Proceedings of Machine Learning Research vol 291:1–25, 2025 38th Annual Conference on Learning Theory

Learning Algorithms in the Limit

Hristo Papazov HRISTO.PAPAZOV@EPFL.CH
Nicolas Flammarion NICOLAS.FLAMMARION@EPFL.CH
EPFL, Lausanne, Switzerland

Editors: Nika Haghtalab and Ankur Moitra

Abstract

This paper studies the problem of learning computable functions in the limit by extending Gold's inductive inference framework to incorporate *computational observations* and *restricted input sources*. Complimentary to the traditional Input-Output Observations, we introduce Time-Bound Observations, and Policy-Trajectory Observations to study the learnability of general recursive functions under more realistic constraints. While input-output observations do not suffice for learning the class of general recursive functions in the limit, we overcome this learning barrier by imposing computational complexity constraints or supplementing with approximate time-bound observations. Further, we build a formal framework around observations of *computational agents* and show that learning computable functions from policy trajectories reduces to learning rational functions from input and output, thereby revealing interesting connections to finite-state transducer inference. On the negative side, we show that computable or polynomial-mass characteristic sets cannot exist for the class of linear-time computable functions even for policy-trajectory observations.

Keywords: learning in the limit, inductive inference, algorithmic learning theory, recursion theory

1. Introduction

Nearly a century ago, Hilbert and Ackermann (1928) formulated "the most general problem of mathematics" – the so-called *Entscheidungsproblem*[1] – which sought an *effective procedure* for deciding the provability of any first-order logical sentence from a given set of axioms. Despite Hilbert's steadfast optimism, progress eluded the Entscheidungsproblem for almost a decade because, perhaps ironically, "the main problem of mathematical logic" lacked a mathematical formulation. Indeed, the term *effective procedure* stubbornly evaded formal treatment and only represented the intuitive idea of a finitely described finite process whose execution requires no spark of creativity. The breakthrough came with the work of Church (1936) and Turing (1936), who formalized the intuitive predicate *effective* through *λ-calculability* and *Turing-computability*, respectively. This paradigm shift ultimately led to the resolution of the Entscheidungsproblem in the negative.

Interestingly, the death of Hilbert's program elucidated a remarkable coincidence in mathematics: Every effort to formalize the intuitive notion of an *effectively computable* function appears to single out exactly the same set of functions. Initially proved for λ-calculability and Turing-computability, this extensional equivalence now captures all known reasonable[2] formalisms of computation. Due to the sheer and exhaustive abundance of equivalent models of computation, nowadays, computer scientists, philosophers, and mathematicians widely accept the **Church-Turing Thesis** (CTT) which

1. See (Copeland, 2023b) for a brief history of the problem.
2. Some unreasonable formal models of computation, which lead to hypercomputation, utilize infinite precision arithmetic of real numbers and mechanisms for accelerating computation, see (Piccinini and Maley, 2021). We refer the curious reader to (Aaronson, 2005) for a survey of how hypercomputation clashes with the known laws of physics.

states that Turing machines (TMs) can *simulate* all "intuitively computable" functions/processes. In our paper, we adopt the prevailing view that the umbrella term "intuitively computable" encompasses other informal notions such as "effectively computable," "computable by an algorithm," and "computable by a harnessable physical process."[3] Hence, we take for granted that a TM can simulate any reasonable computational model (see Appendix A.3 for a discussion on simulations and the CTT).

Learning Computable Functions in the Limit. Now, a computational model M working on an enumerable domain $\mathcal{D}$ may not halt for all inputs $x \in \mathcal{D}$. Thus, when considering computation at the function level, we make a distinction between the family of total recursive functions $\mathcal{C}_{\mathcal{D}}$ – computable on the full domain $\mathcal{D}$, and the family of general recursive functions $G_{\mathcal{D}}$ – computable on subsets of $\mathcal{D}$.[4] Having briefly argued that every function computable by the laws of nature or the complex processes of the human brain ultimately resides in $G_{\mathcal{D}}$ for the relevant domain $\mathcal{D}$, in this paper, we set out to prove that the family of general recursive functions are learnable in the limit from *computational observations*.

