Simulation Invariance and Its Limits in Higher-Topos Computability

Anonymous

31 October 2026 — Version 1.0

Manuscript prepared for Higher Categorical Semantics

Abstract

We study whether effective computability eliminates the distinction between a directly executed algorithm and a simulated algorithm. The unrestricted assertion that every formal definition of reality and simulation is equivalent is false: a computable provenance bit already gives a counterexample. We establish a precise replacement. An adequate universal interpreter preserves partial input-output behavior, and every predicate that descends to that behavior is invariant under simulation. In an arbitrary infinity topos, descent along an effective epimorphism characterizes these predicates exactly by invariance on its kernel pair. We instantiate the result with coded Turing and lambda computations in an explicit presheaf infinity topos, and separate this construction from Hyland's effective topos. Higher semantics retains coherent identifications and structured observations without making equivalence automatic. We give countermodels involving provenance, cost, and group actions, and a relative Rice theorem showing that nontrivial extensional Boolean classifiers are not uniformly computable. The result is an observation-relative non-distinguishability theorem, not an identification of physical reality with formal simulation.

Keywords. Computability; universal interpretation; lambda calculus; infinity topos; effective descent; realizability; observational equivalence.
Mathematics Subject Classification (2020). 03D10, 03G30, 18N60, 68Q55.

1 The question and its necessary qualification

The motivating proposal is that, in an effective higher-topos semantics, there is no mathematical difference between a real algorithm and a simulated algorithm. To evaluate it, we must specify both the objects being compared and the observations allowed. Three claims are easily conflated:

(i) a universal formalism can represent every computation in a specified class;

(ii) an adequate simulation has the same specified behavior as its source;

(iii) every formal predicate, including predicates about provenance and resources, takes the same value on source and simulator.

The first claim concerns expressive power. The second requires a correctness theorem for a translation or interpreter. The third additionally requires that the predicates forget every distinction that the semantics forgets. Computability alone supplies no such requirement.

Our principal conclusion is

$$q(p) = q(\mathrm{sim}(p)) \quad \Longrightarrow \quad D(q(p)) \simeq D(q(\mathrm{sim}(p))), \tag{1}$$

where q records a chosen behavior and D is a predicate, or more generally a family of types, on the space of behaviors. We prove how the premise arises from effective simulation and characterize exactly when a predicate on implementations factors through q. The counterexamples are part of the result: omitting them would turn a valid invariance statement into an invalid universal collapse statement.

The underlying computability facts originate in Church, Turing, and subsequent recursion theory [1, 2, 3]. The categorical machinery is standard higher-topos descent [5]. Our contribution is an explicit synthesis, the separation of its quantifiers, and a collection of exact boundary statements. We do not claim a new proof of the Church–Turing thesis, a new model of higher realizability, or a new undecidability theorem.

We use classical external mathematics, with universes large enough for the categories in question. Arguments identified as internal use the intuitionistic internal language of an infinity topos. The metatheoretic Rice argument is stated separately. No theorem in this manuscript has been checked by a proof-assistant kernel.

2 Effective machines, lambda terms, and an actual simulator

Definition 2.1 (Acceptable computation presentation)**.** Fix an acceptable numbering $(\varphi_e)_{e\in\mathbb{N}}$ of the unary partial recursive functions. Its universal evaluator is a partial recursive map

$$U : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}, \qquad U(e, x) = \varphi_e(x).$$

We assume a computable pairing and a total computable specialization function s with

$$\varphi_{s(a,z)}(x) \simeq_{\mathrm{par}} \varphi_a(\langle z, x\rangle).$$

Here $f(x) \simeq_{\mathrm{par}} g(x)$ means that both sides are undefined or both are defined with the same value. Invalid program encodings are assigned the nowhere-defined computation.

This is a statement about finite descriptions and unbounded abstract execution, not a bound on a physical device. Acceptability is stronger than an arbitrary enumeration of programs: the uniform specialization operation will be used explicitly.

