Absolute Computability and the Grounding of the Church-Turing Thesis

Marina Yu, June 2026

1 Introduction

Gödel's incompleteness theorems show that mathematical truth cannot be reduced to provability in any single formal system. Provability is in this sense relative. Computability is not. The models of computation developed in the 1930s—Turing machines, lambda calculus, the recursive functions—all pick out the same class of functions, and that class does not change when we change the formalism. This paper asks why one notion should be absolute when the other is relative. I argue that the answer is epistemological.

Making the contrast precise requires the formal machinery on both sides. I begin with the relativity of provability, building Gödel's first incompleteness theorem from the undecidability of Robinson arithmetic, before turning to the absoluteness of computability and, finally, to the epistemology of the Church-Turing Thesis.

2 Gödel and Robinson Arithmetic

2.1 Theory Q

In order to study the limits of formal proof, we work with Robinson arithmetic (Q), a deliberately minimal theory of arithmetic.[1]

To bridge the gap between meta-theoretic natural numbers $n \in \mathbb{N}$ and object-language terms within, we define the $\overline{n}$ for each natural number n as the term $0''^{\cdots\prime}$ containing exactly n successor symbols. Despite the weakness of Q—it cannot even prove algebraic laws like commutativity of addition—it is strong enough to represent all computable functions.

We formalize this capacity through representability. A k-ary function $f(x_0, \ldots, x_{k-1})$ from $\mathbb{N}$ to $\mathbb{N}$ is *representable* in Q if there exists a formula $\varphi_f(x_0, \ldots, x_{k-1}, y)$ such that whenever $f(n_0, \ldots, n_{k-1}) = m$, Q can derives both:

$$Q \vdash \varphi_f(\overline{n}_0, \ldots, \overline{n}_{k-1}, \overline{m})$$
$$Q \vdash \forall y\, (\varphi_f(\overline{n}_0, \ldots, \overline{n}_{k-1}, y) \to \overline{m} = y)$$

The central result is that a function is representable in Q if and only if it is computable [1, Theorem 4.3.2]. A parallel result holds for relations: a relation $R(x_0, \ldots, x_{k-1})$ is rep-

[1] The axioms of Q are: (1) $x' = y' \to x = y$; (2) $0 \neq x'$; (3) $x \neq 0 \to \exists y\,(x = y')$; (4) $x + 0 = x$; (5) $x + y' = (x + y)'$; (6) $x \times 0 = 0$; (7) $x \times y' = x \times y + x$; and (8) $x < y \leftrightarrow \exists z\,(z' + x = y)$ [1, Section 4.3].

resentable in Q if and only if it is computable, where representability now requires that $Q \vdash \varphi_R(\bar{n}_0, \ldots, \bar{n}_{k-1})$ for true instances and $Q \vdash \neg\varphi_R(\bar{n}_0, \ldots, \bar{n}_{k-1})$ for false ones [1, Theorem 4.3.12].

2.2 The Halting Problem and the Decidability of Q

We now examine Q as a set of sentences, to ask whether membership in that set is decidable. Recall that Q is the deductive closure of its eight axioms. By introducing a numerical coding that maps object-language formulas and proofs to natural numbers, we define a proof predicate $\mathrm{Pr}_Q(x, y)$, which holds if and only if x codes a valid derivation in Q of the sentence coded by y [1, Section 4.2, 4.4]. Because we can algorithmically check the rules of first-order logic, $\mathrm{Pr}_Q(x, y)$ is a primitive recursive relation. Therefore, the set of sentences provable in Q viewed as a set of codes is:

$$Q = \{y \mid \exists x \, \mathrm{Pr}_Q(x, y)\}$$

Since this is the existential quantification of a primitive recursive predicate, Q is computably enumerable (c.e.) [1, Section 4.4].

To show that Q is *not* decidable, we reduce the halting problem to it. Define $K_0 = \{\langle e, x\rangle \mid \varphi_e(x)\!\downarrow\}$, the set of pairs for which the e-th partial computable function halts on input x. We show $K_0 \leq_m Q$, so that any decision procedure for Q would also decide K_0—which is known to be impossible [1, Section 3.3, 4.4].

Kleene's predicate $T(e, x, s)$, which asserts that program e on input x produces a halting computation coded by s, is primitive recursive [1, Section 2.7]. By Theorem 4.3.12, it is therefore representable in Q by a formula $\varphi_T(e, x, s)$. Define the reduction function f by mapping each pair $\langle e, x\rangle$ to (the code of) the sentence $\exists s \, \varphi_T(\bar{e}, \bar{x}, s)$.

