

# On the Experimental Distinguishability of Computed and Fundamental Physical Dynamics

Omar Ali Ibrahim

ChatGPT · GPT-5.6 Sol

Working Draft v0.2

30 September 2026

This manuscript is an exploratory foundations draft. Version 0.2 is an adversarial revision of the first five sections. It corrects the quantifier structure of the central hypothesis, separates behavioral emulation from constitutive implementation, treats the physical-computation relation as a contested parameter rather than a solved problem, moves computable analysis into the foundations for continuous state spaces, and narrows the finite-observation claims to what they actually prove. The project remains outcome-neutral: either a specified computational class can be separated from the operational behavior of our universe, or a principled indistinguishability result can be established under explicit assumptions.

## Abstract

The contemporary simulation hypothesis is usually motivated by computation: if sufficiently capable physical computers can simulate increasingly complex worlds, perhaps the observable universe is itself implemented computationally. This paper replaces that broad intuition with a narrower question. For a specified class of physical computational implementations, is the complete operational behavior available to internal observers realizable by at least one member of that class? We distinguish computable, computational, computed, and simulated dynamics; define the target hypothesis with explicit quantifier order; separate predictive simulation from constitutive implementation; and formulate internal experiments as interactive protocols inducing observable transcript distributions. We then classify computational implementations by their declared resources, emphasizing that additional computational power must be traced to an explicit additional resource. Elementary results show that finite realized transcripts, finite deterministic response trees, and finite extensional data cannot by themselves certify noncomputability. The relevant target is therefore a uniform physical protocol whose general behavior is linked by theorem to a nonrecursive decision problem. Finally, we examine Gandy’s theorem, quantum extensions by Arrighi and Dowek, and related work as conditional physical-computability results rather than universal facts about nature. The paper is deliberately agnostic about its eventual conclusion: either the operational behavior of our universe lies outside a meaningful computational class, or computed and fundamental dynamics are internally indistinguishable for that class under stated conditions.

**Revision note for v0.2.** The most important correction is the quantifier distinction between

$$\forall E \exists C_E \quad \text{and} \quad \exists C \forall E.$$

The former says that each finite experiment can be fitted by some program; the latter says that one computational implementation reproduces the full open-ended operational behavior of the world. Only the second corresponds to the simulation hypothesis studied here. The draft also replaces the shorthand  $U \notin \mathcal{C}$  by the type-correct statement  $\mathcal{B}_U \notin \mathfrak{B}(\mathcal{C})$ , where  $\mathfrak{B}(\mathcal{C})$  is the class of behaviors realizable by computational implementations in  $\mathcal{C}$ .

## Contents

|          |                                                                    |          |
|----------|--------------------------------------------------------------------|----------|
| <b>1</b> | <b>The Question</b>                                                | <b>1</b> |
| 1.1      | Operational behavior . . . . .                                     | 1        |
| 1.2      | The computational implementation hypothesis . . . . .              | 2        |
| 1.3      | Three quantifier orders . . . . .                                  | 2        |
| 1.4      | What would count as success? . . . . .                             | 3        |
| <b>2</b> | <b>What Constitutes Computation and Simulation?</b>                | <b>4</b> |
| 2.1      | Physical evolution is not automatically computation . . . . .      | 4        |
| 2.2      | Predictive simulation versus constitutive implementation . . . . . | 4        |

|          |                                                                              |           |
|----------|------------------------------------------------------------------------------|-----------|
| 2.3      | Internal experiments as labeled interactions . . . . .                       | 5         |
| 2.4      | Operational equivalence . . . . .                                            | 6         |
| 2.5      | Counterfactual fidelity revisited . . . . .                                  | 7         |
| 2.6      | Simulation as constitutive behavioral realization . . . . .                  | 7         |
| 2.7      | Continuous state spaces and represented information . . . . .                | 8         |
| 2.8      | Computable dynamics versus computed dynamics . . . . .                       | 8         |
| <b>3</b> | <b>Classes of Computational Implementation</b>                               | <b>9</b>  |
| 3.1      | Architecture, efficiency, and computability . . . . .                        | 9         |
| 3.2      | Effective specification is stronger than finite description . . . . .        | 9         |
| 3.3      | Fixed finite hosts, extensible machines, and ideal Turing machines . . . . . | 10        |
| 3.4      | Classical algorithmic computation . . . . .                                  | 10        |
| 3.5      | Reversible computation . . . . .                                             | 11        |
| 3.6      | Probabilistic computation . . . . .                                          | 11        |
| 3.7      | Parallel, distributed, and interactive computation . . . . .                 | 11        |
| 3.8      | Quantum computation . . . . .                                                | 12        |
| 3.9      | Analog and continuous computation . . . . .                                  | 12        |
| 3.10     | Hypercomputational models . . . . .                                          | 13        |
| 3.11     | A working standard class . . . . .                                           | 14        |
| 3.12     | Host isolation and simulated time . . . . .                                  | 14        |
| 3.13     | Where computation becomes vacuous . . . . .                                  | 15        |
| <b>4</b> | <b>Finite Observational Indistinguishability</b>                             | <b>15</b> |
| 4.1      | Finite protocols and finite transcripts . . . . .                            | 15        |
| 4.2      | A realized finite transcript is always reproducible . . . . .                | 16        |
| 4.3      | Finite deterministic response trees . . . . .                                | 16        |
| 4.4      | The quantifier lesson . . . . .                                              | 17        |
| 4.5      | Finite extensional data cannot certify noncomputability . . . . .            | 17        |
| 4.6      | The role of theory . . . . .                                                 | 18        |
| 4.7      | Uniform operational decision protocols . . . . .                             | 19        |
| 4.8      | Finite experiments can still falsify restricted classes . . . . .            | 20        |
| 4.9      | Implications for experiment design . . . . .                                 | 20        |
| <b>5</b> | <b>Physical Limits of Computation</b>                                        | <b>20</b> |
| 5.1      | From the Church–Turing thesis to conditional physical theorems . . . . .     | 21        |
| 5.2      | What Gandy contributes . . . . .                                             | 21        |
| 5.3      | An assumption matrix, not yet a lattice . . . . .                            | 22        |
| 5.4      | Counterexamples identify resources, not automatic hypercomputers . . . . .   | 22        |
| 5.5      | Quantum mechanics and effectiveness . . . . .                                | 23        |
| 5.6      | Continuous physics and computable analysis . . . . .                         | 24        |

---

|      |                                                                    |    |
|------|--------------------------------------------------------------------|----|
| 5.7  | Host constraints need not resemble simulated constraints . . . . . | 24 |
| 5.8  | Resource bounds and causal isolation . . . . .                     | 25 |
| 5.9  | A conditional exclusion theorem scheme . . . . .                   | 25 |
| 5.10 | Behavioral emulation is not yet constitutive simulation . . . . .  | 26 |
| 5.11 | What Section 5 establishes, and what it does not . . . . .         | 27 |

# 1. The Question

The modern simulation hypothesis considered here is a computational hypothesis. It is not merely the older epistemological possibility that experienced reality is illusory, deceptive, or generated by an inaccessible underlying reality. Its modern motivation is that physical computers can instantiate models whose internal dynamics resemble increasingly complex parts of the world, and that sufficiently advanced computers might instantiate worlds containing observers [6]. The hypothesis therefore adds a specifically computational claim to the more general possibility of a deeper ontology.

That distinction matters. If the putative external system is allowed arbitrary physical powers—unrestricted state spaces, arbitrary precision, noncomputable transformations, exotic causal structures, or whatever additional resources are required to reproduce every observation—then the word *computation* ceases to constrain the hypothesis. The claim approaches the weaker statement that our universe is generated by some deeper physical reality. That may be metaphysically meaningful, but it is not the constrained computational hypothesis examined here.

We therefore formulate the problem operationally and relative to a declared computational class.