Theorem 2.2 (Effective universal simulation)**.** *There is a total computable function* $\mathrm{sim} : \mathbb{N} \to \mathbb{N}$ *such that*

$$\forall e, x \in \mathbb{N}, \qquad \varphi_{\mathrm{sim}(e)}(x) \simeq_{\mathrm{par}} \varphi_e(x).$$

Moreover, for every finite k*, the* k*-fold iteration* $\mathrm{sim}^k(e)$ *has the same partial behavior as* e*.*

Proof. Choose an index u for the algorithm that, on input $\langle e, x\rangle$, runs the universal evaluator $U(e, x)$. Put $\mathrm{sim}(e) = s(u, e)$. Specialization gives

$$\varphi_{\mathrm{sim}(e)}(x) \simeq_{\mathrm{par}} \varphi_u(\langle e, x\rangle) \simeq_{\mathrm{par}} U(e, x) \simeq_{\mathrm{par}} \varphi_e(x).$$

This equality includes divergence: the interpreter returns exactly when the interpreted computation returns, with the same value. Iteration follows by induction on k and transitivity of partial equality. □

The construction gives an interpreted implementation by specializing an evaluator. It does not promise a distinct numerical code for every wrapper, identical running time, or an infinite tower that completes a computation. Finite towers are covered; a limit of towers is a different construction.

2.1 The lambda-calculus presentation

Fix a deterministic weak call-by-value untyped lambda calculus with finite, effectively coded terms and effective capture-avoiding substitution. Terms have grammar $t ::= x \mid \lambda x.t \mid t\,t$, values are abstractions, and evaluation contexts are $K ::= [\,] \mid K\,t \mid v\,K$. The rule is $(\lambda x.t)v \to t[v/x]$ in such a context, with no reduction under an abstraction. Use canonical Scott numerals $\overline{0} = \lambda s.\lambda z.z$ and $\overline{n+1} = \lambda s.\lambda z.s\,\overline{n}$, with variable names chosen to avoid capture. Apply a closed program to $\overline{n}$, evaluate, and syntactically decode a canonical numeral value; direct a noncanonical result or malformed code to divergence. This effective observation protocol defines a partial numeric function. It prevents a silent change from numeric computation to arbitrary normalization or to syntactic beta equality.

We use the established effective simulations between Turing machines and this lambda formalism: there are total computable translations c and d satisfying

$$\lambda_{c(e)}(x) \simeq_{\mathrm{par}} \varphi_e(x), \qquad \varphi_{d(a)}(x) \simeq_{\mathrm{par}} \lambda_a(x). \tag{2}$$

The existence of these translations is a cited adequacy input, not a compiler construction proved afresh here. Forster, Kunze, Smolka, and Wuttke give a mechanized mutual simulation with time bounds [4]; we need only the weaker preservation of partial numeric results. Our use of their theorem does not make the present paper machine checked.

Proposition 2.3 (Equivalence of extensional presentations)**.** *The maps c and d induce mutually inverse maps between Turing codes modulo equality of partial numeric behavior and lambda codes modulo the same equality. The composite $e \mapsto d(c(e))$ is a behavior-preserving simulation of Turing programs.*

Proof. Equation (2) shows that equivalent inputs have equivalent translated outputs, so the maps are well-defined on the quotients. It also gives $\varphi_{d(c(e))} \simeq_{\mathrm{par}} \varphi_e$ and $\lambda_{c(d(a))} \simeq_{\mathrm{par}} \lambda_a$ pointwise. Thus both composites are identities on the quotients. □

Neither raw syntax nor reduction histories are identified by this argument. In particular, numerical extensional equality of lambda programs is not the same relation as beta convertibility, equality of their costs, or equality in every reflective program context.

3 An explicit higher-topos semantics of computation

An infinity topos does not by itself specify which arrows are computable. We therefore state effectivity as additional presentation data rather than build it into the word "topos."

