Synthese (2026) 207:99
https://doi.org/10.1007/s11229-026-05482-4

Varieties of confluence arguments, part 1: practical applications

Jason Zesheng Chen[1]

Received: 11 January 2025 / Accepted: 31 January 2026

Abstract
This paper is the self-contained first part of a two-part series that examines the wide varieties of ways that mathematicians and philosophers have appealed to confluence phenomena in their work. In this part, we focus on the practical roles such phenomena can play, paying special attention to how they facilitate the communication of mathematical ideas (proofs, definitions, and conjectures) in actual practice. Through surveying the wide array of such arguments, I shall eventually hone in on two subtly distinct facets concerning the uses of confluence arguments (and another that is revealed to be rather trivial upon analysis), which are sometimes conflated in philosophical discussions: one (dubbed Rigor Assurance) guarantees the formal rigor of a proof, while the other (Coding Invariance) ensures the results obtained are not a mere artifact of the coding that is used. I will then attempt to tease these two facets apart with actual examples in print, citing crucial evidence witnessing the distinction. With this setup in place, we will see how certain recent philosophical debates about the practical use of the Church-Turing Thesis may naturally dissolve, as the combatants do not share the same assumptions of the roles confluence arguments play in each case, and thus end up merely talking past each other.

Keywords Church-Turing thesis · Confluence · Philosophy of mathematics · Mathematical practice · History of mathematical logic · Computability theory · Borel equivalence relations

✉ Jason Zesheng Chen
zeshengc@uci.edu

[1] Department of Logic and Philosophy of Science, University of California, Irvine, CA, USA

Published online: 23 February 2026

1 Introductory overview

1.1 Confluence phenomena and confluence arguments: What do they do?

The Church-Turing Thesis states that any function intuitively computable by an effective method can be computed by a Turing machine. The received wisdom is that the thesis expresses the conviction that our formalization of effectivity by means of Turing machines is *correct*. Aside from Turing's insightful analysis of the meaning of effective method, the thesis is crucially justified by a certain phenomenon of *confluence*: all attempts to formalize the notion of effective method have led to the same class of functions.

For the present context, confluence[1] will be fairly liberally understood to encompass situations whereby a number of distinct attempts (to define, formalize, solve, etc.) have all produced results that are in some sense equivalent, or conversely that a particular object or concept admits numerous seemingly distant equivalent definitions. Confluence arguments, then, refer to any argument that appeals to such confluence phenomena as evidence for something. That is:

Confluence Argument (template). The fact that we have all these equivalences or otherwise similar results constitutes evidence for …

The prototypical example of a confluence argument manifested itself in the justification of the Church-Turing Thesis that we previously alluded to. The homonymous entry in the Stanford Encyclopedia of Philosophy (Copeland, 2024) cites the following remark by Church:

> The fact … that two such widely different and (in the opinion of the author) equally natural definitions of effective calculability[2] … turn out to be equivalent adds to the strength of the reasons adduced below for believing that they constitute as general a characterization of this notion as is consistent with the usual intuitive understanding of it. (Church, 1936)

The seminal textbook by Kleene (1952) adds many other equivalences to the list. Jointly, the growing list of equivalences is meant to convince one of the truth of the Church-Turing Thesis. Kleene cites this phenomenon, among others, as evidence for Church's Thesis:

> Several other characterizations of a class of effectively calculable functions … have turned out to be equivalent to general recursiveness, i.e. the classes of functions which they describe are coextensive. … The fact that several notions

[1] To borrow a term from Gandy (1988). Gandy's notion of confluence has recently been significantly elaborated by Kennedy (2013, 2020), especially in relation to entanglement with formalisms. Elsewhere, especially in the sciences, the term *consilience* is often used to mean the same. I opted for the former to align with the logic community.

[2] Referring to λ-definability and the Turing machines.

> which differ widely lead to the same class of functions is a strong indication that this class is fundamental. (Kleene, 1952, p. 319)

Following Gandy, we will occasionally refer to this line of thinking in justifying the Church-Turing Thesis as "(Church's or Kleene's) argument by confluence". Gödel, in his 1946 Princeton Bicentennial Address (see Kennedy (2013)), similarly attributes the success of Turing's analysis of computability to this confluence: "It seems to me that this importance [of Turing computability] is largely due to the fact that with this concept one has succeeded in giving an absolute definition of an interesting epistemological notion, i.e., one not depending on the formalism chosen." Our modern vantage point has only made this argument more forceful for Copeland:

> All attempts to give an exact characterization of the intuitive notion of an effectively calculable function have turned out to be equivalent, in the sense that each characterization offered has been proved to pick out the same class of functions, namely those that are computable by Turing machine. The equivalence [i.e., confluence] argument is often considered to be very strong evidence for the thesis, because of the diversity of the various formal characterizations involved. (Copeland, 2024)

For the philosophically minded, the confluence arguments above serve to justify the Church-Turing Thesis, ultimately a thesis about correct formalizations. To complicate matters a little, however, we note that this is not how mathematicians usually engage with confluence arguments in actual practice. Despite an apparently singular philosophical application in the case of the Church-Turing Thesis, confluence arguments turn out to serve a few distinct practical purposes for the working mathematician[3].

To name one such purpose, in his classic computability textbook, Rogers recognizes that the Church-Turing Thesis "has come to play a somewhat broader role" (Rogers, 1987, p. 20) than the philosophical one in the foregoing, that is, it "permit[s] us to avoid cumbersome detail and to isolate crucial mathematical ideas from a background of routine manipulation" ibid.). Such sentiment has well persisted into more modern texts - Gao (2008), for instance, declares from the outset that the text will not deal with the details of Turing machine definitions and accepts (on grounds of the Thesis) that informal proofs are just as good.

Granted, this safety is putatively guaranteed by the philosophical lesson of the Church-Turing Thesis. We will later see that this need not be the case. In other words, in this capacity, such safety need not be tied to the Church-Turing Thesis proper, but rather to the confluence of definitions that scaffolds it.

To echo this appeal to confluence, and to point out that this mode of argument extends beyond just the Church-Turing Thesis, let us also take a hint from Maddy's classic paper on axiom justification (Maddy, 1988). There, surveying the various rules of thumb by which set-theoretic axioms came to be accepted, she notes Kanamori and Magidor's remarks (Kanamori & Magidor, 1978) that the many equivalent defini-

[3] And it shall become clear as the paper progresses that these are sometimes conflated with applications of the Church-Turing Thesis proper.

tions of weakly compact cardinals serve as "good evidence for the naturalness and efficacy of the concept". Recalling that the same pattern of argument was observed in early discussions of the naturalness of the concept of general recursiveness, Maddy writes:

> … [T]he property of weakly [sic] compactness is equivalent to the compactness of the language $L_{\kappa,\kappa}$, and to a certain tree property, and to an indescribability property, and to several other natural properties … This convergence has led some writers to *diversity*, another rule of thumb. (Maddy, 1988, p. 504)

Later in the same paper, Maddy highlights another instance of appeal to this rule of thumb by Kanamori and Magidor, this time on the subject of zero-sharp and the list of statements equivalent to its existence:

> … [The assumption that zero-sharp exists] turns out to be equivalent to a determinacy assumption and to an elementary embedding assumption. This prompts Kanamori and Magidor to another application of diversity: '[This] is a list of equivalences, much deeper than the confluence seen at weak compactness.' Thus the implication of the existence of the sharps provides a very appealing extrinsic support for the Axiom of Measurable Cardinals. (ibid., p. 507)

It's fair to say that in each of the preceding examples, some evidential import is being communicated by a confluence argument, although it is decreasingly clear how and of what. Of course, the philosophical case of the Church-Turing thesis is relatively straightforward. The definitions were intended specifically for a common goal: to capture the informal notion of effective feasibility. This goal had sufficiently significant motivation: both Hilbert's tenth problem and his Entscheidungsproblem turned on its satisfactory resolution. And the common lesson drawn from the equivalence between, say, the λ-definable functions and the Turing-computable ones is that we have the *correct* formalization of the informal notion of effective/mechanical computability. Against this backdrop, the Church-Turing thesis has a clear philosophical role to play: it is the expression of (perhaps our confidence in) the correctness of our formal conceptual analysis of idealized effective feasibility. And the confluence of definitions serves as evidence for this correctness. So far so good.

1.2 What this paper sets out to do

And yet other examples above remain puzzling: the clear (philosophical) case of the Church-Turing Thesis notwithstanding, this *pas de trois* of confluence, evidence, and justification turns out to be rather pervasive across numerous mathematical disciplines, serving a diverse range of purposes largely orthogonal to each other. This paper and its planned sequel are undertaken to present a taxonomy of what those are, how they work, and the ways they differ from each other. Jointly, they will articulate a taxonomical framework through which confluence arguments (as well as claims about theses of the Church-Turing type) can be better understood.

Secondary to this goal is an attempt to untether successful instances of confluence arguments from the premise of an underlying background notion or something of this sort, which often goes hand-in-hand in the discussions of the Church-Turing Thesis. If I am successful in this regard, I will have shown that, in many cases, confluence phenomena can and are invoked substantively in ways that do not hinge on the business of notion-capturing - their thrust comes from somewhere else.

As an overview, this paper will study three practical purposes putatively served by confluence arguments:

1. **Conjecture Heuristic**: in this form with the least philosophical baggage, confluence motivates conjectures of the form "every such-and-such will be so-and-so", on the basis that a large class of (natural, practical) examples are so-and-so. A typical example is Martin's Conjecture in recursion theory. Although this is something mundane and, strictly speaking, not characteristic of confluence arguments per se, it nevertheless serves to reveal some of the inductive force behind confluence-as-evidence.
2. **Rigor Assurance**: in terms of theorems and proofs, confluence allows mathematicians to be confident that formal rigor is not loss in translation between informal and formal language or translation between language used in different fields of mathematics. This allows for freedom to move at once between different levels of rigor and conceptual frameworks.
3. **Coding Invariance**: Confluence also suggests that our theorems are not a result of superficial coding peculiarities. The same result is obtained regardless of the coding of the objects/concepts, indicating that we have gained bona fide knowledge about the objects/concepts themselves rather than being misled by formalisms.