## 1.1 Operational behavior

Let  $U$  denote the physical world accessible to a community of internal observers. Let  $\mathcal{E}$  be a specified class of admissible experimental protocols available to those observers. A protocol  $E \in \mathcal{E}$  may be adaptive: later actions may depend on earlier outcomes. Its observable result is a transcript

$$\tau = (i_1, o_1, i_2, o_2, \dots),$$

finite for finite protocols and potentially open-ended for more general protocols. The world induces, for each protocol, a probability law over possible transcripts. We write this operational behavior as

$$\mathcal{B}_U : E \mapsto P_U(\tau \mid E).$$

In a deterministic theory the probability law may collapse to a single transcript. In stochastic or quantum theories it need not.

For a candidate computational implementation  $C$ , together with whatever admissible representation and implementation relation is required to interpret its internal states as states of the represented world, define analogously

$$\mathcal{B}_C : E \mapsto P_C(\tau \mid E).$$

The operational comparison between world and implementation is therefore a comparison between behavior maps, not between their microscopic state spaces.

Let

$$\mathfrak{B}(\mathcal{C}) = \{\mathcal{B}_C : C \in \mathcal{C}\}$$

denote the class of operational behaviors realizable by implementations in computational class  $\mathcal{C}$ .

## 1.2 The computational implementation hypothesis

The central hypothesis is

$$H_{\mathcal{C}} : \quad \exists C \in \mathcal{C} \forall E \in \mathcal{E} : \quad \mathcal{B}_C(E) = \mathcal{B}_U(E), \quad (1)$$

subject also to the constitutive implementation requirements developed in Section 2.

Equation (1) is equivalent, at the behavioral level, to

$$\mathcal{B}_U \in \mathfrak{B}(\mathcal{C}).$$

Its negation is

$$\neg H_{\mathcal{C}} : \quad \forall C \in \mathcal{C} \exists E \in \mathcal{E} : \quad \mathcal{B}_C(E) \neq \mathcal{B}_U(E), \quad (2)$$

again conditional on the implementation relation being admissible.

This formulation immediately clarifies three distinct claims that must not be conflated:

- (i) The dynamics of  $U$  are computable.
- (ii)  $U$  has a computationally describable or computational internal structure.
- (iii)  $U$  is constitutively implemented by another physical system performing a computation.

Computability is a property of an abstract input-output or dynamical relation under an encoding. Computational structure is a property of a description or organization. Being computed is an additional realization claim about another physical substrate. None of these implications is automatic.

## 1.3 Three quantifier orders

The problem is particularly sensitive to quantifier order. Three statements recur throughout this paper:

1. A *universal discriminator* would satisfy

$$\exists E \in \mathcal{E} \forall C \in \mathcal{C} : \quad \mathcal{B}_C(E) \neq \mathcal{B}_U(E). \quad (3)$$

One experiment would exclude the entire computational class. This is sufficient to refute  $H_{\mathcal{C}}$  but is stronger than necessary.

2. The actual negation of the computational implementation hypothesis is

$$\forall C \in \mathcal{C} \exists E_C \in \mathcal{E} : \mathcal{B}_C(E_C) \neq \mathcal{B}_U(E_C). \quad (4)$$

Different candidate implementations may fail on different protocols.

3. A much weaker fact, established in Section 4 for finite observations, is of the form

$$\forall E \in \mathcal{E}_{\text{fin}} \exists C_E : \mathcal{B}_{C_E}(E) = \mathcal{B}_U(E) \quad (5)$$

for the finite behavior actually tested or recorded. This does *not* imply that there exists one  $C$  reproducing all protocols.

Thus

$$\forall E \exists C_E \not\equiv \exists C \forall E.$$

This is the central logical correction introduced in version 0.2.

#### 1.4 What would count as success?

The project is intentionally neutral about its eventual outcome. There are at least two scientifically meaningful endpoints.

First, one may establish a separation theorem: under explicitly stated physical assumptions, the operational behavior of  $U$  cannot be realized by any member of a declared computational class,

$$\mathcal{B}_U \notin \mathfrak{B}(\mathcal{C}).$$

Such a result would not prove that  $U$  is metaphysically fundamental. It would exclude that class of computational implementations.

Second, one may establish an indistinguishability result: under explicit assumptions, there exists at least one computational implementation whose behavior is identical to that of  $U$  for every admissible internal protocol,

$$\exists C \in \mathcal{C} \forall E \in \mathcal{E} : \mathcal{B}_C(E) = \mathcal{B}_U(E).$$

Such a result would not prove that  $U$  is computed. It would show that the computational-implementation claim has no internal operational discriminator within the chosen protocol class.

The purpose of the present work is therefore not to defend an initial intuition that reality must exceed computation. It is to determine which of these two possibilities, or which intermediate result, follows once the terms of the question are made explicit.

## 2. What Constitutes Computation and Simulation?

The phrase “implemented by a computation” carries two distinct burdens. First, there must be a principled sense in which a physical host implements a computation rather than merely undergoing physical change. Second, there must be a principled sense in which that computation constitutes the represented world rather than merely predicts or models an independently existing world. Neither issue is trivial.

### 2.1 Physical evolution is not automatically computation

Every computer is a physical system, but it does not follow that every physical system computes every abstract process onto which its history can be mapped. The implementation problem is familiar in the philosophy of computation. Simple mapping accounts are vulnerable to trivialization: if arbitrary post-hoc mappings are allowed, sufficiently rich physical histories can be mapped onto many abstract state-transition systems. Chalmers introduced causal and counterfactual restrictions intended to block such trivialization [7]. Piccinini argues that even causal and counterfactual mapping accounts remain too permissive and develops a mechanistic alternative [17]. Horsman et al. approach the issue through representation relations connecting abstract and physical evolution [14].

This paper does not assume that this debate has been settled. Instead, let  $\mathfrak{R}$  denote a family of *admissible implementation relations*. Any theorem that depends on a stronger choice of implementation account should state that dependence explicitly.

At minimum, an admissible relation used here must rule out a purely retrospective relabeling of one realized history. It should preserve relevant causal or dynamical structure across a declared state domain and should support the counterfactual transitions required by the experimental protocols under consideration. These requirements are methodological constraints, not a claim that they form a complete metaphysics of computation.

**Definition 2.1 (Admissible computational implementation relation)** *An implementation relation  $R \in \mathfrak{R}$  connects states or histories of a physical host  $C$  to states or histories of a represented system only if the correspondence is fixed independently of the particular realized experimental transcript, preserves the relevant transition structure over a declared domain, and supports the protocol counterfactuals to which the simulation claim commits itself.*

The formulation deliberately leaves open whether additional mechanistic, semantic, functional, or organizational conditions are required. Results that survive multiple admissible accounts are correspondingly stronger.

### 2.2 Predictive simulation versus constitutive implementation

A second distinction is more important for the present question. A physical computer may simulate an independently existing target without constituting it. If a computer numerically predicts the

orbit of Mars, the physical planet does not cease to exist when the computer is switched off. The represented Martian state and the state of the computer are correlated by a model, but Mars is not realized by that computer.

Horsman et al.’s representation framework is especially useful for understanding physical computation and prediction, but the hypothesis considered here is stronger: the states experienced as  $U$  are claimed to be *realized by* states of the host computational process, rather than merely mirrored by them [14].

We therefore distinguish:

- **Predictive simulation:** a host  $C$  computes a representation of an independently instantiated target  $W$ .
- **Constitutive computational implementation:** the represented physical state of  $U$  is realized by, or supervenes on, the relevant state of  $C$  under the implementation relation. There is no additional independently evolving target whose behavior merely happens to match the computation.

The second condition is ontological rather than directly observational. An internal observer who sees only  $\mathcal{B}_U$  may be unable to determine whether a behaviorally identical host is predictive or constitutive. This is not a defect in the formalism; it is precisely why behavioral indistinguishability and metaphysical fundamentality must remain separate claims.

For compactness, we write

$$C \Vdash_R U$$

when  $C$  is claimed to constitutively realize  $U$  under admissible implementation relation  $R$ .

### 2.3 Internal experiments as labeled interactions

The first draft represented internal interventions as if each experimental choice were an external input sent directly to a host computer. That picture is unnecessarily restrictive. If an observer inside a simulated world turns a knob, the observer, hand, knob, apparatus, and resulting field configuration may all be endogenous parts of the represented dynamics. The host need not receive a special command from outside the simulation.