Let Fin be a small skeleton of finite sets and all functions. Its objects and arrows have finite codes, and composition is computable. The category

$$\mathcal{X} = \mathrm{Fun}(\mathsf{Fin}^{\mathrm{op}}, \mathcal{S}) \tag{3}$$

is a presheaf infinity topos. Write Δ for the constant-presheaf functor. Constant sets are 0-truncated objects; $N = \Delta\mathbb{N}$ is a natural-numbers object. We interpret program codes by $P = N$. Finite products and the computation relations below are interpreted objectwise. This is a genuine higher ambient category: constant spaces such as ΔBG need not be 0-truncated.

Choose a primitive recursive predicate $T(e, x, y, t)$ asserting that t codes a valid finite halting computation of program e on input x with output y. The explicit trace, rather than just a claimed time bound, makes verification primitive recursive. Define internally

$$G(e, x, y) := \left\| \sum_{t:N} T(e, x, y, t) \right\|_{-1}. \tag{4}$$

The truncation says that a witnessing trace exists without retaining the trace as part of the truth value. Determinism gives

$$G(e, x, y) \wedge G(e, x, z) \Longrightarrow y = z.$$

Before truncation, trace witnesses are retained; after truncation we have a partial-function graph. No total universal evaluator $N \times N \to N$ is asserted.

Let Ω be the subobject classifier. Define B to be the subobject of $\Omega^{N\times N}$ consisting of single-valued relations. It classifies partial numeric graphs, with no assertion that every such graph is computable. The predicate (4) determines

$$P \xrightarrow{q} A \hookrightarrow B, \qquad A = \mathrm{im}(P \xrightarrow{G} B). \tag{5}$$

Thus A contains exactly the behaviors locally represented by the chosen codes, and q is an effective epimorphism in the categorical sense. Here "effective epimorphism" means realization of its Čech nerve; it does *not* mean that an algorithm can decide equality in A.

Proposition 3.1 (Internal adequacy of the simulation)**.** *In the model* (3)*, the map induced by Theorem 2.2 satisfies* $q \circ \mathrm{sim} = q$*. The translations of Proposition 2.3 give the same behavioral image for Turing and lambda presentations.*

Proof. The interpreter equality holds for all numerical codes and inputs, including nontermination. Hence the associated halting graphs are equal. Applying Δ gives equality of their constant relations, and their classifying maps into B agree. Since $A \hookrightarrow B$ is a monomorphism, the factorizations through A agree. The same reasoning applies to both equations (2). □

The model is deliberately explicit and modest. It has noncomputable arrows as well: every set function $\mathbb{N} \to \mathbb{N}$ induces a natural transformation $\Delta\mathbb{N} \to \Delta\mathbb{N}$. Consequently, having an effectively coded indexing category does not make all its presheaves or morphisms computable. We call the displayed data a *computably presented semantics in an infinity topos*; we do not claim to have constructed a canonical object named "the effective infinity topos."

For another ambient infinity topos, the later theorems apply when the interpretation, adequate simulation, and behavioral quotient are supplied. An arbitrary interpretation of an externally acceptable numbering need not automatically validate every external arithmetic theorem. Adequacy must be checked in that interpretation. The concrete constant-presheaf model above discharges this requirement directly.

4 Simulation invariance and exact predicate descent

We now work in any infinity topos $\mathcal{X}$. Let $q : E \twoheadrightarrow A$ be an effective epimorphism. The object E contains implementations, and A contains the observations retained by the semantics. Define its kernel pair

$$R = E \times_A E, \qquad r_0, r_1 : R \rightrightarrows E.$$

All equalities and pullbacks are interpreted homotopically. Statements on generalized elements range over every base object, not just global points.

Definition 4.1 (Behavioral predicate)**.** A predicate $Q \hookrightarrow E$ is behavioral relative to q if there exists $\overline{Q} \hookrightarrow A$ with $Q \simeq q^*\overline{Q}$ as subobjects of E. It is saturated if $r_0^*Q = r_1^*Q$ in $\mathrm{Sub}(R)$.

