The Review of Symbolic Logic
Volume 19, Number 4, December 2026

THE PROOF IS IN THE CHECKING

NICHOLAS J. J. SMITH
The University of Sydney

Abstract. This paper investigates the question: what is a formal logical proof (in general—as opposed to: what is a formal proof in this or that specific proof system)? I begin by setting out a core conception of formal proof and substantiating its designation as 'core'. The heart of this conception is that there must be an effective or mechanical procedure for checking the correctness of a purported proof. I then consider a proposed strengthening of this conception—one that has become standard in the literature on propositional proof complexity—according to which the proof-checking procedure must be able to be carried out in polynomial time. I discuss a variety of arguments in favour of the strengthened conception—and reject them all. My conclusion is that, when it comes to analysing the very idea of formal logical proof, we have no good reason to move to the strengthened conception and should stick with the core conception.

§1. The core conception. What *in general* is a formal proof? It is a straightforward matter to say what is a formal proof in a particular formal proof system: for example, a particular axiomatic (Frege/Hilbert style) proof system for classical first-order logic, or a particular tableaux system for a certain modal logic, or a particular natural deduction system for intuitionistic propositional logic, or a particular sequent calculus for a certain substructural logic, and so on. But what makes all these things formal logical proofs? Or, to look at things from a slightly different angle, what makes all these different systems count as systems of formal logical proof?[1]

Two features of formal proofs are immediately striking. First, a formal proof is always a proof *of* some claim. A proof is never just 'a proof', full stop. It is a proof *that* a certain formula is a tautology, or that a certain argument is valid, or that a certain set of formulas is unsatisfiable, and so on. So one obvious condition on a formal logical proof is that it is a proof of a logical claim as opposed to a claim of some other kind:

Received: August 28, 2025 UTC.
2020 Mathematics Subject Classification: Primary 03A05.
Key words and phrases: proof, formal proof.

[1] For the sake of clarity, it is worth pointing out that the project of this paper—the project of answering the questions just raised—is different from Prawitz's project of 'general proof theory' (which he pursues in numerous publications including Prawitz [32–34, 36]). My concern is with *formal proofs*, which are made up of logical formulas and perhaps other symbols. Prawitz, by contrast, thinks of proofs as abstract objects that are represented by but distinct from formal proofs, in much the same way that propositions are distinct from—and represented or expressed by—sentences. On this sort of view of proofs and their relationship to formal proofs, see also Kreisel [26, 27]. Cf. also Boghossian [5, p. 17].

© The Author(s), 2026. Published by Cambridge University Press on behalf of The Association for Symbolic Logic. This is an Open Access article, distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives licence (https://creativecommons.org/licenses/by-nc-nd/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is unaltered and is properly cited. The written permission of Cambridge University Press must be obtained for commercial re-use or in order to create a derivative work.

doi:10.1017/S1755020326101300

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

for example, of geometry or physics. For purposes of this paper, we need not determine a precise boundary between logical and nonlogical claims, nor even assume that the boundary is precise.

Second, a formal proof is always a proof *in* a certain proof system. A given sequence of well-formed formulas (wffs) is a proof (that, say, the last wff in the sequence is a valid formula in classical first-order logic) in a certain axiomatic proof system—but not in a different system (that has different axioms or different rules of inference). So we are naturally led to the question: what is a formal proof system? According to what I shall call the *core conception* of formal proof—the name will be justified shortly—there are two essential features of a proof system. The first is a specification of (a) the *symbols* that are allowed to appear in proofs and (b) the *format* of proofs in the system—that is, which *arrangements* of the symbols are allowable. I shall (as in Smith [43]) use the term *proof candidate* for any array of the specified symbols in the specified format.[2] The second essential feature of a proof system is a *decision procedure* that takes as input a proof candidate and a claim and returns Yes (this is a proof of that claim) or No (this is not a proof of that claim). It is this second feature—the decision procedure—that is the heart of the matter. Such a decision procedure requires well-defined *inputs* and the first feature of a proof system—the specification of symbols and format—is needed to provide one of these inputs: the proof candidate. The other input to the decision procedure is a logical claim. Each proof system is associated with certain kinds of claim (e.g., this argument of classical propositional logic is valid; or this set of formulas of classical first-order logic is satisfiable; or these two modal formulas are equivalent). These are the kinds of claim of which the proof system can provide proofs. The second input to the decision procedure is a claim of one of these kinds.

A couple of points of clarification about the decision procedure. First, in accordance with standard usage, by 'decision procedure' I mean an *effective* procedure that always (i.e., on all input pairs) terminates after a finite number of steps of computation with the answer Yes or the answer No—and by 'effective procedure' I mean a finite set of instructions, to be followed in a specified order, where each individual instruction is mechanical (no ingenuity or insight is required to carry it out), deterministic (it involves no random devices such as coin tosses) and finitely achievable (only a finite amount of time is required to complete it). Second, we are talking about a procedure for *checking* whether a *given* (as input) proof candidate is indeed a proof of a certain claim (also given as input). We are not talking about a procedure for *finding* or *building* proofs. Some familiar proof systems—for example, tableaux systems for classical propositional logic—come with mechanical procedures for building proofs, while others—for example, axiomatic proof systems—do not: to find an axiomatic proof of a tautology, one has to use ingenuity, insight or trial and error. But all familiar proof systems come (at least implicitly) with a procedure for checking the correctness of a candidate proof (however it was found in the first place)—and it is *this* kind of procedure that, according to the core conception, is central to the very idea of a formal proof system.

According to the core conception, then, a *formal proof* (of claim P, in proof system S) is a proof candidate (of S) that the decision procedure (for S) classifies as a proof (of P). In the remainder of this section I shall substantiate the claim that this is indeed the *core* conception of formal proof in the logical literature.

[2] Cf. 'figure' in Gentzen [14].

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

First, formulations of it can be found in textbooks including—besides Smith [43]—Bostock [6], Hughes and Cresswell [22] and Tennant [44]:

> a proof in the precise sense we are about to explain is often called a *formal* proof). First, there must be a specified set of symbols from which proofs may be constructed. (These will be the symbols of the logical language, together perhaps with some additional symbols, e.g., the lines and crosses in tree proofs.) Any finite array of these proof symbols will then be a candidate for the title of "proof"—it may or may not actually be a proof, but it is at least in the running to be one (unlike something that is not even composed of the right symbols). Second, there must be a decision procedure for determining whether a candidate proof that something is the case really is a proof that that thing is the case. That is, there must be an effective procedure which, given any candidate proof as input, will yield an answer one way or the other (either "yes it is a proof that X" or "no it is not a proof that X") in a finite amount of time. [43, pp. 375–376]
>
> From now on our proofs will be 'formal' proofs, belonging to some specified system of proofs. The system will lay down particular rules of proof, and a proof in that system must use just those rules and no others. It will be a requirement on any system of proof that there is an effective way of telling whether what we have *is* a proof in that system, i.e. a mechanical test to determine whether any particular array of formulae is or is not to count as a proof. It is for this reason that such systems of proof are called 'formal' systems: one does not need any understanding of the formulae involved in order to say what counts as a proof in the system, or to check whether this or that is an example. [6, p. 135]
>
> An *axiomatic basis* for a logical system consists of (a) a specification of the *language* in which the formulae of the system will be expressed—i.e. a list of *primitive symbols*, together with any definitions that may be thought convenient, together with a set of *formation rules* specifying which strings of symbols are to count as wff; (b) a selected set of wff, known as *axioms*; and (c) a set of *transformation rules*, licensing various operations on the axioms, and also (normally) on wff obtained from the axioms by previous applications of the transformation rules. ... An axiomatic basis must be formulated in such a way that we can determine effectively (i) of any arbitrary string of symbols whether or not it is a wff, (ii) of any wff whether or not it is an axiom, and (iii) of any purported application of a transformation rule whether or not it is a genuine application of that rule. We therefore take care that our formulation of formation and transformation rules, and indeed our specification of a system as a whole, can be understood without reference to the interpretation of the symbols [22, pp. 23–24]
>
> In order to demonstrate the validity of an argument one needs a *proof*. A proof of an argument is a detailed representation of valid transition from its premisses, taken as starting points, to its conclusion

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

> as terminus. Within a proof this transition is effected by a series of steps, or 'inferences', each one of which is obviously valid. [¶] A system of proof is a codification of these obviously valid kinds of inference, which we call rules of inference. ... A proof in accordance with these rules must, in order to meet the demands of certainty, satisfy the following conditions.
>
> (i) It must be of finite size.
> (ii) Every one of its steps must be effectively checkable: that is, a machine or some other kind of comprehensively briefed moron must be able to determine of each step, in a finite time, whether it is in accordance with one of the basic rules of inference of the system.
> (iii) We must be able effectively to read off the premisses and conclusion from the proof – otherwise we shall not know what argument's validity the proof establishes. [44, pp. 5–6][3]

Second, the core conception is bound up in the historical origins of the modern notion of formal logical proof: the essential features of formal proofs, according to the core conception, are indeed essential if formal proofs are to serve the purpose for which they were originally introduced. Although, like all major theoretical advances, it has precursors, the idea of formal proof under discussion in this paper essentially goes back to Frege's *Begriffsschrift*:

> No serious attempt was made to reduce mathematical reasoning to Aristotle's syllogisms. The deductions in Euclid's *Elements*, and in other established parts of mathematics, were taken as obviously valid, without this intuition being given formal shape. Nevertheless, Gottfried Leibniz (1646–1716) sought to develop a universal scientific language and logical calculus, which he hoped would, as he wrote in 1679, make it possible to "judge immediately whether propositions presented to us are proved... with the guidance of symbols alone, by a sure and true analytical method." His efforts had only limited practical success. George Boole (1815–64) similarly sought to turn logic into mathematical form, also with the goal of providing a means of verifying the correctness of an argument purely formally, without reference to its subject matter. Boole's algebraic representation of the "connectives" common in reasoning — AND, OR, NOT, and so on — has been of lasting significance. [¶] The most crucial formalization of logic, however, derives from that provided by Gottlob Frege (1848–1925), especially in his 1879 *Begriffsschrift*. [30, p. 8]
>
> The study of Proof Theory is traditionally motivated by the problem of formalizing mathematical proofs; the original formulation of first-order logic by Frege [1879] was the first successful step in this direction. [8, p. 3]

