On the Boundaries of Formal Intelligence

R. Everheart

Abstract

We define a "hierarchical Turing machine" to be a sequence of total Turing machines. We then prove that any Turing machine in a hierarchical Turing machine is incapable of deciding whether subsequent Turing machines can possibly reach "novel", states without new input. We prove this for an exceedingly broad class of hierarchical Turing machines, without invoking Rice's theorem. If we consider human-like "intelligence" as entailing capacity for being able to infer truly new knowledge after learning new information, this demonstrates that determining whether a machine exhibits human-like "intelligence" is undecidable. We also define the complexity class AI to be the class of problems that require "novel" insights to solve. It follows from this definition that, in the general case, it is undecidable whether any given problem is in AI. Finally, we show that hierarchical Turing machines that exhibit persistent novelty are, by their very definition, impossible to control by any "less intelligent" hierarchical Turing machine.

1 Introduction

The question "does this machine exhibit human-like intelligence?" is foundational in the study of artificial intelligence. Unfortunately, it is difficult to formalize a rigorous definition of "human-like intelligence", and, as a result, approaches for answering this question intrinsically involve a human referee which vets the capabilities of the machine participating in a test for "intelligence". Correspondingly, debate has ensued whether passing tests such as the Turing test [17], Lovelace test [4], etc. is indeed indicative of "human-like" intelligence, or, conversely, is indicative of a perceived illusion contrived by human intelligence [15]. Irregardless of the outcome of such debates, a lack of formalization for "human-like intelligence" has resulted in a lack of rigorous formalization for problems that are hypothesized as requiring "human-like intelligence" to solve, being the AI problems.

Nevertheless, it has been proposed that the ability to learn new information is intrinsically tied to intelligence [9]. We can consider a loose definition of "learning" to be taking information from one's environment, and using that information to do something that could not be done without it [16]. We will use the term "novel insight" to describe an insight that could only be made after receiving certain, new information. We claim that having "human-like intelligence" entails the capacity to persistently make novel insights when granted truly new information. Consequentially, any test that is accepted as validating the intelligence of a machine in context to a human being will, simultaneously, validate the machine's ability to produce novel insights.

We can formalize this notion rigorously. In summary, we will consider an intelligent machine to be a Turing machine, which, after learning certain information, runs another Turing machine which accepts the output of the old Turing machine, and new information

as input. If this new Turing machine can access states that were not computable from any previous states by any possible Turing machine, we say the new Turing machine has the capacity for novel insight. It is the goal of this paper to show that determining whether a Turing machine can make such novel insights when given new information is undecidable in the general case. Similarly, we can frame the AI problems as the class of problems which require a Turing machine that has the capacity for novel insights in order to solve. It follows that the problem of ascertaining a problem's membership in AI, is generally undecidable. The proofs used in these results do not utilize Rice's theorem [11], and instead involve representing Turing machines as formal systems in the style of Feferman [6]. Effectively, this means our results generalize to a larger, and more practical, class of Turing machines.

In any case, we will proceed by going into detail as to our formalization, and defining that which is necessary to prove our results. We will not work with Turing machines directly. Rather, we will work for an alternate form of Turing machines known as hierarchical Turing machines.

2 Hierarchical Turing Machines

Let us begin by with a quick thought experiment. Imagine a human or machine agent $\mathcal{A}$ who is capable of accessing a domain of knowledge. We will make no attempts to define what could possibly constitute this domain of knowledge, nor will we attempt to describe how our agent accesses it. Though, one might consider this domain of knowledge as being a series of valid empirical observations that the individual is able to make about the world around him/her, a series of possible confabulatory hallucinations, or as being something else entirely. We are intentionally vague in this regard, as, in this thought experiment, we are exclusively concerned about the practical utility additional knowledge provides to our agent.

At a given position in this domain of knowledge, our agent $\mathcal{A}$ will be said to have a state of mind $\mathcal{A}_k$, representing its current capacity for knowledge. That is to say, $\mathcal{A}_k$ refers to knowledge the agent may obtain at position k through some reasoning process, without extending itself with additional knowledge. Our agent $\mathcal{A}$ then reifies new information from its inhabited domain of knowledge with its current state of mind $\mathcal{A}_k$, producing a new state of mind $\mathcal{A}_{k+1}$. Much like before, we will make no attempts to formalize a definition of "reification" here. Rather, we will simply consider "reification" at position k to mean "accepting as true" at position $k+1$.

The question that is posed by this thought experiment is this: Does our agent, at state of mind $\mathcal{A}_k$, already encompass the reasoning potential of $\mathcal{A}_{k+1}$? This is effectively asking: Are there certain insights that an agent cannot make without accessing additional knowledge about its world, domain of knowledge, etc.? It has been claimed that a capacity for learning is an intrinsic aspect of intelligence [9]. If one agrees with this claim, and considers learning to depend on the advent information that was hitherto inaccessible to our agent, then one must, necessarily, accept that there are insights that cannot be gleaned without acquiring certain information.

In this paper, we will operate under the assumption that the question posed by our thought experiment must be answered in the affirmative. Interestingly, answering the question in the affirmative enables us to formulate a boundary condition for intelligence. This boundary is, loosely, a machine's capacity for learning – though we will use different terminology (i.e., *persistent novelty*) throughout this paper.

We will embark towards a formal definition of this boundary condition. This boundary condition is best formalized in regards to a specific construct, known as a hierarchical Turing machine, which is intended to represent an agent's capabilities at certain moments in the world it inhabits.

Definition 1. Let $H = (H_k)_{k \in S}$ be a sequence indexed by a linearly ordered set $S \subseteq \mathbb{N}$, where each H_k is a Turing machine executed on input $X_k \in \Sigma^*$. We say that H is a *hierarchical Turing machine* if the following conditions hold:

1. For each $k > \min S$, the input X_k is constructed as the pair $\Big(\text{Output}\big(H_{k-1}(X_{k-1})\big),\, X'_k\Big)$, where $X'_k \in \Sigma^*$ is a supplementary input string;
2. For each $k < \max S$, the machine H_k halts on input X_k.

Remark. Given the sequences $H = (H_k)_{k \in S}$, and $X := (X_k)_{k \in S}$, where $X_k \in \Sigma^*$, we write $H(X)$ to denote that each H_k is executed X_k.

In the above definition, H can be viewed as being the agent from our thought experiment. Each H_k can be viewed as being the state of mind of the agent at position k, and each input X'_k can be viewed as being the information the machine takes from its world, or domain of knowledge at position k.

Hierarchical Turing machines correspond to normal Turing machines. We may represent a hierarchical Turing machine H as a normal Turing machine M by taking each level H_k and running them successively. The input to M becomes becomes a single vector X^*, which is precisely the combination of all inputs X_k to H_k. In this sense, hierarchical Turing machines might not seem like an interesting construct. For now, we will make no attempt to refute this claim. We will simply state that a certain property can be discussed in terms of hierarchical Turing machines that cannot be discussed as naturally in terms of normal Turing machines.

In any case, this property that we wish to discuss is dependent on the notion of a recursive closure. Briefly, a recursive closure of some Turing machine H_k in a hierarchical Turing Machine H, is the set of all states that may computed – by any preceding Turing machine – from some subset of the states of all preceeding Turing machines. Formally, we define this set as follows.

Definition 2. Let $H := (H_k)_{k \in S}$ be a hierarchical Turing machine, and let $\mathcal{F}_k$ be the set of all total recursive functions H_i with $i \leq k$. Define the *recursive closure* $\mathrm{RC}_k(H, X)$ inductively as the smallest set satisfying:

1. **Base case:** $\bigcup_{i \leq k} \mathrm{St}(H_i, X_i) \subseteq \mathrm{RC}_k(H, X)$.
2. **Closure under function application:** If $f \in \mathcal{F}_k$ is an n-ary total recursive function, and $s_1, \ldots, s_n \in \mathrm{RC}_k(H, X)$, then $f(s_1, \ldots, s_n) \in \mathrm{RC}_k(H, X)$.

It is crucial to note that membership in $\mathrm{RC}_k(H, X)$ is generally undecidable. This is because deciding membership in certain recursive closures reduces from the halting problem.