Theorem 4.2 (Exact criterion for predicate invariance)**.** *Pullback induces an isomorphism of posets*

$$\mathrm{Sub}(A) \;\cong\; \{Q \in \mathrm{Sub}(E) : r_0^*Q = r_1^*Q\}. \tag{6}$$

In particular, a predicate is definable solely from the retained behavior exactly when it is invariant under every pair of behaviorally equivalent implementations. Its descended subobject is unique.

Proof. If $Q = q^*\overline{Q}$, the equality $qr_0 = qr_1$ gives saturation. Conversely, let Q be saturated and define the internal proposition

$$\overline{Q}(a) := \left\| \sum_{e:E} (q(e) = a) \times Q(e) \right\|_{-1}. \tag{7}$$

For $e_0 : E$, a proof of $Q(e_0)$ yields $\overline{Q}(q(e_0))$ by using (e_0, refl). In the other direction, the goal $Q(e_0)$ is a proposition, so we may eliminate the truncation. A witness (e, p, u) has $p : q(e) = q(e_0)$ and $u : Q(e)$. It determines an element of R; saturation transports u to $Q(e_0)$. Hence $q^*\overline{Q} = Q$.

To prove uniqueness, suppose $q^*S = q^*S'$ for subobjects of A. Since q is an effective epimorphism, every $a : A$ is locally in its image: internally $\|\sum_e q(e) = a\|_{-1}$ holds. Eliminate this truncation into the proposition $S(a) \leftrightarrow S'(a)$, and use the assumed equality over a representative e. Thus $S = S'$. The same argument with an implication proves that pullback reflects order. This proves the poset isomorphism. The descent proof uses no global choice of representatives. □

Corollary 4.3 (Faithful simulation cannot change a behavioral predicate)**.** *Suppose $f, g : C \to E$ have a specified identification $qf = qg$. If Q is saturated, then $f^*Q = g^*Q$. In particular, for a faithful simulator* $\mathrm{sim} : E \to E$ *with* $q\,\mathrm{sim} = q$*, every behavioral predicate satisfies $Q(e) \leftrightarrow Q(\mathrm{sim}(e))$.*

Proof. Write $Q = q^*\overline{Q}$ using Theorem 4.2. Both pullbacks are the pullback of $\overline{Q}$ along the identified maps qf and qg. □

Invariance under one chosen simulator is weaker than saturation under the whole kernel pair. A predicate can be constant along that simulator's orbits and still distinguish other implementations of the same behavior. The theorem therefore does not identify "unchanged by this wrapper" with "extensional."

4.1 Exactly what equality of two definitions would require

Even two behavioral definitions need not be equivalent *to each other*. If $R = q^*\overline{R}$ and $S = q^*\overline{S}$, then

$$R = S \quad \Longleftrightarrow \quad \overline{R} = \overline{S}. \tag{8}$$

This follows from order reflection in Theorem 4.2. A predicate can be invariant under simulation without being equal to another invariant predicate. For example, "returns zero on input zero" and "returns one on input zero" are distinct behavioral predicates. The pointwise statement $R(p) \leftrightarrow R(\mathrm{sim}(p))$ does not entail $R \leftrightarrow S$.

5 Higher structure, type-valued observations, and coherence

The preceding theorem already holds in a higher topos, but a mere proposition cannot display arbitrary higher homotopy. In fact Ω is 0-truncated, so truncation gives

$$\mathrm{Map}(A, \Omega) \simeq \mathrm{Map}(\tau_{\leq 0} A, \Omega).$$

The same holds with the discrete Boolean object $\mathbf{2} = 1 + 1$ in place of Ω. This is a precise limitation of proposition-valued observations, not an assertion that A has no higher structure.

Proposition 5.1 (Transport of structured observations)**.** *Let $F \to A$ be a family of objects. An identification $p : a = a'$ induces an equivalence of fibers $F_a \simeq F_{a'}$, coherently with composition and higher identifications. Therefore a specified equality $q(e) = q(\mathrm{sim}(e))$ preserves every structured observation defined as a family over A.*

Proof. The fibers are pullbacks along the maps $a, a' : 1 \to A$. The identification between these maps identifies their pullback functors up to natural equivalence. Functoriality of pullback supplies coherence. In internal type theory this is path transport; the inverse is transport along p^{-1}. □

In a univalent universe this can also be expressed by applying $F : A \to \mathcal{U}$ to p. Univalence identifies equivalences with universe paths; it does not produce an equivalence merely because two objects have computable presentations. The standard interpretation of such reasoning in higher toposes is justified by Shulman's semantics [6]; see also the exposition by Riehl [7] and the HoTT book [8].