[3] I corrected (what was clearly) a typo in the original.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

Now consider what Frege wrote in the first paragraph of the Preface of *Begriffsschrift*:

> To prevent anything intuitive from penetrating here unnoticed, I had to bend every effort to keep the chain of inferences free of gaps. In attempting to comply with this requirement in the strictest possible way I found the inadequacy of language to be an obstacle; no matter how unwieldy the expressions I was ready to accept, I was less and less able, as the relations became more and more complex, to attain the precision that my purpose required. This deficiency led me to the idea of the present ideography. Its first purpose, therefore, is to provide us with the most reliable test of the validity of a chain of inferences and to point out every presupposition that tries to sneak in unnoticed, so that its origin can be investigated. [10, pp. 5–6]

and consider also his remarks in the Introduction to *Grundgesetze*:

> The proofs themselves contain no words but are carried out entirely in my symbols ... The ideal of a strictly scientific method in mathematics, which I have here attempted to realize, and which might indeed be named after Euclid, I should like to describe as follows. It cannot be demanded that everything be proved, because that is impossible; but we can require that all propositions used without proof be expressly declared as such, so that we can see distinctly what the whole structure rests upon. ... Furthermore, I demand—and in this I go beyond Euclid—that all methods of inference employed be specified in advance; otherwise we cannot be certain of satisfying the first requirement. This ideal I believe I have now essentially attained. ... Because there are no gaps in the chains of inference, every 'axiom', every 'assumption', 'hypothesis', or whatever you wish to call it, upon which a proof is based is brought to light; and in this way we gain a basis upon which to judge the epistemological nature of the law that is proved. Of course the pronouncement is often made that arithmetic is merely a more highly developed logic; yet that remains disputable so long as transitions occur in the proofs that are not made according to acknowledged laws of logic, but seem rather to be based upon something known by intuition. Only if these transitions are split up into logically simple steps can we be persuaded that the root of the matter is logic alone. I have drawn together everything that can facilitate a judgment as to whether the chains of inference are cohesive and the buttresses solid. ... If we find everything in order, then we have accurate knowledge of the grounds upon which each individual theorem is based. [11, pp. 1–3]

Frege's purpose in introducing a system of formal proof is epistemological or foundational. The aim is to be able to see on what exactly a mathematical theorem depends: from what axioms it follows by what rules of inference. (Furthermore, once we see that a theorem follows from clearly acceptable axioms via clearly acceptable rules, we thereby gain justification for believing the theorem.) To this end there must be no gaps in the proof and each step of the proof must be maximally small and

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

simple.[4] Frege repeatedly talks of "the rigorous avoidance of gaps in the chains of inference" [11, p. 7]—or in an alternative translation "strict gaplessness of the chains of inferences" [12, p. x]—and "my gapless procedure" [11, p. 9].[5] The upshot of this is that checking a proof becomes a mechanical matter: one that does not require insight, ingenuity or intuition. If the latter were not the case—if insight, ingenuity or intuition were required to see that the proof *is* a proof—then the former goal would not in fact have been reached: the proof would not be sufficiently gapless. Anything beyond simple mechanical checking that you need to bring to a proof to see that it works—that it is indeed a genuine proof—has to be *built into* the formal proof itself, explicitly, so that checking *it* becomes a *purely* mechanical matter. So in sum: our current idea of formal proof—the idea under discussion in this paper—had its origins in an epistemologico-foundational project, and an essential upshot of a proof's being fit for this purpose is that checking its correctness becomes a purely mechanical matter. Thus the central feature of the core conception—the requirement that there be an effective proof-checking procedure—is embedded right in the historical origin of the idea of formal proof.

Following Frege, the next great figure in the history of formal proof is Hilbert, whose concerns are summed up with admirable concision by Gandy:

> For [Hilbert's] axioms for (various) geometries he had proved consistency by interpreting the axioms in the theory of the reals. But for the reals (and the natural numbers) this method could not be applied. So, starting with [19], he developed his 'Beweistheorie'. The logical apparatus as well as the mathematical subject matter had to be formalized; formal proofs are then finite objects. Metamathematics is the study of these objects, and here Hilbert accepted Kronecker's restrictions to 'finitistic' methods. Operations on formal proofs (such as reduction to a normal form) are to be (in an intuitive sense of the word) recursive; relations between (parts of) proofs are to be decidable; the prime method of metamathematical proof is induction on the construction of formal proofs. [13, pp. 91–92]

Hilbert's motivations are also epistemological or foundational—but they are not exactly the same as Frege's. Frege assumes that his axioms and rules of inference are OK and wants to close all gaps in a piece of reasoning so that we can see conclusively that the conclusion follows from the given axioms by the given rules: nothing else creeps in. But Hilbert does not assume that given axioms are OK: he wants to prove that they are consistent, by proving in finitary mathematics—which he does assume to be OK—that there can be no proof from the axioms of a contradiction.[6] To this end, Hilbert requires that statements about proofs such as 'P is a proof of C' be part of finitary mathematics.[7] But all statements of finitary mathematics are

[4] Cf. Whitehead and Russell [46].

[5] Cf. also Frege's [11, p. 4] critical remarks on Dedekind.

[6] Cf. Hardy [17, p. 14].

[7] Note Kreisel [26, p. 321]: "Hilbert wanted to accept only that part $\mathscr{P}_0$ of mathematical reasoning, of which school mathematics is typical, and without which there could be no science at all ... He called $\mathscr{P}_0$ 'finitist' because he believed that finiteness was essential to its elementary character; but this is in itself a problematic assumption because one works

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

decidable—hence claims of the form '*P* is a proof of *C*' must be decidable.[8] So in sum: for formal proofs to serve their purpose—which (as with Frege) is epistemologico-foundational (although it is not exactly the same such purpose)—claims about formal proofs such as '*P* is a proof of *C*' have to lie within the province of finitary mathematics, and hence must be decidable.[9] Thus the central feature of the core conception—the requirement that there be an effective proof-checking procedure—is embedded right at the core of Hilbert's proof theory.[10]

Third, the core conception of proof is central to many landmark results in modern formal logic concerning what can and cannot be proved. Consider, for example, Gödel's incompleteness theorems. Gödel writes:

---

with *variables* (other names for $\mathscr{P}_0$ reflecting a different philosophical analysis of what is essential to $\mathscr{P}_0$ are (1) combinatorial or (2) concrete). One knows fairly well what $\mathscr{P}_0$ is, but giving a precise definition is the object, not the starting point of research here." Whatever one calls this part of mathematics, and however *exactly* one characterises it, it certainly has one property: "every assertion of $\mathscr{P}_0$ is decidable" (p. 322). Cf. Zach [47, sec. 3]: "When Hilbert, in [20] explicitly introduces the notion of an ideal proposition, and in [21], when he first speaks of *real propositions* in addition to the ideal, he is quite clear that the real part of the theory consists only of decidable, variable-free formulas."

[8] Note Hilbert [20, p. 381]: "the *logical calculus* ... was originally created in an entirely different context, and, accordingly, its signs were initially introduced for purposes of communication only; but we will be consistent in our course if we now divest the logical signs, too, of all meaning, just as we did the mathematical ones, and declare that the formulas of the logical calculus do not mean anything in themselves either, but are ideal propositions. ... contentual inference is replaced by manipulation of signs according to rules." For Frege, we are doing contentual inference and the point is that the steps need to be so small and so precise that they can be mechanically checked, in order to make sure that the inference is free of gaps (in content). For Hilbert we are not doing contentual inference at all. The proof is regarded as an array of signs that can be "given as such to our perceptual intuition" (p. 381) and about which we can have finitary knowledge: "a formalized proof, like a numeral, is a concrete and surveyable object" (p. 383). (Similar formulations can be found in Hilbert [21, pp. 465, 467, and 471].) One might put it this way: for Frege we are reasoning *with* formulas—they are the *vehicle* of enquiry; whereas for Hilbert we are reasoning *about* formulas: they are the *object* of inquiry.

[9] The fact that despite Frege's being a logicist and Hilbert's being a formalist (and criticising Frege on this issue, e.g., in Hilbert [20, p. 376] and [21, pp. 464–465]), both their approaches yield the core conception of proof, demonstrates the robustness of that conception.

[10] Although Hilbert's hopes for a consistency proof for all of mathematics via finitary proof theory were dashed by Gödel's incompleteness theorems and Hilbert's programme was subsequently largely abandoned, the *conception of proof* at the heart of that programme was never threatened. Note, for example, the following characterisation of proofs, written almost 70 years after Gödel [15]: "A proof is a finite structured object; we can also verify it mechanically. If we are looking for a proof, we can search in a space of finite objects. ... logical truth becomes inter-subjectively and effectively checkable when a proof is given. Its overall structure allows direct checking of the relevant syntactic features, instead of infinitely many interpretations. When sufficiently detailed, proofs allow the verification of (the truth of) something being a theorem to become a kind of empirical search and verification in the space of finite combinatorics" [29, p. 121]. Note too that Gentzen's [14] development of natural deduction and sequent calculus arose from Hilbert's programme—and they are now central systems of proof. See also MacKenzie [30] "While Gödel's result was generally taken as showing inherent limitations of the overall formalist program, it did not undermine the notion of proof at the heart of formalism. Quite the opposite." (p. 9) and "the automation of proof has consolidated the triumph of the formal, Hilbertian notion of proof" (p. 24).