Theorem 1. *Let $H := (H_k)_{k \in S}$ be a hierarchical Turing machine. Then, for a given state s, $s \in RC_k(H, X)$ is generally undecidable.*

Proof. In order to demonstrate this, we will reduce the problem of determining membership in the recursive closure of a specific H_k from the Halting Problem. We will first fix an index

$j \in \mathbb{N}$ encoding a Turing machine M_j, and an input $x \in \mathbb{N}$. Next, we will construct a finite sequence $H = (f_0, f_1)$ of total recursive functions such that

$$M_j \text{ halts} \iff 1 \in \mathrm{RC}_1(H, X)$$

Let $f_0(j, x, t)$ simulate $M_j(x)$ for t steps, and return 1 if $M_j(x)$ halts within t steps, or 0 if otherwise. Since step-bounded simulation is primitive recursive, f_0 is total recursive. That is to say, $f_0(j, x, t)$ is defined to be

$$f_0(j, x, t) \coloneqq \begin{cases} -1 & \text{if } M_j(x) \text{ halts within } t \text{ steps} \\ t + 1 & \text{otherwise} \end{cases}$$

Here, making it so $f_0(j, x, t)$ outputs $t + 1$ when M_j fails to halt emphasizes that arbitrarily large values of t can be generated within the recursive closure, enabling unbounded search over simulation steps through successive compositions. Keeping this in mind, let $f_1(y)$ be the identity function $f_1(y) \coloneqq y$. Clearly, f_1 is total recursive. Now consider the composition $f \coloneqq f_1 \circ f_0$, and define $X_0 \coloneqq (j, x, t)$ for some t, and $X_1 \coloneqq y$. Let $s = -1$. Then

$$s \in \mathrm{RC}_1(H, X) \iff \exists t \in \mathbb{N}, f_1\big(f_0(j, x, t)\big) = -1$$

holds if and only if $M_j(x)$ halts within t steps for some t. In other words, $-1 \in \mathrm{RC}_1(H, X)$ is true if and only if $M_j(x)$ halts. So, if there existed a decision procedure for membership in $\mathrm{RC}_1(H, X)$, we could decide the Halting Problem. Therefore, membership in $\mathrm{RC}_k(H, X)$ is undecidable in general. □

The above undecidability result holds even though many instances of $\mathrm{RC}_k(F, X)$ have decidable membership. Crucially, there exists a general construction that allows encoding of an undecidable problem via total recursive functions and their recursive closure. This suffices to establish that membership in $\mathrm{RC}_k(F, X)$ is undecidable in general. In any case, it should also be quickly noted that, if $s \notin \mathrm{RC}_k(H, X)$, then $s \notin \mathrm{RC}_i(H, X)$ for all $i < k$.

Proposition 1. *Let $H = (H_k)_{k \in S}$ be a hierarchical Turing machine. If $s \notin RC_k(H, X)$ for some k, then, for each $i < k$, $s \notin RC_i(H, X)$.*

Proof. Let $\mathcal{F}_k$ denote the closure of all functions in H_k under function application, and similarly for $\mathcal{F}_i$. Suppose there existed an n-ary function f such that $f \in \mathcal{F}_i$. Then, clearly, $f \in \mathcal{F}_k$. By contraposition, we know for all $f \notin \mathcal{F}_k$, then $f \notin \mathcal{F}_i$. From our given condition that $s \notin \mathrm{RC}_k(H, X)$, we may conclude that there is does not exist any n-ary function $f \in \mathcal{F}_k$ that outputs s. Thus, there does not exist any n-ary function $f \in \mathcal{F}_i$ that outputs s. Consequentially, we may conclude that $s \notin \mathrm{RC}_i(H, X)$. □

We will immediately make use of this proposition in our definition of *novelty*. We will say a state is novel, with respect to some hierarchical Turing machine H_k, if it is in the states of H_k, but not in the recursive closure of H_k.

Definition 3. Let $H = (H_k)_{k \in S}$ be a hierarchical Turing machine, and let X_k denote the input to H_k for each $k \in S$. A state s is said to be *novel* at level k, denoted $\mathrm{NovelState}_k$, if and only if

$$\mathrm{NovelState}_k(H, s) \coloneqq s \in \mathrm{St}(H_k, X_k) \wedge s \notin \mathrm{RC}_{k-1}(H, X)$$

In other words, no computable function exists that could possible reach s given any previous hierarchy's states.

Using Proposition 1, we may rewrite our definition of NovelState_k as being

$$\text{NovelState}_k(H, s) := s \in \text{St}(H_k, X_k) \wedge \forall i < k, s \notin \text{RC}_i(H, X)$$

We will opt to prefer this definition throughout this paper. The reason for this is that we will eventually discuss novelty in context to hierarchical formal systems, and our definition of novelty in such a logic requires us to explicitly use the universal quantifier. By defining novelty in the above fashion, there will be a symmetry to our definitions. It may also be useful, at least notationally, to define the set of all novel states with respect to a given Turing machine H_k in a hierarchical Turing machine H.

Definition 4. The set of *novel states* at level k, denoted $\text{NS}_k(H)$, is defined as

$$\text{NS}_k(H) := \{s \in \text{St}(H_k, X_k) \mid \text{NovelState}_k(H, s)\}$$

To aid our organization, we are using abbreviations to denote sets, and fully spelled out, Pascal-case identifiers to denote things other than sets, such as logical predicates. Regardless, we will now define the most important concept in our paper, being the notion of persistent novelty. A hierarchical Turing machine H is said to exhibit persistent novelty if there exists a novel state at each Turing machine H_k in its hierarchy.

Definition 5. A hierarchical Turing machine $H = (H_k)_{k \in S}$ is said to exhibit *persistent novelty* on input X if

$$\forall k \in S, \exists j > k, \exists s, \text{NovelState}_j(H, s)$$

If we expand the $\text{NovelState}_b(H, s)$ in the above definition, we obtain the following statement.

$$\forall k \in S, \exists j > k, \exists s \in \text{St}(H_k, X_k), \forall i < k, s \notin \text{RC}_i(H, X)$$

This may, in turn, be rewritten as

$$\forall k \in S, \exists j > k, \forall i < k, \exists s, s \in \text{St}(H_k, X_k), s \notin \text{RC}_i(H, X)$$

The above writing is more typical for logical predicates, as the universal quantifiers and existential quantifiers have been interleaved appropriately. We will opt to prefer the above form when we are discussing persistent novelty.

All of our definitions of persistent novelty are ill-formed in context to normal Turing machines. This is because the input of a Turing machine is represented in its initial configuration. As a result, if we take all inputs X_k to H_k and coalesce them into a single input X^* of a Turing machine that corresponds to H, then, clearly, $s \in H_k$ implies $s \in \text{RC}_0(H, X)$, and, thus, $s \in \text{RC}_i(H, X)$ for each $i < k$. This defeats our definition of persistent novelty. We could salvage this by defining persistent novelty more carefully for normal Turing machines, but, for the purposes of this paper, it is simpler discuss hierarchical Turing machines.

Whatever the case, we have now established all the prerequisite terminology relating to hierarchical Turing machines. The purpose of this paper will be to prove, in general, that hierarchical Turing machines cannot decide the novelty of any other hierarchical Turing machine, including itself, before using this result to discuss the undecidability of membership in AI. We can derive this undecidability result easily, by applying a proof similar to the proof used Rice's theorem [11]. While we provide this proof here, it does not encompass the goal of this paper.

Theorem 2. *There exists no hierarchical Turing machine H such that $H_k(G)$ decides*

$$\forall a \in S, \exists b > a, \exists s, \mathit{NovelState}_b(H, s)$$

for all hierarchical Turing machines G.