For a general family on E to descend along q, coherent descent data are required. By the van Kampen property of colimits in an infinity topos [5, Theorem 6.1.3.9 and Proposition 6.1.3.10], if $E_\bullet$ is the Čech nerve, with $E_n = E \times_A \cdots \times_A E$ ($n+1$ factors), the relevant principle is

$$\mathcal{X}_{/A} \simeq \lim_{[n] \in \Delta} \mathcal{X}_{/E_n}. \tag{9}$$

The theorem for predicates is simpler because equivalences between subobjects, when present, have no nontrivial choices. Thus their higher compatibility conditions are automatic. A family of types can carry nontrivial automorphisms, and its compatibility conditions cannot be discarded.

Example 5.2 (A finite coherence obstruction)**.** In spaces, let $G = C_2$ and consider the effective epimorphism $* \to BG$. A family of discrete sets over BG is a G-set. The two-point set with the trivial action and the two-point set with the swapping action have identical underlying fibers after

pullback to $*$. They are not equivalent over BG: an equivariant bijection would send a fixed point of the trivial action to a fixed point of the swapping action, but the latter has none.

This example also lives as constant higher data in (3). It shows why equal unstructured fibers do not establish equivalence of higher structured semantics. Likewise $*$ and BG have the same connected-component set but different fundamental groups. Higher formalization can preserve additional distinctions instead of eliminating them.

6 Countermodels to unrestricted reality–simulation collapse

6.1 Computable provenance

Let P be the program object above, and form

$$I = P \times \mathbf{2}, \qquad d(e, b) = q(e).$$

Define two implementations and two predicates by

$$\begin{aligned} \mathrm{direct}(e) &= (e, 0), & \mathrm{virtual}(e) &= (\mathrm{sim}(e), 1), \\ \mathrm{Real}(e, b) &\equiv (b = 0), & \mathrm{Simulated}(e, b) &\equiv (b = 1). \end{aligned}$$

The tag is declared provenance, not a claimed metaphysical detector. It is enough to refute a statement quantified over *every formal definition.*

Theorem 6.1 (Effective separation with identical behavior)**.** *In the concrete higher-topos model,* $d(\mathrm{direct}(e)) = d(\mathrm{virtual}(e))$*, but the total computable projection* $I \to \mathbf{2}$ *distinguishes these implementations. The displayed Real and Simulated predicates are distinct, complementary, inhabited subobjects.*

Proof. The behavioral equality is $q(e) = q(\mathrm{sim}(e))$, already proved. Projection to the second coordinate returns 0 on the direct implementation and 1 on the virtual implementation. Coproduct disjointness gives $0 \neq 1$ in this nondegenerate model. Each predicate has a witness using any program code. Thus neither predicate is saturated on the kernel pair of d, and neither descends to the behavioral image. □

As bare objects, both tagged subobjects are isomorphic to P. They are nevertheless different *subobjects of* I; their inclusions and classification predicates matter. This distinction blocks a second possible ambiguity in the word "equivalent." If equivalence is meant only as equivalence of bare types, there are counterexamples too: take a three-element implementation object, give every element the same constant-zero behavior, designate one element Real and the other two Simulated. Both predicates are decidable, yet their fibers have cardinalities one and two and are not equivalent.

6.2 An observed cost difference

Consider a machine that returns zero in one native transition. An interpreter with fetch, decode, and execute phases can take three transitions to return the same result. If the semantics records only the output, these executions agree. If it records (output, step count), they differ. Both cost functions in this example are computable.