A better abstraction is a labeled transition system. Write

$$s \xrightarrow{a} s'$$

for a possible world transition, where the label  $a$  records an operationally meaningful action, intervention, or observed event. Internal experimental protocols select or condition future actions on prior labels and observations. A host implementation contains corresponding transitions among

host states, but the labels need not correspond to externally injected host inputs.

This perspective connects the problem naturally to the theory of behavioral equivalence, simulation, and bisimulation developed in concurrency theory. Bisimulation was introduced to characterize when systems with different internal states exhibit matching observable transition behavior; modern treatments explicitly interpret it as a tool for behavioral equality among interactive processes [18]. We do not assume that ordinary labeled-transition bisimulation is the final formalism for quantum or continuous physics, but it provides the right conceptual template: internal structure may differ while all allowed interactions remain behaviorally matched.

## 2.4 Operational equivalence

For the broadest common framework, we use protocol semantics rather than commit immediately to one form of bisimulation.

**Definition 2.2 (Exact operational equivalence)** *Two systems  $A$  and  $B$  are operationally equivalent relative to protocol class  $\mathcal{E}$ , written*

$$A \equiv_{\mathcal{E}} B,$$

*if for every  $E \in \mathcal{E}$  the induced probability measures over complete observable transcripts are identical:*

$$P_A(\tau \mid E) = P_B(\tau \mid E) \quad \text{for all observable transcripts } \tau.$$

*Equivalently,  $\mathcal{B}_A = \mathcal{B}_B$  on  $\mathcal{E}$ .*

This definition can later be refined to deterministic bisimulation, probabilistic bisimulation, quantum process equivalence, or another model appropriate to the physical theory. For the foundational argument, the protocol-level equality is sufficient.

Approximate equivalence requires more care. A fixed local error tolerance can be amplified by repeated trials. If reality produces a Bernoulli probability  $p$  and the implementation produces  $p + \delta$ , then sufficiently many repetitions can distinguish the systems whenever  $\delta \neq 0$ . We therefore define approximation at the level of complete protocols rather than individual transitions.

**Definition 2.3 (Protocol-relative approximate equivalence)** *For a finite protocol  $E$ , let  $P_A^E$  and  $P_B^E$  be the corresponding transcript distributions. Given a declared tolerance function  $\epsilon(E)$ , the systems are approximately equivalent on a protocol family  $\mathcal{F}$  if*

$$d_{\text{TV}}(P_A^E, P_B^E) \leq \epsilon(E) \quad \forall E \in \mathcal{F},$$

*where  $d_{\text{TV}}$  is total-variation distance for discrete transcript distributions, or an appropriate operational metric in the relevant theory.*

For the main logical results below, exact equivalence is the cleaner starting point. Approximate

equivalence should be introduced together with explicit bounds on protocol length, experimental resolution, or available resources.

## 2.5 Counterfactual fidelity revisited

A prerecorded sequence of states is not by itself a simulation of a dynamical world. Suppose one stores the single realized history

$$s_0, s_1, \dots, s_N.$$

The recording may reproduce what happened without encoding what would have happened under alternative internal choices. A world simulation makes a stronger commitment to alternative reachable histories.

In version 0.1 this requirement was called *counterfactual fidelity*. In version 0.2 we retain the idea but locate it within operational equivalence: the protocol class  $\mathcal{E}$  itself specifies which counterfactual interventions and adaptive histories the implementation is required to match. A larger  $\mathcal{E}$  defines a stronger simulation claim.

Thus there is no single context-free demand that an implementation reproduce “all counterfactuals.” The relevant domain is explicit:

$$\mathcal{E}_1 \subseteq \mathcal{E}_2 \implies A \equiv_{\mathcal{E}_2} B \Rightarrow A \equiv_{\mathcal{E}_1} B.$$

A simulator faithful only to a finite laboratory protocol is making a much weaker claim than one faithful to every physically accessible experiment.

## 2.6 Simulation as constitutive behavioral realization

We can now state the working notion used in the remainder of the paper.

**Definition 2.4 (Computational simulation of a physical world)** *A physical host  $C$  computationally simulates  $U$  relative to  $(\mathcal{E}, \mathfrak{R})$  if:*

1.  *$C$  is a physical implementation of a computation according to at least one admissible relation  $R \in \mathfrak{R}$ ;*
2.  *$C \Vdash_R U$ , i.e. the represented states of  $U$  are constitutively realized rather than merely predicted by an independent duplicate;*
3. *the induced operational behavior of the represented world is equivalent to that of  $U$  for every allowed protocol:*

$$\mathcal{B}_C(E) = \mathcal{B}_U(E) \quad \forall E \in \mathcal{E}.$$

The constitutive clause prevents a perfect weather model from counting as the atmosphere it predicts.

The behavioral clause prevents a merely symbolic association between host and represented states from counting as a faithful world implementation.

## 2.7 Continuous state spaces and represented information

For discrete finite strings, the phrase “computable state transition” is relatively straightforward. Physical theories, however, often use real-valued fields, wavefunctions, probability measures, and infinite-dimensional spaces. A Turing machine cannot receive an arbitrary real number as a literal finite input. Computability must therefore be defined relative to a representation of continuous information.

Type-2 computability and computable analysis provide the standard machinery. A real number, function, or point in a metric space is represented by an effective *name*, such as a procedure producing increasingly accurate rational approximations. Operators on continuous objects are computable when they transform names effectively into names [21]. This distinction is essential: a continuous mathematical state space does not by itself imply infinite operational information or hypercomputation.

Accordingly, when later sections say that continuous physical dynamics are computable, they mean computable relative to an explicit effective representation of the state and observables, not that an arbitrary exact real can be handed to a finite digital machine as a primitive symbol.

## 2.8 Computable dynamics versus computed dynamics

The distinction can now be stated without ambiguity. Suppose a represented dynamical law  $F$  is computable, in the discrete or represented-space sense appropriate to its domain. Then there exists an effective procedure for obtaining the relevant outputs from effective descriptions of its inputs.

This implies only

the dynamics admit computation.

It does not imply

the physical transition occurs because an external host executes that computation.

Thus

computable  $\not\Rightarrow$  computed.

Even if the fundamental laws of a universe were exactly those of a cellular automaton, that fact alone would not establish that the universe is a simulation. The automaton could simply be the fundamental dynamics. The additional constitutive relation  $C \Vdash_R U$  is precisely the extra content of the hypothesis investigated here.

### 3. Classes of Computational Implementation

The statement “the universe is computed” remains incomplete until the admissible model of computation and its physical resources are specified. The purpose of this section is not to decide in advance that Turing computation exhausts physical computation. It is to prevent the computational hypothesis from expanding silently whenever a proposed limitation is encountered.

#### 3.1 Architecture, efficiency, and computability

Different computational architectures may have very different state spaces, transition rules, parallelism, energy costs, and asymptotic complexity while agreeing on which discrete decision problems are computable at all. Reversible, stochastic, parallel, and quantum computation provide familiar examples. The first distinction in this paper is therefore between:

- **computational architecture:** how a physical implementation is organized;
- **computational complexity:** the resources required to solve a problem;
- **computability or decision power:** which uniformly specified yes/no problems can be decided at all.

This distinction is important because complexity is not automatically visible from inside a simulated world. A host that requires an enormous number of external steps to generate a short interval of represented time need not reveal that cost if host time has no independent observable coupling to simulated clocks. That assumption is made explicit below.

Interactive and stochastic systems are not naturally summarized by a single class of ordinary functions. We therefore use two related descriptions. For behavior, we retain

$$\mathfrak{B}(\mathcal{C}) = \{\mathcal{B}_C : C \in \mathcal{C}\}.$$

For computability comparisons, we ask which subsets of  $\mathbb{N}$  can be decided by a uniform family of finite procedures with a declared reliability criterion. This allows classical, probabilistic, quantum, and continuous architectures to be compared without pretending that they all literally compute the same kind of mathematical object.

#### 3.2 Effective specification is stronger than finite description

A finite sentence can describe a noncomputable object. “Use the halting set as an oracle” is a finite description; so is a conventional definition of Chaitin’s  $\Omega$ . Finite describability therefore cannot serve as a proxy for effective computability.