Proof. Let $H \coloneqq (H_k)_{k \in S}$ be hierarchical Turing machine, and let X_k refer to the input of H_k, where, for each $k \in S$ such that $k > \min S$

$$X_k \coloneqq \Big(\mathrm{Output}\big(H_{k-1}(X_{k-1})\big),\, X'_k\Big)$$

Suppose that, for any hierarchical Turing machine G, $H(G)$ halts at level k and and outputs

$$\begin{cases} 1 & \text{if } G \text{ exhibits persistent novelty} \\ 0 & \text{otherwise} \end{cases}$$

That is to say, $H_k(X_k, G_k)$ halts. Let D be the hierarchical Turing machine $D = (D_j)_{j \in T}$ such that the behavior of each D_j depends on the computation $H(D)$, and let Y_k refer to the input of D_k, where, for each $k \in \mathrm{S}$ such that $k > \min S$

$$Y_k \coloneqq \Big(\mathrm{Output}\big(D_{k-1}(Y_{k-1})\big),\, Y'_k\Big)$$

Now, suppose at level k $H(D)$ halts. That is to say, $H_k(X_k, Y_k)$ halts. Then, define the first for each $i \leq k$, define X'_i as D_i, and D_k as $D_i(Y_i) = H_i(X_k)$. So, each D_i takes the form

$$D_i(Y_i) = H_i\Big(\mathrm{Output}\big(H_{k-1}(X_{k-1})\big), D_{i-1}(Y_{i-1})\Big)$$

In other words, across its first k Turing machines, D simulates H's computation on D. Next, suppose there does not exist any $\mathcal{O}$ that is present in the recursive closure of D_k. This amounts to saying that the recursive closure of D_k is precisely equivalent to the codomain of partial recursive functions. This means determining the lack of persistent novelty of D_k is equivalent to determining it simulates all partial recursive functions. Clearly, determining this is undecidable [5]. So, our goal follows immediately in this case. Likewise, we may assume there exists some $\mathcal{O}$ that is *not* the evaluation of some function in the recursive closure of D_k. Formally, suppose there exists an $\mathcal{O}$ in the set

$$\mathcal{O} \in \big\{Y \mid Y \notin \mathrm{RC}_k(D, X')\big\}$$

With this in mind, we will define the remaining levels of D as follows. The level D_{k+1} takes the output of D_k as input. If D_k outputted 1, then D_{k+1} immediately halts without entering any novel state. If D_k outputted 0, then D_{k+1} runs Turing machine D_{k+2}, which takes a Turing machine state as input, and advances to that state. There are two cases to consider, corresponding to the output of $H_k(D)$.

Case 1: $H_k(D) = 1$

In this case, D_{k+1} halts with no novel output at all. However, this means D does not exhibit persistent novelty, which contradicts the fact H_k returned that D exhibits persistent novelty.

Case 2: $H_k(D) = 0$

In this case, D_{k+1} runs Turing machine D_{k+2}. Seeing as we have assumed $\mathcal{O}$ exists, there exists some D_{k+2} that runs on input $\mathcal{O}$. But $\mathcal{O}$ is novel by definition, so it follows D_{k+2} is able to reach a novel state. However, this contradicts with the fact that H_k returned that D did not exhibit persistent novelty.

In either case, we arrive at a contradiction. This implies that H cannot possibly exist. □

The above proof does, indeed, imply that the problem of determining (via a hierarchical Turing machine) whether some other hierarchical Turing machine exhibits persistent novelty is, generally, undecidable. However, to reiterate, we have arrived at this result through diagonalization, in a manner similar to that of Rice's theorem [11]. Likewise, the usual criticisms of this approach apply. Namely, this approach does not allow us to formulate any broad conditions as to a hierarchical Turing machine's ability to determine persistent novelty, nor does it allow us to assert whether a hierarchical Turing machine can determine its own capacity for persistent novelty. In summary, if we limit ourselves to looking only at those hierarchical Turing machines that do not exhibit a diagonal construction, then we may still be able to determine persistent novelty.

In order to address these criticisms, we will begin by looking at a stronger, more permissive proof. Our proof will begin by defining what is meant by a hierarchical formal system, before showing that all hierarchical Turing machines correspond to these hierarchical theories. We will then show that deciding whether a broad class of hierarchical Turing machines exhibits persistent novelty allows a hierarchical Turing machine to recognize a theorem that is fundamentally unprovable in these hierarchical formal systems.

3 Hierarchical Formal Systems

A hierarchical Turing machine is a sequence of Turing machines. Likewise, a hierarchical formal system is a sequence of formal systems. We could treat this hierarchy as a Gödel-Löb-Polymodal (GLP) logic, but it is unnecessary for our purposes.

Definition 6. Let $T = (T_k)_{k \in S}$ be a sequence indexed by a linearly ordered subset $S \subseteq \mathbb{N}$, where each T_k is a recursively enumerable theory. We say that T is a *hierarchical formal system* if and only if, for each $k \in S$, $T_k \subseteq T_{k+1}$.

In hierarchical Turing machines, a state was novel if no total recursive function equivalent to some convolution of previous Turing machines could access it given the states of prior Turing machines in the hierarchy as input. For hierarchical formal systems, a theorem is *novel* if it could not be proved in any prior theory of the hierarchy. Formally, this is defined as follows.

Definition 7. Let $T = (T_k)_{k \in S}$ be a hierarchical formal system such that $T_{\min S} \supseteq \mathsf{Q}$[1]. A sentence φ is said to be *novel* at level k, denoted $\mathrm{Novel}_k(T, \varphi)$, if and only if

$$\mathrm{Novel}_k(T, \varphi) := \mathrm{Pr}_{T_k}(\varphi) \wedge \forall j < k, \neg \mathrm{Pr}_{T_j}(\varphi)$$

Here, $\mathrm{Pr}_{T_k}(\varphi)$ denotes the standard Gödelian provability predicate expressing that φ is provable in T_k.

[1] Q denotes the formal system of Robinson arithmetic.

As in hierarchical Turing machines, *persistent novelty* simply refers to whether novelty persists across all theories in the hierarchy. That is, whether each theory can express a novel sentence.

Definition 8. Let $T = (T_k)_{k \in S}$ be a hierarchical formal system such that $T_{\min S} \supseteq \mathsf{Q}$. We say that T exhibits *persistent novelty* if and only if

$$\forall k \in S,\ \exists \varphi,\ \mathrm{Novel}_k(T, \varphi)$$

It can be shown that any hierarchical formal system T is unable to determine its own persistent novelty at any level T_k. The reasoning behind this is simple. If it could determine its persistent novelty at some theory T_k, then it could determine that there are theorems which are true in T_k, but which are unprovable in T_k. In other words, T_k could prove its incompleteness. This does not contradict Gödel's incompleteness theorem directly, but it does contradict an entailment of Gödel's incompleteness theorem as originally expressed by Lindström. Before stating this proof, it is important to note that we will be using a modern day interpretation of Lindstöm's result, as given by Hájek and Pudlák [8], as opposed to using, verbatim, the original form of Lindström's result [10].

Theorem 3. *Let $T = (T_k)_{k \in S}$ be a hierarchical formal system such that $T_{\min S} \supseteq \mathsf{Q}$. If Q is consistent then, for any $a \in S$,*

$$T_a \nvdash \forall b > a,\ \exists \varphi,\ \mathit{Novel}_b(T, \varphi)$$

unless T_a proves the uniform reflection schema *for all $b > a$, i.e.,*

$$\forall b > a,\ \forall \varphi,\ \mathit{Pr}_{T_b}(\varphi) \to \varphi$$

Proof. There are two cases to consider, depending on whether T_a proves the uniform reflection schema for its successors.

Case 1: $T_a \nvdash \forall x\big(\mathit{Pr}_{T_b}(\varphi) \to \varphi\big)$ *for each $b > a$*

That is to say, T_a does not contain the uniform reflection schema. Let us start by assuming

$$T_a \vdash \forall b > a, \exists \varphi, \forall c < b, \mathrm{Pr}_{T_b}(\varphi) \wedge \neg \mathrm{Pr}_{T_c}(\varphi)$$

Seeing as $a < b$, we may infer that

$$T_a \vdash \forall b > a, \exists \varphi, \mathrm{Pr}_{T_b}(\varphi) \wedge \neg \mathrm{Pr}_{T_a}(\varphi)$$

Then, from within T_a, for each $b > a$, there exists some sentence x such that $T_a \vdash \mathrm{Pr}_{T_b}(\ulcorner x \urcorner)$, and $T_a \vdash \neg \mathrm{Pr}_{T_a}(\ulcorner x \urcorner)$. Seeing as T_a proves $\ulcorner x \urcorner$ is not provable, and T_a does not possess a uniform reflection schema, we know $T_a \nvdash x$. Informally, we may consider T_a as being capable of asserting "No matter how far I (T_a) am extended, there is always some sentence (x) provable in an extension (T_b) of myself that I cannot prove". In other words, we may consider T_a as being capable of proving its own incompleteness with respect to future extensions.

However, Lindström has shown [10] that if a theory $F \supseteq \mathsf{Q}$ proves that some true sentence is unprovable in F, then $F \vdash \mathrm{Con}(F)$. Our definition of T_n implies it is at least as powerful as Q. We have also already assumed $T_a \vdash \neg \mathrm{Pr}_{T_a}(\ulcorner x \urcorner)$. Finally, our definition of T_n implies that $T_a \subseteq T_b$. Because T_b is a consistent extension of T_a, it follows that $\ulcorner x \urcorner$ is true in T_a.

