1)\big) \wedge \\
& \bigwedge\!\!\!\bigwedge_{i=1}^{k} \Big\{ Contents_i(t_2,z_i) \wedge BeforeContents_i(t_1,u_i) \wedge \\
& \quad Contents_i(t_i',z_i') \wedge Contents_i(t_i'',z_i'') \wedge \\
& \quad Left_i(t_i',t_2) \wedge Left_i(t_2,t_i'') \wedge Less(t_i',t_2) \wedge Less(t_i'',t_2)\Big\} \wedge \\
& \bigwedge\!\!\!\bigwedge_{i=1}^{k} \Big[\Big\{ ExitMiddle_i(t_2) \wedge u_i = z_i \wedge Left_i(t_i',t_1) \wedge Left_i(t_i,t_i'')\Big\} \\
& \quad \vee \Big\{ ExitLeft_i(t_2) \wedge LeftS(z_i,z_i',u_i) \wedge Left_i(t_1,t_2) \\
& \qquad \wedge \exists t(Left_i(t,t_i') \wedge Left_i(t,t_1))\Big\} \\
& \quad \vee \Big\{ ExitRight_i(t_2) \wedge RightS(z_i,z_i'',u_i) \wedge Left_i(t_2,t_1) \\
& \qquad \wedge \exists t(Left_i(t_i'',t) \wedge Left_i(t_1,t))\Big\}\Big]\Big)
\end{aligned}$$

$$\begin{aligned}
OK2(t) \Leftrightarrow\ & \exists \vec{u},\vec{x} \bigvee_{q,q'} \Big(BeforeState_q(t) \wedge State_{q'}(t) \wedge Yields(\vec{u},q,\vec{x},q') \\
& \quad \wedge \bigwedge\!\!\!\bigwedge_{i=1}^{k} \Big[BeforeContents_i(t,u_i^L{}^\frown Rev(u_i^R)) \wedge \\
& \qquad Contents_i(t,x_i^L{}^\frown Rev(x_i^R)) \wedge \\
& \qquad \Big\{(x_i^L = \epsilon \wedge ExitLeft_i(t)) \\
& \qquad\ \vee (x_i^R = \epsilon \wedge ExitRight_i(t)) \\
& \qquad\ \vee (EqLen(x_i^L{}^\frown \$, x_i^R) \wedge ExitMiddle_i(t))\Big\}\Big]\Big)
\end{aligned}$$

$$OK0 \quad \Leftrightarrow \quad \bigwedge_{i=1}^{k} \Big\{ \exists w(BeginBlock(w) \wedge Contents_i(\epsilon, w)) \Big\} \wedge Left_i(\epsilon, \epsilon)$$

$$OK(t) \quad \Leftrightarrow \quad OK0 \wedge OK1(t) \wedge OK2(t)$$

To complete the proof Theorem 16, we must establish: (a) if ϕ_x is valid then M_0 accepts x and (b) if M_0 accepts x then ϕ_x is provable in $O(n^2 2^n)$ symbols. We omit the detailed proof of (a); essentially, we constructed ϕ_x so that this would hold. However, a formal proof of (a) can be sketched as follows: if ϕ_x is valid, then there is a w witnessing ϕ_x's existential quantifier and this w must encode an accepting computation of M_0. To prove (b), recall that M_0 was chosen to have runtime $n2^n$ so that if M_0 accepts x, then M_0 accepts x within $O(2^n)$ block steps. Let w be a string of length $n2^n$ symbols which encodes a computation of M_0 accepting x with w a balanced term consisting of a tree of applications of H with terms $g_\sigma(\cdots)$ at the leaves. Thus there are (not all disjoint) subterms $w_{r,s}$ of w, with $0 \leq s \leq n$ and $r2^s < 2^n$, so that $w = w_{0,n}$ and so that $w_{r,s} = H(w_{2r,s-1}, w_{2r+1,s-1})$ for $s \geq 1$ and such that each $w_{r,0}$ is of the form $g_\sigma(\cdots)$. From axioms (ED-2), there is a conjunction of formulas, denoted ED^w_{r+1}, so that

$$ED(w_{r,0}, \underline{r}, \underline{r+1}) \rightarrow ED^w_{r+1}.$$

Here ED^w_{r+1} is the conjunction of the literals asserting the values of the *BeforeState*, *State*, *BeforeContents*, *Contents*, *ExitX* and *Left* predicates that w encodes for block step $r+1$.

The formula ϕ_x is of the form $\exists y \psi(y)$. For w the above string, we show that $\psi(y)$ has a proof of $O(n^2 2^n)$ symbols by establishing (α)-(ϵ):

(α) The formula $ED(w, \underline{0}, \underline{2^n}) \rightarrow \bigwedge_{t=1}^{2^n} ED^w_t$ has a proof of $O(n^2 2^n)$ symbols.