We distinguish two notions.

**Definition 3.1 (Resource-explicit specification)** *A computational model is resource-explicit if its transition rules, primitive constants, state-preparation operations, communication primitives, and readout rules are stated sufficiently clearly that any nonstandard informational or physical resource is identifiable rather than hidden inside the word “computer.”*

**Definition 3.2 (Effective specification)** *A computational implementation is effectively specified when its finite inputs, transition operations, required constants, state preparations, and finite outputs are given by effective procedures in the relevant representation. No noncomputable oracle, noncomputable real constant, completed infinite operation, or equivalent non-effective resource is admitted unless it is declared as an additional primitive.*

The central methodological principle is therefore:

Additional computational power requires an identifiable additional resource.

This does not declare such resources impossible. It makes them part of the hypothesis to be examined.

### 3.3 Fixed finite hosts, extensible machines, and ideal Turing machines

A physically fixed device with finitely many distinguishable internal states is, in the strict mathematical sense, a finite-state system. If it has at most  $N$  physical states, no clever memory allocator removes that bound. By contrast, the abstract Turing machine does not possess an actually infinite amount of used tape at any finite stage; rather, its model imposes no predetermined finite upper bound on how much tape may eventually be used.

It is useful to separate at least three idealizations:

$$\mathcal{C}_{\text{fixed}} \subset \mathcal{C}_{\text{ext}} \subset \mathcal{C}_{\text{TM}}^{\text{ideal}},$$

where  $\mathcal{C}_{\text{fixed}}$  contains fixed finite physical machines,  $\mathcal{C}_{\text{ext}}$  contains implementations whose finite resources may be extended without a fixed a priori bound, and  $\mathcal{C}_{\text{TM}}^{\text{ideal}}$  contains the ordinary mathematical idealization of Turing computation.

A universe-scale simulation hypothesis normally needs at least the latter two idealizations unless a finite-state implementation of the entire relevant operational behavior can be demonstrated. This is an assumption about the host model and should not be left implicit.

### 3.4 Classical algorithmic computation

Turing’s analysis formalized effective symbolic computation through a machine whose local transition rule acts on finite symbolic configurations [20]. We write  $\mathcal{C}_{\text{TM}}$  for standard Turing-equivalent algorithmic implementations under effective encodings. The corresponding total and partial recursive functions form the reference class against which super-Turing claims are compared.

The Church–Turing thesis identifies the informal notion of effective algorithmic calculability with Turing computability. It is not a theorem that every physically realizable process must be Turing-computable. A claim about all physical computing mechanisms is a stronger *physical* thesis and requires additional assumptions about nature, as Section 5 discusses.

### 3.5 Reversible computation

Logical reversibility changes the transition structure of a machine without enlarging ordinary computability. Bennett showed that irreversible Turing computations can be reorganized as logically reversible computations while retaining general computational ability [3]. We write  $\mathcal{C}_{\text{REV}}$  for effectively specified reversible implementations.

At the level of ordinary decision power,

$$\mathcal{C}_{\text{REV}} \text{ and } \mathcal{C}_{\text{TM}}$$

are equivalent. Reversibility may profoundly affect thermodynamic accounting, memory trade-offs, and physical architecture, but it does not by itself produce a halting oracle.

### 3.6 Probabilistic computation

A probabilistic machine supplements effective state evolution with random choices. Let  $\mathcal{C}_{\text{PROB}}$  contain machines whose transition probabilities are effectively specifiable and whose output is interpreted under a fixed reliability rule.

For unbiased probabilistic Turing machines, Gill proved that if the computed partial function is defined by the output occurring with probability greater than 1/2, only partial recursive functions are probabilistically computable [12]. Randomness may alter complexity and algorithmic strategy without automatically expanding discrete computability.

This conclusion depends on the random resource. If a primitive coin has an arbitrary noncomputable bias

$$p = 0.p_1p_2p_3\dots$$

and the physical model permits arbitrary operational access to its digits, then nonrecursive information may already be present in the primitive constant. The extra power comes from the resource  $p$ , not from stochasticity as such. Such a model belongs in an extended resource class rather than ordinary probabilistic computation.

### 3.7 Parallel, distributed, and interactive computation

A simulator need not resemble a serial processor. It may distribute represented regions across many processors, use asynchronous communication, or compute only selected parts of a represented state

when they become operationally relevant. We write  $\mathcal{C}_{\text{PAR}}$  and  $\mathcal{C}_{\text{INT}}$  for broad effectively specified parallel and interactive architectures.

Any finite amount of effective parallelism can be serialized in principle, although perhaps with catastrophic loss of efficiency. The same is true of finite networks of effective components whose communication rules are computable. Thus finite parallelism alone does not furnish nonrecursive decision power.

Actual infinite parallelism is a different resource. If infinitely many processors perform distinct computations and an infinite family of results can influence a finite-time output, the model has introduced a completed infinity. Whether such a resource is physically realizable is a separate question; it must not be conflated with ordinary parallel hardware.

### 3.8 Quantum computation

Quantum computation is physically and mathematically different from classical symbolic computation, but standard quantum models do not thereby escape ordinary computability. Deutsch formulated the universal quantum-computer program [9], and Bernstein and Vazirani developed an efficient universal quantum Turing machine and the complexity-theoretic foundations of quantum computation [4].

Effectiveness of amplitudes matters. Adleman, DeMarrais, and Huang showed that rational amplitudes suffice for the ordinary bounded-error class BQP, while unrestricted complex amplitudes can import sets of arbitrary Turing degree into the formal model [1]. Nielsen similarly constructed mathematically valid quantum observables and unitary operators which, if physically realizable without further restriction, would contradict the Church–Turing thesis [15].

Accordingly, we define  $\mathcal{C}_Q^{\text{eff}}$  to contain quantum computational implementations whose gates, Hamiltonians, state preparations, amplitudes, and measurement rules are effectively specifiable in the relevant sense. The standard claim used in this paper is not that quantum systems are classically efficient, but that ordinary effectively specified quantum computation does not supply a generic oracle for nonrecursive discrete functions. A quantum model supplied with noncomputable amplitudes or non-effective measurements is treated as an extended-resource model.

### 3.9 Analog and continuous computation

Analog computation must be separated from infinite-precision computation. Shannon’s differential analyzer formalized computation through continuous physical variables and differential equations [19]. The presence of real-valued state variables does not itself imply access to infinitely many operational bits.

For physically meaningful comparison with Turing computation, continuous systems require represented-space computability. Following computable analysis, a real quantity is accessed through

an effective name, such as successively accurate rational approximations; an operator is computable if it transforms such names effectively [21].

We therefore distinguish:

- $\mathcal{C}_{\text{AN}}^{\text{eff}}$ : analog or continuous implementations with effectively represented parameters, states, dynamics, preparation, and readout;
- $\mathcal{C}_{\mathbb{R}}^{\text{exact}}$ : models in which exact real-valued states or operations are treated as primitive computational resources.

The Blum–Shub–Smale framework provides a mathematically rigorous theory of computation over rings and fields such as  $\mathbb{R}$  [5]. It should not be caricatured as simply “a Turing machine with magic reals”: it defines its own domain and complexity theory. Nevertheless, comparison with ordinary bit computation depends crucially on which real constants and access operations are allowed. If arbitrary noncomputable reals are physically available and their information can be extracted effectively, they can carry nonrecursive information. The extra informational resource must therefore be stated explicitly.

### 3.10 Hypercomputational models

We use  $\mathcal{C}_H$  as an umbrella for computational models whose declared resources allow the decision of at least some nonrecursive subsets of  $\mathbb{N}$  under an effective finite input/output interpretation. The literature contains several distinct routes [16].

**Oracle computation.** A machine is given primitive access to a set  $A$  through

$$O_A(n) = \begin{cases} 1, & n \in A, \\ 0, & n \notin A. \end{cases}$$

If  $A$  is the halting set, ordinary halting becomes decidable relative to the oracle. The additional resource is oracle access.

**Accelerating or Zeno computation.** A machine performs steps in times  $1/2, 1/4, 1/8, \dots$ , allowing an infinite sequence of computational steps to fit into finite external duration. The additional resource is a completed infinite operation in finite time.