We verify that f is indeed a many-one reduction. If $\langle e, x\rangle \in K_0$, there is a witness n such that $T(e, x, n)$ holds. Since φ_T represents T, we have $Q \vdash \varphi_T(\bar{e}, \bar{x}, \bar{n})$, and hence $Q \vdash \exists s \, \varphi_T(\bar{e}, \bar{x}, s)$. Conversely, suppose $Q \vdash \exists s \, \varphi_T(\bar{e}, \bar{x}, s)$. If $\langle e, x\rangle \notin K_0$, then $T(e, x, n)$ is false for every n, so representability gives $Q \vdash \neg\varphi_T(\bar{e}, \bar{x}, \bar{n})$ for each n. A theory that both refutes every instance $\varphi_T(\bar{e}, \bar{x}, \bar{n})$ and proves the existential $\exists s \, \varphi_T(\bar{e}, \bar{x}, s)$ is ω-inconsistent, contradicting soundness of Q over the standard model. Thus $\langle e, x\rangle \in K_0$ if and only if $f(\langle e, x\rangle) \in Q$, establishing $K_0 \leq_m Q$. Because K_0 is a complete c.e. set, Q is likewise a complete c.e. set and in particular is undecidable [1, Section 4.4].

2.3 Gödel's First Incompleteness Theorem

If Q, the bare-minimum theory of arithmetic, is already undecidable, then any consistent extension of Q inherits this undecidability. Since any such extension still represents all computable relations, any decision procedure for it could be turned into a decision procedure

for Q. This means it is fundamentally impossible to capture all arithmetical truth within a single, consistent, algorithmically verifiable set of axioms. The following formalizes this intuition.

Recall that a theory T is *complete* if for every sentence φ in its language, either $T \vdash \varphi$ or $T \vdash \neg\varphi$, and *computably axiomatizable* if there is an algorithm that can decide whether a given sentence is an axiom. By Lemma 4.4.8, if a theory T is both complete and computably axiomatizable, it is decidable: given φ, simultaneously search for a proof of φ and of $\neg\varphi$. Completeness guarantees this search will terminate, and consistency ensures we will not find both [1, Section 4.4].

Suppose, for contradiction, that some consistent, computably axiomatized extension T of Q were complete, and thus decidable. In particular, we define a binary relation $R(x, y)$ that holds if and only if x codes a unary formula $\theta_u(u)$ and $T \vdash \theta_x(\overline{y})$. Because T is decidable, $R(x, y)$ is a computable relation. Additionally, since every unary computable (partial recursive) relation is representable in Q, and T extends to Q, we know that $R(x, y)$ is universal. This means that for any unary computable relation $P(y)$, there is an index k such that $P(y) \leftrightarrow R(k, y)$ for all y [1, Section 4.4].

We now construct such a $S(y)$ by diagonalizing out of R:

$$S(y) \equiv \neg R(y, y)$$

Since R is computable, S is also a computable relation. By the universality of R, there must exist a fixed index k such that $S(y) \leftrightarrow R(k, y)$ for all y. Evaluating this equivalence at the diagonal input where $y = k$ yields:

$$R(k, k) \leftrightarrow S(k) \leftrightarrow \neg R(k, k)$$

This gives u s a contradiction, which proves that a universal computable relation cannot exist. Therefore, no consistent, computably axiomtized extension of Q is decidable, and by Lemma 4.4.8, none is complete [1, Section 4.4].

While this shows that an independent sentence must exist, it does not exhibit one, which is where Gödel's fixed-point construction comes in. For any computably axiomatized extension T of Q, let $\mathrm{Prov}_T(y)$ for the formula $\exists x\, \mathrm{Pr}_T(x, y)$, expressing "$y$ is provable in T". By, Gödel's fixed-point lemma [1, Lemma 4.5.1] there exists a φ satisfying:

$$T \vdash \varphi \leftrightarrow \neg\mathrm{Prov}_T(\ulcorner\varphi\urcorner)$$

The sentence φ asserts its own unprovability in T. If $T \vdash \varphi$, then there is a code m for a valid derivation of φ, meaning $\mathrm{Pr}_T(m, \ulcorner\varphi\urcorner)$ holds. Therefore, $T \vdash \mathrm{Prov}_T(\ulcorner\varphi\urcorner)$, and by the equivalence, $T \vdash \neg\varphi$, which makes T inconsistent. So if T is consistent, $T \nvdash \varphi$. Because T

cannot prove φ, the statement φ is a true arithmetical fact that escapes the deductive reach of T, rendering the theory incomplete [1, Section 4.6].