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

> the concept of formal system, whose essence it is that reasoning is completely replaced by mechanical operations on formulas. ... for any formal system provability is a predicate of the form $(Ex)x\mathfrak{B}y$, where $\mathfrak{B}$ is primitive recursive [and hence decidable (p.44)]. [16, p. 72]

It is crucial to Gödel's entire line of thought that the relation that holds between a formal proof P and a theorem C when P is a proof of C has the kind of concrete simplicity that Hilbert took it to have[11] —or as Gödel puts it more precisely, that the relation is primitive recursive. But then this relation must be decidable. So again, the central feature of the core conception—the requirement that there be a decision procedure for the question whether X is a proof of Y—is embedded right at the heart of Gödel's incompleteness theorems. To jump from an early example to a recent one, Kripke's [28] derivation of the Church–Turing thesis from Gödel's completeness theorem for first-order logic with identity[12] turns essentially on the assumption that claims of the form 'P is a proof of C' must be decidable.[13]

So the core conception deserves its name: it is found in the textbooks, rooted in the history and embedded in the results of formal logic as we know it.[14]

§2. The strengthened conception. Nevertheless, a strengthened conception of formal proof has emerged in the last 50 or so years—in particular in the literature on propositional proof complexity. The strengthened conception imposes an additional requirement on a formal proof system: not only must there be (as on the core conception) an *effective* or *mechanical* proof-checking procedure, but furthermore this procedure must be *feasible* in the sense of being able to be carried out in *polynomial time*. That is, the running time of the procedure must be bounded above by a polynomial function of the length of its input. Recall that on the core conception, the input to the proof-checking procedure is a proof candidate plus a claim, and the procedure returns Yes (this is a proof of that claim) or No (this is not a proof of that claim) after a finite number of steps of computation. The strengthened conception imposes an upper bound on this finite number—or more precisely on the rate at which this number grows as the length of the input to the procedure increases.[15]

---

[11] Cf. Kreisel [24, p. 154]: "one of the major purposes of considering formal systems (and it is formal systems which Gödel considers) is that a clear combinatorial (geometric) meaning is given to a formula being provable."

[12] Note: not Gödel's incompleteness theorems, which I just discussed.

[13] See, for example, "Granted that the proof relation of such a system is recursive (computable)" (p. 81) and "Granted that the proof predicate of such a system is recursive/computable" (p. 82).

[14] The essence of the core conception—the idea that claims of the form 'P is a proof of C' must be decidable—is also expressed in numerous works about the nature of proof, for example, Kreisel [25, p. 202] and Prawitz [35, p. 98].

[15] The *running time* of the procedure is not a *number* of steps of computation but a *function* from lengths (of input to the procedure) to numbers (of steps of computation: the *maximum* number of steps required by the procedure across *all* inputs of the given length). The extra condition added to the core conception concerns the *rate of growth* of this function: it must be bounded above by some *polynomial* function.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

Here are some statements of the strengthened conception of formal proof:[16]

> What exactly is a propositional proof? Cook and Reckhow were possibly the first to make this and related questions precise. ... we identify a proof system with a polynomial time procedure that checks the correctness of proofs. [4, pp. 43–44][17]
>
> One can consider many different formalizations and it is difficult to find a classification schema which would cover all. There is however one basic property which all formalizations of the concept of a proof must satisfy: it must be computable in polynomial time whether a given sequence is a proof of a given formula. [37, p. 549]
>
> The literature of mathematical logic contains a very wide variety of proof systems. ... we need a general definition of a proof system. ... The definitions are adapted from Cook and Reckhow [9]. ... The class $\mathcal{L}$ of polynomial-time computable functions is a way of making precise the vague notion of "feasibly computable function."[18] ... The crucial property of a proof system ... is that, given an alleged proof, there is a feasible method for checking whether or not it really is a proof, and if so, of what it is a proof. [45, pp. 425–426]
>
> three properties that seem necessary for any method of certifying tautologies: Every tautology has a proof, only tautologies have proofs, and valid proofs are computationally easy to verify.[19] [40, p. 418]

I have explained in §1 why there must be a mechanical proof-checking procedure if formal proofs are to do the work for which they were introduced in the first place and if central results about what can and cannot be proved are to remain valid. So there are good reasons behind the core conception of proof. What about the proposed strengthening? *Why* should we think that the proof-checking procedure must be computable *in polynomial time*? In the remainder of this paper I shall discuss four arguments for the strengthened conception: one in each of §§3–6. Some of the

---

[16] More precisely, these are statements of the idea behind a formal definition of a formal proof system. The actual formal definitions differ somewhat across the literature, although they are all interchangeable in the sense that for any formal proof system in the sense of one definition there is another formal proof system in the sense of any of the other definitions. The original formal definition in Cook and Reckhow [9, p. 37] is as follows. Where Σ is a finite alphabet and $L \subseteq \Sigma^*$ is a language, a *proof system* for L is an onto function $f : \Sigma^* \to L$ that can be computed by a deterministic Turing machine in time bounded by a polynomial in the length of the input. The essential difference here from the core conception is that (assuming the Church–Turing thesis—see n. 18) the latter would merely require f to be computable by a deterministic Turing machine (with no time bound on the computation).

[17] Cf. also Beame [3, p. 202].

[18] The identification of the ordinary notion of a feasibly computable function with the precise notion of a polynomial time computable function has become standard in the computational complexity literature. It is known as the Cobham-Edmonds thesis and is often compared with the Church-Turing thesis in computability theory, which identifies the ordinary notion of an effectively computable function with the precise notion of a Turing computable function (or a provably equivalent precise notion such as that of a recursive function).

[19] 'Easy' is standardly used as a shorthand for 'polynomial time' in the computational complexity literature and it is quite clear from the definition that follows this quotation that Segerlind is using the term in this sense.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

arguments are drawn from the literature and some are of my own construction. They are all the reasonable arguments in favour of the strengthened conception that I can find or devise. I shall reject them all. My conclusion will be that we have no good reason to move beyond the core conception of formal proof to the strengthened conception.

§3. The argument from convenience. Consider an example of a central theorem from the literature on propositional proof complexity—one proven in the paper in which the polynomial-time constraint on proof-checking entered the literature [9]:

> There is a proof system in which every tautology α has a short proof iff $\mathsf{NP} = \mathsf{coNP}$.[20]

This claim would not be true if it were not built into the meaning of 'proof system' that the proof-checking procedure *runs in polynomial time*. That is, the existence of a proof system that yields *short* proofs of all tautologies is only equivalent to the identity of NP and coNP provided that all proofs in the system can be *checked* in polynomial time. (Why this is so—specifically, why the 'only if' direction depends on polynomial checkability—is explained in the Appendix.)

This is not an isolated example: many results in the propositional proof complexity literature are stated in a way that requires for their truth that proofs are always checkable in polynomial time. And this is no surprise, because the standard definition of 'formal proof system' in this literature builds in the requirement of *polynomial time* checkability (not just *effective* checkability, as on the core conception). But our concern now is *why* we should adopt this definition. I can imagine two lines of thought stemming from the observation I have just made about the formulations of results in this literature—but as I shall now explain, neither of them yields a good argument for the conclusion that polynomial-time (as opposed to merely effective) checkability is or should be part of the very idea of what makes something a formal proof system.

First, someone might think that authors in the literature on propositional proof complexity are not in the business of contributing to the project of analysis of the very idea of formal proof. They are simply setting up formal definitions in a way that is convenient and that allows them to prove interesting results. Note that the main preexisting proof systems studied in this literature—Frege–Hilbert style axiomatic proof systems, natural deduction systems, sequent calculi and so on—all in fact meet the requirement that proofs in these systems can be checked in polynomial time. The formal definition of 'proof system' simply abstracts away certain features of these existing proof systems—and there's nothing more to it than that.

If this is true, then for purposes of this paper there is nothing to see here and we can move on. Our interest here is (analysis of) the very idea of formal logical proof. If the definition of 'formal proof system' in the propositional proof complexity literature is not actually intended as a proposal to strengthen the core conception of formal proof but is merely a convenient definition for 'internal use only' within this literature, then it poses no challenge to the core conception and we can simply pass on. However it is

[20] A *short proof* is a proof whose length is bounded by a (fixed, for that proof system) polynomial function of the length of α.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

worth noting that this is not the way the literature actually reads.[21] See, for example, the quotes in §2 and note the following formulations (my emphases): "one *basic* property which *all* formalizations of the concept of a proof *must* satisfy"; "The *crucial* property of a proof system"; "three properties that seem *necessary* for *any* method of certifying tautologies."

Second, someone might think that we should adopt the strengthened definition of proof because it is more convenient: it allows for simpler statements of results. One can, for example, present the result above in the way I did, rather than as follows:

> There is a proof system in which every tautology α has a short proof *that can be checked in polynomial time* iff $\mathsf{NP} = \mathsf{coNP}$.

But this is pretty obviously a terrible argument for altering the very definition of 'formal proof system'. All it warrants is (sticking with the core conception and) adding a rider: "For simplicity, it will henceforth be assumed that all proof systems under discussion have polynomial-time proof-checking procedures." Furthermore, that way of proceeding would be in keeping with the treatment of other notions in the literature on computational complexity. One standardly talks, for example, of *feasibly* (i.e., polynomial-time) computable functions, using 'feasible' as an explicit qualifier rather than building it into the very definition of 'computable'. There is something oddly asymmetric and out of keeping with general usage in treating 'short' (i.e., polynomial *length* bounded) as an explicit qualifier on 'proofs' but building 'feasible' (i.e., polynomial-*time* bounded) into the word 'proof' itself, rather than adding it explicitly—whether on individual occasions, or as an initial overall rider.[22]

---

[21] More subtly—and I am grateful here to an anonymous referee for helpful comments—it is not the way the contributions of at least some authors read. Other authors may be more concerned with specialised technical matters (such as the Cook–Reckhow definition of a propositional proof system or the notion of a feasible propositional certificate) than with the general philosophical thesis that polynomial-time checkability is part of the very idea of formal proof. The extent to which this philosophical view has to be extrapolated from their words, as opposed to being propounded explicitly, and their general level of engagement with philosophical issues may well vary from author to author—but for those of us whose concern *is* the very idea of formal proof simpliciter, many remarks in the propositional proof complexity literature certainly suggest the view that I set out in §2 and labelled the strengthened conception of formal proof. As the referee put it, the present paper "need not show that propositional proof complexity contains a fully worked-out philosophy of proof. It is enough to show that some influential formulations in that literature encourage, or at least invite, such a philosophical reading."