More generally, replacing q by a richer semantics $\widetilde{q}$ refines its kernel pair. A proof that $q\,\mathrm{sim} = q$ says nothing about $\widetilde{q}\,\mathrm{sim} = \widetilde{q}$ until the newly recorded observations are checked. Polynomial simulation overhead, including the bounds in [4], is not equality of exact costs.

6.3 Why a fully universal collapse is inconsistent

If an inhabited implementation object I had all its predicates equivalent, the constantly true and constantly false predicates would coincide on I. Pulling back along a point $1 \to I$ would identify the full and empty subobjects of 1, forcing $0 \simeq 1$. Thus this unrestricted schema collapses a nondegenerate topos. No improvement in computational representation can repair the missing restriction on predicates.

7 Effectivity and the separate Rice obstruction

Semantic invariance is different from decidability. In the chosen Turing presentation, consider an external total computable classifier $b : \mathbb{N} \to \{0, 1\}$ satisfying

$$\varphi_e \simeq_{\text{par}} \varphi_f \Longrightarrow b(e) = b(f). \tag{10}$$

Theorem 7.1 (Rice's obstruction, relative to this presentation)**.** *Every total computable classifier satisfying* (10) *is constant. Equivalently, a nontrivial property of partial recursive behavior has no total computable decision procedure on all program codes.*

Proof. Suppose b is nonconstant. Let $\perp$ be the nowhere-defined partial function. By replacing b with $1 - b$ if necessary, assume its value on indices of $\perp$ is zero. Choose a program a with $b(a) = 1$. Uniformly in e, construct the partial program

$$h_e(x) : \quad \text{first run } \varphi_e(e);\ \text{if it halts, run } \varphi_a(x).$$

Specialization provides a total computable index map $e \mapsto k(e)$ for h_e. If $\varphi_e(e)$ does not halt, then $\varphi_{k(e)} = \perp$ and $b(k(e)) = 0$. If it halts, then $\varphi_{k(e)} \simeq_{\text{par}} \varphi_a$ and $b(k(e)) = 1$. Hence $b \circ k$ decides the diagonal halting set, a contradiction. This is the classical Rice argument [9] in the form needed here. □

The result applies to a Boolean observation on A whenever its pullback to codes is a total computable function in the specified presentation. It does not say that every internally definable predicate is decidable or trivial. For example,

$$H(e) := \left\| \sum_{y,t:N} T(e, 0, y, t) \right\|_{-1}$$

is a nontrivial extensional predicate, but has no total computable characteristic function. A general subobject is classified by Ω, whereas a decidable subobject admits a characteristic map to $\mathbf{2}$. Confusing these classifiers would invalidate the proposed collapse theorem.

7.1 The corresponding realizability statement

Hyland's effective topos is an elementary realizability topos, built from Kleene application [10]. To see the same separation concretely, let PC be the assembly whose underlying set consists of unary partial recursive functions and whose realizers for f are

$$E(f) = \{e \in \mathbb{N} : \varphi_e = f\}.$$

Each set of realizers is nonempty. A morphism $\mathsf{PC} \to \mathbf{2}$ is tracked by a partial recursive function defined on every realizer; since every natural number indexes a partial recursive function, this tracker is total on $\mathbb{N}$. It is extensional, so Theorem 7.1 makes the underlying Boolean classifier constant. In contrast, the code assembly paired with a Boolean tag has a nonconstant tracked projection to $\mathbf{2}$.

This calculation uses only assemblies and their fully faithful embedding in the effective topos. It establishes that genuine realizability does not erase provenance. We make no identification of Hyland's topos with the 0-truncation of (3), and we do not infer that simplicial objects in an arbitrary elementary topos automatically form a Grothendieck infinity topos. A particular higher-realizability construction would require its own existence and adequacy theorems.

8 Scope, evidence, and consequences

8.1 Which models are covered

The descent theorem is independent of computability and holds for every infinity topos and every effective epimorphism. The simulation theorem requires an acceptable universal computation presentation, or explicitly supplied adequate translations. Restricted finite-state or resource-bounded languages need not contain a universal simulator for their own class within the same resource budget. Relative-oracle models require the corresponding relative notion of computability; they do not become ordinary Turing computation merely by being formally specified.