Given my rather peculiar focus on the confluence arguments, a word about methodology is in order: familiar readers will note at this point that the setup is heavily inspired by Maddy's poignant classification (Maddy, 2019) of what "foundational" means in mathematical or philosophical discourse comparing the roles played by set theory, category theory, and homotopy type theory[4]. As such, this work is going to be as much about the nature of various instances of confluence arguments as it is about the actual mathematical practice (and the goals thereof) of invoking them to do one thing or another. This choice of focus engenders a particular methodology - the raw data here, as it were, will be written remarks in the literature by the working mathematicians, which I will take at face value as attestation to actual mathematical practices[5] and proceed to investigate them as such.

This paper will be structured as follows. Sections 2, 3, 4 will each be devoted to a particular practical purpose that can be served by appealing to confluence. These are each introduced by one or two examples I consider representative, and further

[4] So this article might as well be titled "*What do we want confluence arguments to do?*".

[5] At places, it would seem I am allowing myself excessive interpretive latitude as to what constitutes confluence. The reader is kindly reminded of this methodological principle here - I am trying to stay as close to the mathematician's usage as possible.

attesting examples in print will be surveyed. For each of these purposes, I will make an effort to identify its defining characteristic and attempt to tease them apart via an analysis of the presented examples. At appropriate junctures throughout the later parts of the paper (mostly Sects. 3 and 4), I will also point to existing dabates in the literature[6] and show how they may be fruitfully resolved under the framework developed here.

Some readers will likely notice an omission from my treatment: that of the most philosophically significant case of confluence arguments, i.e., to justify that computability is Turing-computability simpliciter; and perhaps relatedly, omissions of the kinds of diversity considerations in the opening references to the set-theoretic literature. I would like to clarify at the outset that such philosophically involved confluence arguments are delegated to Part 2 in this series, where similar distinctions will be carefully delineated and distinguished, this time between confluence serving as evidence of correctness (of a formalized concept), fruitfulness (of a particular mathematical pursuit), and robustness (of a mathematical definition).

For now, let me simply acknowledge the salient presence such arguments command in the philosophical literature, which has been the occasion to furnish this work with a standalone sequel. In the meantime, the present Part 1 focuses squarely on the practical roles such arguments can play, paying special attention to how they facilitate the communication of mathematical ideas (proofs, definitions, and conjectures) in actual practice. Concretely, this involves putting under the lens various mathematical practices related to maxims like the Church-Turing Thesis, from which the present work derives much of its motivations. Inevitably, therefore, the locus of such an investigation will gravitate towards computability theory and its (more or less distant) cousins, so let us start there.

2 Conjecture Heuristic

Let us begin with something that carries the least philosophical baggage[7]. I will freely concede in advance that this will probably not end up being a serious use of confluence arguments, for it will likely turn out to be a fancy term for something mundane that mathematicians are all too accustomed to. This does not mean, however, that I am going through the pains of drawing out this thread just to discard it at the end; rather, the content of this section serves to 1) reflect on traces of confluence-like arguments at the emergence of the Church-Turing Thesis and the later development of computability theory, and, more importantly, 2) to reveal an important underlying mechanism that enables the more involved uses of confluence arguments in the later sections.

[6] For example, San Mauro (2018)'s examination of the Church-Turing Thesis in practice, or Hamkins (2016)'s call for philosophical grounds supporting what he calls Gao's Thesis

[7] Least for the present context, anyway. By this I mean the way that confluence phenomena figure in what goes on in this section will be shown to be likely insignificant, if not outright trivial, having nothing to do with the sort of "joint-carving" vindication in the Church-Turing Thesis. I nevertheless acknowledge that probabilistic thinking and conjecturing in mathematical practice remains a tricky topic in the philosophy literature.

In his recollection of the early days of λ-calculus, Kleene recalls coming up with the definition of the predecessor function on the natural numbers[8] in 1932 (at the dentist's office, of all places) and showing the result to Church, who "had just about convinced himself there wasn't a predecessor function [definable in λ-calculus]" (Crossley, 2006, p. 5). This and other related results inspired great confidence in the range of functions that could be defined in λ-calculus. Church, for example, gradually came around and asked "whether [they] had not really got *all* the effectively calculable functions."

At any rate, that eventually became the prevailing sentiment at Princeton:

> There was no idea [before the discovery of the λ-definable predecessor function] that [the class of λ-definable functions] was going to be all effectively calculable functions. But I [Kleene] kept taking it as a challenge and everything I tried I could work [sic], and [having been able to define the integers] we got the idea that this could represent all calculable functions … for us the first idea that λ-definability was general was after … discovering that everything you thought of that you wanted to prove λ-definable, you could. (Crossley, 2006, pp. 5–6)

Keeping in mind that this development took place in what can be said to be the prehistoric era of the Church-Turing Thesis, prior to any knowledge of Turing's work in 1936, the confluence at play here is a particularly simple one: all the effectively calculable functions one could think of were λ-definable. And this seemed to have increased the credence that the latter captures all of them.

To be fair, the quoted passage still has quite a bit in common with the later Church-Turing Thesis. For one, it seems to express that they had done something right in their formalization of effective calculability. However, I would like to entertain the possibility that something else was conveyed alongside this vindication. To see what, let me quote from a letter from Church to Kleene, dated November 29, 1935, according to which Gödel regarded Church's proposal of identifying effective calculability with λ-definability as "thoroughly unsatisfactory" (Davis, 1982, p. 9), presumably sometime around his visit to the Institute for Advanced Study in the fall of 1933. Despite this, Church recalled remaining confident: "If [Gödel] would propose any definition of effective calculability which seemed even partially satisfactory, [Church] would undertake to prove that it was included in lambda-definability [sic]." (ibid.)

Gödel did in fact propose a definition of effective calculability. In a series of lectures at the IAS[9] in 1934, Gödel presented what is known today as the Herbrand-Gödel general recursive functions. Kleene recounts questions being asked along the lines of "Does this embrace all effectively calculable functions, and is it equivalent to λ-definability?" (Crossley, 2006, p. 6) For us, this draws out a useful distinction: suspending judgment on the correctness of one's own definition, one can still independently conjecture that it is equivalent to another plausible definition.

[8] The motivation arose out of Kleene's approach to deriving the Peano axioms in Church's original formalism (which was eventually shown inconsistent by Kleene and Rosser), in particular the injectivity of successor. See Kleene (1981, p. 56) for the history.

[9] Transcribed in Gödel (1934) and reprinted in Gödel (Göodel, 2001, pp. 338–371).

Even though Gödel at the time[10] was "not at all convinced that [his] concept of recursion comprises all possible recursions" (Davis, 1982, p. 8), Church, who had "explicitly come out with [Church's Thesis]" (Crossley, 2006, p. 6), responded by proving the equivalence between the two notions together with Kleene (according to whom, "then it was a simple matter to prove the equivalence of the two notions" ibid.)).

Here lies plainly the first purpose that confluence arguments serve: along with the conviction that λ-definable functions are the correct formalization of effective calculability, Church was also motivated to conjecture that other plausible definitions will be (either weaker or) equivalent[11]. This seems straightforward enough: if λ-definability is the right definition for the effective calculability, then any other candidate definition worth their salt should be equivalent to it. The point is that, after seeing just the mini-confluence (all the functions they could think of were λ-definable), Church gained not only credence in the correctness of his definition, but also a heuristic for conjecturing that other definitions will be equivalent to it.

This kind of **Conjecture Heuristic** was first documented in print by Post (1936), when he tried to provide yet another definition of effective calculability (now known as the Post-Turing machines, one that he considered to be of "logical potency" and "psychological fidelity"). Acknowledging the earlier equivalence between Church's λ-definable functions and Herbrand-Gödel general recursive functions, Post ends his paper by speculating that his machines are also equivalent: "The writer expects the present formulation to turn out to be logically equivalent to recursiveness in the sense of the Gödel-Church development."

Of course, the modern-day reader will know that virtually all attempts to define effective calculability have ended up equivalent. With the Church-Turing Thesis fully fledged, we see the same kind of Conjecture Heuristic at work when Wolfram conjectured that a particular class of cellular automata can simulate any Turing machines. The SEP entry *Cellular Automata*[12] documents one such conjecture:

> "The basic feature a cellular automaton needs to perform computations is the capacity of its transition rule of producing … localized, stable, but non-periodic configurations of groups of cells … seen as encoding packets of information … [which] can propagate in time and space without undergoing important decay. The amount of unpredictability in the behavior of Class_4 rules also hints at computationally interesting features: by the Halting Theorem … it is a key feature of universal computation that one cannot in principle predict whether a given computation will halt given a certain input. These insights led Wolfram to conjecture that Class_4 [cellular automata] were … capable of universal com-

[10] For an evolution of Gödel's view on computability, see Sieg (2006). Gödel would go on to be convinced by Turing's 1936 analysis. We will return to this in later sections.

[11] In an abstract of Church (1936) submitted to the *Bulletin of AMS* in 1935, Church wrote: "... it is maintained that the notion of an effectively calculable function ... should be identified with that of a [general] recursive function, since other plausible definitions of effective calculability turn out to yield notions which are either equivalent to or weaker than [general] recursiveness." (Davis, 1982, p. 10)

[12] To which I refer the reader for a survey of the requisite background.

> putation. … Rule 110 [a particular cellular automaton] was indeed proved to be computationally universal. (Berto & Tagliabue, 2023)

The preceding example is meant to further tease apart the two types of sentiment somewhat present in Church's letter. Church was confident on the one hand that his definition was correct, and on the other hand this warrants the conjecture that other similar definitions would be equivalent. Wolfram's conjecture, however, does not seem to presuppose one way or another whether he had any correct definition of effective calculability. That is, having the history of confluence of computability notions in the background and the precedent that certain cellular automata can indeed simulate universal Turing machines (e.g., Conway's Game of Life), one is already rationally licensed to conjecture that a cellular automaton equipped with rules that make it resemble effective computation is Turing-complete.

This heuristic for conjecturing that certain rules of cellular automaton make it able to simulate (perhaps all) Turing machine computations is exactly as Rogers spelled out in his seminal textbook on computability:

> In fact, if certain general (and reasonable) formal criteria are laid down for what may constitute a [specification of what counts as computation], it is possible to show that the class of partial functions obtained is always a subclass of the maximal class of all partial recursive functions. (Rogers, 1987, p. 18)

The point I want to make is a simple one. If one sees a number of (natural) attempts or instances have all converged on a particular result, then one is motivated to conjecture that the next one is going to yield the same result, or even better, that all such attempts/instances will end up being in some sense equivalent.