[22] Razborov [38, p. 100] expresses things in a more balanced way (my emphases): "Propositional proof complexity ... plays as important a role in the theory of *feasible* proofs as the role played by the complexity of Boolean circuits in the theory of *efficient* computations." Cf. also Pudlák [37, p. 548]: "Our subject could be called *the quantitative study of the proofs*. In contrast with the classical proof theory we want to know not only whether a theorem has a proof but also whether the proof is feasible, i.e., can be actually written down or checked by a computer." However while Pudlák here talks as if being feasibly checkable is a property that proofs may *or may not* possess, one page later he describes it as a "trivial observation" that "There is however one basic property which all formalizations of the concept of a proof must satisfy: it must be computable in polynomial time whether a given sequence is a proof of a given formula."

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

Before moving on, it is worth discussing a line of thought that connects Gödel's famous letter to von Neumann and the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem and turns on *polynomial-time* proof-checking. Gödel writes:

> Clearly we can construct a Turing machine, which for each formula F of the predicate calculus and every natural number n allows to decide if F has a proof of length n. [Length = number of symbols]. Let $\Psi(F, n)$ denote the number of steps required by the machine, let $\phi(n) = \max_F \Psi(F, n)$. The question is, how fast grows $\phi(n)$ for an optimal machine. One can show that $\phi(n) \geq K \cdot n$. If in fact a machine with $\phi(n) \sim K \cdot n$ (or even with $\sim K \cdot n^2$) existed, it would have very important consequences. It would apparently mean that in spite of the unsolvability of the Entscheidungsproblem, the reasoning of mathematicians about yes-or-no questions can be completely replaced by machines. One should just choose n so large that, if the machine does not yield a result, there is no reason to think about it. [18, p. 6][23]

Aaronson [1, p. 7] writes:

> If we replace the "$\sim K \cdot n$ or $\sim K \cdot n^2$" in Gödel's challenge by $\sim K \cdot n^c$ for an *arbitrary* constant c, then we get precisely what computer science now knows as the P versus NP problem. ... If $\mathsf{P} = \mathsf{NP}$, then the ability to *check* the solutions to puzzles efficiently would imply the ability to *find* solutions efficiently.[24]

I take it that Gödel's line of thought is as follows. To see whether F has a proof of length n, go through all the strings of symbols of length n and for each one, check whether it is a proof of F. This part of the reasoning depends on the fact that proof-checking is a mechanical matter: it is decidable whether or not a certain string is a proof of F. Now suppose the Turing machine that does this checking of strings runs in linear (in n) time (or even quadratic time)—and suppose we want to know whether F. To warrant the answer Yes, we need a proof of F. We ask the machine if there is a proof of length up to some very large n—so large that proofs longer than this are beyond our grasp (i.e., even if there is such a proof, it will never warrant us in believing F, because we cannot even read through the whole proof). If the machine says Yes then we know F is true. If it says No, then the question whether F is *in effect* closed: perhaps there does exist some astronomically long proof, or perhaps there is no proof of F at all, but either way, as far as we are concerned, we are not and cannot ever be warranted in believing or asserting F, and so the question whether F is effectively terminated.

---

[23] The square brackets and the remark they enclose are in the original. I corrected an obvious typo ('Entscheidengsproblem'). For another translation see Sipser [41, p. 612].

[24] Cf. Arora and Barak [2, p. 58]: "If **P** = **NP** ... Mathematicians could be replaced by efficient theorem-discovering programs (a fact pointed out in Kurt Gödel's 1956 letter and recovered two decades later). In general, for every search problem whose answer can be efficiently verified (or has a short certificate of correctness), we will be able to find the correct answer or the short certificate in polynomial time."

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

I take it that Aaronson's line of thought is as follows. Suppose that the 'proves' relation (between proof candidates and formulas) is not merely decidable but polynomially so. Consider the set G of pairs (α, x) of a formula α and a number x (encoded in unary) such that α has a proof of length x or less. This set is in NP: the membership certificate (see the Appendix) for a pair (α, x) is just the proof of α of length at most x—and we have just supposed that this can be checked (by a deterministic Turing machine) in polynomial time. Now suppose also that $\mathsf{P} = \mathsf{NP}$. Then we cannot merely *confirm* in polynomial time (given the proof as a certificate) that $(\alpha, x) \in G$: we can, given only (α, x), *decide* in polynomial time whether $(\alpha, x) \in G$. So if $\mathsf{P} = \mathsf{NP}$ then one can feasibly (i.e., polynomially) determine for any formula α whether or not it has a proof of length up to some n so large that proofs longer than this are beyond our grasp.

Note that Aaronson's line of thought does not require that it be part of the very idea of a proof system that proofs can be checked in polynomial time. All we need to reach the conclusion that we can feasibly determine for any formula α whether or not it has a proof (in system S) of length up to some big n is the supposition that $\mathsf{P} = \mathsf{NP}$ together with the fact that we have *some* proof system S in which proofs can be checked in polynomial time. It would not undermine the argument at all if we did not build polynomial-checkability into the *very idea* of a proof system—that is, if we allowed the possibility of other (legitimately so-called) proof systems in which proof-checking is mechanical but not polynomial.

§4. The argument from intuition. A second possible argument in favour of the strengthened definition of proof is that it is simply intuitively obvious (at least once attention was drawn to the matter by Cook and Reckhow) that a proof must be *easy* to check (in the sense of 'easy' discussed in n. 19). Many of the writers quoted in §2 seem to present the new definition this way: they give no argument for it but write as if it is simply obvious:

> The crucial property of a proof system [45, p. 426]
> one basic property which all formalizations of the concept of a proof must satisfy ... This trivial observation [37, p. 549]
> properties that seem necessary for any method of certifying tautologies [40, p. 418][25]

In response to this (argument from) intuition, I present the following exhibits. First, Hughes and Cresswell [22, pp. 24–25] introduce an axiomatic proof system **K**:

> Its axioms consists of all valid wff of PC [Propositional Calculus], i.e. all the wff specified by the following axiom schema,
> **PC** If α is a valid wff of PC, then α is an axiom

[25] It's worth noting however that some other authors are more half-hearted—e.g., Razborov [38, p. 101]: "Also, it is conceivable that for any reasonable calculus the function P will be polynomial time computable."

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

They then write "We shall first prove two theorems in full detail" [p. 26] and then present the following proof [p. 27]:

PC1	(1)	$(p \wedge q) \supset p$
(1) × N	(2)	$L((p \wedge q) \supset p)$
K	(3)	$L(p \supset q) \supset (Lp \supset Lq)$
(3)$[p \wedge q/p, p/q]$	(4)	$L((p \wedge q) \supset p) \supset (L(p \wedge q) \supset Lp)$
(2), (4) × MP	(5)	$L(p \wedge q) \supset Lp$
PC2	(6)	$(p \wedge q) \supset q$
(6) × N	(7)	$L((p \wedge q) \supset q)$
(3)$[p \wedge q/p]$	(8)	$L((p \wedge q) \supset q) \supset (L(p \wedge q) \supset Lq)$
(7), (8) × MP	(9)	$L(p \wedge q) \supset Lq$
PC3	(10)	$(p \supset q) \supset ((p \supset r) \supset (p \supset (q \wedge r)))$
(10)$[L(p \wedge q)/p, Lp/q, Lq/r]$	(11)	$(L(p \wedge q) \supset Lp) \supset ((L(p \wedge q) \supset Lq) \supset (L(p \wedge q) \supset (Lp \wedge Lq)))$
(5), (11) × MP	(12)	$(L(p \wedge q) \supset Lq) \supset (L(p \wedge q) \supset (Lp \wedge Lq))$
(9), (12) × MP	(13)	$(L(p \wedge q) \supset (Lp \wedge Lq))$ **Q.E.D.**

Note the lines in this proof annotated 'PC1', 'PC2' and 'PC3'. To *check* the proof, you need to check that each of the wffs on these lines really is a tautology of propositional logic. We have no polynomial-time procedure for that—and if $\mathsf{P} \neq \mathsf{NP}$ then there is no such procedure.[26] So according to the strengthened conception of proof, this is not a formal proof. Yet, I submit, it clearly *is* a perfectly good formal proof: Hughes and Cresswell thought it was one, and generations of their readers have accepted this without demur.

Note that Hughes and Cresswell describe this as a proof *in full detail*. They do *not* regard it as a placeholder for a full formal proof, that *could* be spelled out into a formal proof if one wanted to by inserting extra lines before each wff labelled 'PC': extra lines mirroring a proof of that wff in a standard axiomatic proof system for propositional logic. They regard this as a formal proof in itself, as it stands.[27] Furthermore, they say quite explicitly:

> The fact that the axiom schema **PC** gives us infinitely many axioms does not conflict with our requirements for a satisfactory set of axioms, since we have (in the truth-table method, for example), an effective way of determining whether any given wff is a valid wff of PC or not. [22, p. 49, n.2]

[26] For suppose there were a procedure that checks in polynomial time whether α is a tautology. Then the following is a procedure that checks in polynomial time whether α is satisfiable (i.e., that decides SAT): append a negation to α and then check whether $\neg\alpha$ is a tautology; if it is, then α is not satisfiable; if it isn't, then α is satisfiable. But SAT is NP-complete (see the Appendix) so then $\mathsf{NP} \subseteq \mathsf{P}$—which together with the known (and trivial) fact that $\mathsf{P} \subseteq \mathsf{NP}$ implies that $\mathsf{P} = \mathsf{NP}$. Whether or not $\mathsf{P} = \mathsf{NP}$ is "one of the greatest unsolved problems in theoretical computer science and contemporary mathematics" [42, p. 298]. I think it is fair to say however that most theorists working in the field suspect that $\mathsf{P} \neq \mathsf{NP}$.