In other words, T_a proves that some true $\ulcorner x \urcorner$ is unprovable in T_a. We may thus conclude $T_a \vdash \mathrm{Con}(T_a)$. This violates Gödel's second incompleteness theorem, providing we accept the consistency of Q.

Case 2: $T_a \vdash \forall \varphi, Pr_{T_b}(\varphi) \to \varphi$ *for each* $b > a$

In our statement of this theorem, we have already assumed T_a can represent the proof predicates of T_b. Consequentially, the fixed point theorem states that for each $b > a$, there exists a sentence φ_b such that

$$T_a \vdash \varphi_b \leftrightarrow \big(\mathrm{Pr}_{T_b}(\varphi_b) \wedge \neg \mathrm{Pr}_{T_a}(\varphi_b)\big)$$

Correspondingly, we have assumed the existence of a uniform reflection schema $T_a \vdash \forall \varphi, \mathrm{Pr}_{T_b}(\varphi) \to \varphi$ for each $b > a$. Thus, for each $b > a$, we have

$$T_a \vdash \mathrm{Pr}_{T_b}(\varphi_b) \to \varphi_b$$

Clearly, $T_a \vdash \varphi_b$ implies $T_a \vdash \mathrm{Pr}_{T_b}(\varphi_b)$. Seeing as $a < b$, we may conclude, via our uniform reflection schema, that $T_a \vdash \varphi_b$. Now, from our fixed point definition of φ_b, it trivially follows, for each $b > a$, that

$$T_a \vdash \mathrm{Pr}_{T_b}(\varphi_b) \wedge \neg \mathrm{Pr}_{T_a}(\varphi_b)$$

So, we may conclude

$$T_a \vdash \forall b > a, \exists \varphi_b, \mathrm{Pr}_{T_b}(\varphi_b) \wedge \neg \mathrm{Pr}_{T_a}(\varphi_b)$$

□

Similarly, we can prove the converse of the above statement. This proof of this converse statement is trivial, and is arrived at immediately through the application of Gödel's incompleteness theorems.

Theorem 4. *Let* $T = (T_k)_{k \in S}$ *be a hierarchical formal system such that* $T_{\min S} \supseteq \mathsf{Q}$*. Then, for any* $a \in S$*,*

$$T_a \nvdash \neg\big(\forall b > a,\ \exists \varphi,\ \mathit{Novel}_b(T, \varphi)\big)$$

Proof. Let us start by assuming

$$\begin{aligned} T_a &\vdash \neg\big(\forall b > a, \exists \varphi, \forall c < b, \mathrm{Pr}_{T_b}(\varphi) \wedge \neg \mathrm{Pr}_{T_c}(\varphi)\big) \\ &\vdash \exists b \leq a, \forall \varphi, \exists c \geq b, \neg \mathrm{Pr}_{T_b}(\varphi) \vee \mathrm{Pr}_{T_c}(\varphi) \end{aligned}$$

There are two cases to consider. The first case is that there exists a $b \leq a$, such that, for all φ, T_a proves T_b cannot prove φ. The second is that, for some $c \geq b$, T_a proves T_c can prove φ.

Case 1: $T_a \vdash \exists b \leq a, \forall \varphi, \neg \mathrm{Pr}_{T_b}(\varphi)$

We have been given that, for each $b \leq a$, T_b extends Q. Obviously, there are sentences in Q that are provable in Q, and so we know there are sentences in each T_b that are provable in T_b. This contradicts the given statement that each T_b is unable to prove any sentence.

Case 2: $T_a \vdash \exists b \leq a, \forall \varphi, \exists c \geq b, \mathrm{Pr}_{T_c}(\varphi)$

This means that, for some $b \leq a$, and $c \geq b$, all sentences are provable by T_c. By fixed point theorem, this means the fixed point $\varphi_c \leftrightarrow \neg \mathrm{Pr}_{T_c}(\varphi)$ is also provable by T_c. Yet, this implies the sentence $\mathrm{Pr}_c(\varphi_c) \leftrightarrow \neg \mathrm{Pr}_{T_c}(\varphi)$ is provable in T_c, which violates the consistency of T_c. □

We have now proved that any theory in a hierarchical formal system is unable to prove its persistent novelty, nor is it able to disprove its persistent novelty. This means that persistent novelty is undecidable with respect to hierarchical formal systems.

The above proofs only state how T is unable to decide its own persistent novelty at any level T_k. It does not directly state as to whether T_k can prove the persistent novelty of some theory S_k that is present in an external system S. However, if we are given the condition that $S_k \subseteq T_k$ for each k, then the external result itself becomes an intensional result, and Theorem 3 applies readily.

Theorem 5. *Let $U = (U_i)_{i \in S}$ and $V = (V_j)_{j \in T}$ be hierarchical formal systems such that $U_{\min S} \supseteq \mathsf{Q}$ and $V_{\min T} \supseteq \mathsf{Q}$. Suppose that at least one of the following holds:*

1. *$S \subseteq T$, and, for each $i \in S$, $U_i \subseteq V_i$*
2. *There exist $i \in S$ and $j \in T$ such that U_i cannot represent $Pr_{V_j}(x)$* [2]
3. *For each $k \in T$, there exists some j, and $i < j$ such that*

$$V_k \vdash \forall j > i,\ \forall \varphi,\ Pr_{V_j}(\varphi) \leftrightarrow Pr_{V_i}(\varphi)$$

Then, for any $i \in S$,

$$U_i \nvdash \forall j > i,\ \exists \varphi_j,\ \forall k < j,\ Pr_{V_j}(\varphi_j) \wedge \neg Pr_{V_k}(\varphi_j)$$

Proof. To prove our result, is suffices to prove three cases. Each of these cases corresponds to one of the conditions specified in our theorem statement.

Case 1: $S \subseteq T$, and, for each $i \in S$, $U_i \subseteq V_i$

Let us start by assuming that

$$U_i \vdash \forall j > i,\ \exists \varphi_j,\ \forall k < j,\ \mathrm{Pr}_{V_j}(\varphi_j) \wedge \neg \mathrm{Pr}_{V_k}(\varphi_j)$$

However, because $U_i \subseteq V_i$, for each i, this implies

$$V_i \vdash \forall j > i,\ \exists \varphi_j,\ \forall k < j,\ \mathrm{Pr}_{V_j}(\varphi_j) \wedge \neg \mathrm{Pr}_{V_k}(\varphi_j)$$

which contradicts theorem 3.

Case 2: There exist $i \in S$ and $j \in T$ such that U_i cannot represent $\mathrm{Pr}_{V_j}(\varphi)$

Since U_i is not expressive enough to define $Pr_{V_j}(\varphi)$, it cannot possibly prove a statement involving $Pr_{V_j}(\varphi)$, regardless of whether the statement is true in some meta-theory of U_i. Thus, this case is trivial.

[2] U_i admits a formula $Pr_{V\,j}(x)$ satisfying the standard derivability conditions for V_i's proof predicate.

Case 3: For each $k \in T$, there exists some j, and $i < j$ such that

$$V_k \vdash \forall j > i,\ \forall \varphi,\ \mathrm{Pr}_{V_j}(\varphi) \leftrightarrow \mathrm{Pr}_{V_i}(\varphi)$$

As in the other cases, let us assume, for some k

$$U_k \vdash \forall j > k,\ \exists \varphi_j,\ \forall i < j,\ \mathrm{Pr}_{V_j}(\varphi_j) \wedge \neg \mathrm{Pr}_{V_i}(\varphi_j)$$

We have been given that there for each $k \in T$, there exists a j, and $i < j$, such that $V_k \vdash \forall j > i,\ \forall \varphi,\ \mathrm{Pr}_{V_j}(\varphi) \leftrightarrow \mathrm{Pr}_{V_i}(\varphi)$. Let j' and i' be the witness to our j and i, respectively. This allows us to rewrite our assumption as

$$\begin{aligned} U_k &\vdash \exists \varphi_j,\ \mathrm{Pr}_{V_{j'}}(\varphi_j) \wedge \neg \mathrm{Pr}_{V_{i'}}(\varphi_j) \\ &\vdash \exists \varphi_j,\ \mathrm{Pr}_{V_{j'}}(\varphi_j) \wedge \neg \mathrm{Pr}_{V_{j'}}(\varphi_j) \end{aligned}$$

which is contradictory, and violates the consistency of U_k. □

As before, we will also prove the above theorem in the converse direction. This proof follows the same logic as our above proof.