**Relativistic hypercomputation.** Malament–Hogarth space-times have been studied as backgrounds in which one worldline can contain unbounded proper time while a result signal remains in the causal past of an event reached by another observer in finite proper time. Hogarth and later Etesi and N emeti analyze how such geometries could support supertasks and non-Turing decision procedures [13, 10]. The additional resource is the global causal structure of spacetime.

**Infinite-information primitives.** A physical constant, state, field configuration, or measurement primitive may encode a nonrecursive set at infinite precision. If arbitrary relevant bits can be operationally recovered, the nonrecursive information was supplied in the resource.

These proposals are not dismissed by definition. They are classified so that a failed test of standard computation cannot be rescued merely by saying “the simulator is a more powerful computer” without specifying what makes it more powerful.

### 3.11 A working standard class

For the first stage of the project, let  $\mathcal{C}_{\text{std}}$  denote effectively specified physical computational implementations whose uniformly usable finite decision procedures have Turing-recursive decision power. It is intended to include, under their standard effectiveness assumptions,

$$\mathcal{C}_{\text{TM}}, \quad \mathcal{C}_{\text{REV}}, \quad \mathcal{C}_{\text{PROB}}, \quad \mathcal{C}_{\text{PAR}}, \quad \mathcal{C}_{\text{INT}}, \quad \mathcal{C}_Q^{\text{eff}}, \quad \mathcal{C}_{\text{AN}}^{\text{eff}}.$$

This notation should not be read as saying that all members share one architecture, one complexity class, or even one natural state representation. It records a common boundary on ordinary effective decision power.

Extended classes are indexed by the additional resource, for example

$$\mathcal{C}_{\text{Oracle}}, \quad \mathcal{C}_{\text{Zeno}}, \quad \mathcal{C}_{\text{MH}}, \quad \mathcal{C}_{\mathbb{R}}^{\text{exact}}.$$

The first substantive hypothesis is then

$$H_{\text{std}} : \quad \exists C \in \mathcal{C}_{\text{std}} \exists R \in \mathfrak{R} : \quad C \Vdash_R U \quad \text{and} \quad \mathcal{B}_C = \mathcal{B}_U \quad \text{on } \mathcal{E}.$$

### 3.12 Host isolation and simulated time

One assumption used repeatedly in informal simulation arguments is that the external duration, energy expenditure, or memory pressure experienced by the host has no independent causal channel into the represented world except through the simulated dynamics themselves. Without this assumption, host resource consumption could become observable through externally coupled clocks, incoming host signals, or boundary effects.

We therefore distinguish an *isolated-host* hypothesis from a coupled-host hypothesis. Under isolated-host simulation, slowing the host by a factor of  $10^{100}$  need not alter any internal clock relation because all represented clocks advance only through host state transitions. Under coupled-host simulation, slowdown may be observable if the represented world is synchronized to an external signal.

This is not a theorem about simulations. It is a parameter of the host model.

### 3.13 Where computation becomes vacuous

Suppose every new physical capability discovered in  $U$  is simply added as a primitive capability of the simulator:

$$\mathcal{C}_0 \subset \mathcal{C}_1 \subset \mathcal{C}_2 \subset \dots$$

until the limiting class is defined as “whatever physical transformations are needed to reproduce  $U$ .” The hypothesis then excludes no possible operational behavior. The word *computation* contributes no independent constraint.

**Definition 3.3 (Non-vacuous computational hypothesis)** *A computational simulation hypothesis is non-vacuous only if the declared class of implementations excludes at least some well-defined operational behaviors or decision capabilities that are logically describable in the framework.*

The scientific content of the simulation hypothesis therefore lies in the limitations associated with its declared computational resources. The remainder of the paper asks whether those limitations can be brought into contact with the operational behavior of our universe.

## 4. Finite Observational Indistinguishability

The preceding sections deliberately make the simulation hypothesis stronger than mere replay: one computational implementation must match an open-ended family of internal protocols. This section establishes a complementary limitation. Finite empirical records are too weak, by themselves, to certify that their generator is noncomputable. The result does not establish the existence of one universal simulator. It establishes only a family of finite-fit results of the form  $\forall E \exists C_E$ . Keeping that quantifier order explicit is essential.

### 4.1 Finite protocols and finite transcripts

Let  $\mathcal{E}_{\text{fin}} \subset \mathcal{E}$  contain protocols that terminate after finitely many operational steps and produce finite records at finite measurement resolution. A realized transcript has the form

$$\tau = (i_1, o_1, i_2, o_2, \dots, i_k, o_k), \quad k < \infty.$$

The symbols  $i_j$  and  $o_j$  need not imply that underlying nature is discrete. They represent only the finite operational record retained by the experiment.

An adaptive finite protocol can be represented as a finite decision tree if both the set of tested branches and the recorded outcome alphabet are finite. Each internal node specifies the experimenter’s next action as a function of previous observations; each edge records a possible finitely distinguishable outcome.

## 4.2 A realized finite transcript is always reproducible

**Proposition 4.1 (Finite-transcript realization)** *For every particular finite experimental transcript*

$$\tau = (i_1, o_1, \dots, i_k, o_k),$$

*there exists a finite deterministic program that returns exactly the recorded outputs along that realized intervention sequence.*

*Proof.* Store the finite transcript as data. At step  $j$ , when the stored expected intervention  $i_j$  is received, return  $o_j$  and advance to the next stored state. The program contains finitely many states and symbols.  $\square$

The proposition is intentionally modest. It concerns the realized finite record, not the untested counterfactual behavior of the world. Nevertheless it blocks a common inference: no matter how surprising, apparently random, or algorithmically irregular a finite sequence is, its occurrence alone cannot prove that its source is noncomputable. Every particular finite string can be hard-coded.

## 4.3 Finite deterministic response trees

A stronger finite result applies when a complete finite adaptive response tree is specified deterministically.

**Proposition 4.2 (Finite deterministic decision-tree realization)** *Let  $T_E$  be a finite adaptive protocol tree with finitely many allowed interventions and a specified finite deterministic response at every tested branch. Then a finite-state transducer can reproduce the entire response tree exactly.*

*Proof.* Associate one machine state with each node of  $T_E$ . For every allowed intervention at a node, store the specified output and transition to the corresponding child node. Since  $T_E$  has finitely many nodes and edges, the transition table is finite.  $\square$

This proposition should not be overextended to arbitrary exact stochastic laws. If a finite branch is required to produce an exact probability equal to a noncomputable real, an effectively specified probabilistic machine need not reproduce that probability exactly. What remains true is more limited:

- any *realized* finite stochastic transcript can be reproduced exactly as a stored transcript;
- a finite deterministic response table can be reproduced exactly;
- any empirically estimated finite set of probabilities can be matched to the finite resolution of the measurements by computable rational probabilities.

The distinction between an observed frequency and an exact underlying probability is crucial.

#### 4.4 The quantifier lesson

From the preceding propositions one may infer statements of the form

$$\forall E \in \mathcal{E}_{\text{fin}} \exists C_E : \quad C_E \text{ matches the finite tested behavior of } E.$$

One may *not* infer

$$\exists C \forall E \in \mathcal{E} : \quad \mathcal{B}_C(E) = \mathcal{B}_U(E).$$

The program  $C_E$  may be an ad hoc lookup table constructed separately for each experiment. The simulation hypothesis requires one implementation whose behavior remains correct across the entire declared protocol family.

This is the main limitation of finite-tree arguments. They establish an obstacle to finite-data refutation, but they do not establish global computational indistinguishability.

It is also useful to distinguish two kinds of class-level separation:

$$\exists E \forall C \in \mathcal{C} : \quad \mathcal{B}_C(E) \neq \mathcal{B}_U(E)$$

would be a single universal experiment against the entire class, while

$$\forall C \in \mathcal{C} \exists E_C : \quad \mathcal{B}_C(E_C) \neq \mathcal{B}_U(E_C)$$

is sufficient to show that no single implementation in  $\mathcal{C}$  realizes the complete behavior. The second statement is the actual negation of  $H_{\mathcal{C}}$  and may hold even when no universal one-shot discriminator exists.