[27] For further discussion, see the parenthetical remark at the end of n. 36.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

The requirement on a satisfactory set of axioms that they are talking about here is the one already quoted in §1: "An axiomatic basis must be formulated in such a way that we can determine effectively... of any wff whether or not it is an axiom" [p. 23]. So Hughes and Cresswell are quite explicit that the *truth table method* is a perfectly good effective procedure for checking a proof (or more precisely, a perfectly good sub-procedure, that is, *part of* the overall proof-checking procedure: the part that checks whether a given line of a proof in system **K** is an instance of the axiom schema **PC**). Yet it is well known that the truth table procedure runs in exponential time. So Hughes and Cresswell are extremely close here to denying *explicitly* that the proof-checking procedure must run in polynomial time.

In this connection, note Kleene [23]. On p. 224 he says—in accordance with the core conception—that the question "Is a given finite sequence of formulas a proof?" is decidable:

> For a given formal system **S**, consider the following three general questions, i.e. (countably infinite) classes of particular questions: "Is a given formal expression a formula?", "Is a given finite sequence of formulas a proof?", "Is a given formula provable?". ... To answer any given question of the second class, we merely consider each formula in the given sequence in order, and examine whether it is an axiom or follows from earlier formula(s) by one of the rules of inference.

In n. 163 on the next page, he then notes that even when we have a decision procedure for answering a certain question:

> *Practically*, the answer to the question might still be beyond our reach by the means described. For, in applying the given decision procedure ... more space and time than we have available might be required to carry out the finitely many steps that lead to the decision.

He immediately continues:

> For what we do in this book, it is not essential that we concern ourselves with this matter of whether a given decision procedure for a class of questions is or is not practical to use in answering certain questions of the class. (That falls under "computer sciences".)

Kleene is effectively saying here that the core conception is all we need: claims of the form 'P is a proof of C' must be decidable, but it does not matter whether or not the decision procedure is feasible.

Burgess [7] does the same sort of thing as Hughes and Cresswell: when defining the system $\mathbf{L}_0$ of minimal temporal logic, one of his axiom schemas is "A, where A is any tautology" [p. 21]. Furthermore, he also introduces a rule of inference—the *rule of tautological consequence* (Taut)—defined as follows: "from $A_1, \ldots, A_n$ to infer any

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

tautological consequence *B*" [p. 22].[28] He goes on to present the following proof [p. 28]:

i	$\mathsf{GP}A \to \mathsf{GGP}A$	Ax
ii	$A \to \mathsf{GP}A$	Ax
iii	$A \to \mathsf{GGP}A$	i, ii, Taut
iv	$\mathsf{PP}A \to \mathsf{PPGGP}A$	iii, Becker
v	$\mathsf{PGGP}A \to \mathsf{GP}A$	Ax
vi	$\mathsf{PPGGP}A \to \mathsf{PGP}A$	v, Becker
vii	$\mathsf{PGP}A \to \mathsf{P}A$	Ax
viii	$\mathsf{PP}A \to \mathsf{P}A$	iv, vi, vii, Taut

Similar remarks apply to this proof as to the proof of Hughes and Cresswell discussed earlier. To check line iii (and mutatis mutandis for line viii), you need to check that it really follows logically from i and ii. We have no polynomial-time procedure for that—and if $\mathsf{P} \neq \mathsf{NP}$ then there is no such procedure.[29] So according to the strengthened conception of proof, this is not a formal proof.

There are two ways to take the material in this section: as an appeal to your intuitions—look at the proofs from Hughes and Cresswell and from Burgess and agree that they are satisfactory formal proofs—or as an appeal to the authority of the authors discussed (Hughes and Cresswell, Kleene and Burgess). Either way, one clearly cannot maintain that it is just *intuitively obvious* that proofs have to be checkable *in polynomial time*.

Before moving on, I can imagine someone objecting: the strengthened definition of proof comes from the literature on *propositional* proof complexity—but Hughes and Cresswell and Burgess are talking about modal and tense logics, not propositional logic. The latter observation is true—but it is hard to see how it is supposed to help the argument from intuition for the strengthened conception. Is the point supposed to be that the strengthened conception applies to all formal logical proofs—but the intuitions that support it concern only propositional proofs? Setting aside (for now)[30] the issue of whether we really have any such intuitions (i.e., about *propositional* proofs specifically), how could they possibly justify an analysis of proof that rules out as formal proofs things that seem to be perfectly good formal proofs, such as the proofs of Hughes and Cresswell and Burgess? That broad thesis would clearly require broader support. Or is the idea supposed to be that the strengthened analysis applies only to proofs in propositional logic? But then the proponents of the strengthened (but now narrower in scope) analysis have actually given themselves more work to do: now they must explain what is special about *propositional* proofs that makes them subject to more stringent conditions than other logical proofs. The (alleged) existence of intuitions concerning the easy checkability of propositional proofs plus the non-existence of intuitions concerning the easy checkability of non-propositional proofs (vide Hughes

[28] He does not take this to be a primitive rule of inference—but what he says about this rule would be no less acceptable if he had done so.

[29] For suppose there were a procedure that checks in polynomial time whether β follows from $\alpha_1, \ldots, \alpha_n$. Then the following is a procedure that checks in polynomial time whether α is satisfiable: check whether $\beta \wedge \neg\beta$ follows from α; if it does, then α is not satisfiable; if it doesn't, then α is satisfiable. But then (see n. 26) $\mathsf{P} = \mathsf{NP}$.

[30] In n. 36 I'll give an example of a proof system for *propositional* logic that does seem, intuitively, to warrant the title 'proof system', even though its proofs are not checkable in polynomial time.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

and Cresswell and Burgess) does not add up to a compelling reason for a two-sorted analysis of formal proof that adheres to the core conception for non-propositional proofs and the strengthened conception for propositional proofs: such a disjunctive analysis would require more compelling support. But here the prospects seem dim: when we move from formal logical proofs in propositional logic to formal logical proofs in predicate logic or modal logic or any other kind of logic, there seem only to be changes in the details of the languages and proof systems under consideration—not a change in the fundamental nature of the objects (formal proofs) under consideration which requires a different analysis of formal proof once we move outside the realm of classical propositional logic.

§5. The argument from no-self-proof. In introducing (what I am calling) the strengthened conception of formal proof, Beame and Pitassi [4, p. 43] write:

> Since there are only finitely many truth assignments to check, why not allow the statement itself as a proof? What extra value is there in a filled out truth table or a derivation using some axiom/inference scheme? The key observation is that a proof is easy to check, unlike the statement itself.

The final sentence could be read as an expression of an intuition that a proof must be easy to check. That approach was covered in the previous section. But taken as a whole, the passage could be read as offering an argument for the strengthened conception that goes beyond the argument from intuition:

1. If we do not impose the additional requirement of feasibility of proof-checking on formal proofs then we get the result that a formula of propositional logic counts as a proof of itself (of its own tautologousness, or satisfiability, or unsatisfiability, as the case may be).
2. This is the wrong result: intuitively, a formula cannot prove itself or act as its own proof.
3. Therefore we should impose the feasibility requirement.

The argument from intuition argues for the feasibility requirement by appealing directly to its intuitive obviousness. This new argument appeals only to a more focussed intuition: that formulas cannot prove themselves.[31]

My response to this argument will be to show that even if we impose the feasibility requirement, we *still* get a pervasive class of self-proving wffs—so avoiding self-proving formulas is not an adequate motivation of the feasibility requirement.[32]

[31] A proponent of this argument might want to allow certain isolated special cases of self-proof. One could define an axiomatic proof system in which proofs are lists of wffs, each of which is an axiom or follows from earlier wffs on the list by a rule of inference, and identify a one-membered list with its only entry—or instead define a proof to be either a single axiom, or a two-or-more-membered list of wffs where each wff is an axiom or follows from earlier wffs on the list by a rule of inference; either way, an axiom (the wff, by itself) would be a proof of its own tautologousness. Likewise, one could define a tableaux system in which a single literal (the wff, by itself) is a proof of its own satisfiability (i.e., a tree that is finished—there being no tree rules for literals—and has an open path).

[32] The class of self-proving wffs that we shall see below is far too extensive to be admitted via the kind of loophole discussed in n. 31.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

p_3	p_1	p_4	$(p_3 \rightarrow p_1) \leftrightarrow (p_1 \rightarrow p_4)$	$(p_3 \wedge p_1 \wedge p_4) \vee (\neg p_3 \wedge p_1 \wedge p_4) \vee (\neg p_3 \wedge \neg p_1 \wedge p_4) \vee (\neg p_3 \wedge \neg p_1 \wedge \neg p_4)$
1	1	1	1	1
1	1	0	0	0
1	0	1	0	0
1	0	0	0	0
0	1	1	1	1
0	1	0	0	0
0	0	1	1	1
0	0	0	1	1

Figure 1. Truth table and cdnf.

In order to show this, we start with some definitions. Write the basic propositions of the language (of propositional logic) as: $p_1, p_2, p_3, \ldots$.[33] A *literal* is a basic proposition or a negation of a basic proposition. A *conjunction of distinguished literals* (cdl) is a conjunction of one or more literals in which no basic proposition occurs more than once—for example:

$$p_1 \quad \neg p_3 \quad (p_1 \wedge \neg p_2) \quad (\neg p_2 \wedge \neg p_4) \quad (p_7 \wedge p_8 \wedge \neg p_6).$$

The *signature* of a cdl is a sequence of positive integers comprising the subscripts on the basic propositions in the order in which they occur in the cdl. So the signatures of the example cdls above are, respectively:

$$(1) \quad (3) \quad (1,2) \quad (2,4) \quad (7,8,6).$$

Two cdls are *cosigned* if they have the same signature. A set of cdls is cosigned if its members are pairwise cosigned; in this case we also say that the cdls in the set are cosigned. A *cosigned disjunctive normal form* (cdnf) is a disjunction of zero or more *distinct* cosigned cdls—for example:

$$(p_2 \wedge p_7 \wedge \neg p_5) \vee (p_2 \wedge p_7 \wedge p_5) \vee (p_2 \wedge \neg p_7 \wedge \neg p_5).$$

We stipulate that $\perp$ is the unique disjunction of zero cdls and is false on every model.