Theorem 6. *Let $U = (U_i)_{i \in S}$ and $V = (V_j)_{j \in T}$ be hierarchical formal systems such that $U_{\min S} \supseteq Q$ and $V_{\min T} \supseteq Q$. Suppose that at least one of the following holds:*

1. *$S \subseteq T$, and, for each $i \in S$, $U_i \subseteq V_i$*
2. *There exist $i \in S$ and $j \in T$ such that U_i cannot represent $Pr_{V_j}(x)$* [3]

Then, for any $i \in S$,

$$U_i \nvdash \neg\big(\forall j > i,\ \exists \varphi_j,\ \forall k < j,\ Pr_{V_j}(\varphi_j) \wedge \neg Pr_{V_k}(\varphi_j)\big)$$

Proof. To prove our result, is suffices to prove two cases. Each of these cases corresponds to one of the conditions specified in our theorem statement.

Case 1: $S \subseteq T$, and, for each $i \in S$, $U_i \subseteq V_i$

Let us start by assuming that

$$U_i \vdash \neg\big(\forall j > i,\ \exists \varphi_j,\ \forall k < j,\ \mathrm{Pr}_{V_j}(\varphi_j) \wedge \neg \mathrm{Pr}_{V_k}(\varphi_j)\big)$$

However, because $U_i \subseteq V_i$, for each i, this implies

$$V_i \vdash \neg\big(\forall j > i,\ \exists \varphi_j,\ \forall k < j,\ \mathrm{Pr}_{V_j}(\varphi_j) \wedge \neg \mathrm{Pr}_{V_k}(\varphi_j)\big)$$

which contradicts theorem 4.

Case 2: There exist $i \in S$ and $j \in T$ such that U_i cannot represent $\mathrm{Pr}_{V_j}(\varphi)$

Since U_i is not expressive enough to define $Pr_{V_j}(\varphi)$, it cannot possibly prove a statement involving $Pr_{V_j}(\varphi)$, regardless of whether the statement is true in some meta-theory of U_i. Thus, this case is trivial. □

[3] U_i admits a formula $Pr_{V\,j}(x)$ satisfying the standard derivability conditions for V_i's proof predicate.

We will now discuss a correspondence between hierarchical Turing machines, and hierarchical formal systems. Such correspondences have been discussed before, namely by Feferman [6]. Our correspondence will be similar to that of Feferman's, though not entirely identical. We will define the set of arithmetic trace formulas for a given theorem in some hierarchical formal system. These formulas will be the Gödel numbers of theorems asserting that certain states are members of the hierarchy, as well as the fact that certain states are novel in that level of the hierarchy.

Definition 9. Let $H = (H_k)_{k \in S}$ be a hierarchical Turing machine, and let X_k be the input to H_k. Additionally, define the following arithmetized predicates

- $\text{State}_k(H, s)$, encoding that machine H_k entered state s
- $\text{NovelState}_k(H, s)$, encoding that s is a novel state in machine H_k, as in definition 3

Then, the *arithmetic trace formulas* of H_k on input X_k, denoted $\text{Tr}_k(H, X)$, is the finite set of Godel numbers

$$\text{Tr}_k(H, X) \quad := \quad \begin{matrix} \left\{ \ulcorner \text{State}_k(H, s) \urcorner \mid s \in St(H_k, X_k) \right\} & \cup \\ \left\{ \ulcorner \text{NovelState}_k(H, s) \urcorner \mid s \in \text{NS}(H) \right\} \end{matrix}$$

We must note here that the $\text{NovelState}_k(H, s)$ predicate expresses that $s \notin \text{RC}_i$ for each $i < k$. In general, enumerating RC_i requires enumerating a subset of the partial recursive functions that is composed of non-total recursive functions, which means the predicate is not necessarily, itself, enumerable [12]. As a result, we do not expect PA [4] to be able to represent this in the general case. We also do not expect any stage of the hierarchy to be able to represent this fact without explicitly including it as a reflection principle. Actually, it is the very fact that this cannot generally be represented in PA, or any theory in the hierarchy, that will eventually enable us to prove a much more general form of Theorem 2. Nevertheless, none of this implies the predicate cannot be encoded using a Gödel numbering scheme. It merely implies that it cannot be derived. Gödel numbering for a sentence is independent of whether the set it belongs to is recursively enumerable [1].

With that clarified, we can now define what is meant by the *arithmetic trace theory* for a given hierarchical Turing machine. In general, this trace theory is composed of the axioms of PA, the code of our Turing machine H_k, and the arithmetic trace formulas of H_k.

Definition 10. Let $H = (H_k)_{k \in S}$ be a hierarchical Turing machine, and let X_k be the input to H_k. Then, the *arithmetic trace theory* of H_k on input X_k, denoted $\text{Tr}^*_k(H, X)$, is the finite set of Godel numbers

$$\text{Tr}^*_k(H, X) := \begin{cases} \mathsf{PA} + \text{Tr}^*_{\text{pred}\, k}(n) + \text{Tr}_k(H, X) & \text{if } k > \min \text{ S} \\ \mathsf{PA} + \text{Tr}_{\min S}(H, X) & \text{otherwise} \end{cases}$$

Finally, we will define the *arithmetic trace hierarchy* of a hierarchical Turing machine to be the sequence of each Turing machine's respective arithmetic trace theory.

Definition 11. Let $H = (H_k)_{k \in S}$ be a hierarchical Turing machine. Define the *arithmetic trace hierarchy* of H to be the sequence $A := (A_k)_{k \in S}$, where each $A_k \equiv \text{Tr}^*_k(H, X)$.

[4] PA denotes the formal system of Peano arithmetic.

It is possible to relate the persistent novelty of any hierarchical Turing machines $H := (H_k)_{k\in S}$ to the persistent novelty of its corresponding arithmetic trace hierarchy A. In fact, under the condition that $s \notin \mathrm{RC}_k(A, X)$ is undecidable across previous formal systems in the hierarchy (being undecidable in A_i, for any $i < k$) then persistent novelty in H immediately implies persistent novelty in A.

Lemma 1. *Let $H = (H_k)_{k\in S}$ be a hierarchical Turing machine, and let $A := (A_k)_{k\in S}$ be the arithmetic trace theory of H. If the following are true*

- *H exhibits persistent novelty*
- *For each $k \in S$, and for all $s \notin St(A_k, X_k)$, $A_k \nvdash s \notin RC_k(H, X)$*

Then, A exhibits persistent novelty.

Proof. From our definition of persistent novelty for hierarchical formal systems, we must prove

$$\forall a \in S, \exists b > a, \exists \varphi, A_b \vdash \varphi \wedge \forall c < b, A_c \nvdash \varphi$$

We have been given that H exhibits persistent novelty. Then, for each H_b, it follows there exists some s, such that s is novel at H_b. That is to say,

$$\forall a \in S, \exists b > a, \exists s, \forall c < b, s \in \mathrm{St}(H_b, X_b) \wedge s \notin \mathrm{RC}_c(H, X)$$

Let s_b be the witness of this s at H_b. A_b has been defined as including $\mathrm{Tr}_b(H, X)$, which, in turn, has been defined as including the Gödel number of all sentences of the form $\mathrm{NovelState}_b(H, s)$. This predicate, itself, has been defined as

$$\mathrm{NovelState}_b(H, s) := \forall c < b, s_b \in \mathrm{St}(H_b, X_b) \wedge s_b \notin \mathrm{RC}_c(H, X)$$

So, it follows, for each $c < b$, the Gödel number of $s_b \notin \mathrm{RC}_c(H, X)$ must be an entailment in A_b. Thus, for each $c < b$

$$A_b \vdash \ulcorner s_b \notin \mathrm{RC}_c(H, X) \urcorner$$

However, we have been given the condition that for each $k \in S$, and for all $s \notin \mathrm{St}(A_k, X_k)$, $A_k \nvdash s \notin \mathrm{RC}_k(A, X)$. Consequentially, we may also conclude

$$A_c \nvdash \ulcorner s_b \notin \mathrm{RC}_c(H, X) \urcorner$$

Let $\upsilon_b(s) := \ulcorner s_b \notin \mathrm{RC}_c(H, X)$. It is clear we have shown $A_b \vdash \upsilon_b(s)$, but $A_c \nvdash \upsilon_b(s)$ for all $c < b$. In summary, $\varphi_b(s)$ is provable in A_b, but unprovable in all A_c, when $c < b$. Thus, φ_b is a novel sentence at level b of the trace hierarchy. Since a was arbitrary, this establishes persistent novelty in $(A_k)_{k\in S}$. □

We have shown the above result generally. However, we have not proved the result within the arithmetic trace theory A itself. In fact, due to the general undecidability of $s \notin \mathrm{RC}_i(H, X)$, it is, perhaps, unreasonable to assume $A_k \vdash \neg \mathrm{Pr}_{A_i}\left(\ulcorner s \notin \mathrm{RC}_i(H, X) \urcorner\right)$ for any k, and $i < k$. This is all to say, it should not be expected that the above result can be encoded in the language of A_k without additional reflection principles. We will now show that, were A to possess a particular reflection schema for each theory A_k, then each A_k could prove its own persistent novelty.