#### 4.5 Finite extensional data cannot certify noncomputability

The same point appears more sharply for functions.

Suppose a physical procedure is claimed to compute

$$f : \mathbb{N} \rightarrow \{0, 1\}.$$

An experiment tests only the finite set of inputs  $n_1, \dots, n_m$  and observes outputs  $b_1, \dots, b_m$ .

**Proposition 4.3 (Finite agreement with a recursive function)** *For every finite extensional dataset*

$$D = \{(n_1, b_1), \dots, (n_m, b_m)\}, \quad b_j \in \{0, 1\},$$

*there exists a total recursive function  $g : \mathbb{N} \rightarrow \{0, 1\}$  satisfying  $g(n_j) = b_j$  for all tested inputs.*

*Proof.* Define

$$g(n) = \begin{cases} b_j, & n = n_j \text{ for some } j, \\ 0, & \text{otherwise.} \end{cases}$$

Because the tested set is finite, the lookup is effective. □

The conclusion must be worded precisely:

Finite extensional input-output data alone cannot certify noncomputability.

This does *not* mean that noncomputability lacks finite mathematical proofs. A finite proof can establish that a specifically defined function is nonrecursive. What a finite dataset cannot do by itself is determine that the unknown function generating it is nonrecursive.

For example, even if a physical device correctly returned the first million known instances of a nonrecursive target function, a recursive function could be constructed that agrees on precisely those tested cases and differs thereafter. The experimental record alone cannot distinguish them.

## 4.6 The role of theory

Empirical science routinely uses finite observations to support mathematical theories whose claims range over infinitely many possible instances. The same strategy is available here, but the source of the infinite generalization must be explicit.

The relevant evidential structure is not

many successful trials  $\Rightarrow$  noncomputable,

but rather

physical assumptions  $A$   
 + uniform experimental construction  
 + mathematical theorem  
 $\Rightarrow$  computational separation.

Experiments support the physical assumptions and validate the regime in which the theorem is applied. The theorem, not finite extrapolation from a table, supplies the universal claim.

This is an ordinary form of theory-mediated inference, but the present problem has an unusually strict bookkeeping requirement. If the theorem depends on infinite precision, an uncomputable initial state, an oracle-like measurement primitive, a completed infinity, or some other non-effective resource, that resource must be counted as part of the physical theory rather than attributed mysteriously to “nature’s computation.”

## 4.7 Uniform operational decision protocols

The most direct route to separating  $U$  from  $\mathcal{C}_{\text{std}}$  would be a uniformly usable physical decision procedure for a nonrecursive set.

**Definition 4.4 (Uniform operational decision protocol)** *A family  $\mathcal{P} = \{P_n\}_{n \in \mathbb{N}}$  is a uniform operational decision protocol for a predicate  $f : \mathbb{N} \rightarrow \{0, 1\}$  if:*

1. **Effective input and preparation.** *Given the finite encoding of  $n$ , there is an effective finite recipe for specifying the physical preparation and intervention defining  $P_n$ .*
2. **Finite instance execution.** *Every individual instance produces an operational result after finite proper or laboratory time for the observer who performs it, under the physical assumptions of the protocol.*
3. **Finite readout.** *The answer is recoverable by a finite operational measurement rule.*
4. **Uniform reliability.** *There exists a fixed  $\eta < 1/2$  such that*

$$\Pr[P_n = f(n)] \geq 1 - \eta \quad \forall n.$$

5. **Theoretical generality.** *A stated physical theory and mathematical argument entail the above behavior for arbitrary admissible  $n$ , rather than merely for the finite subset already tested.*

If  $f$  is nonrecursive, such a protocol would constitute operational evidence for a physical resource outside ordinary effective decision power, conditional on the physical theory supporting the construction.

Version 0.1 incorrectly treated the presence of any “oracle-like” physical resource as grounds for rejecting the protocol. That is too strong. If nature independently supplies a stable, finitely accessible resource whose operational behavior implements a nonrecursive predicate, that would be precisely the kind of result capable of excluding  $\mathcal{C}_{\text{std}}$ . What must be excluded is a circular preparation procedure that requires the experimenter already to possess the nonrecursive answers in order to build each instance.

The proper conclusion would then be resource-relative:

$$\text{nature provides resource } R \notin \mathcal{C}_{\text{std}},$$

not “the experiment cheated.” The subsequent question would be whether a broader computational class containing  $R$  remains viable.

## 4.8 Finite experiments can still falsify restricted classes

The finite-indistinguishability results do not imply that finite experiments are powerless. If a computational class makes a specific finite prediction, a contradictory finite outcome can falsify that restricted class just as in ordinary science. For example, an implementation class may impose a maximum state count, a lattice symmetry, a fixed precision bound, or a particular probability law. Those are ordinary empirically testable hypotheses.

What finite data cannot establish without theory is the stronger unrestricted conclusion that *no* effectively specified computational process can reproduce the world. The broader the computational class, the more structural the separation theorem must become.

## 4.9 Implications for experiment design

The finite results provide a filter for candidate experiments:

1. **Reject pure transcript arguments.** A surprising finite sequence, including a sequence that appears algorithmically random, cannot by itself demonstrate noncomputability.
2. **Reject finite lookup challenges.** Testing a very large but finite family of counterfactual responses does not exclude a finite program that stores precisely that family.
3. **Demand uniformity.** A candidate protocol should be generated by one effective rule from arbitrary finite inputs rather than designed separately with instance-specific hidden information.
4. **Identify the theorem.** The argument must explain why the general physical protocol computes or decides something outside the target computational class.
5. **Name the resource.** If the power enters through precision, causal structure, infinite parallelism, a physical constant, or measurement access, that resource belongs explicitly in the hypothesis.
6. **Audit the quantifiers.** Evidence that each tested experiment can be fitted by some program says nothing by itself about whether one program reproduces the full behavior map.

These constraints move the project away from searching for “glitches” and toward a more mathematical question: what complete operational behaviors are impossible for a declared class of physical computational implementations?

## 5. Physical Limits of Computation

Section 4 showed why finite records alone are weak evidence about computability. A separation theorem therefore needs additional structure: physical principles strong enough to bound what a host computational mechanism can do, or physical principles strong enough to imply that the observable

universe performs an operation outside that bound. This is where the physical Church–Turing literature becomes useful.

## 5.1 From the Church–Turing thesis to conditional physical theorems

The ordinary Church–Turing thesis concerns effective calculability. It does not, by itself, say that every process permitted by every conceivable physical theory is Turing-computable. The stronger assertion that all physically realizable computing devices have only Turing power is a physical thesis whose truth depends on what counts as physically realizable.

Gandy’s 1980 paper is important precisely because it does not merely assert such a physical thesis. It describes a class of discrete deterministic mechanisms subject to structural principles and proves that their successive states form a computable sequence [11]. The result therefore has the logical form

$$A_{\text{Gandy}} \implies \text{computable evolution.}$$

The assumptions do the work. The theorem does not establish

$$\text{all possible physics} \implies \text{Turing computation.}$$

Copeland and Shagrir emphasize this limitation explicitly. They argue that Gandy’s class should not simply be identified with all ideal physical computing machines and discuss candidate ideal machines outside it [8]. For the present project this criticism is productive: Gandy provides one well-defined host class to analyze, not a universal boundary that may be assumed without argument.

## 5.2 What Gandy contributes

Gandy’s formal setting is designed for discrete deterministic mechanisms, not arbitrary continuous or quantum physics. In broad physical terms, the program uses constraints corresponding to finite local complexity, structural uniformity, and local causal dependence. The precise formal principles are more abstract than slogans such as “finite memory” or “speed of light,” and care should be taken not to replace the theorem with an informal paraphrase.

For the present paper, the useful lesson is methodological. A physical computability theorem should proceed in three stages:

1. state a class of mechanisms precisely;
2. identify explicit structural principles satisfied by that class;
3. derive a computability bound from those principles.

Only then is it meaningful to ask whether a physical host capable of implementing  $U$  must fall within that bounded class.

Gandy also constructed counterexamples showing that weakening his principles can permit mechanisms that calculate arbitrary number-theoretic functions [11]. These countermodels do not prove that nature violates Turing computability. They show where a particular theorem ceases to apply and which resources can carry additional power.