To get a feeling for what cdnfs are—and to see that every wff of propositional logic is equivalent to a cdnf—take any formula, say, for example, $(p_3 \rightarrow p_1) \leftrightarrow (p_1 \rightarrow p_4)$, and consider its truth table: the middle column in Figure 1. For each row in which the formula is true, write down a 'description' of that row in the form of a cdl. For example, the wff is true in row 1, in which p_3 is true and p_1 is true and p_4 is true, so the description of this row is $(p_3 \wedge p_1 \wedge p_4)$; and in row 5, in which p_3 is false and p_1 is true and p_4 is true, so the description of this row is $(\neg p_3 \wedge p_1 \wedge p_4)$; and so on. Note that (a) each description is true in and only in the row it describes and (b) these descriptions will always be *cosigned* cdls. Now write down the disjunction of all these

[33] It would make no essential difference to what follows if you wrote them in some other way, for example, as $A, B, C, \ldots$: the statement of some definitions would just be a bit less clear and simple.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

cdls: because of (a) (and the truth conditions of $\vee$) it will be equivalent to the original formula and because of (b) it will be a cosigned cdnf. In the case where the original formula was false in all rows, the resulting cdnf will be $\bot$.

A cdnf is its own proof. More precisely, a cdnf α is (simultaneously) a proof *candidate* of the claims:

- α is unsatisfiable;
- α is satisfiable;
- α is a tautology;

and is a *proof* of the claim(s) for which it passes the following proof-checking test—which runs in *polynomial* time:

1. Is α the wff $\bot$?
   Yes: α is (a correct proof that α is) unsatisfiable.
   No: α is (a correct proof that α is) satisfiable; go to 2.
2. Count the number n of basic propositions in α's first disjunct and the number m of disjuncts in α. Can you halve m (recursively) n times?
   Yes: α is (a correct proof that α is) a tautology.
   No: α is (a correct proof that α is) not a tautology.

There are a couple of things to explain here. First, there is nothing odd about a certain object—in this case a formula—being a proof candidate of *multiple* claims. This situation is very familiar. For example, a truth table for a formula α is simultaneously a proof candidate for the three claims bullet-pointed above and a tableau (tree) beginning with the formula $\neg\alpha$ is simultaneously a proof candidate for the claims:

- $\neg\alpha$ is unsatisfiable;
- $\neg\alpha$ is satisfiable;
- α is a tautology.

The truth table is a *proof* of the claim(s) for which it passes the associated test (respectively: check whether α is false on all rows; check whether α is true on some rows; check whether α is true on all rows) and similarly for the tableau (where the tests are respectively: check whether all paths close; check whether some path remains open; check whether all paths close).[34]

Second, to explain the test itself. The point behind step 1 is that $\bot$ is unsatisfiable and every other cdnf is satisfiable. To see the latter, note that a disjunction of one or more disjuncts is true if at least one of those disjuncts is true. In this case the disjuncts are cdls: conjunctions of *distinguished* literals—that is, no basic proposition occurs more than once. So every cdl is satisfiable, and indeed shows on its face how to assign truth values to basic propositions in such a way as to make it true: a basic proposition that occurs unnegated should be assigned True (1) and a basic proposition that occurs negated should be assigned False (0). Therefore every cdnf with one or more disjuncts—that is, every cdnf other than $\bot$—is satisfiable. The idea behind step 2 is as follows. There are n basic propositions in the cdnf so there are 2^n ways of assigning truth values to those basic propositions. The wff as a whole is a tautology if it is true on all those valuations.

[34] Note that all these tests run in polynomial time (i.e., polynomial in the length of the input—that is, the input to the test—which is a test of the proof candidate, not of the original formula), for their input is the entire truth table or tableau (not simply the wff α or $\neg\alpha$).

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

Each disjunct describes exactly one such valuation—and different disjuncts describe different valuations, because a cdnf is a disjunction of *distinct* cdls and no two distinct cdls with the same signature describe the same valuation. Thus if there are 2^n disjuncts then *every* valuation is covered: the cdnf is true on all of them and hence a tautology. If there aren't that many then it isn't a tautology. So we want to know whether $m = 2^n$. We do not simply calculate 2^n and then check whether it equals m because we want to ensure that the procedure runs in polynomial time. Instead we start with m, noting that $m = 2^n$ iff you can *halve* m (recursively) n times (i.e., halve m, then halve the result, and so on), where the halving procedure *fails* if you attempt to perform it on an odd number.

The foregoing considerations can readily be extended from single formulas to *sets* of formulas and *arguments*. Say that the *signature* of a cdnf is the signature of any (and all) of its disjuncts (and that the signature of $\bot$ is the empty sequence of integers). A *set* of cdnfs is cosigned if its members all have the same signature. A cosigned set of cdnfs is a proof of its own un/satisfiability: we just need to check whether there is a disjunct that appears in every formula in the set. If so, the set is satisfiable; if not, the set is unsatisfiable. Say that an *argument* is cosigned if its premisses and conclusion are cdnfs that all have the same signature. A cosigned argument is a proof of its own in/validity: we just need to check whether there is a disjunct that appears in every premise and does not appear in the conclusion. If so, the argument is invalid; if not, the argument is valid.

So, even if we demand that proof-checking procedures must run in polynomial time, we still have self-proving formulas (and sets of formulas and arguments): so many of them, indeed, that every wff of the language of propositional logic is equivalent to one (or more). We can even consider a language in which *only* cdnfs count as wffs. It has the same expressive power as the standard language of propositional logic, because every wff of the standard language has an equivalent cdnf. In this language, *every* formula serves as its own proof—so we do not need separate proof objects at all. The formulas themselves are all we need: the language is simultaneously a language of formulas and a language of proofs.[35] Thus, requiring that proofs be *easy* to check does not ban wffs'

[35] As discussed in the Appendix, if there exists a proof system in which every tautology has a short proof that can be checked in polynomial time, then NP = coNP. One might think that the proof system just mentioned (in which each tautologous cdnf serves as its own proof) counts as such a proof system, because a cdnf (considered as a proof) is certainly short relative to itself (considered as the tautology proved)—it is exactly the same length—and (as discussed above) we can check in polynomial time whether a cdnf is (a proof that it itself is) a tautology. This would be a mistake however. Any α in $TAUT$ is equivalent to some cdnf α'. α' is a short, polynomially checkable proof that $\alpha' \in TAUT$. But it is not necessarily a short, polynomially checkable proof that $\alpha \in TAUT$. First, α' might not be short relative to α. For example, suppose α is $p_1 \vee \neg p_1 \vee p_2 \vee p_3 \vee p_4 \vee p_5 \vee p_6 \vee p_7 \vee p_8$ and α' is the formula obtained by writing up α's truth table and reading off a cdnf equivalent in the manner illustrated in Figure 1. (The truth table will have 256 rows and α' will have 256 disjuncts.) Second, to check $\alpha \in TAUT$ we have to check both $\alpha' \in TAUT$ and $\alpha \equiv \alpha'$ (i.e., that α is equivalent to α'), and there might not be a *polynomial-time* procedure for performing the latter equivalence check. If, alternatively, we take as the proof of α not just α', but α' *together with* a proof of its equivalence to α (say, a truth table proof, or a tableaux proof), then this proof of α will be polynomially checkable, but we no longer have any guarantee that it will be short. A second mistake someone might make at this point is to think that because (as we have seen in this paper) there exists a proof system in which every *cdnf* tautology has a short proof that can be checked in polynomial time (the system of proofs being the

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

being their own proofs—nor even a language in which *every* wff is its own proof. So an intuition against endemic self-proof is not an adequate motivation of the strengthened conception of proof over the core conception.[36]

---

cdnfs themselves), it follows (directly) that NP = coNP. (The first mistake, just discussed, was thinking that it follows that there exists a proof system in which *every* tautology has a short proof that can be checked in polynomial time—from which it *would* indeed follow that NP = coNP.) This does not follow—because while $TAUT$ is coNP-complete, essentially the same obstacles discussed earlier in this footnote stand in the way of showing that the set of cdnf tautologies is coNP-complete (even if we consider the complement of this set relative to the set of all cdnfs, as opposed to the set of all formulas of propositional logic). Similarly, we know that SAT is NP-complete (see the Appendix), but we do not know that the set of satisfiable cdnfs is NP-complete (and indeed there are reasons to suspect it isn't), because we do not know a way of finding a cdnf equivalent to a given wff *in polynomial time* and (so) do not know how to reduce SAT to the set of satisfiable cdnfs.