Indeed, such a heuristic is so common that it is almost paradigmatic of much of the later work in computability theory. To point to a prominent example, the most famous open problem in computability theory[13]:

Conjecture 1 (Martin's Conjecture) Every Turing-invariant[14] Borel function is either constant on a cone or Turing-equivalent to an iterate (maybe transfinite) of the Turing jump on a cone.

The heuristic for this conjecture partially follows from an empirical observation. Having surveyed a number of natural examples of uncomputable sets, Montalbán (2019) summarizes:

[13] I have chosen to present a lighter version aimed at general mathematical audience, taken from Montalbán (2019), which differs from the stronger and more general version found in the specialist literature. The reason is that the latter contains more set-theoretic jargons and subtleties that will distract us from the current exposition, with little gain in the lesson.

[14] A function $F : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ is Turing-invariant if and only if it maps Turing-equivalent reals to Turing-equivalent reals, i.e., $x \equiv_T y$ implies $F(x) \equiv_T F(y)$. For example, the Turing jump is Turing-invariant. A cone is a set of the form $\mathrm{Cone}(a) := \{y \mid a \leq_T y\}$. We say F is constant on a cone if there is a cone whose image under F contains only one Turing degree.

> [The examples ordered under Turing-reduction] do seem to form a hierarchy. There are many more examples one can get from elsewhere in mathematics and many more from computability theory that are still very natural … All the examples we know are ordered in a line. Furthermore, we know of no natural example strictly in between 0 and K [the halting problem], and all natural examples strictly in between 0 and COF [the set of polynomials in $\mathbb{Z}[x, y_0, y_1, y_2, ...]$ that have integer solutions for all but at most finitely many values of x] are Turing equivalent to either K or TF [finite presentations of torsion-free groups]. We know many natural examples between COF and TA [truth of arithmetic], though they are nicely ordered in a line like the ordering of the natural numbers. We know many natural examples between TA and WF [codes for programs computing well-founded trees], and they are nicely ordered in a well-ordered line

Here one hears echoes of confluence: *all the natural examples of uncomputable sets end up being linearly ordered by Turing-reduction*. Considering the diverse fields that these examples come from, the following observation of further confluence surely reminds one of Kanamori and Magidor's appeal to *diversity*:

> The halting problem K has degree 0', and one can show that so do WP [word problem for finitely generated groups] and HTP [Hilbert's tenth problem]. And if we apply the Turing jump again, we get 0". It turns out that TF is equivalent to 0". If we take another jump, we get to 0"', which happens to be Turing equivalent to COF. (ibid.)

All the natural examples of uncomputable sets end up being linearly ordered. All of them are Turing-equivalent to iterates of Turing jumps. This so-called "linearity phenomenon" marks yet another example of confluence that has led to a conjecture. According to Montalban, "Martin's conjecture is a formal statement trying to capture the essence of … the following empirical observation: while the infinite sequences in $\mathbb{N}^{\mathbb{N}}$ are not linearly ordered by Turing computability, the naturally occurring sequences are." (ibid.)

Conjectures similar in spirit to Martin's conjecture abound in mathematical logic. The confluence engendered by the linearity phenomenon[15] has a particularly close relative in axiomatic theories[16]. The abstract of Friedman, Rathjen, and Weiermann (2013) begins with the proclamation: "The fact that 'natural' theories, i.e. theories which have something like an 'idea' to them, are almost always linearly ordered with regard to [consistency] strength has been called one of the great mysteries of the

[15] For a proposed counterpoint, see Hamkins (2025).

[16] There is yet another close relative, i.e., all natural theories extending ZF are equiconsistent with large cardinal axioms and well-ordered by consistency strength. This phenomenon has not motivated overarching conjectures like Martin's conjecture, but rather piecemeal consensus, e.g., that the determinacy hypotheses would be equiconsistent with some large cardinals (which we now know) or that Proper Forcing Axiom will be equiconsistent with a supercompact cardinal (which we don't). So I would like to consider this phenomenon as serving some other purpose that we will return to later, rather than guiding conjectures, although it certainly does so at points.

foundation of mathematics." This observation led Montalban and Walsh to prove an analogue in Montalbán and Walsh (2019)[17].

Elsewhere, Bagaria (2023) has shown that a variety of large cardinal principles are equivalent to instances of what he calls the general principle of Structural Reflection[18]. For Bagaria, the equivalences serve to induce confidence in the assertion that "all large cardinals are in fact different manifestations of a single general reflection principle." And eventually this led to the following conjecture in Bagaria and Ternullo (2025): "Conjecture 1. every known large cardinal notion is equivalent to some (form of) [structural reflection]."

Let me pause here and recognize that the kinds of reasoning surveyed above amount to mostly just ordinary Humean induction. The usual tropes of prediction, inductive evidence, and empirical confirmation and fallibility[19] are all at work here. Is it fair to consider them confluence arguments? Let me bite the bullet here and say yes. Note, however, that I am not equating this kind of things with the kind of confluence undergirding the Church-Turing Thesis, although I do want to say that they are closely related in this case. Instead, I am simply stipulating that my liberal understanding of the term "confluence" is meant to encompass both situations (and more). I can offer two justifications for this move: first, the latter was clearly responsible for the former in guiding the conjectures of e.g., Church and Post in the early days of computability theory; and second, such interpretive latitude will later turn out useful in understanding the divorce between successful instances of confluence arguments and the premise of an underlying background notion.

Recall that Church was confident that whatever definition Gödel proposed would be equivalent to his own. That confidence stemmed partly from his belief that he had the right definition. In contrast, Bagaria supplies a disclaimer to the effect that he "[does] not wish to claim that the [structural reflection] principle is what Gödel had in mind when talking about reflection of internal structural properties of the membership relation." So for Bagaria, structural reflection principles are not intended as an attempt to capture any pre-theoretic notion, but rather they are an interesting family of principles, motivated from what Cantor and Ackermann had said about absolute infinity, to which many large cardinal notions happen to be equivalent. It is this "complex, and multi-faceted, ramification of principles, and the equivalence results with large cardinal notions obtained" that led him to conjecture that all large cardinals are equivalent to some form of structural reflection.

For a final example of Conjecture Heuristic provided by confluence, let us turn to another most famous open problem: P vs.NP. In the survey Aaronson (2016), Aaronson makes numerous cases for conjecturing $P \neq NP$. In doing so, he managed to sum up, in my view, what it is that provides Conjecture Heuristic from confluence:

[17] I refer the interested reader to recent works by James Walsh, in particular Walsh (2025) where both the philosophy and the mathematics are elaborated.

[18] This refers to principles of the form "For every definable, in the first-order language of set theory, possibly with parameters, class $\mathcal{C}$ of relational structures of the same type there exists an ordinal α that reflects $\mathcal{C}$, i.e., for every A in $\mathcal{C}$ there exist B in $\mathcal{C} \cap V_\alpha$ and an elementary embedding from B into A" (Bagaria, 2023, p. 30).

[19] Section 4.2 of Copeland (2024) surveys skepticisms about these inductive and equivalence arguments in the case of the Church-Turing Thesis.

> To my mind, however, the strongest argument for $P \neq NP$ involves the thousands of problems that have been shown to be NP-complete, and the thousands of other problems that have been shown to be in P. If just one of these problems had turned out to be both NP-complete and in P, that would've immediately implied $P = NP$. Thus, we could argue, the hypothesis has had thousands of chances to be "falsified by observation". (Aaronson, 2016, p. 25)

Certainly, here Aaronson is telling the usual story of inductive confirmation that we are adequately used to in the empirical sciences. But the added strength from diversity makes it resemble Kleene's argument by confluence. That is, for all the discoveries of NP-complete and P-problems, all from diverse areas, no one has been able to equate the two classes[20].

At the beginning of this section, I described Conjecture Heuristic as carrying the least philosophical baggage. I hope it is now clear why I said so, considering what I have sketched is something all too familiar for the working mathematician: once a pattern occurs numerous times, from numerous sources, one is motivated in conjecturing that this pattern will continue.

Note that this is not to say that the pattern is the decisive reason to make such a conjecture[21]. Of the myriad of NP-complete problems, the assertion that any one of them has a polynomial-time solution is equivalent to $P = NP$. This list of equivalences does not seem to carry any thrust for conjecturing that P is indeed equal to NP. After all, mathematicians are of course free to conjecture on whatever grounds they see fit - natural curiousity, wild guesses, or friendly challenge to colleagues. My point is simply that, under favorable circumstances, the observed confluence makes it a good rational guess that facts of the matter will turn out to be one way rather than another. It is this credence-bolstering aspect of confluence that I want to highlight here, which will play a part in later sections.

Now, confluence was at least partially responsible for Church's and Post's conjectures that certain definitions of effective calculability would be equivalent. Indeed, the repeated confirmation of such conjectures formed a firm basis of our acceptance of the later Church-Turing Thesis. Going one step further, common interpretation of the thesis entails that it does not really matter which definition one chooses. This particular attitude has materially influenced how mathematicians go about their work, the manner of which I shall proceed to analyze in the next two sections.

[20] Notice, however, that Kleene's argument by confluence appeals to positive evidence: all definitions have been proven to be equivalent; whereas here Aarsonson is appealing to negative evidence: none of the putative ramifications of $P = NP$ has ever been observed.

[21] I thank an anonymous referee for pointing this out.

3 Rigor Assurance

3.1 Another look at practices in computability theory

Despite its deeply philosophical nature, in mathematical practice the term "Church-Turing Thesis" most frequently appears in the phrase "proof by Church(−Turing)'s Thesis", as a practical means to an end. Again, this is a markedly different use of confluence arguments than, say, the one in justifying the received wisdom of the Church-Turing Thesis, alluded to in the Introduction. In particular, such appeals are not intended to signal that we have uncovered new truths of the One True Notion of Computable (though that might very well be the case). Instead, when mathematicians invoke the Church-Turing Thesis in their work, such an invocation serves as warrants for something else. The goal of the remainder of the paper is to pin down precisely the range of possibilities as to what that something else may be, and to argue that the main thrust of such invocations is not to be found in the Church-Turing Thesis (understood as a philosophical thesis about the nature of computation), but rather in the broader interplay between confluence, evidence, and justification.