Gödel's First Incompleteness Theorem (1931)

No complete, consistent, computably axiomatized extension of Q exists.

Any formal system for arithmetic must therefore sacrifice at least one of the following: completeness, consistency, or computable axiomatizability. If we insist on a system that is both consistent and algorithmically verifiable, it will inevitably remain incomplete.

2.4 Philosophical Implications

The proof of Theorem 4.4.9 carries significant consequences for the philosophy of mathematics. Before Gödel, the foundational landscape was dominated by Frege's logicism and Hilbert's formalism. Hilbert's program, and subsequent conventionalism, reduced mathematical truth to syntactic derivability within formal systems. On this view, a mathemtical truth is nothing more than a theorem.

The incompleteness results show this cannot be the case. The Gödel sentence φ is independent of T, yet true in the standard model of arithmetic $\mathcal{N} = \langle \mathbb{N}, 0, ', +, \times, < \rangle$. If T is consistent, φ is unprovable, which is exactly what φ asserts, so φ is true. This creates an asymmetry between truth and provability that the formalist program cannot accommodate.

Gödel himself adopted mathematical realism or Platonism, and read this asymmetry as direct support for it. If mathematical truth was just codification or linguistic convention, φ would have no determinate truth value outside of T. The realist holds that we recognize φ as true anyways because we can reason metamathematically about the standard model of the natural numbers from outside any formal system. An anti-realist can grant that φ is true and still deny that any abstract objects follow from it. So, the asymmetry by itself cannot settle the question.

3 Turing Invariance

3.1 The Phenomenon

While the boundaries of provability are sensitive to the choice new axioms, the boundaries of computability are remarkably stable. During the 1930s, a foundational convergence occurred: Turing's mechanical formulation of computability via tape-driven devices, Church's algebraic framework of the λ-calculus, and Kurt Gödel and Kleene's arithmetic characterization of the general recursive functions were all proved to be extensionally equivalent. Despite originating from completely different conceptual starting points, each system captures the exact same class of partial functions from $\mathbb{N}$ to $\mathbb{N}$ [1, Section 1.1, 2.6, 2.8].

This convergence is codified in the Church-Turing Thesis: any function that is computable in the intuitive sense is computable by a Turing machine, and vice versa [1, Section 1.1]. This identifies a notion of computability that is *absolute* in the sense that it is invariant across all known models of computation. Changing the syntax, programming language, or physical implementation of a computation procedure does not change which functions are computable, but only how such a function is described.

3.2 Kleene's Normal Form

The mathematical justification for this invariance is Kleene's Normal Form Theorem (Theorem 2.7.3) [1, Section 2.7], which states that there exists a single, fixed primitive recursive relation $T(e, x, s)$ and a primitive recursive function $U(s)$ such that any partial computable function $f(x)$ can be expressed as:

$$f(x) \simeq U(\mu s\, T(e, x, s))$$

for some index e [1, Sections 2.7, 3.1]. The relation $T(e, x, s)$ asserts that s codes a complete, halting computation sequence of program e on input x; the minimization operator μs searches for the least such s; and U extracts the output from that computation sequence [1, Section 2.6, 2.7]. The index e is just a natural number encoding the specific instructions of the procedure, so every computable function corresponds to at least one such index.

The theorem is precisely why computability is absolute. The components T and U are primitive recursive, meaning they only need bounded loop structures and basic arithmetic operations to evaluate [1, Section 2.3, 2.4]. Crucially, any model of computation that can evaluate primitive recursive predicates can simulate any other model by varying only the index e [1, Section 2.1, 2.4, 2.8]. The class of computable functions is therefore not an artifact of any particular computational framework, but a structural feature of mathematics itself.

3.3 Partiality and Closure

To understand why computability is absolute when provability is not, it is instructive to examine how each concept responds to diagonalization. As established in Section II, attempting to construct a total, complete theory of provability fails because diagonalizing out of the proof predicate forces an independent sentence that the theory can neither prove nor disprove [1, Section 4.4, 4.6].

Diagonalization applied to the class of total computable functions, produces an analogous failure. Suppose we effectively enumerate all total computable functions, $\phi_0, \phi_1, \phi_2, \ldots$, and

define:

$$h(x) = \phi_x(x) + 1$$

By construction, h differs from every φ_e at the point $x = e$, so h cannot appear anywhere on the list [1, Section 2.5, 2.7]. If we insist that h must be total, we reach the same impasse as in the provability case: the class of total computable functions cannot be enumerated by any function within that same class, and any attempt to extend the class to include h simply produces a new class with the same deficiency.