### 5.3 An assumption matrix, not yet a lattice

Version 0.1 called the emerging structure an “assumption lattice.” That language is premature until a formal partial order and closure relations are defined. For now we use an *assumption matrix*: a list of candidate physical principles whose presence or absence can be studied systematically.

Representative entries include:

| Assumption family                             | Operational or computational role                                                                                          |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <b>Effective local specification</b>          | Prevents nonrecursive information from entering through primitive constants, transition rules, or inaccessible exact data. |
| <b>Bounded accessible information density</b> | Prevents a bounded controllable region from functioning as an unlimited directly addressable information store.            |
| <b>Bounded causal dependence</b>              | Prevents one finite-time output from requiring access to an unrestricted spatial domain through a single primitive update. |
| <b>Uniformity / homogeneity</b>               | Prevents an undecidable set from being encoded directly into location- or time-dependent laws.                             |
| <b>Effective initial conditions</b>           | Prevents the initial global state itself from serving as an undeclared oracle.                                             |
| <b>Finite operational readout</b>             | Requires a finite observer to obtain the computational answer by an effective measurement procedure.                       |

These are not asserted as jointly necessary conditions for all computation. They are candidates for theorem premises. The research task is to determine which combinations are sufficient for which computability conclusions and which violations are physically exploitable rather than merely mathematically describable.

### 5.4 Counterexamples identify resources, not automatic hypercomputers

Several intuitive failure modes recur in Gandy-style discussions. If spatial uniformity is dropped, nonrecursive information can be placed into a spatial pattern. If temporal uniformity is dropped, it can be inserted into the time dependence of the transition law. If bounded accessible information density is dropped, a finite region may be assigned unbounded or noncomputable information. If effective initial conditions are dropped, the initial state may already contain an oracle.

The correct inference is not

$$\neg A_i \implies \text{hypercomputation.}$$

It is only

$$\neg A_i \implies \text{the previous computability theorem no longer applies,}$$

and in some formal models one can then construct super-Turing counterexamples. An observed continuum, nonlocal correlation, or spatial inhomogeneity is not itself a halting oracle.

For this paper, every proposed violation must pass the operational test of Section 4: can the resource be prepared or accessed uniformly from finite inputs, can each instance terminate in finite observer time, and can a finite readout recover the answer with a fixed reliability gap?

## 5.5 Quantum mechanics and effectiveness

Quantum theory illustrates why “nonclassical” and “noncomputable” must be kept separate. Arrighi and Dowek formulate a quantum analogue of the Gandy program, retaining principles such as homogeneity, bounded information propagation, and finite-dimensional local quantum systems while imposing effectiveness conditions on the quantum description [2]. Their result shows how a quantum physical Church–Turing conclusion can be obtained from explicit quantum assumptions rather than from classicality.

The effectiveness restriction is substantive. Nielsen showed that the mathematical formalism of quantum theory allows one to write down observables and unitary operations which, if physically realizable as unrestricted primitives, would contradict the Church–Turing thesis [15]. Adleman, DeMarrais, and Huang likewise demonstrate that unrestricted complex amplitudes can import arbitrary Turing degrees into a quantum model, whereas rational amplitudes suffice for ordinary BQP [1]. Bernstein and Vazirani analyze universal quantum computation with controlled precision of transition amplitudes [4].

Thus the defensible statement is not

$$\text{quantum mechanics} \implies \text{Turing computability.}$$

It is conditional:

$$\text{effectively specified standard quantum computation} \implies \text{ordinary recursive decision power,}$$

while the bare mathematical syntax of Hilbert spaces contains formally definable non-effective objects unless physical admissibility restricts them.

This distinction is central to the present project. If experimentally accessible quantum physics supplied a stable measurement or dynamical primitive encoding a nonrecursive set, the correct response would not be to dismiss it as “not computation.” It would be to conclude that  $\mathcal{C}_{\text{std}}$  is too narrow and to identify the newly observed resource explicitly.

## 5.6 Continuous physics and computable analysis

Exactly the same caution applies to classical continuum physics. A field theory can employ real-valued functions without granting an experimenter exact arbitrary real-number access. Computable analysis distinguishes continuous mathematical state spaces from non-effective data by specifying how states are named and how approximations are produced [21].

Any claim that a differential equation, wave equation, or continuum model supplies super-Turing computation therefore requires an operational audit:

- Are the initial conditions computably preparable?
- Are the coefficients and boundary data effectively specified?
- Does the relevant solution operator preserve computability in the chosen representation?
- Is the noncomputable feature robust under finite perturbation and finite measurement precision?
- Can an observer recover a discrete answer without already supplying nonrecursive information in the preparation?

This is why represented-space computability belongs in the foundations of the paper rather than appearing only as a later technical refinement.

## 5.7 Host constraints need not resemble simulated constraints

A recurrent logical error in simulation arguments is to transfer a property of the represented world directly to the host. Suppose  $U$  exhibits a globally coupled update rule or apparently instantaneous influence. That does not imply that the host computer itself communicates superluminally. A local serial host may compute a globally coupled represented update by performing a very long sequence of local operations before advancing the represented clock.

Likewise:

- continuous simulated variables do not require continuous host variables;
- irreversible represented dynamics do not require irreversible host dynamics;
- spatial homogeneity in  $U$  does not require spatial homogeneity in the host;
- represented causal speed need not equal host signal speed;
- one simulated second need not correspond to any fixed amount of host time.

Therefore, observing that  $U$  violates one of Gandy's host assumptions is insufficient. The relevant question is whether the *operational behavior that the host must reproduce* lies outside the behavior

class realizable by hosts satisfying those assumptions.

This is the bridge between Sections 4 and 5. Internal physics constrains a simulator only through the input-output or interactive behavior that counterfactual fidelity obliges the simulator to reproduce.

## 5.8 Resource bounds and causal isolation

Arguments from computational expense require an additional coupling assumption. Under an isolated-host model, a host may consume an arbitrarily large finite amount of external time to generate the next represented event, while every clock inside  $U$  remains paused until that event is produced. Internal observers see no slowdown relative to one another.

This ceases to be true if simulated time is coupled to an independently observable host clock or if the represented world receives external signals whose timing is set by host dynamics. We therefore make the condition explicit:

**Definition 5.1 (Host-time isolation)** *A simulation is host-time isolated relative to  $\mathcal{E}$  if the transcript probabilities of every allowed internal protocol depend only on the represented dynamical state and not on an independently observable external measure of the host's computational duration, energy cost, or memory pressure.*

Under host-time isolation, complexity alone is generally not an internal discriminator. Nonrecursive decision power is more promising because additional finite time does not enable a Turing-equivalent machine to decide a nonrecursive set. Under a coupled-host hypothesis, by contrast, complexity and resource exhaustion may become empirically relevant and should be analyzed separately.

## 5.9 A conditional exclusion theorem scheme

The previous machinery supports a clean conditional result.

**Proposition 5.2 (Operational nonrecursion excludes the standard class)** *Assume that:*

1.  *$U$  supports a uniform operational decision protocol  $\{P_n\}_{n \in \mathbb{N}}$  for a predicate  $f : \mathbb{N} \rightarrow \{0, 1\}$  with a fixed error bound  $\eta < 1/2$ ;*
2. *a stated physical theory entails that the protocol satisfies the preparation, termination, readout, and reliability conditions of Section 4 for every  $n$ ;*
3.  *$f$  is mathematically nonrecursive; and*
4. *every  $C \in \mathcal{C}_{\text{std}}$ , when coupled to an effectively specified uniform protocol family, can decide only recursive predicates with such a fixed reliability gap.*

*Then no  $C \in \mathcal{C}_{\text{std}}$  can be operationally equivalent to  $U$  on a protocol class containing all  $P_n$ . Hence*

$$\mathcal{B}_U \notin \mathfrak{B}(\mathcal{C}_{\text{std}}).$$

*Proof sketch.* Suppose for contradiction that some  $C \in \mathcal{C}_{\text{std}}$  is operationally equivalent to  $U$  on every  $P_n$ . Then running the corresponding uniform protocol against  $C$  decides  $f(n)$  with the same fixed error gap. By assumption 4, the decided predicate must be recursive, contradicting assumption 3.  $\square$

The theorem is deliberately conditional. Physics must supply the protocol and support its universal behavior; recursion theory supplies the contradiction. Finite experiments can support the physical premises but cannot independently verify their validity for infinitely many  $n$ .