For a first example, consider the following passage from the first chapter of André Nies's influential textbook *Computability and Randomness*. Having defined computable functions in terms of Turing machines, Nies is quick to note that

> [M]any other formal definitions for the intuitive notion of a computable function were proposed. All turned out to be equivalent. This lends evidence to the Church-Turing thesis[22] which states that any intuitively computable function is computable in the sense of [Turing machines]. More generally, each informally given algorithmic procedure can be implemented by a Turing program. We freely use this thesis in our proofs: we give a procedure informally and then take it for granted that a Turing program implementing it exists. (Nies, 2012, p. 3)

Note that Nies is making a confluence argument here. And yet, despite making a case resembling Church's and Kleene's, Nies is not, at least not in the present context, applying the thesis to promise the reader that the computability notion studied in the book is the One True Notion (Nies, 2012 is of course more of a mathematics textbook than a philosophical treatise). Rather, it serves to anticipate a popular practice in the computability theory literature: that when it comes to proofs and arguments, informal language will be preferred over formal manipulations. Furthermore, the reader is assured that no loss of rigor will be incurred in doing so.

The preceding example illustrates what I call **Rigor Assurance**. The following passage from Rogers is almost tailor-made to characterize it:

> These techniques [of relying on informal descriptions of algorithms] have been developed to a point where (a) a mathematician can recognize whether or not an

[22] It is worth mentioning that this is the first place in the book where the term "the Church-Turing thesis" occurs.

> alleged informal algorithm provides a partial recursive function … and where (b) a logician can go from an informal definition for an algorithm to a formal definition … [S]uch methods … permit us to avoid cumbersome detail and to isolate crucial mathematical ideas from a background of routine manipulation. We shall see that much profound mathematical substance can be discussed, proved, and communicated in this way … Of course, [anyone arguing this way] … must be prepared to supply formal details if challenged. Proofs which rely on informal methods have, in their favor, all the evidence accumulated in favor of Church's Thesis. (Rogers, 1987, p. 20)

Admittedly, the pull of Rigor Assurance partially derives from simple inductive credence (this point is part of the reason for having the previous section): once one has seen a certain amount of formalizations of informal proofs, one gets accustomed to such translation procedures, and can be assured that similar things can be done in the next case.

But I claim there is more to it than that. Sure, inductive credence will for the most part safeguards translations from informal description of (say) some read-write-transition procedure to formal Turing machines. And yet the computability theory parlance is blessed with a wealth of colorfully informal language, some having no familiar machine model within reach - with such gems as computerized squirrels (Mansfield & Weitkamp, 1985, p. 43) and blinking creatures (Salo, 2020). If anything, such informality has only aided the communication of mathematical ideas, not hindered it. The point is, having seen the vast confluence of equivalent definitions of "computable", one can be confident that informal descriptions of an algorithm can be formalizable as the formal algorithms, be it of Turing machines or whatever else; and in the event that one's formalization departs from the usual definitions, one is justified in believing that the difference is harmless, as long as the end result seems to obey reasonable criteria[23] of what counts as computation[24].

3.2 Examples outside of computability, and what they tell us about a debate in the philosophical literature

The practice of proving something by Church's Thesis is well-documented in the literature. Its practical and philosophical merits are extensively studied[25], for example, in San Mauro (2018), which will now serve as my main interlocutor in the remainder of this paper. My reason for engaging with San Mauro this extensively is so that I can leverage certain subtleties in San Mauro's analysis, which will help me draw out and clarify the distinction between what I call Rigor Assurance here and Coding Invariance later, a finer distinction that is often overlooked in the literature.

[23] Recall Rogers (1987, p. 18)'s optimism along the same lines in the previous section.

[24] In the present context, this serves only to warrant trust in using informal language in proofs. But I shall claim that this invariance under different choice of formalisms is itself an additional facet to confluence, which will be the theme of the next section.

[25] For more general discussions on the relationship between informal proofs, formal proofs, and rigor, see e.g., Hamami (2022), Stanley Tanswell (2024)

San Mauro (2018) is concerned with how the Church-Turing Thesis is used in computability theory literature - more specifically whether it is just a matter of offloading straightforward but tedious tasks like many other fields of mathematics. To start, San Mauro conspicuously calls the aforementioned practice "the practical side of Church-Turing Thesis", referring namely to the collection of all the appeals to the thesis that appear in some steps of some mathematical proof[26]. Although I will eventually argue that there is an additional dimension to this seemingly uniform practice, let me cite San Mauro's identification of the following principle ("The Church-Turing Bridge") allowing rigor to be assured:

(The Church-Turing Bridge) If any informal description of an algorithm can be formally implemented in each model of computation (e.g., assuming the Church-Turing Thesis) then, in order to prove that something is computable, it is sufficient to describe an informal way to compute it and then make reference to the Church-Turing Thesis.

For an instance of (The Church-Turing Bridge) in practice[27], here is Goldbring and Hart forecasting their choice of the informal:

> There are many approaches to formalizing [computability] (e.g. Turing-machine computable functions and recursive functions) and all known formalizations can be proven to yield the same class of functions. This latter fact gives credence to the Church-Turing thesis, which states that this aforementioned class of functions is indeed the class of functions that are computable in the naïve sense described above. In the rest of this paper, we will never argue about this class of functions using any formal definition but will only argue informally in terms of some kind of algorithm or computer; this is often referred to as arguing using the Church-Turing thesis. Goldbring and Hart (2021)

Presumably, Goldbring and Hart (2021) are not primarily concerned with discovering what is and what is not computable in the intuitive sense, the way perhaps the earliest works on the Entscheidungsproblem or Hilbert's tenth problem were. It merely treats computability as a tool to solve problems in the model theory of operator algebras. In other words, the role played by the Church-Turing Thesis in their paper is merely to assure rigor and simplify presentation. The evidence from confluence accrued for the Church-Turing Thesis is being used to support the belief that this type of informality will not lead to error.

And indeed, to assure rigor, there is no need to appeal to the Church-Turing Thesis at all. Consider historically the first paper relating computability and measure theory. The article Leeuw et al. (1956) is concerned with machines equipped with access to outcomes of random experiments and aims to show how a machine may or may

[26] I refer the reader to standard computability theory textbooks or to San Mauro (2018) for examples.

[27] Supplementing San Mauro's comment that "the most immediate source of observations regarding how such practice has to be intended comes from the kind of expository remarks [in textbooks]", we will see plenty of such instances in the specialist journal articles too.

not be enhanced to take advantage of them[28]. Before delving into the technical arguments, the authors announce:

> [The proofs in the paper] can be considered to be indications of how the theorems, if they were stated formally in the language of recursive function theory, could be proved formally using the tools of that theory. This formalization is not carried out in the paper since it would detract from the conceptual content of the proofs. However, it should be clear to anyone familiar with recursive function theory that the formalization can be carried out

Let us focus now on de Leeuw et al.'s justification of using informal natural language in their proofs. Here we see the same kind of Rigor Assurance as in Nies's textbook and Goldbring and Hart's article, this time without explicitly invoking the Church-Turing Thesis. To the authors, formalization serves more to distract than to illuminate, and hence it ought to be kept at minimum. Furthermore, doing so is safe, because the vast body of work in computability theory has assured the reader that they can trust that such formalizations are possible.

In contexts more detached from classical computability theory, the ability to translate from the informal to the formal is deemed as a facilitating virtue. For example, in pioneering works of infinitary computation (*not* classical computability theory), Rigor Assurance allowed Hamkins and Lewis to "avoid getting bogged down in … minutiae … reading what would otherwise resemble computer code." Hamkins and Lewis (2000) In Carl's survey of the later works of infinite-time decidable equivalence relations, this is considered an advantage: "… ITTM-reducibility has the advantage [of having] rather intuitive informal descriptions which can be used in proofs … One may say that 'proof by Church-Turing thesis' is more readily available for ITTM-reducibility than for Borel reducibility" (Carl, 2019, p. 263).

To really highlight the role Rigor Assurance (and to see that this role can really be decoupled from other commitments brought by the Church-Turing Thesis), consider α-recursion theory. Greenberg (2020) opens with a survey of the array of equivalent definitions of admissible computability[29] and an intricate network of related facts showing how α-computability shares many of the analogous results of ordinary computability. Having primed readers of the multi-faceted nature of α-computaility, Greenberg writes:

> In general, working in α-computability, with experience, we apply some kind of Church-Turing thesis to α-computable functions … we eventually cease to write down precise Σ_1 formulas … Instead, we develop an intuition as to what constitutes "legal" α-computable manipulations of α-finite objects (elements

[28] We will return to these technical results in Part 2.

[29] For example, for an admissible ordinal α and $A \subseteq \alpha$ the following are equivalent: 1. A is $\mathbf{\Sigma}_1$-definable in L_α. 2. A is computably enumerable by Koepke's α-Turing machines. 3. A is semi-decidable by Koepke's α-register machines. See Koepke and Seyfferth (2009) and Carl (2019, Theorem 3.3.3) for definitions and proofs.

of L_α), and get a sense of the "time" that a process takes; if it takes fewer than α steps, then it "halts"

What "kind of Church-Turing thesis" is at work here? The "intuition" that Greenberg says the reader develops is certainly not one about some far-fetched notion of computability that deals with the transfinite, waiting to be captured by α-computability, but rather, it is one about how to translate informal language used in proofs to formal arguments. Perhaps more accurately, having seen the confluence of definitions for α-computability and the appropriate generalizations of the familiar results from ordinary computability, the reader is expected to develop confidence in the use of informal language. In the absence of a conspicuous informal notion to be captured by α-computability, Greenberg's appeal to the "kind of Church-Turing thesis" in studying the α-computable functions can only be interpreted to mean an appeal to Rigor Assurance.

These examples attest to the ubiquity of Rigor Assurance in practice. I hope to have shown that many of these are not really invocations of the received wisdom of the Church-Turing Thesis proper, but rather appeals to confluence of definitions as means of crossing San Mauro's Church-Turing Bridge.

Much of the remainder of San Mauro (2018) is concerned with adjudicating a controversy about the significance of this practice. San Mauro identified a tension between the ubiquity and usefulness of this practical side of the Church-Turing Thesis on the one hand, and on the other hand its seeming triviality and lack of depth. He characterizes this latter attitude by what he calls the Standard View, attributing it to his interlocutors (San Mauro, 2018, 236):

1. The Church-Turing Thesis allows us to rely on informal methods (by (The Church-Turing Bridge));
2. Yet, these methods are in the end just a matter of convenience: informal definitions point towards formal ones, and we could theoretically substitute the former with the latter without any significant loss or gain of information;
3. This operation is analogous to what happens in most parts of mathematics.