For interactive, probabilistic, concurrent, or reflective programs, a partial numeric graph may discard essential observations. The same descent argument can be used with trace, contextual, or distributional semantics, but the simulation's adequacy must be established for that chosen semantics. Mutual representability alone does not prove full abstraction for an arbitrary collection of contexts.

8.2 Reproducibility and proof status

The accompanying dependency-free Python artifact implements a small register-machine transition system and a separate three-phase interpreter. It checks agreement of outputs and the exact three-to-one transition count on terminating examples, includes bounded nontermination examples marked as such, and exhaustively checks the finite predicate-descent countermodel. A second finite check distinguishes trivial and swapping C_2-actions.

These finite checks illustrate the hypotheses and expose the counterexamples; they are not evidence that an arbitrary interpreter is correct, a decision procedure for nontermination, or a kernel verification of the categorical proofs. The general results depend on the proofs above and on the cited standard adequacy and descent results. The publication package includes the manuscript source, journal-format text, validation results, build instructions, and checksums.

8.3 What can be said about reality

A running interpreter is itself a computation, and a faithful one realizes the source algorithm's selected behavior. There is no additional "reality bit" determined by that behavior that can separate a program from a behaviorally identical interpreter execution. A supplied provenance bit can separate them, precisely because it is additional information.

To reach a claim about physical reality, one would need a separately justified interpretation relating physical systems, observations, causal interventions, and resource limits to the formal model. No such interpretation is supplied by universal computation, descent, or univalence alone. The mathematical result concerns what follows from an explicitly stated observational interface; it neither proves nor disproves a general metaphysics of simulation.

9 Final statement

For the effective Turing and lambda presentations constructed here, direct and faithfully simulated implementations determine the same partial-function graph. Every predicate descending to that graph, and every structured observation transported along the resulting identification, is preserved. Predicate descent is equivalent to invariance on the entire semantic kernel pair. Arbitrary formal definitions can retain provenance, cost, or higher structure and therefore need not be equivalent. This is the strongest general claim established by the stated hypotheses.

Authorship and automation. Anonymous manuscript. Generative AI assisted with drafting, mathematical analysis, reference checking, and preparation of the accompanying executable examples. The text provides explicit assumptions and proofs but has not undergone independent peer review or proof-assistant verification. No institutional affiliation or named author is asserted. Submission and publication are separate from preparation of this manuscript.

References

[1] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics* 58(2), 345–363, 1936. Full text.

[2] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* s2-42(1), 230–265, 1937 (received 1936). doi:10.1112/plms/s2-42.1.230.

[3] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987 reprint of the 1967 edition. Publisher record.

[4] Y. Forster, F. Kunze, G. Smolka, and M. Wuttke. A mechanised proof of the time invariance thesis for the weak call-by-value λ-calculus. *ITP 2021*, LIPIcs 193, 19:1–19:20, 2021. doi:10.4230/LIPIcs.ITP.2021.19.

[5] J. Lurie. *Higher Topos Theory*. Annals of Mathematics Studies 170, Princeton University Press, 2009. Chapters 5–6, especially Sections 6.1 and 6.2. Author's updated text.

[6] M. Shulman. All $(\infty, 1)$-toposes have strict univalent universes. 2019. arXiv:1904.07004.

[7] E. Riehl. On the ∞-topos semantics of homotopy type theory. *Bulletin of the London Mathematical Society*, 2024. doi:10.1112/blms.12997; arXiv:2212.06937.

[8] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. Chapters 2, 3, and 6. `https://homotopytypetheory.org/book/`.

[9] H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society* 74(2), 358–366, 1953. Publisher full text.

[10] J. M. E. Hyland. The effective topos. In A. S. Troelstra and D. van Dalen (eds.), *The L. E. J. Brouwer Centenary Symposium*, Studies in Logic and the Foundations of Mathematics 110, 165–216, North-Holland, 1982. Author-hosted retypeset text.