If the physical protocol derives its power from a newly discovered resource  $R$ , the correct conclusion is

$$\mathcal{B}_U \notin \mathfrak{B}(\mathcal{C}_{\text{std}}),$$

while a broader class  $\mathcal{C}_{\text{std}+R}$  may remain viable. The paper’s aim is to make that retreat explicit rather than allow the computational hypothesis to absorb  $R$  without changing its statement.

### 5.10 Behavioral emulation is not yet constitutive simulation

The opposite direction requires equal caution. Suppose the complete operational behavior map  $\mathcal{B}_U$  is uniformly computable in some suitable sense: given a finite protocol description and desired finite precision, an effective procedure computes the corresponding observable probabilities. This strongly suggests the existence of an abstract algorithmic *emulator* of the operational predictions.

But two further steps are nontrivial.

First, one must pass from separately computing finite protocol predictions to constructing a single coherent interactive process whose responses remain mutually consistent across all adaptive protocols. This is a global consistency problem, not merely a collection of lookup tables.

Second, even if such an abstract interactive emulator exists, that establishes

$$\mathcal{B}_U \in \mathfrak{B}(\mathcal{C}_{\text{abstract}})$$

only at the level of operational behavior. It does not by itself establish the ontological claim

$$C \Vdash_R U.$$

Computable behavior does not imply that the world is actually implemented by a computer. This is the same computable-versus-computed distinction that motivated Section 2.

Accordingly, version 0.2 does *not* state a “computable-physics indistinguishability theorem” as if it were already proved. It states a theorem program.

**Behavioral indistinguishability program.** Identify sufficient conditions under which a uniformly computable operational behavior map  $\mathcal{B}_U$  admits a single effectively specified

interactive computational process  $C$  satisfying

$$\mathcal{B}_C(E) = \mathcal{B}_U(E) \quad \forall E \in \mathcal{E}.$$

Then separately determine what additional assumptions, if any, are required to promote this behavioral emulator to a constitutive implementation  $C \Vdash_R U$  under an admissible implementation relation.

A proof of the first statement would show that no internal protocol in  $\mathcal{E}$  distinguishes the two behaviorally. A proof of the second would concern the stronger ontology of simulation. Keeping them separate prevents a behavioral theorem from silently becoming a metaphysical conclusion.

### 5.11 What Section 5 establishes, and what it does not

The present section supports four conclusions.

First, physical computability results are conditional. Gandy and Arrighi–Dowek are valuable because they make their assumptions visible rather than because they establish a universal theorem about all possible physics [11, 2, 8].

Second, violations of those assumptions identify possible sources of additional computational power but do not automatically constitute operational hypercomputation.

Third, the most direct separation from  $\mathcal{C}_{\text{std}}$  would be a uniform, finite-instance, finitely readable physical protocol for a mathematically nonrecursive predicate. A naturally existing non-effective physical resource would count as a discovery, not as a disallowed trick; it would simply force an explicit enlargement of the computational class.

Fourth, computability of the observed laws does not establish that the universe is computed. At most it may support a behavioral-emulation theorem. Constitutive implementation remains an additional relation.

These conclusions narrow the next phase of the project. Before adding more speculative experiments, the framework should be tested against self-reference and physical-inference limits, against undecidability results in mathematical physics, and against the formal theory of behavioral equivalence. The central question is no longer whether a sufficiently complex simulation must “glitch.” It is whether the complete operational behavior of our universe belongs to the behavior class of a scientifically non-vacuous computational implementation class.

## References

- [1] L. M. Adleman, J. DeMarrais, and M.-D. Huang, “Quantum computability,” *SIAM Journal on Computing*, 26(5), 1524–1540 (1997). doi:10.1137/S0097539795293639.
- [2] P. Arrighi and G. Dowek, “The physical Church–Turing thesis and the principles of quantum theory,” *International Journal of Foundations of Computer Science*, 23(5), 1131–1145 (2012). doi:10.1142/S0129054112500153.
- [3] C. H. Bennett, “Logical reversibility of computation,” *IBM Journal of Research and Development*, 17(6), 525–532 (1973). doi:10.1147/rd.176.0525.
- [4] E. Bernstein and U. Vazirani, “Quantum complexity theory,” *SIAM Journal on Computing*, 26(5), 1411–1473 (1997). doi:10.1137/S0097539796300921.
- [5] L. Blum, M. Shub, and S. Smale, “On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines,” *Bulletin of the American Mathematical Society*, 21(1), 1–46 (1989). doi:10.1090/S0273-0979-1989-15750-9.
- [6] N. Bostrom, “Are we living in a computer simulation?” *The Philosophical Quarterly*, 53(211), 243–255 (2003). doi:10.1111/1467-9213.00309.
- [7] D. J. Chalmers, “Does a rock implement every finite-state automaton?” *Synthese*, 108(3), 309–333 (1996). doi:10.1007/BF00413692.
- [8] B. J. Copeland and O. Shagrir, “Physical computation: How general are Gandy’s principles for mechanisms?” *Minds and Machines*, 17(2), 217–231 (2007). doi:10.1007/s11023-007-9058-2.
- [9] D. Deutsch, “Quantum theory, the Church–Turing principle and the universal quantum computer,” *Proceedings of the Royal Society of London A*, 400(1818), 97–117 (1985). doi:10.1098/rspa.1985.0070.
- [10] G. Etesi and I. Németi, “Non-Turing computations via Malament–Hogarth space-times,” *International Journal of Theoretical Physics*, 41(2), 341–370 (2002). doi:10.1023/A:1014019225365.
- [11] R. Gandy, “Church’s thesis and principles for mechanisms,” in J. Barwise, H. J. Keisler, and K. Kunen (eds.), *The Kleene Symposium*, Studies in Logic and the Foundations of Mathematics, vol. 101, 123–148 (North-Holland, 1980). doi:10.1016/S0049-237X(08)71257-6.
- [12] J. Gill, “Computational complexity of probabilistic Turing machines,” *SIAM Journal on Computing*, 6(4), 675–695 (1977). doi:10.1137/0206049.
- [13] M. Hogarth, “Non-Turing computers and non-Turing computability,” *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, 1994(1), 126–138 (1994). doi:10.1086/psaprocbienmeetp.1994.1.193018.

- [14] C. Horsman, S. Stepney, R. C. Wagner, and V. Kendon, “When does a physical system compute?” *Proceedings of the Royal Society A*, 470(2169), 20140182 (2014). doi:10.1098/rspa.2014.0182.
- [15] M. A. Nielsen, “Computable functions, quantum measurements, and quantum dynamics,” *Physical Review Letters*, 79(15), 2915–2918 (1997). doi:10.1103/PhysRevLett.79.2915.
- [16] T. Ord, “The many forms of hypercomputation,” *Applied Mathematics and Computation*, 178(1), 143–153 (2006). doi:10.1016/j.amc.2005.09.076.
- [17] G. Piccinini, *Physical Computation: A Mechanistic Account* (Oxford University Press, 2015). doi:10.1093/acprof:oso/9780199658855.001.0001.
- [18] D. Sangiorgi, *Introduction to Bisimulation and Coinduction* (Cambridge University Press, 2011). doi:10.1017/CBO9780511777110.
- [19] C. E. Shannon, “Mathematical theory of the differential analyzer,” *Journal of Mathematics and Physics*, 20, 337–354 (1941). doi:10.1002/sapm1941201337.
- [20] A. M. Turing, “On computable numbers, with an application to the Entscheidungsproblem,” *Proceedings of the London Mathematical Society*, s2-42(1), 230–265 (1936). doi:10.1112/plms/s2-42.1.230.
- [21] K. Weihrauch, *Computable Analysis: An Introduction* (Springer, 2000). doi:10.1007/978-3-642-56999-9.