San Mauro's interlocutors are authors who do not see much difference between "proof by Church's Thesis" and other routine mathematical drudgeries. Odifredi, for example, calls Rigor Assurance "avoidable … does not even require a Thesis: it is just an expression of a general preference, widespread in mathematics, for informal (more intelligible) arguments, whenever their formalization appears to be straightforward, and not particularly informative" (Odifreddi, 1989, as cited in San Mauro, 2018)

He also describes Epstein and Carnielli as the proponents of the Standard View, who have judged[30] in their computability textbook that "To invoke Church's thesis

[30] A side note: San Mauro is not being entirely fair to Epstein and Carnielli, who enthusiastically call the equivalences of definitions of computability *The Most Amazing Fact* in Epstein and Carnielli (2008), and commendably dedicate a large portion to the discussion of the Church-Turing Thesis and its attendant evidence and philosophies of mathematics. Here Epstein and Carnielli is responding to Daniel Cohen, who described Church's Thesis as "... more than a philosophical statement about the nature of computability. It is a useful tool in proofs. To show that the function is partial recursive [formally] ... [is] not likely to give

when "the proof is left to the reader" is meant amounts to giving a fancy name to a routine piece of mathematics while at the same time denigrating the actual mathematics" (Epstein & Carnielli, 2008), as cited in (San Mauro, 2018)

At any rate, both Odifredi's and Cohen's sentiments are concrete demonstrations of a key thesis of the present paper, that many invocations of the Church-Turing Thesis are really appeals to something else much less thesis-looking, and Epstein and Carnielli is just one step short of making this point: Cohen is indeed pointing out the salient practice of invoking Church's Thesis in proofs, but what he is referring to is merely Rigor Assurance in disguise.

San Mauro (2018) eventually arrives at the diagnosis that the Standard View omits a key part of computability and hence should be rejected: "[The Standard View] fails to represent a central phenomenon of Computability, that of conceiving most constructions as absolute, i.e. independent from the background formalism and yet not to be regarded as incomplete" (San Mauro, 2018, p. 246). To make this point, San Mauro considers the proof of the existence of a simple set[31]. As with most proofs in computability theory, the proof begins with a numbering of the partial computable functions (i.e., a c.e. surjection from natural numbers to programs), and then proceeds to construct a simple set from this numbering. San Mauro notes that the proof does not specify the background enumeration on which the proof is based, and yet the proof is considered complete, because a set remains simple under all acceptable (in a technical sense) numberings.

And this places considerable burden on proponents of the Standard View, who would claim that, barring further details like what coding is used, such a semi-formal proof "has to be intended as a sort of a prototype, to be completed by specifying a certain numbering" and "would correspond to a general method for describing, for each acceptable numbering, a corresponding simple set". This view is mistaken, says San Mauro:

> … not specifying the background numbering … subsumes the kind of practical use of [the Church-Turing Thesis] that we have extensively discussed … The theory of simple sets is invariant with respect to the acceptable numbering we choose to work with, making this very choice superfluous. So, against the Standard View, … [the simple set constructed] does not refer to any of its formal definitions … [and] although our informal proof does provide a method for producing, for all numberings, a given simple set … [T]o collapse the meaning of such proof to this method would correspond with claiming that an implicit reference to numberings is somehow needed to make complete sense of our

any insight ... [and is] usually replaced by an appeal to Church's Thesis." (Cohen, 1987). p. 104; as cited in (Epstein & Carnielli, 2008), p0.231 In my view, San Mauro should have attributed the Standard View to Cohen, who argues most forcefully for it in the quoted passage, and not to Epstein and Carnielli, who are merely pointing out that it is mistaken to call this practice "invoking Church's Thesis", especially when the latter is in fact much more involved.

[31] A simple set is a co-infinte, computably enumerable set of natural numbers that meets every infinite, computably enumerable set. The reader is not required to know the definition or the proof, only that the existence of a simple set is not a trivial fact, and that simplicity is invariant under different choice of enumerating the computably enumerable sets.

> construction … Rather, the notion of simplicity is better understood as an absolute one, i.e. independent from the chosen formalism. (ibid., pp. 242–243)

I see the tension identified by San Mauro as a disagreement about commitment. For San Mauro, invoking Church's Thesis in a proof commits its participants (i.e., its writers and readers) to more. His point is that, by invoking the Church-Turing Thesis in a proof, we see ourselves as dealing with an absolute notion of computability (The One True Notion), instead of merely backing away from the cumbersome details of formalization and leaving them to the reader. Most investigations in computability theory are like this (that is, studying objects invariant under choice of coding). Rogers (1987), for instance, seems to agree that Church's Thesis is committing us to dealing with The One True Notion, when he wrote the following after introducing the concept of Gödel numbering of programs:

> The use of codings raises an immediate question of invariance. Once a coding is chosen, will the formal concept partial recursive function on code numbers correspond to the informal notion algorithmic mapping on the uncoded expressions? As the latter notion is informal, the answer must be, in part, empirical. Church's Thesis provides an affirmative answer.(p. 28)

Proponents of the Standard View, on the other hand, are characterized as thinking there is no such commitment, seeing as they reduce the practice of invoking Church's Thesis to a matter of time-saving practicality, something routine with no additional significance. And in doing so, they miss out on this key formalism-free aspect of computability.

Contrary to San Mauro's characterization, however, I will claim that "not specifying the background numbering" should not subsume "the kind of practical use of the Church-Turing Thesis" considered so far (i.e., Rigor Assurance), and that the quoted passage from Rogers above is in fact pointing to another distinct use of confluence arguments, in establishing a kind of **Coding Invariance** (the subject of the next section). Furthermore, I shall argue that the tension between the practical side of the Church-Turing Thesis and proponents of the Standard View arises from a misalignment of what role each party is expecting Church(−Turing)'s Thesis to play; or better yet, what each party is trying to achieve with confluence arguments. Both sides are right, and there is in fact no serious disagreement. I will elaborate this point in the next section.

4 Coding Invariance

4.1 What is it, how is it different from Rigor Assurance, and how this distinction resolves San Mauro's debate

By way of transition, let us notice that Rogers is making a subtly different point in the quoted passage above. Recall when he promised Rigor Assurance, he meant "a logician can go from an informal definition for an algorithm to a formal definition

… [S]uch methods … permit us to avoid cumbersome detail." This is assurance that some formalization exists and is easily available, whereas his remark on codings guarantees that it does not matter which one.

Kleene (1952) seems to assume that these are two different facets to the confluence of various definitions. Having made his argument by confluence, he lists the stability under different codings as a separate piece of evidence for Church's Thesis: "Of less weight, but deserving mention, is the circumstance that several formulations of the main notions are equivalent; i.e. the notions possess a sort of 'stability'."(p. 320)

Kleene proceeds to cite the proofs of equivalences of different ways of formalizing the same idea. I should stress that, crucially, Kleene is not citing (say) the equivalence between general recursiveness and λ-definability here - he has already made that argument before - he is citing the equivalences of (say) different variants of λ -definability:

> The notion of λ-definability has the variants λ-K-definability … and λ-δ-definability … also there is a parallel development, started by [Schönfinkel, Curry, and Rosser], which leads to a notion that we may call combinatory definability, proved equivalent to λ-definability by Rosser.(p. 321)

Kleene would go on to mention similar developments in other families of formalizations. But his main point is clear: invariance under the choice of codings is different from guarantee that formalization is possible in principle.

Why does this invariance matter? In classical computability theory, just as in its higher cousin hyperarithmetic theory, a common practice is to talk about things in terms of their codes. This is in some sense inevitable, because things like ordinals or well-orderings are not a native part of the theory's vocabulary. So one finds need to utilize codings for constructive ordinals or the H_a in hyperarithmetic theory[32]. And for Moschovakis (2016), such uses provide additional insight:

> Codings are useful for expressing succinctly uniform properties of coded sets. Their general theory is technically messy, not very interesting mathematically and certainly not worth putting here … It is clear that propositions like Lemmas 5.3.1 and 5.3.2 which hold uniformly for a certain coding also hold uniformly for every equivalent coding—and for some of them the proof might be easier[17] [Moschovakis's footnote]. We exploit this idea by establishing an elegant characterization of [the class of hyperarithmetic sets] which produces a coding for it equivalent to the classical one in (5.23) but much simpler. (pp. 120–121)

The latter part of Moschovakis's remark, which he conspicuously titled Coding Invariance, is further explained in footnote 17, where he references exactly the kind of "central phenomenon of Computability" that San Mauro champions:

[32] See Moschovakis (2016) for details. The reader only need to know that these are fundamental concepts in hyperarithmetic theory.

> For a classical example, consider the coding of recursive partial functions specified by [Kleene's Normal Form Theorem]. Its precise definition depends on the choice of computation model that we use, Turing machines, systems of recursive equations or whatever [that is, each choice of model gives rise to a different enumeration of the recursive partial functions], but all these codings are equivalent and so uniform propositions about them are coding invariant. (ibid.)

Note also, that by "technically messy, not very interesting mathematically", Moschovakis is certainly not referring to the kind of informal-to-formal translation that proponents of the Standard View are claiming to be messy and routine. A brief glance at Moschovakis (2016) will show that he makes very little use of informal language characteristic of classical computability theory, mostly because he has no need to describe any algorithm at all! So the worry here is certainly not about the validity of proofs (something that would otherwise be assuaged by Rigor Assurance), but that we might be led to pseudo-insights that are really just byproducts of the peculiarities in our choice of coding.

Immediately following Moschovakis (2016) in the same collection is Boker and Dershowitz (2016), who take the perils of coding peculiarities as a point of departure and proposed a framework for characterizing them. Boker & Dershowitz motivate this issue of *honest* versus *dishonest* coding with a few examples. To name one, consider the usual practice in (theoretical) computer science of encoding a graph as some kind of data structure, e.g., a list of nodes and edges. The point Boker and Derschowitz make is that if one insists or happens to encode a graph by listing the nodes in the order of a Hamiltonian path (if one exists), then deciding the existence of such a path, a classic NP-complete problem, can be solved in linear time. This is a dishonest coding, because it is the choice of encoding that makes the problem easy. And in fact, this will have the unwelcomed consequence that one "will only be able to quickly answer the specific question whether there is a Hamiltonian path, whereas she would have a much harder time performing basic graph operations, such as adding an edge."