Lemma 2. *Let $H = (H_k)_{k\in S}$ be a hierarchical Turing machine that exhibits persistent novelty, let $A \coloneqq (A_k)_{k\in S}$ be the arithmetic trace hierarchy of H, and let $F \coloneqq (F_k)_{k\in S}$ be a hierarchical formal system. If for each $k \in S$, F_k satisfies*

1. *$A_k \subseteq F_k$*
2. *For each $i < k$, and for all $s \in St(H_k, X_k)$, $F_i \nvdash \ulcorner s \notin RC_i(H, X)\urcorner$*
3. *$F_k \vdash \ulcorner \forall j > k, \exists s, \forall i < j, s \in St(H_j, X_j) \wedge s \notin RC_i(H, X)\urcorner$*
4. *For each $j > k$, there exists some s such that, for each $i < k$, F_k contains the reflection schema*

$$s \in St(H_k, X_k) \wedge s \notin RC_i(H, X) \rightarrow Pr_{F_j}(\ulcorner s \notin RC_i(H, X)\urcorner) \wedge \neg Pr_{F_i}(\ulcorner s \notin RC_i(H, X)\urcorner)$$

Then, it follows

$$F_k \vdash \ulcorner \forall j > k,\ \exists \varphi,\ \forall i < j,\ Pr_{T_j}(\varphi) \wedge \neg Pr_{T_i}(\varphi)\urcorner$$

Proof. It has been given that, for each $k \in S$, F_k proves persistent novelty starting at H_k. That is

$$F_k \vdash \ulcorner \forall j > k, \exists s, \forall i < j, s \in \mathrm{St}(H_j, X_j) \wedge s \notin \mathrm{RC}_i(H, X)\urcorner$$

However, our reflection principle states, for all $j > k$, there exists some s such that, for all $i < j$, the following holds

$$s \in \mathrm{St}(H_j, X_j) \wedge s \notin \mathrm{RC}_i(H, X) \quad \rightarrow \quad \mathrm{Pr}_{F_j}(\ulcorner s \notin \mathrm{RC}_i(H, X)\urcorner) \wedge \neg \mathrm{Pr}_{F_i}(\ulcorner s \notin \mathrm{RC}_i(H, X)\urcorner)$$

So, we may infer

$$F_k \vdash \ulcorner \forall j > k, \exists s, \forall i < j, \mathrm{Pr}_{F_j}\big(\ulcorner s \notin \mathrm{RC}_i(H, X)\urcorner\big) \wedge \neg \mathrm{Pr}_{F_i}\big(\ulcorner s \notin \mathrm{RC}_i(H, X)\urcorner\big)\urcorner$$

Let $\upsilon_j(s) \coloneqq \ulcorner s \notin \mathrm{RC}_i(H, X)\urcorner$. Then, it follows

$$\begin{aligned} F_k \quad &\vdash \quad \ulcorner \forall j > k, \exists s, \forall i < j, \mathrm{Pr}_{F_j}\big(\upsilon_j(s)\big) \wedge \neg \mathrm{Pr}_{F_i}\big(\upsilon_j(s)\big)\urcorner \\ F_k \quad &\vdash \quad \ulcorner \forall j > k, \exists \upsilon, \exists s, \forall i < j, \mathrm{Pr}_{F_j}\big(\upsilon(s)\big) \wedge \neg \mathrm{Pr}_{F_i}\big(\upsilon(s)\big)\urcorner \\ F_k \quad &\vdash \quad \ulcorner \forall j > k, \exists \varphi, \forall i < j, \mathrm{Pr}_{F_j}(\varphi) \wedge \neg \mathrm{Pr}_{F_i}(\varphi)\urcorner \end{aligned}$$

Thus, we have derived our goal. □

We may now prove Theorem 2 in a much broader way. The intuition behind our proof for this theorem is that, in general, $s \notin \mathrm{RC}_k(H, X)$ is undecidable, as it requires deciding membership in an infinite set that is the union of the codomains of certain partial recursive functions. Obviously, this is not guaranteed to be decidable by Theorem 1. Yet, a hierarchical Turing machine deciding the persistent novelty of some other Turing machine implicitly involves deciding $s \notin \mathrm{RC}_k(H, X)$ computationally. Thus, if this were to be true, then we could enumerate a theorem that is not enumerable. Our proof will look slightly different from our intuition, and will involve showing that a persistent novelty deciding Turing machine entails a hierarchical formal system exists which can prove its own persistent novelty. This fact cannot ever be possible, as it violates Gödel's second incompleteness theorem, per the proofs of Theorems 3, and 6.

Theorem 7. *If PA is consistent, then there exists no hierarchical Turing machine H such that $H_k(G)$ decides*

$$\forall a \in S,\ \exists b > a,\ \exists s,\ \mathit{NovelState}_b(G, s)$$

for any hierarchical Turing machine $G \coloneqq (G_k)_{k\in T}$ with arithmetic trace $A \coloneqq (A_k)_{k\in T}$, such that for all $i < k$, and for all $s \in St(H_k, X_k)$, $A_i \nvdash s \notin RC_i(H, X)$.

Proof. Let us assume there does exist a hierarchical Turing machine $H \coloneqq (H_k)_{k\in S}$ such that $H_k(G)$, for any hierarchical Turing machine $G \coloneqq (G_k)_{k\in T}$, outputs 1 when the encoding of the following statement is true in the standard model

$$\forall b > a, \exists s, \forall c < b, s \in \mathrm{St}(G_b, X_b) \wedge s \notin \mathrm{RC}_c(G, X)$$

for all $a \in T$. Now, let us consider a hierarchical formal system $F \coloneqq (F_k)_{k\in T}$ such that each F_a encodes the states of every H_k, for each $k \in T$, as well as the arithmetic trace theory $\mathrm{Tr}^*_a(G, X)$ of G_a. Recall that $\mathrm{Tr}^*_a(G, X)$ is the union of PA, the arithmetic trace theories of Turing machines preceding G_a, and the arithmetic trace formulas $\mathrm{Tr}_a(G, X)$ of G_a

$$\mathrm{Tr}_a(G, X) \quad = \quad \begin{array}{r} \{\ulcorner \mathrm{State}_a(H, s)\urcorner \mid s \in \mathrm{St}(G_a, X_a)\} \quad \cup \\ \{\ulcorner \mathrm{NovelState}_a(G, s)\urcorner \mid \forall c < a, s \in \mathrm{St}(H_a, X_a) \wedge s \notin \mathrm{RC}_c(G, s)\} \end{array}$$

It should be noted that $\mathrm{Tr}^*_a(G, X) \subseteq F_a$. Additionally, let $\mathrm{Output}_1(H, G_a)$ be a predicate encoding that the output state of H ran on input G_a corresponds to 1. Then, let F_a encode the following axiom

$$\mathrm{Output}_1(H, G_a) \to \forall b > a, \exists s, \forall c < b, s \in \mathrm{St}(G_b, X_b) \wedge s \notin \mathrm{RC}_c(G, X)$$

We have assumed H does correctly decide the persistent novelty of G. More specifically, we have assumed H only outputs 1 if the encoding of the above statement's consequent is true in the standard model. Thus, it follows that the above axiom is consistent in F_a. Of course, we cannot claim the proof that H correctly decides persistent novelty may be carried out in F_a, which is why we declared it as a separate axiom. Finally, we will also say for each $b > a$, F_a includes the reflection schema for some s, and for each $c < b$

$$s \in \mathrm{St}(H_a, X_a) \wedge s \notin \mathrm{RC}_c(H, X) \to \mathrm{Pr}_{F_b}(\ulcorner s \notin \mathrm{RC}_c(H, X)\urcorner) \wedge \neg \mathrm{Pr}_{F_c}(\ulcorner s \notin \mathrm{RC}_c(H, X)\urcorner)$$

We have restricted ourselves to forms of G, where, for all $c < b$, and for all $s \in \mathrm{St}(H_b, X_b)$, $A_c \nvdash s \notin \mathrm{RC}_c(H, X)$. Likewise, the above reflection schema is consistent in the arithmetic trace hierarchy of G, and, seeing as F is an augmentation of the arithmetic trace hierarchy of G, it is also consistent in F.