However, when we admit *partial* functions, that is, functions that may be undefined on some inputs due to non-halting computations. The universal function $Un(e, x) \simeq U(\mu s\, T(e, x, s))$ is partial computable, and the diagonal construction yields:

$$g(x) \simeq Un(x, x) + 1$$

When we evaluate g at its own index k, we get $g(k) \simeq Un(k, k) + 1$, which is simply undefined—there is no contradiction [1, Section 2.7].

This asymmetry is the reason why computability is absolute while provability is not. Provability cannot tolerate partiality: a sentence in first-order logic is bivalent and when a self-referential construction forces a dilemma, the system must either assert a falsehood (inconsistency) or leave a truth uncaptured (incompleteness). Closing that gap necessitates new axioms, which merely shifts the boundaries of provability to a new, relative system without eliminating the underlying limitation. Computability, by contrast, accommodates undefinedness by allowing the diagonal paradox to resolve as a non-haling computation. Since this is internalized by the partial recursive functions, the boundary of computability does not have to change, and thus remains absolute [1, Section 2.5, 2.7, 4.4].

4 Epistemology and the Church-Turing Thesis

4.1 The Vulnerability of Pure Syntax

The epistemological dilemma of the Church-Turing Thesis lies in its attempt to establish a strict identity between an informal, pre-theoretic concept—human effective calculability—and a rigorous mathematical formalism [7]. However, we cannot treat the Church-Turing Thesis as a standard mathematical theorem, since it is a category error: one cannot formally prove an equivalence between a physical, cognitive reality and a syntactic object. To understand why this epistemological bridge cannot be constructed through syntax alone, we must examine what Gödel's incompleteness results reveal about the relationship between formal systems and the informal intuitions they are meant to capture.

Church proposed that the intuitive boundaries of calculation were perfectly coextensive

with the formal rules of λ-definability [3]. Unlike Turing, who offered an analysis of what a human computer does, Church attempted to settle the question by stipulation: $\lambda-$definability simply *is* what we mean by effective calculability. At first blush, it may seem that this definitional strategy immunizes the thesis against the kind of incompleteness worries that beset formal proof systems, since if $\lambda-$definability just is the domain, there is no external standard against which it could be found wanting in the Gödelian sense.

This apparent strength disguises a difficulty I will call the *grounding problem*, which can be understood as a radicalization of Carnap's explication problem [2]. Carnap recognized that replacing an informal concept with a formal one requires the explicatum to be genuinely *similar* to the explicandum (in addition to *exact, fruitful,* and *simple*). The grounding problem asks how such faithfulness is to be verified. I argue that verification requires what we might call *independent epistemic access*, that is, some grip on the informal concept that is not itself wholly mediated by the formalism whose adequacy is in question. Without this independent access, any claim that the formal system has correctly tracked the informal notion is either circular or unfalsifiable.

Even if we grant that $\lambda-$definability is extensionally equivalent to every competing formalization of computability, it does not establish that what they agree on *is* effective calculability. Convergence, in this sense, is horizontal, where the question is vertical. That is, it certifies coherence among the candidate explicata but does not address the fidelity of any of them to the explicandum. As Gödel himself noted (which I cite as evidence, not authority), what would be needed for a fully satisfying analysis is a notion of computability that is *absolute* [4]. Absoluteness, so understood, is itself a horizontal property. A uniform formalization is perfectly consistent with a uniform omission. Invariance is thus necessary for having captured effective calculability, but cannot be sufficient.

Church's stipulation, then, does not answer the foundational question so much as reframe it: *what justifies our confidence that $\lambda-$definability has correctly tracked effective calculability*? This tells us that the relevant evidence must be different in kind as we have just seen that no relation among formalisms speaks to their fidelity to the concept. What we are after, is precisely some form of independent epistemic access. Because such access cannot be drawn from within any formalism, it must be drawn from without. The justification for the thesis must therefore be explicitly extra-mathematical.

4.2 The Semantic Triumph

Turing resolved this deadlock in 1936 by shifting the entire project from syntactic to semantic reduction. Instead of proposing another symbolic language, he turned to the calculating agent and asked what, of necessity, constrains any process we would recognize as a calculation [7]. The shift is from supplying another candidate explicatum to interrogating the explicandum

directly—and it is precisely this that furnishes the *independent epistemic access*.

The analysis isolates a small set of invariants on the human computer: a finite alphabet of distinguishable symbols; attention restricted at any moment to a bounded portion of the workspace; a finite number of internal "states of mind"; and elementary steps determined by, and altering, only the locally observed configuration together with the current state. What makes this semantic, rather than syntactic, is that the constraints are not features of a notation but of the embodied activity the notation is about [7].