They go on to make a point strongly resembling Moschovakis's, that ultimately Coding Invariance safeguards us from such unnecessary worries:

> The [complexity] of a problem turns out to be essentially independent of the particular encoding scheme and computer model used for determining time complexity … In fact, the standard encoding schemes used in practice for any particular problem always seem to differ at most polynomially from one another. It would be difficult to imagine a "reasonable" encoding scheme for a problem that differs more than polynomially from the standard ones (Garey & Johnson, 1979 as cited in Boker & Dershowitz, 2016, p. 153)

Appealing to this Coding Invariance aspect of confluence is exactly how Sipser dispels worries about choice of computation models after introducing the complexity class P in his classic textbook on computational complexity:

> In complexity theory, we classify computational problems according to their time complexity. But with which model do we measure time? The same language may have different time requirements on different models
>
> Fortunately, time requirements don't differ greatly for typical deterministic models. So, if our classification system isn't very sensitive to relatively small differences in complexity, the choice of deterministic model isn't crucial … All reasonable deterministic computational models are polynomially equivalent. That is, any one of them can simulate another with only a polynomial increase in running time. (Sipser, 2013, p. 286)

Let us take stock and a moment of reflection. The existence of, say, a universal enumerating programs is not due to some whimsical definition of a program or a Turing machine, which might smuggle in a particular way of encoding the computable functions. It is a substantive (or absolute, in San Mauro's terms) property of the partial computable functions that there is one that enumerates all of them, which holds no matter how one codifies Turing machines. This perhaps also explains the painstaking length that Rogers (1987) has gone to, again and again whenever a new concept is introduced, in showing that the concept is invariant in one way or another. The point is, anything worth their computability-theoretic salt will not technically depend on identifying these computable functions as the ones implemented by a specific formalization of Turing machines; any model of computation will allow us to prove the same thing[33].

So, to summarize the difference in somewhat slogan form: Rigor Assurance guarantees no rigor is lost in translation. Coding Invariance guarantees no extraneous peculiarities is smuggled in by codification.

This key insight allows us to resolve the tension identified by San Mauro (2018). Recall that the proponents of the Standard View claims that "proofs by Church's Thesis" is a completely routine and humdrum practice, nothing more significant than saying "the proof is left to the reader." San Mauro, on the other hand, argues that the Standard View fails to capture the central phenomenon of computability, that of considering most constructions as absolute, i.e., invariant under the choice of codings. I claim this is merely a misunderstanding: Rigor Assurance with the use of informal language is of course routine and unremarkable, it is simply the kind of thing that

[33] An anonymous referee has pointed out to me that this point may be contentious. Say one is to construct a particular set witnessing a certain property, then starting from two different fixed coding schemes there is no prior guarantee that the two versions of the constructed set will be e.g., Turing equivalent. This seems to be quite the opposite of invariance. To address this point, I claim that there are two meanings of invariance at work here. On the one hand, the constructed set may very well be different in each coding. On the other hand, the desired property that this set is meant to satisfy is always true of the set (otherwise one would not have the right proof!). This distinction is the content of Rogers's remark in Rogers (1987, p. 22): "Our fixed choice of Gödel numbering ... appears to be a rather noninvariant feature of our theory. We shall see ... that our results possess an invariant significance independent of this choice. In this respect, our use of a particular Gödel numbering is much like the use of a particular coordinate' system to establish coordinate free results in geometry." An entire (Chapter 4) there is devoted to this notion of invariance (e.g., under computable bijections). See also the recent Grabmayr (2021) for a more focused discussion

licenses professional mathematicians to choose not to spell out every last detail of a proof or leave it to the reader. Coding Invariance, on the other hand, is a much more profound and substantive property of computability, one that is characteristic of it and not many other fields of mathematics (at least not in every one where someone has left an exercise to the reader). And it is this that is being invoked when we say that the proof of the existence of a simple set is complete without specifying the background numbering.

Thus, the tension identified by San Mauro stems from a misalignment of expectations on both parties. They are misaligned on what specific goal each party takes the other to be characterizing. Proponents of the Standard View would like to reduce Rigor Assurance to proof-left-to-reader, something that is routine and unremarkable, while San Mauro wants to claim that Coding Invariance carries nontrivial significance, in the form of committing us to the substantive parts of computability theory that are formalism-independent. Both parties are of course right. However, on appearance they seem to be arguing about a singular goal of a singular practice - which I hope to have demonstrated to not be the case. Confluence arguments are actually being used to serve two different purposes, and that is the key insight.

4.2 More on the distinction between Rigor Assurance and coding Invariance, with examples outside of classical computability

Returning to the matters at hand, we observe that Coding Invariance extends beyond just the computable and the hyperarithmetic. Recall the following Spector-Gandy-type characterization of Σ^1_2 sets, with which Chong et al. (2019) set up the rest of their paper:

Theorem 1 Given a set $A \subseteq 2^\omega$, the following are equivalent

1. A is Σ^1_2, that is, definable in second-order arithmetic by a formula of the form $\exists f \forall g \varphi$, where f, g are real number variables and φ only quantifies over the naturals.
2. There is a Σ_1-formula φ such that for all reals x, $x \in A \iff L_{\omega_1^{L[x]}}[x] \vDash \varphi(x)$.
3. There is a Σ_1-formula φ such that for all reals x, $x \in A \iff L_{\delta^1_2(x)}[x] \vDash \varphi(x)$, where $\delta^1_2(x)$ is sup of the ordertypes of $\Delta^1_2(x)$ well orderings of ω.

Having reminded the reader of the classic theorem that will be used throughout their paper, Chong et al. make an appeal to a Church-Turing-type thesis: "Theorem 1.1 enables one to use recursion-theoretic arguments to study Σ^1_2-sets. It follows that there is a version of 'Church-Turing Thesis' for Σ^1_2-sets that we can appeal to in the construction of such sets."

Chong et al. (2019) serves as yet another instructive example in distinguishing Rigor Assurance and Coding Invariance. Here, again, they are not anticipating the use of informal description of algorithms. They are instead preparing the reader for their liberal context-switching, utilizing different codings or representations of the Σ^1_2 sets as they see fit. Granted, this confluence at Σ^1_2 licenses the authors to talk freely about, for example, "effectively computing an index of a Π^1_1 set" and "fix[ing]

a L_{ω_1}-effective enumeration of perfect trees". This much is Rigor Assurance, similar to Greenberg's application of his "Church-Turing Thesis" for admissible computability (Greenberg, 2020). But the key point here is not that the reader needs assurance that such use of informal language will not render the proofs invalid. Quite on the contrary, the reader is promised that translation is always available, not between informal language and formal language, but between "languages" from different fields of mathematics, so the proof is not led off-topic and end up talking about something else entirely.

For another example of how coding peculiarities might smuggle in pseudo-insight, consider the following well-known fact in proof theory and recursion theory[34]. The theorem concerns transfinite progressions of consistency statements. Noting that one can only talk about limit stages of such progressions by talking about codes of computable well-orderings, we define $T_0 := \mathsf{ZFC},\ T_{2^n} := T_n + \mathsf{Con}(T_n),\ T_{3\cdot 5^e} = \mathsf{ZFC} \cup \bigcup T_{\Phi_e(n)}$. Now coding peculiarity follows:

Theorem 2 (Turing's completeness theorem) *For every true Π^0_1 sentence φ, there exists a notation in d in Kleene's $\mathcal{O}$ such that φ is provable in T_d.*

Roughly, if we want to study the strength of iterated consistency statements, then the strength of the limit theories are extremely sensitive to the choice of codings. Every true Π^0_1 sentence is provable in some T_d, a sensible candidate for encoding a limit theory. This is an obvious lack of Coding Invariance. The same coding peculiarities also underlie the famous Kreisel's Counterexample (cf. Theorem 7.5.1 in Pohlers (1989)) attesting to the delicacy in treating the notion of proof-theoretic ordinals.

This facet of Coding Invariance is at play in a particularly amusing episode in the history of quantum computation. In one of the earliest papers defining quantum Turing machines and its complexity classes, appearing in *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing* (Bernstein & Vazirani, 1993), the authors made no restriction on what transition amplitudes were allowed. This enabled Adleman et al. (1997) to prove that "the class of sets which are decidable [by Bernstein and Vazirani (1993)'s machines] with bounded error in polynomial time has uncountable cardinality and contains sets of all Turing degrees." This is certainly an artifact of coding peculiarity, and the culprit was quickly identified and corrected - in the journal version (Bernstein & Vazirani, 1997) of Bernstein and Vazirani (1993) added a uniform computability requirement to what quantities are allowed as transition amplitudes, acknowledging Adleman et al. (1997), which was published in the same journal:

> we need to constrain the entries allowed in the transition function of our probabilistic TM. Otherwise, it is possible to smuggle hard-to-compute quantities into the transition amplitudes, for instance by letting the ith bit indicate whether the ith deterministic TM halts on a blank tape … As in the case of probabilistic

[34] The standard reference is Feferman (1962). See also the recent article Pakhomov et al. (2025)

> TM, we must limit the transition amplitudes [of quantum Turing machines] to efficiently computable numbers. (Bernstein & Vazirani, 1997, p. 1417)

Careful choices were then made to ensure that, in terms of computability (i.e., what are computable simpliciter), the resulting machines are equivalent to classical Turing machines, citing Adleman et al. (1997) and an unpublished manuscript by Solovay and Yao. The resulting definition of quantum Turing machines is now the canon. This prompts Peter Shor[35] to ask if a mathematical definition can be wrong (Shor, 2020).

At this point, I hope to have convinced the reader that Coding Invariance is sufficiently distinct from Rigor Assurance, as two distinct purposes that are served by confluence arguments in mathematical practice. Despite not making the distinction, San Mauro (correctly, in my view) elucidates this facet of Coding Invariance by citing Burgess's notion of *indifferentism* (Burgess, 2008, 2015). For Burgess,

> [T]he general phenomenon of the indifference of working mathematicians to certain kinds of decisions that have to be made in any codification of mathematics … two analysts who wish to collaborate do not need to check whether they were taught the same definition of 'real number'. (Burgess, 2015)

In the theory of Borel equivalence relations, Gao's thesis is an epitome of confluence-driven indifferentism. The rich theory of Borel equivalence relations aims to provide a complexity hierarchy to the various classification problems that arise in other areas in mathematics. The theory, also known as invariant descriptive set theory[36], incorporates tools from effective and classical descriptive set theory, and, making substantial use of measure and category machinery, applies them to show how one classification problem is reducible or irreducible to another.