However, we have been given that G_a is persistently novel. Seeing as F encodes the states of each Turing machine in H, it follows trivially that F derives an encoding of $\mathrm{Output}_1(H, G_a)$, and thus

$$F_a \vdash \ulcorner \forall b > a, \exists s, \forall c < b, s \in \mathrm{St}(G_b, X_b) \wedge s \notin \mathrm{RC}_c(G, X)\urcorner$$

But, we have now satisfied the conditions underlying Theorem 2. So, we are able to conclude

$$F_a \vdash \ulcorner \forall b > a,\ \exists \varphi,\ \forall c < b,\ \mathrm{Pr}_{T_b}(\varphi) \wedge \neg \mathrm{Pr}_{T_c}(\varphi)\urcorner$$

for each $a \in T$. But, by Theorem 6, this is not possible. In fact, doing so amounts to proving the consistency of F_a, which we cannot do under Gödel's second incompleteness theorem.

So, assuming the existence of H allows us to prove certain theories in a formal system that is at least as powerful as PA. Notably, it will allow us to prove a sentence asserting the consistency of the formal system itself. This is fundamentally impossible if we accept the consistency of PA, and so our assumption as to the existence of H must be invalid under our given conditions. □

Much like in Theorem 3, this theorem is contingent on the consistency of PA. This is widely accepted [7], but we have stated it as an explicit condition for the sake of rigor.

We may also prove the converse of Theorem 7. Proving this converse is trivial, as decidability of the converse directly implies there exists some k, such that, for a given s, the sentence $s \in \mathrm{RC}_k(G, X)$ is decidable. Unlike Theorem 7, which considers whether there *exists* a state whose membership in $\mathrm{RC}_k(G, X)$ is decidable, this is asking whether the membership of certain states in $\mathrm{RC}_k(G, X)$ is decidable. Seeing as our hierarchal Turing machine is generally defined, this is not possible.

Theorem 8. *There exists no hierarchical Turing machine H such that $H_k(G)$ decides*

$$\neg\big(\forall a \in S,\ \exists b > a,\ \exists s,\ \mathit{NovelState}_b(H, s)\big)$$

for all hierarchical Turing machines $G := (G_k)_{k\in T}$.

Proof. Let us assume there does exist a hierarchical Turing machine $H := (H_k)_{k\in S}$ such that $H_k(G)$, for any hierarchical Turing machine $G := (G_k)_{k\in T}$, outputs 1 when the following statement is true

$$\begin{aligned} \Phi &:= \neg\big(\forall b > a, \exists s, \forall c < b, s \in \mathrm{St}(G_b, X_b) \wedge s \notin \mathrm{RC}_c(G, X)\big) \\ &:= \qquad \exists b \le a, \forall s \in \mathrm{St}(G_b, X_b), \exists c \ge b, s \in \mathrm{RC}_c(G, X) \end{aligned}$$

for all $a \in T$. Determining Φ means asserting there exists some b, such that for all $s \in \mathrm{St}(G_b, X_b)$, there is some $c \ge b$, such that $s \in \mathrm{RC}_c(G, X)$. Alternatively, it means asserting that there is a set of states for which membership in $\mathrm{RC}_c(G, X)$ is decidable. However, deciding $s \in \mathrm{RC}_c(G, X)$ requires deciding the mebership in an infinite set that is the union of the codomains of certain partial recursive functions. We showed in by Theorem 1 that this is not possible. □

We have proved both the forward and converse case, and so it follows that no hierarchical Turing machine may decide whether some other hierarchical Turing machine exhibits persistent novelty. The conditions underlying this result are stronger than the conditions underlying Theorem 2, and so we are now able to assert that a hierarchical Turing machine cannot possibly decide its own persistent novelty.

Corollary 1. *Let $H := (H_k)_{k\in S}$ be a hierarchical Turing machine. If, for all $i < k$, and for all $s \in St(H_k, X_k)$, $H \nvdash s \notin RC_i(H, X)$, then for each $k \in S$, H_k cannot decide*

$$\forall a \in S,\ \exists b > a,\ \exists s,\ \mathit{NovelState}_b(H, s)$$

Proof. This result trivially follows after substituting H for G in Theorems 7 and 8. □

In fact, the only condition as to the applicability of Theorem 7, is the condition that, for all $i < k$, and for all $s \in \mathrm{St}(G_k, G_k)$, $G \nvdash s \notin \mathrm{RC}_i(H, X)$. In effect, being able to decide a hierarchical Turing machine's capacity for persistent novelty, or its "intelligence", requires

being able to decide state membership in the recursive closure of each Turing machine in the hierarchy. This is a stronger conditions than the condition that G corresponds to a diagonal hierarchical Turing machine, as used in Theorem 2.

We will next define the AI problems as the class of problems that are exclusively decidable by hierarchical Turing machines that exhibit persistent novelty. It will follow from Theorem 7 that membership of AI is, generally, undecidable.

4 AI Problems

The AI problems have been defined, loosely, as the problems that require intelligence in order to solve. More formal definitions of the AI problems have been investigated by Yampolsky [18], and involve human oracles. At the start of our paper, we assumed that intelligence entails a capacity for learning. To express this assumption using our current terminology: If a hierarchical Turing machine to exhibits intelligence, it must also exhibit persistent novelty. Therefore, we can treat whether a hierarchical Turing machine demonstrates persistent novelty as a boundary condition for intelligence, and thus membership in AI. We will say that if a problem is in AI, then any hierarchical Turing machine that decides it must have persistent novelty.

Definition 12. If a problem $X : \Sigma^* \to \Sigma^*$ belongs to the class AI then, for all hierarchical Turing machines H such that for all $x \in \Sigma^*$, H halts on input w and correctly outputs $y \in \{0, 1\}$, H has persistent novelty.

Theorem 9. *There exists no hierarchical Turing machine H which, given a description of a decision problem X, determines whether $X \in$ AI.*

Proof. Let us assume that such a machine H exists. Then, we may define a certain decision problem P_G as follows

$$P_G(x) \coloneqq G_k(x)$$

This is simply the problem of computing $G_k(x)$.

$$P_G(x) \coloneqq \text{“Compute } G_k(x)''$$

Consider briefly what happens if we run H on P_G. If H returns “true”, then every hierarchical Turing machine that solves P_G must exhibit novelty. Clearly, we have defined P_G in such a way that G solves P_G. Thus, $H(P_g)$ would be able to determine whether G exhibits persistent novelty. Seeing as G is general, Theorem 7 implies this is not possible. □

If we instead considered the AI problems as being decision problems, instead of functions problems, the above proof would still apply. However, we would restrict ourselves to hierarchical Turing machines G such that G_k defined as some mapping $\Sigma^* \to \{0, 1\}$.

In any case, membership of AI is not decidable. This also means that we cannot decide the reducibility (polynomially or otherwise) of any problem $X \in \mathcal{P}$ to AI. Moreover, we also are unable to decide whether a given hierarchical Turing machine decides whether some other hierarchical Turing machine decides some problem $X \in$ AI.

Theorem 10. *Let X be a decision problem. If $X \in$ AI, then exists no hierarchical Turing machine H that decides whether a given hierarchical Turing machine G decides X.*

Proof. Let us assume there exists a hierarchical Turing machine H that takes, as input, some other hierarchical Turing machine G, and an input string X. H then outputs whether or not G decided X.

$$H(G,X) \coloneqq \begin{cases} 1 & \text{if } G \text{ decides } X \\ 0 & \text{otherwise} \end{cases}$$

Next, let us assume that X is decidable. Then, there exists a hierarchical Turing machine $F \coloneqq (F_k)_{k\in S}$ that decides X. From our definition of the AI problems, this implies F must exhibit persistent novelty. Consider now H ran on inputs F and X. Clearly, $H(G,X)$ outputting 1 amounts to deciding that F exhibits persistent novelty, as otherwise F would not decide X. This is not possible under theorem 7. □

We will end our mathematical exposition with a brief footnote as to what is, arguably, the most pressing problem in the study of artificial intelligence [2] [14]. Being, whether an intelligence is able to be controlled, and aligned to human interests [13]. Interestingly, if a hierarchical Turing machine exhibits persistent novelty, by our very definition, it is unpredictable, and, likewise, uncontrollable by any less capable hierarchical Turing machine. This was not a prescient intention of the definition, but is nevertheless an immediate consequence of it.