In simplest terms, the learning-in-the-limit framework of Gold (1967) refers to the scenario where a learning algorithm $\mathcal{L}$ at time $t \in \mathbb{N}$ observes a piece of information i_t about an unknown function f from a concept class Ω and comes up with a hypothesis based on everything observed so far: $h_t \leftarrow \mathcal{L}(i_1, \dots, i_t)$. If for every ground truth $f \in \Omega$ and every valid sequence of observations $(i_t)_{t=1}^{\infty}$, the learner always converges on the correct function after finitely many mistakes (i.e., $h_t = f,\ \forall t \geq t^\star$), then we say that $\mathcal{L}$ learns Ω in the limit and define Ω as learnable in the limit.

Though concise, the above description left out some important details. First, clearly $\Omega \subseteq G_{\mathcal{D}}$ for some enumerable domain of computation $\mathcal{D}$. Second, with the notation $h_t \leftarrow \mathcal{L}(i_1, \dots, i_t)$, we meant that the learning algorithm at time t outputs a representation R_t belonging to a computable set of representations $\mathcal{R}$[5] such that R_t generates the function h_t when executed on a simulator, like a Universal TM. Third, each example $i_t = (x_t, f(x_t), \alpha(M, x_t))$ includes an input x_t, a function value $f(x_t)$, and auxiliary information $\alpha(M, x_t)$[6] about the computation $x_t \mapsto f(x_t)$ by some computational model M. Crucially, any valid sequence of observations $(i_t)_{t=1}^{\infty}$ must exhaust all possible inputs: i.e., $\forall x \in \mathcal{D}\ \exists t \in \mathbb{N}$ s.t. $i_t = (x, f(x), \alpha(M, x))$.[7] Finally, the requirement that $\mathcal{L}$ must correctly learn f in the limit for any ordering of the example set $\{(x, f(x), \alpha(M, x)) : x \in \mathcal{D}\}$ exists to facilitate genuine learning. Otherwise, an ordering function could encode M as the first input, allowing $\mathcal{L}$ to learn f without meaningful generalization.

Characteristic Sets. Learning-in-the-limit algorithms should correctly identify concepts for any adversarial ordering of examples. Nevertheless, one could still study the sample efficiency of such algorithms in the presence of favorable curricula through the notion of *characteristic sets*. First introduced by Gold (1978) for finite-state transducers (FSTs) and later generalized by De La Higuera (1997) for arbitrary concept classes, characteristic sets represent a core set of inputs linked to a representation which once observed allow the learner to identify the ground truth.

3. For a broad disambiguation between different interpretations of the CTT, we refer the reader to (Piccinini and Maley, 2021; Copeland, 2023a). Remarkably, the CTT lives at the intersection of mathematics, physics, and philosophy.
4. Note that $\mathcal{C}_{\mathcal{D}} \subsetneq G_{\mathcal{D}}$ as proved by Turing (1936) through the undecidability of the Halting Problem.
5. For example, one could think of $\mathcal{R}$ as the set all finite-state automata, cellular automata, or TMs.
6. Without $\alpha(M, x)$, we arrive at Gold (1967)'s setup of *learning in the limit with an informant*.
7. We relax this requirement in our paper. Instead, if all observed inputs come from an input source $I \subseteq \mathcal{D}$, we will require a successful learner $\mathcal{L}$ to learn a representation for a function h in the limit s.t. $h_{|I} = f_{|I}$.

Additional Information. Gold proved that, when the observations include no additional information about the computational process taking x to $f(x)$ (i.e., $\alpha(M, x) = \varepsilon$ – the empty string), no algorithm can learn the concept class of totally recursive functions $\mathcal{C}_{\mathcal{D}}$ in the limit (Gold, 1967, Theorem I.5). Although, we revisit this input-output-observations (IOOs) framework in Section 3, we believe that intelligent agents hardly ever learn new skills from such impoverished observations. In the real world, learning systems observe computational processes, not the mere results of computation. For example, in human learning, teachers guide students through the process of reaching the correct answer and explain numerous intermediate steps. Similarly, research in machine learning (Wei et al. (2022); Anil et al. (2022)) demonstrates that both pretraining and in-context prompting with chain-of-thought (CoT) data significantly improve reasoning accuracy and facilitate length generalization. Motivated by these considerations, we propose two types of *computational observations* as an extension to the original framework:

- ∘ *Time-Bound Observations (TBO).* The learner $\mathcal{L}$ only gains a general sense of the task's hardness through a rough upper bound $\alpha_{\mathrm{TB}}(M, x) \in \mathbb{N}$ on the number $t_M(x)$ of discrete computation steps the computational model M takes to compute $f(x)$. That is, we assume that up to some model-dependent scale, the upper bound $t_M(x) = O_M(\alpha_{\mathrm{TB}}(M, x))$ holds.