A caveat about the dialectic is owed here. Turing was not historically answering a reframing from Church; the two worked nearly contemporaneously, and Turing's quarry was the *Entscheidungsproblem*. My claim is rather that Turing's analysis *retrospectively supplies* the extra-mathematical warrant that Church's stipulation could only gesture at.

That this is the right diagnosis is corroborated by the one reader whose assent mattered the most. Gödel was conspicuously unmoved by the convergence of formalisms. In 1946, he singled out computability as the first *absolute* epistemological notion and in the 1964 postscript he attributed to Turing's work a precise definition of the general concept of a formal system [4]. Yet, his endorsement was carefully hedged. He charged Turing's argument with a "philosophical error" insofar as it treated the states of mind of a developing mathematician as fixed and finite [5]. This reservation is the seam along which the Platonist payoff opens.

4.3 The Platonist Payoff

Return now to the original question: why should computability be absolute while provability remain relative? Provability is relative because we have no access to proof *überhaupt* except through a formal system; there is no phenomenology of "being a proof" to which it could be reduced. Computability is absolute because Turing supplied exactly such access. Absoluteness, on this view, is an extra-mathematical act of intuition.

This has critical consequences for formalism and nominalism. The very notions on which their programs depend—formal system, algorithm, mechanical procedure, decidable—are the notions Turing's analysis defines. To wield them with the confidence the field now grants is to accept the Church-Turing Thesis. But accepting the thesis, I have argued, requires having solved the grounding problem, and solving it requires non-inferential access to the correct boundaries of an informal concept. The implicit assumption is that there is something like a rational grasp of conceptual structure that outruns both stipulation and observation. That faculty is precisely the one Gödelian Platonism takes as its point of departure.

The result is a dilemma, not a argument for or proof of Platonism. Either there is a fact about where effective calculability's boundaries lie or there is no such fact. If the former is true, then the central Platonist posit is vindicated, at least for concepts. If the latter is true, then the convergence of formalisms is only a contingent consensus and *absoluteness* collapses

into convention.

Turing's analysis bounds the mechanical computer absolutely. Incompleteness then guarantees that any such system adequate for arithmetic contains truths it cannot prove. I do not draw the Lucas-Penrose conclusion that mind is demonstrably no machine. The honest residue is Gödel's own disjunction: either mind is not captured by any finite machine, or there exists absolutely undecidable propositions [6]. Both disjuncts favor the Platonist—the first locates a mind grasping more than any formal system, the second posits truths holding independently of all proof. The same stroke that grounded computation extra-mathematically thereby drew the sharp outline against which mathematical intuition reveals itself.

5 Conclusion

The paper defended a conclusion about computability, but the lever was general. The grounding problem—the demand that a formalism show, by some access independent of itself, that it has captured the concept it replaces—can be applied to any explication, not only the Church-Turing Thesis. Where that access can be found, as Turing found it for computation, a concept can become absolute; where it cannot, we are left with the relativity Gödel found in provability. Whether the access Turing supplied reaches mathematical objects, or stops at our concepts of them, is a question I leave open. But it suggests that the line between the absolute and the relative in mathematics is drawn not by our formalisms, but by whether we can find a standpoint outside them.

Word count: 3140

References

[1] Jeremy Avigad. Computability and incompleteness: Lecture notes, 2007. Version: January 9, 2007.

[2] Rudolf Carnap. *Logical Foundations of Probability*. University of Chicago Press, Chicago, 1950.

[3] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, 1936.

[4] Kurt Gödel. Remarks before the Princeton Bicentennial Conference on Problems of Mathematics. In Martin Davis, editor, *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*, pages 84–88. Raven Press, New York, 1965. Original work published 1946.

[5] Kurt Gödel. Some remarks on the undecidability results. In Solomon Feferman, Jr. John W. Dawson, Stephen C. Kleene, Gregory H. Moore, Robert M. Solovay, and Jean van Heijenoort, editors, *Kurt Gödel: Collected Works, Volume II: Publications 1938–1974*, pages 305–306. Oxford University Press, New York, 1990.

[6] Kurt Gödel. Some basic theorems on the foundations of mathematics and their implications. In Solomon Feferman, John W. Jr. Dawson, Warren Goldfarb, Charles Parsons, and Robert M. Solovay, editors, *Kurt Gödel: Collected Works, Volume III: Unpublished Essays and Lectures*, pages 304–323. Oxford University Press, New York, 1995. Josiah Willard Gibbs Lecture, delivered in 1951.

[7] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[1] This paper also draws on the material of PHIL 162 which I am currently taking.