[36] Further to n. 35, let $TAUT_{cdnf}$ be the set of cdnf tautologies, that is, the intersection of $TAUT$ and the set of cdnfs. As discussed, even on the strengthened conception of formal proof, any $\alpha \in TAUT_{cdnf}$ is a proof that $\alpha \in TAUT$ (i.e., a cdnf tautology is a proof that it itself is a tautology). On the core conception of proof—but not on the strengthened conception—for any $\beta \in TAUT$, a formula $\beta' \in TAUT_{cdnf}$ with $\beta \equiv \beta'$ is a proof that $\beta \in TAUT$. That is, the formula β' all by itself is the proof. To *check* this proof you have to check that $\beta \equiv \beta'$. We know of effective procedures for doing that, but no polynomial-time procedure: hence the divergent judgements of the core and strengthened conceptions concerning whether β' is a proof that $\beta \in TAUT$. I think intuition is on the side of the core conception here, not the strengthened conception. Think about what happens when you go from regarding β' as a proof that $\beta' \in TAUT$ to regarding β' as a proof that $\beta \in TAUT$. (For the sake of simplicity of formulation, I'll write as if there is no polynomial-time procedure for checking whether $\beta \equiv \beta'$. This is suspected but not known.) On the strengthened conception, you go from having a proof to not having a proof at all. On the core conception, you go from having a proof with a certain property (it is checkable in polynomial time) to a proof that does not have this extra property (it is still effectively checkable, but not in polynomial time). The latter judgement seems the more natural, intuitive one. (Note also that it fits with the customary usage of terminology in other parts of the computational complexity literature, noted in §3, where 'feasible' is treated as a qualifier rather than a *sine qua non*.) To pump the intuition a bit more, imagine a context in which *space* is at a premium. Suppose, for example, that a group is including some logical formulas together with proofs that they have certain properties (unsatisfiability, satisfiability, and tautologousness) on something like the Voyager Golden Record. Note that the cdnf equivalent α' read off from the truth table for a wff α (see Figure 1) is more compact than the full truth table yet contains almost the same information: in particular it tells us exactly the situations in which α is true. Once you see the cdnf equivalent of a wff you can immediately see certain facts about the wff. So I think it would be perfectly natural for those compiling such a record to regard the cdnfs as proofs of the associated formulas. They might think it unimportant that an alien would require substantial time to verify the proof; or they might note that the particular examples given on the record are small enough to be verifiable in a reasonable amount of time, even by a non-polynomial-time procedure (see further the discussion in §6)—cf. for example, known SAT-solvers that do not run in polynomial time and yet are useful in practice on problem instances encountered in their areas of application; or they might leave open the possibility that aliens have some sort of nonclassical computer that can verify the proofs in polynomial time. The core conception permits this attitude: for there *is* an effective procedure for checking $\alpha' \equiv \alpha$ (as well as the properties of α' itself). The strengthened conception does not: its asymmetric fixation on *temporal* efficiency in proof-checking prevents it from accepting these *spatially* compact condensations of truth tables as proofs. (On the issue of *spatially* compact proofs: one reason for regarding the proofs of Hughes and Cresswell as full formal proofs, rather than shorthand for something that could be expanded into a formal proof—by including a proof in a standard axiomatic proof system for propositional logic

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

§6. The argument from epistemology. I turn now to a final argument for the strengthened conception (one of my own devising—not drawn from the literature) which runs as follows. *Proof* is an epistemological notion. A proof of a theorem is what enables you to *know* the theorem. But you do not know the theorem just in virtue of holding the proof in your hands: you have to work through it, step by step. But this process of going through the proof and seeing that each step follows from the previous steps until ultimately the theorem is reached, is exactly the same as the procedure of *checking* the proof for correctness. So if theorems are to be knowable, this checking procedure must be feasible—that is, it must be polynomial-time computable.

I think there is a correct argument in there—and also an incorrect one. To distinguish them clearly I'll need to use words carefully. So far I've been following standard practice in the computational complexity literature (see n. 18 and n. 19) and using 'feasible' (and 'easy') as a shorthand for 'computable in polynomial time'. I'm going to stop doing that now because—at least in the present context—it would foster confusion between two very different notions. One is the notion of an *efficient procedure*. This is what complexity theorists mean by 'feasible'.[37] The other is the notion of an *achievable task*. This is what *ordinary folk* often mean by 'feasible'.

The key underlying distinction here is between *procedures* and *tasks*. By a 'task' I mean a *singular* (one-off) item of work.[38] By a 'procedure' I mean a *general* process or set of instructions that can be followed in order to perform a task.[39] For example, adding up 67 and 58 (on some occasion) is one task and adding up 324 and 42 (on some occasion) is a different task—but you can follow the same procedure to carry out both tasks.[40] Note that even though you follow the same procedure, the number of steps involved in carrying out that procedure will be greater on the second occasion than on the first.

An *efficient* procedure is one whose running time (in the sense of n. 15) has a rate of growth that is bounded above by a polynomial function. An *achievable* task is one that does not involve *too many steps*[41] (or steps that are too difficult—but when we are

---

of each line marked 'PC'—is that we have no system of *short* proofs for tautologies, so the expanded proof could be unworkably long. The core conception allows us to favour brevity in formal proofs in a way that the strengthened conception does not.)

[37] They also apply the term by extension to problems or languages: *sets* of finite objects such as numbers, formulas, strings and so on. For example, a *set* of formulas—a *language*—can be described as feasible if there is a feasible *procedure* for deciding membership in the set.

[38] The term 'task' can in ordinary language also be used in a general sense for a collection of tasks in the singular sense—for example, "Your task is to prepare breakfast each morning"—but I shall always use it in the singular sense.

[39] Problems and languages (cf. n. 37) are also general, in the sense that they are *sets* of instances, each of which is a singular object.

[40] The procedure is a procedure for solving the general problem $x + y$: the problem has many instances and the procedure for solving it is a procedure for solving any and all of its instances.

[41] Cf. the first quotation from Kleene [23, p. 225, n.163] in §4 above. This notion of 'too many'—or the closely related notion of 'too big'—is the (complement of the) one expressed by the term 'feasible' in the literature on strict finitism: see, for example, Parikh [31] "Here "feasible" and "reasonable length" are intuitive notions and we appeal to the reader's common sense to agree that a proof with over 10^{1000} symbols is not itself feasible" and Sazonov [39, p. 30] "How to formalize the intuitive notion of *feasible numbers*? To see what feasible numbers are, let us start by counting: 0, 1, 2, 3, and so on. At this point, A.S. Yesenin-Volpin (in his "Analysis of potential feasibility," 1959) asks: *"What does this 'and so on' mean?" "Up to what extent 'and so on'?"* And he answers: *"Up to exhaustion!"*"

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

talking about *effective* procedures, as we are in the case of proof-checking procedures, every step is simple and mechanical and hence there is no question of any step's being too difficult to perform). Note that these are different notions. A task can be achievable even though we have no efficient procedure for carrying it out; and a task can fail to be achievable even though we do have an efficient procedure for carrying it out. As an example of the former, consider the task of writing out a truth table for a formula with five basic propositions. We have an *effective* procedure for doing this, but no *efficient* procedure. Nevertheless the task is perfectly achievable: it is the kind of thing one might set students as a homework exercise. As an example of the latter, consider the task of checking a 30000-odd page truth table (i.e., roughly as many pages as *Encyclopaedia Britannica*) for a wff involving twenty propositional letters. We *do* have an efficient procedure for doing that,[42] yet the task of checking the table is not achievable. No normal person has the time available to do it: it would simply *take too long*.[43]

Now returning to the argument at the beginning of this section, supposing (what is extremely plausible—and can at the very least be granted for purposes of argument) that to get to *know* a theorem one must work through a proof of it, we may indeed conclude that if you are to know the theorem, the *task* of checking its proof must be *achievable*. But this is not the conclusion that the proponents of the strengthened conception need. They want to conclude that the *procedure* for checking the proof must be *efficient*. And that simply does not follow. As we have just seen, a task's being achievable is a different thing from the existence of an efficient procedure for performing the task: a task can be perfectly achievable even though we have no efficient procedure for performing it. So the argument from epistemology fails.

§7. Conclusion. I began in §1 by setting out a core conception of formal proof and then substantiating its designation as 'core' by showing that it is found in the textbooks, rooted in the history and embedded in the results of formal logic as we know it. The essence of this conception is the idea that there must be an effective or mechanical procedure for checking the correctness of a purported proof. I then in §2 presented a proposed strengthening of this conception that has become standard in the literature on propositional proof complexity. The essence of this strengthened conception of formal proof is that the proof-checking procedure must not only be effective but must furthermore be able to be carried out in polynomial time. I then in §§3–6 considered all reasonable arguments in favour of the strengthened conception that I could find in the literature or develop myself. I rejected them all. My conclusion is that, when it comes

[42] Note that I'm talking about checking a truth table that you have been given, not producing it in the first place. The input to the checking procedure is the entire truth table—whereas the input to the procedure for producing it is merely the formula with 20 propositional letters.

[43] Note that the classification of *procedures* as 'infeasible' (or 'inefficient' as I am saying at present) is in fact grounded in judgements about the unachievability of *tasks*. A procedure is inefficient precisely because the task of following it on a particular input quickly (as the length of the input increases) becomes unachievable. In explaining why an exponential time procedure is unfeasible, complexity theorists will say things such as: executing it on an input of only moderate length would take more steps of calculation than there are atoms in the observable universe, or milliseconds since the Big Bang. Note how what gets a grip on our intuition here is the fact that a certain *task* (following the procedure on a given input, of moderate size) has *too many steps* for us to be able to carry out.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

to analysing the very idea of formal logical proof, we have no good reason to move to the strengthened conception and should stick with the core conception.

Appendix. A language L is in NP iff each $x \in L$ has a 'membership certificate' whose length is bounded by a (fixed, for that language L) polynomial function of the length of x (i.e., it is 'short' in the sense discussed in n. 20) and which can be verified by a deterministic Turing machine in polynomial time. A language L is in coNP iff its complement L^c is in NP. (Note that $L^c = \Sigma^* \backslash L$, where Σ is an assumed background alphabet such that $L \subseteq \Sigma^*$.)

A language L_1 is NP-complete iff (i) $L_1 \in$ NP and (ii) any language $A \in$ NP is reducible to L_1, that is, there is a polynomial-time computable function f such that $x \in A$ iff $f(x) \in L_1$. Similarly, a language L_2 is coNP-complete iff (i-co) $L_2 \in$ coNP and (ii-co) any language $B \in$ coNP is reducible to L_2, that is, there is a polynomial-time computable function g such that $x \in B$ iff $g(x) \in L_2$.