A straightforward calculation shows that w has $O(n2^n)$ symbols. Recall conjunctions are always balanced. The proof uses a 'divide-and-conquer' method: with axioms (ED-1), prove

$$ED(w, \underline{0}, \underline{2^n}) \rightarrow \bigwedge_{r=1}^{2^{n-s}} ED(w_{r,s}, \underline{(r-1)2^s}, \underline{r2^s})$$

for $s = n, n-1, \ldots, 2, 1, 0$. Then use axiom (ED-2) 2^n times to prove the desired formula. It is easily checked that this proof can have $O((n2^n)n) = O(n^2 2^n)$ symbols.

(β) For each t, $1 \leq t \leq 2^n$, there are block steps $t_{t,1}, \ldots, t_{t,2k+2}$ such that

$$\bigwedge_{j=1}^{2k_2+2} ED^w_{t_{t,j}} \rightarrow OK(t)$$

has a proof of $O(n)$ symbols.

To prove this, we let $t_{t,1}, \ldots, t_{t,2k+2}$ consist of the block steps t, $t-1$ and the (up to) $2k$ earlier block steps which have current blocks immediately to the left or right of the current blocks of time t. These $2k$ block steps are the t'_i's and t''_i's from (ED-2).

(γ) The formula

$$Hyp'_x \wedge \bigwedge_{i=1}^{2^n} \Big(\bigwedge_{j=1}^{2k+2} ED^w_{t_{i,j}} \Big) \rightarrow \forall t(LessEq(\underline{1}, t) \wedge LessEq(t, \underline{2^n}) \rightarrow OK(t))$$

has a proof of $O(n^2 2^n)$ symbols.

To prove (γ), use (β) and Lemma 17.

(δ) The formula

$$\bigwedge_{i=1}^{2^n} ED^w_i \rightarrow \bigwedge_{i=1}^{2^n} \Big(\bigwedge_{j=1}^{2k+2} ED^w_{t_{i,j}} \Big)$$

has a proof of $O(n^2 2^n)$ symbols.

This fact is a corollary of Theorem 12 of Bonet [1].

(ϵ) The formula

$$Hyp'_x \wedge ED(w, \underline{0}, \underline{2^n}) \rightarrow \forall t(LessEq(\underline{1}, t) \wedge LessEq(t, \underline{2^n}) \rightarrow OK(t))$$

has a proof of $O(n^2 2^n)$ symbols.

This follows directly from (α)-(δ).

From (ϵ), it is not difficult to see that $\psi(y)$ itself has a proof of $O(n^2 2^n)$ symbols. This is because $InputIs(x)$ is easily proved from $ED^w_1, \ldots, ED^w_p$ and $State_{Accept}(\underline{2^n})$ is part of $ED^w_{2^n}$; now ϕ_x follows by a single existential quantification from $\psi(w)$.

That completes the proof of Theorem 16.

References

[1] M. L. Bonet, *Number of symbols in Frege proofs with and without the deduction rule*, in Arithmetic, Proof Theory and Computational Complexity, P. Clote and J. Krajíček, eds., Oxford University Press, 1993, pp. 61–95.

[2] S. R. Buss, *The undecidability of k-provability*, Annals of Pure and Applied Logic, 53 (1991), pp. 75–102.

[3] ——, *On Gödel's theorems on lengths of proofs I: Number of lines and speedup for arithmetics*, Journal of Symbolic Logic, 59 (1994), pp. 737–756.

[4] P. Clote and J. Krajíček, *Arithmetic, Proof Theory and Computational Complexity*, Oxford University Press, 1993.

[5] S. A. Cook, *Computational complexity of higher type functions*, in Proceedings of the International Congress of Mathematicians, 1990, pp. 55–69.

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

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[8] J. Hartmanis, *Gödel, von Neumann and the P=?NP problem*, Bulletin of the European Association for Theoretical Computer Science (EATCS), 38 (1989), pp. 101–107.

[9] J. Hopcroft, W. Paul, and L. Valiant, *On time versus space*, J. Assoc. Comput. Mach., 24 (1977), pp. 332–337.

[10] S. C. Kleene, *Introduction to Metamathematics*, Wolters-Noordhoff and North-Holland, 1971.

[11] R. A. Reckhow, *On the Lengths of Proofs in the Propositional Calculus*, PhD thesis, Department of Computer Science, University of Toronto, 1976. Technical Report #87.

[12] J. I. Seiferas, M. J. Fisher, and A. R. Meyer, *Separating nondeterministic time complexity classes*, J. Assoc. Comput. Mach., 25 (1978), pp. 146–167.

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

[14] R. Statman, *Complexity of derivations from quantifier-free Horn formulae, mechanical introduction of explicit definitions, and refinement of completeness theorems*, in Logic Colloquium '76, R. Gandy and M. Hyland, eds., Amsterdam, 1977, North-Holland, pp. 505–517.

Department of Mathematics
University of California, San Diego
La Jolla, CA 92093-0112, USA
sbuss@ucsd.edu