In the literal sense, what is being proved is the (non-)existence of Borel (or otherwise definable) reductions[37] between Borel equivalence relations on Polish spaces. The significance of such technical results crucially depends on the choice of coding: the choice of topology, which provides a Borel structure, and the choice of how specific mathematical objects are coded as elements of certain Polish spaces. In the absence of any sort of motivating story like Turing's, there is no prior safeguard against coding peculiarities.

In practice, however, this choice rarely matters. Reaching the end of the book and reflecting on the vast amount of application to concrete classification problems, Gao summarizes, having just finished a long proof of "the satisfactory correspondence" between two such actual choices:

[35] whom I thank for this curious bit of history and the exact references.

[36] The eponymous textbook Gao (2008) covers the state of the art, to which we refer the interested reader. We will presently be concerned with what Gao has to say about the practice of invariant descriptive set theorists.

[37] Let E be a Borel equivalence relation on a standard Borel space X and similarly F a Borel equivalence relation on a standard Borel space Y. We say E is *Borel reducible* to F (written as $E \leq_B F$) iff there is a Borel function $f : X \to Y$ such that $x_1 E x_2 \Leftrightarrow f(x_1) F f(x_2)$. Such a function f is called a *Borel reduction* of E to F.

> The existence of such an isomorphism means that the two approaches to equip [the space in question] with a standard Borel structure are equivalent. Note that it is mathematically nontrivial to prove this equivalence. However, in practice, whenever we have different approaches to equip standard Borel structures on hyperspaces … they always end up to be equivalent despite the difficulty of the proof. (Gao, 2008, p. 328)

This motivates Gao's Thesis[38]:

> (Gao's Thesis). For any class $\mathcal{H}$ of mathematical structures, if (X_1, Ω_1) and (X_2, Ω_2) are two standard Borel spaces naturally coding elements of $\mathcal{H}$, then there exists a Borel isomorphism $f : X_1 \to X_2$ such that $f(x)$ and x are isomorphic as mathematical structures for every $x \in \mathcal{H}$.

For a light application of Gao's Thesis, consider Gao's indifference to what he takes countable groups to be when he studies reductions among equivalence relations between them: A majority of our equivalence relations are defined on $2^{\mathbb{N}}$ or some variations such as $2^{\mathbb{N}\times\mathbb{N}}$. We view elements of this space to be either functions from $\mathbb{N}$ to $2 = \{0, 1\}$ or subsets of $\mathbb{N}$; and we freely switch our point of view without explicit mention.. (Gao, 2004) Many other such examples can be found in Kaya (2016, Chap. 3).

Gao expresses a sentiment about Coding Invariance which we should now be thoroughly familiar with:

> For readers familiar with computability theory, the statement is similar to the Church-Turing Thesis … To a large extent such work [i.e., proving equivalences between coding] is mathematically insignificant and irrelevant to the understanding of the main problem. For our purpose the objective is to understand the complexity of various classification problems for, say, Polish metric spaces. While the theorems in this section are nontrivial and their proofs contain important ideas that can be used later to tackle the classification problems, the statements of the theorems themselves are not helping in our understanding of the complexity of the classification problems. (Gao, 2008, p. 328)

By now we have seen the same attitude already expressed by Moschovakis, Garey & Johnson, and to some extent Boker & Dershowitz. Telling is the contrast with Gao's patent appeal to Rigor Assurance in the same textbook, occurring earlier during the computability preliminaries, anticipating the use of informal language:

> All formal definitions of computability have been shown to be equivalent to the above one [i.e., general recursiveness]. In fact, when working with computable functions we will not deal with the details of the above definition, but will rather adopt the Church-Turing Thesis, which states that any of the formal

[38] The name Gao's Thesis is given by Hamkins (2016). The version here is taken from Kaya (2016, p. 35), a version that both Hamkins and Kaya agree stays closer to how the thesis is actually used in practice.

> definitions of computable functions captures the intuitive notion of computability. Thus if a function is intuitively computable by an informal algorithm then by the Church-Turing Thesis we may conclude that it is formally computable without checking the details of the formal definitions. (Gao, 2008, p. 24)

I take this contrast to be decisive evidence that Rigor Assurance and Coding Invariance are definitively distinct, although they are sometimes conflated in the case of classical computability. The kind of Coding Invariance that motivated Gao's Thesis guarantees that interesting results pertaining to the hierarchy of Borel equivalence relations truly reflects the structure of that hierarchy, rather than being the result of some clever (one might say deviant) encoding/representation of the Borel structure of the spaces in question. The Rigor Assurance engendered by Gao's appeal to the Church-Turing Thesis, on the other hand, assures that our proofs' validity is not compromised by the use of informal language. In spite of their surface similarity, the two theses serve distinct purposes.

In sum, Rigor Assurance safeguards us from worries that a mathematical proof is rendered invalid by the use of informal/natural language; whereas Coding Invariance warrants that the insights obtained from the theorems are not peculiarities smuggled in by the choice of coding/representation. Once the distinction is recognized, the tension identified by San Mauro can, as I have argued above, be satisfactorily resolved by seeing that both parties are right in their respective understanding of purposes that their confluence arguments are (or are not) doing.

Now, the importance of Gao's Thesis motivated Hamkins (2016) to ask for a counterexample on the one hand (this remains unanswered), and for a philosophical analysis for how Gao's Thesis is analogous to the Church-Turing Thesis, as well as whether one can provide a Turing-esque philosophical grounds in support of the thesis. We have partially addressed this latter question, but much of the philosophical territory awaits exploration. Part 2 in this series of papers will provide more conceptual tools to figure out how the two theses are alike exactly, and how they are not. Eventually I shall argue that such a Turing-esque motivating story is not necessary, and the empirical observation of these equivalences alone suffices for the purpose at hand, as Gao's Thesis does not engender all the same kinds of philosophical commitments as the Church-Turing Thesis, but only this very practical aspect of Coding Invariance.

5 Conclusion

To sum this paper up, taking mathematical practice at face value, I have attempted to survey a number of confluence arguments that have appeared in print, and I have tried to characterize **Rigor Assurance**, **Coding Invariance**, and (less seriously) **Conjecture Heuristic** as purposes served by these arguments which are relatively light on philosophical commitments but which facilitate progress in mathematical works. It has hopefully been shown that such classification not only provides insights into our mathematical practice, but also may naturally resolve certain debates in the philosophical literature.

Now, in addition to being a versatile practical tool and a safeguard for formal rigor, confluence arguments can also be very enticing philosophically. The familiar Church-Turing Thesis is already a prime example of this, with many authors maintaining that confluence speaks to the fundamentality, correctness, and robustness of the underlying computability concept. This is far from being an isolated instance: philosophical arguments that are driven by confluence also abound in the literature. One such is the tenability of a Church-Turing-style thesis for algorithmic randomness that has been recently discussed in Franklin (2021), Porter (2021), as well as *Rudolph's Thesis* about the confluence of natural models for measure-preserving transformations (Foreman, 2010). Elsewhere, the convergence of evidence regarding determinacy hypotheses has played a key role in their justification (Koellner, 2006; Rittberg, 2020), and the confluence of generalized constructibility (Kennedy, 2020; Kennedy et al., 2021) has been the crux of recent interests in the idea of formalism-freeness. Again, confluence will play a variety of roles there, in ways that are orthogonal to each other, which are almost disjoint from the ones considered in this paper.

All these more philosophically involved confluence arguments will be the main theme of the sequel to this paper, where a similar taxonomy will be provided for them. One message that Part 2 shall share with the present Part 1 is that, for confluence to be productively appealed to, it is not necessary to presuppose a single underlying informal notion awaiting to be captured, as the case of the Church-Turing Thesis might otherwise suggest. I hope to have gestured in this part that, once such a point of view is adopted, a satisfactory analysis emerges, cutting across a wide range of mathematical practices that are shown to be closely related.

Acknowledgements The author would like to thank Joel David Hamkins, Penelope Maddy, and especially Toby Meadows for extensive discussions while working on this paper. He would also like to thank the Pen Pals, the audiences at APA Central 2024 and Notre Dame's Logic Reading Group, and the logic groups at Fudan University and Peking University for useful feedback.

Author contributions Jason Zesheng Chen is the sole author of this paper.

Funding No funding was received for this work.

Data availability This is a theoretical paper that engages with the philosophical and mathematical literature. No empirical data or materials were used in the preparation of this paper.

Declarations

Conflict of interest The author declares that they have no conflicts of interest to disclose.

References

Aaronson, S. (2016). P=? NP. In J. N. J. Forbes, & M. T. Rassias (Eds.), *Open problems in mathematics* (pp. 1–122). Springer International Publishing. https://doi.org/10.1007/978-3-319-32162-2_1

Adleman, L. M., DeMarrais, J., & Huang, M.-D. A. (1997). Quantum computability. *SIAM Journal on Computing*, *26*(5), 1524–1540. https://doi.org/10.1137/S0097539795293639

Bagaria, J. (2023). Large cardinals as principles of structural reflection. *Bulletin of Symbolic Logic*, *29*(1), 19–70. https://doi.org/10.1017/bsl.2023.2

Bagaria, J., & Ternullo, C. (2025). Intrinsic justification for large cardinals and structural reflection. *Philosophia Mathematica*, *33*(2), 123–154. https://doi.org/10.1093/philmat/nkaf006

Bernstein, E., & Vazirani, U. (1993). Quantum complexity theory. In *Proceedings of the twenty-fifth annual ACM symposium on theory of computing* (pp. 11–20). Association for Computing Machinery. California, USA. https://doi.org/10.1145/167088.167097

Bernstein, E., & Vazirani, U. (1997). Quantum complexity theory. *SIAM Journal on Computing*, *26*(5), 1411–1473. https://doi.org/10.1137/S0097539796300921

Berto, F., & Tagliabue, J. (2023). Cellular automata. In E. N. Zalta, & U. Nodelman (Eds.), *The Stanford encyclopedia of philosophy* (Winter 2023 ed.). Metaphysics Research Lab, Stanford University.