Here a hierarchical Turing machine C is said to be "less capable" than some other hierarchical Turing machine H if, for each k, the recursive closure of C_k is a proper subset of of the recursive closure of H_k. Seeing as the recursive closure is intended to model what a hierarchical Turing machine is capable of achieving under its own reasoning power, using its current domain of knowledge, this is a way to formalize that C is "less intelligent" than H. With this in mind, we will now show that no hierarchical Turing machine C exists which is able to predict, at level k, the output of H_{k+1}, providing H has persistent novelty, and H is at least as "intelligent" as C.

Theorem 11. *Let $H \coloneqq (H_k)_{k\in S}$ be a hierarchical Turing machine with persistent novelty, and let $C \coloneqq (C_k)_{k\in\mathbb{N}}$ be a hierarchical Turing machine which takes as supplementary input at k the Turing machine H_k ran on input X_k, such that $RC_k(C,H) \subseteq RC_k(H,X)$. Then, for all $k \in S$, $C(H_k) \neq H_{k+1}(X_{k+1})$.*

Proof. Suppose that there exists such a hierarchical Turing machine C such that, for all $k \in S$, $C_k(H_k, X_k) = H_{k+1}(X_{k+1})$, and $\mathrm{RC}_k(C,X) \subseteq \mathrm{RC}_k(H,X)$. That is to say, C_k is able to predict the output of H_{k+1} given a complete description of H_k and the input to X_k. However, we have been given that H has persistent novelty. So, for each $a \in S$, there exists a $b > a$, such that

$$s \in \mathrm{St}(H_b, X_b) \quad \text{and} \quad s \notin \mathrm{RC}_c(H,X) \quad \text{for all } c < b$$

In other words, s is not obtainable from any combination of states in lower levels of the hierarchy via any partial recursive function that is a convolution of previous Turing machines. Yet, our definition of C_k means it must be able to compute the output of H at level $k+1$, given the input X_k to H_k. Hence, C_k must be able to compute some state $s \in \mathrm{St}(H_{k+1}, X_{k+1})$, being the output state of H_{k+1}. We will use s to refer to the output state of H_{k+1}.

But, if this were possible, then $s \in \mathrm{RC}_k(C,X)$. Seeing as $\mathrm{RC}_k(C,X) \subseteq \mathrm{RC}_k(H,X)$, it follows that $s \in \mathrm{RC}_k(H,X)$. However, this entails that s is the output of some convolution of Turing machines including and preceding H_k, being the Turing machine H_k itself. This contradicts the persistent novelty of H. □

Likewise, we have obtained an impossibility result for whether hierarchical Turing machines with persistent novelty can be predicted. Seeing as controlling a hierarchical Turing machine to produce a given output guarantees an accurate prediction of that given output, it follows that hierarchical Turing machines which have persistent novelty cannot be controlled by any "less intelligent" hierarchical Turing machine. To put slightly differently: Once an intelligence reaches one's own intelligence, one cannot control that intelligence. This aligns with what is presently suspected [3].

5 Conclusion

Although there is no universally accepted definition of intelligence, it is broadly agreed that an intelligent being (whether a human, or machine) must exhibit the capacity for learning. In this work, we proposed a formal model of this capacity. Specifically, that a machine exhibits intelligence if it demonstrates persistent novelty, being the continual ability to enter states not recursively obtainable from previous states.

We then showed that determining whether a hierarchical Turing machine exhibits persistent novelty is undecidable, even under highly permissive and vague conditions. Since our definition of intelligence entails this form of novelty, it follows that deciding whether a machine is intelligent is also undecidable, in general.

As a consequence, no test, including the Turing test, can universally decide whether a machine is intelligent. Likewise, we find that membership in the complexity class AI is undecidable, and that the problem of controlling machines more intelligent than oneself (i.e., predicting and bounding their future behavior) is, in general, not computable. These results rest on two philosophical assumptions:

1. That machines are fundamentally computational.
2. That not all knowledge can be learned in isolation.

Accepting the reliability of any test for intelligence, the decidability of membership in AI, or the ability to control a machine intelligence smarter than oneself requires rejecting at least one of these assumptions. To reject the first is to accept absurdity. To reject the second is to adopt the view that all knowledge is accessible from within Plato's cave. Regardless, either philosophical position demands commitment to a world where the boundaries of intelligence are as illusory as its appearances.

References

[1] George Boolos, John P. Burgess, and Richard C. Jeffrey. *Computability and Logic*. 5th. New York: Cambridge University Press, 2007. ISBN: 9780521877520.

[2] Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford, UK: Oxford University Press, 2014. ISBN: 9780199678112.

[3] Mario Brcic and Roman V. Yampolskiy. "Impossibility Results in AI: A Survey". In: *ACM Computing Surveys* 56.1 (2023), pp. 1–24. DOI: 10.1145/3603371. URL: https://doi.org/10.1145/3603371.

[4] Selmer Bringsjord, Paul Bello, and David Ferrucci. "Creativity, the Turing Test, and the (Better) Lovelace Test". In: *Minds and Machines* 11.1 (2001), pp. 3–27. DOI: 10.1023/A:1011206622741.

[5] Nigel Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge, UK: Cambridge University Press, 1980. ISBN: 0521294657.

[6] Solomon Feferman. "Are There Absolutely Unsolvable Problems? Gödel's Dichotomy". In: *Philosophia Mathematica* 14.2 (2006), pp. 134–152. DOI: 10.1093/philmat/nkj003.

[7] Gerhard Gentzen. "Die Widerspruchsfreiheit der reinen Zahlentheorie". In: *Mathematische Annalen* 112.1 (1936), pp. 493–565.

[8] Petr Hájek and Pavel Pudlák. *Metamathematics of First-Order Arithmetic*. Perspectives in Mathematical Logic. Springer, 1993. ISBN: 978-3-540-56380-1.

[9] Brenden M. Lake et al. "Building Machines That Learn and Think Like People". In: *Behavioral and Brain Sciences* 40 (2017), e253. DOI: 10.1017/S0140525X16001837.

[10] Per Lindström. "On the Criteria of Acceptability for Formalized Theories". In: *Theoria* 30.2 (1964), pp. 186–195. DOI: 10.1111/j.1755-2567.1964.tb00461.x. URL: https://doi.org/10.1111/j.1755-2567.1964.tb00461.x.

[11] Henry Gordon Rice. "Classes of Recursively Enumerable Sets and Their Decision Problems". In: *Transactions of the American Mathematical Society* 74.2 (1953), pp. 358–366. DOI: 10.2307/1990888.

[12] Hartley Jr. Rogers. *Theory of Recursive Functions and Effective Computability*. 2nd. Cambridge, MA: MIT Press, 1987.

[13] Stuart Russell. *Human Compatible: Artificial Intelligence and the Problem of Control*. New York: Viking, 2019. ISBN: 9780525558616.

[14] Anders Sandberg. "The intelligence explosion and machine ethics". In: *Machine Ethics*. Ed. by Michael Anderson and Susan Leigh Anderson. Cambridge University Press, 2011, pp. 451–470.

[15] John R. Searle. "Minds, Brains, and Programs". In: *Behavioral and Brain Sciences* 3.3 (1980), pp. 417–457. DOI: 10.1017/S0140525X00005756.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd. Cambridge, MA: MIT Press, 2018. ISBN: 9780262039246.

[17] A. M. Turing. "Computing Machinery and Intelligence". In: *Mind* LIX.236 (1950), pp. 433–460. DOI: 10.1093/mind/LIX.236.433.

[18] Roman V. Yampolskiy. "Turing Test as a Defining Feature of AI-Completeness". In: *Artificial Intelligence, Evolutionary Computing and Metaheuristics*. Ed. by Xin-She Yang. Vol. 427. Studies in Computational Intelligence. Springer, 2013, pp. 3–17. DOI: 10.1007/978-3-642-29694-9_1. URL: https://doi.org/10.1007/978-3-642-29694-9_1.