The set $TAUT$ of tautologies of propositional logic is coNP-complete. (i-co): If a formula α is *not* a tautology, that is, $\alpha \in TAUT^c$, it has a short membership certificate in $TAUT^c$ that can be verified in polynomial time: the certificate for α is an assignment of truth values to α's component basic propositions under which α comes out false. So $TAUT^c \in$ NP, and $TAUT = (TAUT^c)^c$, hence $TAUT \in$ coNP. (ii-co): Given $B \in$ coNP, by definition $B^c \in$ NP. The Cook–Levin theorem (see, e.g., Sipser [42, chap. 7]) states that the set SAT of satisfiable formulas of propositional logic is NP-complete. So B^c is reducible to SAT: there is a polynomial-time computable function f such that $x \in B^c$ iff $f(x) \in SAT$. And SAT is reducible to $TAUT^c$: $\alpha \in SAT$ iff $\neg\alpha \in TAUT^c$, so there is a polynomial-time computable function f_2 such that $\alpha \in SAT$ iff $f_2(\alpha) \in TAUT^c$, namely the function that maps α to its negation. So there is a polynomial-time computable function f_3, namely the composite function $f_2 \circ f$, such that $x \in B^c$ iff $f_3(x) \in TAUT^c$. Negating both sides of the latter biconditional gives $x \notin B^c$ iff $f_3(x) \notin TAUT^c$. Now the left side of the latter biconditional is equivalent to $x \in B$ and the right side is equivalent to $f_3(x) \in TAUT$ so we have $x \in B$ iff $f_3(x) \in TAUT$. So f_3 serves as the g we need to establish (ii-co).

So if we can show $TAUT \in$ NP, it follows that coNP $\subseteq$ NP. From there it follows that NP $\subseteq$ coNP and hence NP = coNP. For let A be any language in NP. By definition of coNP, $A^c \in$ coNP. Now from coNP $\subseteq$ NP it follows that $A^c \in$ NP. Now by definition of coNP again, $(A^c)^c \in$ coNP, that is, $A \in$ coNP. So $A \in$ NP $\Rightarrow A \in$ coNP, that is, NP $\subseteq$ coNP.

Now if there exists a proof system in which every tautology has a short proof that can be checked in polynomial time, then indeed $TAUT \in$ NP: these proofs are the required membership certificates. However this does *not* follow if all we have are *short* proofs that cannot be checked *in polynomial time*. In that case the proofs do not serve as membership certificates; so it does not follow that $TAUT \in$ NP; and so it does not follow that NP = coNP.

BIBLIOGRAPHY

[1] Aaronson, S. (2011). Why philosophers should care about computational complexity. Technical Report 108 (Revision 2), Electronic Colloquium on Computational Complexity.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

[2] Arora, S., & Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge: Cambridge University Press.

[3] Beame, P. (2004). Proof complexity. In Rudich, S., and Wigderson, A., editors. *Computational Complexity Theory*. Providence: American Mathematical Society, pp. 199–246.

[4] Beame, P., & Pitassi, T. (2001). Propositional proof complexity: Past, present, and future. In Păun, G., Rozenberg, G., and Salomaa, A., editors. *Current Trends in Theoretical Computer Science: Entering the 21st Century*. Singapore: World Scientific, pp. 42–70.

[5] Boghossian, P. (2014). What is inference? *Philosophical Studies*, **169**, 1–18.

[6] Bostock, D. (1997). *Intermediate Logic*. Oxford: Clarendon Press.

[7] Burgess, J. P. (2009). *Philosophical Logic*. Princeton: Princeton University Press.

[8] Buss, S. R. (1998). An introduction to proof theory. In Buss, S. R., editor. *Handbook of Proof Theory*. Amsterdam: Elsevier, pp. 1–78.

[9] Cook, S., & Reckhow, R. (1979). The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, **44**(1), 36–50.

[10] Frege, G. (1879). Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought. In van Heijenoort, J., editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, MA: Harvard University Press, pp. 1–82.

[11] ———. (1964). *The Basic Laws of Arithmetic: Exposition of the System*. Berkeley: University of California Press. Originally published as Grundgesetze der Arithmetik in 1893 (volume I) and 1903 (volume II).

[12] ———. (2013). *Basic Laws of Arithmetic: Derived Using Concept-Script*. Oxford: Oxford University Press.

[13] Gandy, R. (1995). The confluence of ideas in 1936. In Herken, R., editor. *The Universal Turing Machine: A Half-Century Survey*. Vienna: Springer-Verlag, pp. 51–102.

[14] Gentzen, G. (1969). Investigations into logical deduction. In Szabo, M. E., editor. *The Collected Papers of Gerhard Gentzen*. Amsterdam: North-Holland, pp. 68–131. Originally published (in German) in 1935.

[15] Gödel, K. (1931). On formally undecidable propositions of *Principia Mathematica* and related systems I. In van Heijenoort, J., editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, MA: Harvard University Press, pp. 596–616.

[16] ———. (1965). On undecidable propositions of formal mathematical systems. In Davis, M., editor. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Hewlett: Raven Press, pp. 41–74.

[17] Hardy, G. H. (1929). Mathematical proof. *Mind*, **38**(149), 1–25.

[18] Hartmanis, J. (1989). Gödel, von Neumann and the P=?NP problem. Technical Report TR 89-994, Department of Computer Science, Cornell University.

[19] Hilbert, D. (1904). On the foundations of logic and arithmetic. In van Heijenoort, J., editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, MA: Harvard University Press, pp. 129–138.

[20] ———. (1925). On the infinite. In van Heijenoort, J., editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, MA: Harvard University Press, pp. 369–392.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

[21] ———. (1927). The foundations of mathematics. In van Heijenoort, J., editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, MA: Harvard University Press, pp. 464–479.

[22] Hughes, G. E., & Cresswell, M. J. (1996). *A New Introduction to Modal Logic*. London: Routledge.

[23] Kleene, S. C. (1967). *Mathematical Logic*. New York: John Wiley & Sons.

[24] Kreisel, G. (1958). Review of Wittgenstein's Remarks on the Foundations of Mathematics. *British Journal for the Philosophy of Science*, **9**(34), 135–158.

[25] ———. (1962). Foundations of intuitionistic logic. In Nagel, E., Suppes, P., and Tarski, A., editors. *Logic, Methodology and Philosophy of Science: Proceedings of the 1960 International Congress*. Stanford: Stanford University Press, pp. 198–210.

[26] ———. (1968). A survey of proof theory. *Journal of Symbolic Logic*, **33**(3), 321–388.

[27] ———. (1971). A survey of proof theory II. *Studies in Logic and the Foundations of Mathematics*, **63**, 109–170.

[28] Kripke, S. A. (2013). The Church-Turing "thesis" as a special corollary of Gödel's completeness theorem. In Copeland, B. J., Posy, C. J., and Shagrir, O., editors. *Computability: Turing, Gödel, Church, and beyond*. Cambridge, MA: MIT Press, pp. 77–104.

[29] Lolli, G. (1998). Logical completeness, truth, and proofs. In Dales, H. G., and Oliveri, G., editors. *Truth in Mathematics*. Oxford: Clarendon Press, pp. 117–129.

[30] MacKenzie, D. (1995). The automation of proof: A historical and sociological exploration. *IEEE Annals of the History of Computing*, **17**(3), 7–29.

[31] Parikh, R. (1971). Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, **36**(3), 494–508.

[32] Prawitz, D. (1971). Ideas and results in proof theory. In Fenstad, J. E., editor. *Proceedings of the Second Scandinavian Logic Symposium*. Amsterdam: North-Holland Publishing Company, pp. 235–307.

[33] ———. (1973). Towards a foundation of a general proof theory. In Suppes, P., Henkin, L., Joia, A., and Moisil, G. C., editors. *Logic, Methodology and Philosophy of Science IV: Proceedings of the Fourth International Congress for Logic, Methodology and Philosophy of Science, Bucharest, 1971*. Amsterdam: North-Holland Publishing Company, pp. 225–250.

[34] ———. (1974). On the idea of a general proof theory. *Synthese*, **27**, 63–77.

[35] ———. (2015). Explaining deductive inference. In Wansing, H., editors. *Dag Prawitz on Proofs and Meaning*. Cham: Springer, pp. 65–100.

[36] ———. (2019). The fundamental problem of general proof theory. *Studia Logica*, **107**, 11–29.

[37] Pudlák, P. (1998). The lengths of proofs. In Buss, S. R., editor. *Handbook of Proof Theory*. Amsterdam: Elsevier, pp. 547–637.

[38] Razborov, A. A. (2002). Proof complexity of pigeonhole principles. In Kuich, W., Rozenberg, G., & Salomaa, A., editors. *Developments in Language Theory: 5th International Conference, DLT2001*. Berlin: Springer, pp. 100–116.

[39] Sazonov, V. Y. (1995). On feasible numbers. In Leivant, D., editor. *Logic and Computational Complexity: International Workshop LCC '94*. Berlin: Springer, pp. 30–51.

[40] Segerlind, N. (2007). The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, **13**(4), 417–481.

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press

[41] Sipser, M. (1992). The history and status of the P versus NP question. In. *STOC '92: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*. New York: Association for Computing Machinery, pp. 603–618.

[42] ———. (2013). *Introduction to the Theory of Computation* (third edition). Boston, MA: Cengage.

[43] Smith, N. J. J. (2012). *Logic: The Laws of Truth*. Princeton: Princeton University Press.

[44] Tennant, N. (1990). *Natural Logic*. Edinburgh: Edinburgh University Press.

[45] Urquhart, A. (1995). The complexity of propositional proofs. *Bulletin of Symbolic Logic*, **1**(4), 425–467.

[46] Whitehead, A. N., & Russell, B. (1910). *Principia Mathematica*, Vol. I. Cambridge: Cambridge University Press.

[47] Zach, R. (2023). Hilbert's program. In Zalta, E. N., and Nodelman, U., editors. *The Stanford Encyclopedia of Philosophy*, Winter edition, Metaphysics Research Lab. Stanford, CA: Stanford University.

THE UNIVERSITY OF SYDNEY
AUSTRALIA
E-mail: nicholas.smith@sydney.edu.au

https://doi.org/10.1017/S1755020326101300 Published online by Cambridge University Press