Boker, U., & Dershowitz, N. (2016). Honest computability and complexity. In E. G. Omodeo, & A. Policriti (Eds.), *Martin Davis on computability, computational logic, and mathematical foundations* (Vol. 10, pp. 151–173). Springer International Publishing. (Series Title: Outstanding Contributions to Logic) https://doi.org/10.1007/978-3-319-41842-1_6

Burgess, J. P. (2008). *Mathematics, models, and modality: Selected philosophical essays*. Cambridge University Press.

Burgess, J. P. (2015). *Rigor and structure*. Oxford university press.

Carl, M. (2019). *Ordinal computability: An introduction to infinitary machines.* De Gruyter. https://doi.org/10.1515/9783110496154

Chong, C. T., Wu, L., & Yu, L. (2019). Basis theorem for Σ^1_2-sets. *The Journal of Symbolic Logic*, *84*(1), 376–387. https://doi.org/10.1017/jsl.2018.81

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, *58*(2), 345. https://doi.org/10.2307/2371045

Cohen, D. E. (1987). *Computability and logic*. Chichester, West Sussex, E. Horwood; Halsted Press.

Copeland, B. J. (2024). The church-turing thesis. In E. N. Zalta, & U. Nodelman (Eds.), *The Stanford encyclopedia of philosophy* (Spring 2024 ed.). Metaphysics Research Lab, Stanford University.

Crossley, J. N. (2006). Reminiscences of logicians. In *Algebra and logic: Papers from the 1974 summer research institute of the Australian mathematical society, monash university, australia* (pp. 1–62). Springer.

Davis, M. (1982). Why Gödel didn't have church's thesis. *Information and Control*, *54*(1–2), 3–24. https://doi.org/10.1016/S0019-9958(82)91226-8

Epstein, R. L., & Carnielli, W. A. (2008). *Computability: Computable functions, logic and the foundations of mathematics* (3. ed.). Advanced Reasoning Forum.

Feferman, S. (1962). Transfinite recursive progressions of axiomatic theories. *Journal of Symbolic Logic*, *27*(3), 259–316. https://doi.org/10.2307/2964649

Foreman, M. (2010). Models for measure preserving transformations. *Topology and Its Applications*, *157*(8), 1404–1414. https://doi.org/10.1016/j.topol.2009.06.021

Franklin, J. N. Y. (2021). A church-turing thesis for randomness? In L. D. Mol, F. M. A.Weiermann, & D. Fernández-Duque (Eds.), *Connecting with computability* (pp. 217–226). Springer International Publishing. https://doi.org/10.1007/978-3-030-80049-9_20.

Friedman, S.-D., Rathjen, M., & Weiermann, A. (2013). Slow consistency. *Annals of Pure and Applied Logic*, *164*(3), 382–393.

Gandy, R. (1988). The confluence of ideas in 1936. In *A half-century survey on the universal turing machine* (pp. 55–111). Oxford University Press, Inc.

Gao, S. (2004). The homeomorphism problem for countable topological spaces. *Topology and its applications*, *139*(1–3), 97–112. https://doi.org/10.1016/j.topol.2003.09.005

Gao, S. (2008). *Invariant descriptive set theory* (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781584887942

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman & Co.

Gödel, K. (1934). *On undecidable propositions of formal mathematics systems*. Institute for Advanced Study. In S. C. Klein, & J. B. Rosser (Eds.), *(Notes from lectures at the Institute for Advanced Study*.
Goldbring, I., & Hart, B. (2021). Operator algebras with hyperarithmetic theory. *Journal of Logic and Computation*, *31*(2), 612–629. https://doi.org/10.1093/logcom/exaa059
Göodel, K. (2001). *Collected works. 1: Publications 1929 - 1936* (1. issued as paperback ed.). Oxford University Pr.
Grabmayr, B. (2021). On the invariance of Gödel's second theorem with regard to numberings. *The Review of Symbolic Logic*, *14*(1), 51–84. https://www.cambridge.org/core/product/identifier/S1755020320000192/type/journal
Greenberg, N. (2020). Two applications of admissible computability. In *Contemporary logic and computing* (pp. 604–637). College publications.
Hamami, Y. (2022). Mathematica rigor and proof. *The Review of Symbolic Logic*, *15*(2), 409–449. https://doi.org/10.1017/S1755020319000443
Hamkins, J. D. (2016). *Is there a class of mathematical structures with non-isomorphic natural representations as a standard Borel space?* MathOverflow. https://mathoverflow.net/q/234585)
Hamkins, J. D. (2025). Nonlinearity and illfoundedness in the hierarchy of large cardinal consistency strength. *Monatshefte für Mathematik*, *208*(4), 687–728. https://doi.org/10.1007/s00605-025-02082-1
Hamkins, J. D., & Lewis, A. (2000). Infinite time turing machines. *Journal of Symbolic Logic*, *65*(2), 567–604. https://doi.org/10.2307/2586556
Kanamori, A., & Magidor, M. (1978). The evolution of large cardinal axioms in set theory. In G. H. Müller, & D. S. Scott (Eds.), *Higher set theory* (pp. 99–275). Springer.
Kaya, B. (2016). *Cantor minimal systems from a descriptive perspective* (Doctoral Dissertation). Rutgers University.
Kennedy, J. (2013). On formalism freeness: Implementing Gödel's 1946 Princeton bicentennial lecture. *Bulletin of Symbolic Logic*, *19*(3), 351–393. https://doi.org/10.2178/bsl.1903030
Kennedy, J. (2020). *Gödel, Tarski and the lure of natural language: Logical entanglement, formalism freeness* (1st ed.). Cambridge University Press. https://doi.org/10.1017/9780511998393
Kennedy, J., Magidor, M., & Väänänen, J. (2021). Inner models from extended logics: Part 1. *Journal of Mathematical Logic*, *21*(2), 2150012. https://doi.org/10.1142/S0219061321500124
Kleene, S. C. (1952). *Introduction to metamathematics*. Van Nostrand.
Kleene, S. C. (1981). Origins of recursive function theory. *IEEE Annals of the History of Computing*, *3*(1), 52–67. https://doi.org/10.1109/MAHC.1981.10004
Koellner, P. (2006). On the question of absolute undecidability†. *Philosophia Mathematica*, *14*(2), 153–188. https://doi.org/10.1093/philmat/nkj009
Koepke, P., & Seyfferth, B. (2009). Ordinal machines and admissible recursion theory. *Annals of pure and applied logic*, *160*(3), 310–318. https://doi.org/10.1016/j.apal.2009.01.005
Leeuw, K. D., Moore, E. F., Shannon, C. E., & Shapiro, N. (1956). Computability by probabilistic machines. In C. E. Shannon, & J. McCarthy (Eds.), *Automata studies. (AM-34)* (pp. 183–212). Princeton University Press. https://doi.org/10.1515/9781400882618-010
Maddy, P. (1988). Believing the axioms. I. *The Journal of Symbolic Logic*, *53*(2), 481–511. https://doi.org/10.2307/2274520
Maddy, P. (2019). What do we want a foundation to do? In S. Centrone, D. Kant, & D. Sarikaya (Eds.), *Reflections on the foundations of mathematics: Univalent foundations, set theory and General thoughts* (pp. 293–311). Springer International Publishing. https://doi.org/10.1007/978-3-030-15655-8_13
Mansfield, R., & Weitkamp, G. (1985). *Recursive aspects of descriptive set theory (No.11)*. Oxford University Press; Clarendon Press.
Montalbán, A. (2019). Martin's conjecture: A classification of the naturally occurring Turing degrees. *Notices American Mathematics Society*, *66*(8), 1209–1215.
Montalbán, A., & Walsh, J. (2019). On the inevitability of the consistency operator. *The Journal of Symbolic Logic*, *84*(1), 205–225.
Moschovakis, Y. N. (2016). Hyperarithmetical sets. In E. G. Omodeo, & A. Policriti (Eds.), *Martin Davis on computability, computational logic, and mathematical foundations* (Vol. 10, pp. 107–149). Springer International Publishing. (Series Title: Outstanding Contributions to Logic) https://doi.org/10.1007/978-3-319-41842-1_5
Nies, A. (2012). *Computability and randomness*. OUP Oxford.
Odifreddi, P. (1989). *Classical recursion theory: The theory of functions and sets of natural numbers*. Elsevier Science Pub. Co.

Pakhomov, F., Rathjen, M., & Rossegger, D. (2025). Feferman's completeness theorem. *The Bulletin of Symbolic Logic*, *31*(3), 462–487. https://doi.org/10.1017/bsl.2025.2

Pohlers, W. (1989). *Proof theory (Vol. 1407)*. *Springer Berlin Heidelberg*. https://doi.org/10.1007/978-3-540-46825-7

Porter, C. (2021). The equivalence of definitions of algorithmic randomness. *Philosophia Mathematica*, *29*(2), 153–194. https://doi.org/10.1093/philmat/nkaa039

Post, E. L. (1936). Finite combinatory processes—formulation I. *Journal of Symbolic Logic*, *1*(3), 103–105. https://doi.org/10.2307/2269031

Rittberg, C. J. (2020). Mathematical practices can be metaphysically laden. In B. Sriraman (Ed.), *Handbook of the history and philosophy of mathematical practice* (pp. 1–26). Springer International Publishing. 22-1. https://doi.org/10.1007/978-3-030-19071-2_22-1

Rogers, H. (1987). *Theory of recursive functions and effective computability* (1st ed.). MIT Press.

Salo, V. (2020). *Answer to "What is the borel complexity of this set?"*. MathOverflow. https://mathoverflow.net/a/363475/147031

San Mauro, L. (2018). Church-turing thesis, in practice. In M. Piazza, & G. Pulcini (Eds.).*Truth, existence and explanation: FilMat 2016 studies in the philosophy of mathematics* (pp. 225–248). Springer International Publishing. https://doi.org/10.1007/978-3-319-93342-9_13

Shor, P. (2020). *Can a mathematical definition Be wrong? MathOverflow.* https://mathoverflow.net/q/31358/147031

Sieg, W. (2006). Gödel on computability. *Philosophia Mathematica*, *14*(2), 189–207. https://doi.org/10.1093/philmat/nkj005

Sipser, M. (2013). *Introduction to the theory of computation* (3rd ed.). Cengage Learning.

Stanley Tanswell, F. (2024). *Mathematical rigour and informal proof* (1st ed.). Cambridge University Press. https://doi.org/10.1017/9781009325110

Walsh, J. (2025). On the hierarchy of natural theories. *The Bulletin of Symbolic Logic*, 1–34. https://doi.org/10.1017/bsl.2025.10137