- ∘ *Policy-Trajectory Observations (PTO).* The learner $\mathcal{L}$ observes the full interaction of the computational agent M with the environment without observational access to M's internal state updates. Instead, $\mathcal{L}$ perceives only external behavior – analogous to a student looking at a teacher write on a blackboard without access to the teacher's neuronal activity. Thus, $\alpha_{\mathrm{PT}}(M, x)$ represents the external, readily observable part of M's computational trajectory.[8]

Note that PTOs provide strictly more information than TBOs since the learner observes the exact number of computational steps. For our *TBO-Learning* results, we allow observations over arbitrary computational models. However, extracting meaningful information from PTOs requires a fixed model family. Indeed, inferences from external behavior to internal computation only work if the learner knows the model architecture in advance. Thus, all our *PTO-Learning* results concern TMs.

Paper Contributions and Organization. Section 2 develops the formal foundation, defining computational models and agents; learning in the limit with α-observations and restricted inputs; and learning efficiency through characteristic sets. The next sections contain our main contributions:

- *Learning from IOOs.* In Section 3, we generalize the learnability results from (Gold, 1967) by proving that any parameterized complexity class of general recursive functions is learnable in the limit from IOOs. In particular, polynomial-time computable functions are learnable in the limit.

- *Learning from TBOs.* In Section 4, we prove using the Extended Church-Turing Thesis (ECTT) that regardless of the computation domain $\mathcal{D}$ and the model M_f computing the ground-truth f, the concept class $G_{\mathcal{D}}$ is learnable in the limit from TBOs. Notably, without referencing the ECTT, we prove that the family of Turing-computable functions is learnable in the limit from TBOs.

- *Learning from PTOs.* In Section 5, we prove that no algorithm for learning the family of Turing-computable functions in the limit can have polynomial characteristic sets even with PTOs. Additionally, we reduce the problem of learning general recursive functions with PTOs to the problem of learning FSTs from input-restricted IOOs.

8. In Section 5, we will draw connections to Learning from Demonstrations – a subgenre of Reinforcement Learning, and show that $\alpha_{\mathrm{PT}}(M, x)$ truly constitutes observations of trajectories of the agent's policy.

Finally, Section 6 concludes with implications and open problems, and the appendices include detailed technical proofs and supporting discussion.

Related Work. Our work builds upon prior research in formal language learning and recent progress in learning computable functions. Here, we outline key contributions relevant to our framework.

Concerning FSTs, Gold (1978) proved that the class of finite-state transducers is *identifiable in the limit from polynomial time and data* (IPTD). Subsequently, Oncina and Garcia (1992) introduced the RPNI algorithm – a polynomial-time state-merging method for learning FSTs in the limit. Thereafter, De La Higuera (1997) formalized the notion of characteristic sets, demonstrating that context-free grammars and other formal classes are not IPTD. Furthermore, De La Higuera (1997) noticed that the RPNI algorithm implies that FSTs are polynomially T/L-teachable – a stronger learnability condition developed by Goldman and Mathias (1993). Building on this line of work, Parekh and Honavar (2001) proved that FSTs are PAC-learnable under simple distributions and established several equivalences between active and passive learning frameworks.

Recently, Malach (2024) showed that any computable function over fixed-length binary inputs is PAC-learnable by a linear autoregressive model when trained on CoT sequences. We extend this result by proving that computable functions over arbitrary input lengths are exactly learnable from a finite number of CoT observations. Finally, Kleinberg and Mullainathan (2024) revisited the learning-in-the-limit framework in the context of large language models, highlighting that *generation is easier than identification* through a proof that learning to generate from a ground truth language is possible—under broad conditions—when only observing positive examples.

2. Formal Framework

In this section, we build the formal bedrock for our learning-in-the-limit results. We group the topics thematically into two subsections dedicated to computational models and learning theory, respectively. Before proceeding, we introduce key notation. Let ε denote the empty string. For any set S, the Kleene plus S^+ denotes $\cup_{n=1}S^n$, while the Kleene star $S^\star$ stands for $\{\varepsilon\} \cup S^+$. For an enumerable domain $\mathcal{D}$, we define the families of general recursive $G_\mathcal{D} = \{f : \mathcal{D} \rightharpoonup \mathcal{D} \mid f -\text{ general recursive}\}$ and total recursive, or computable, functions $\mathcal{C}_\mathcal{D} = \{f : \mathcal{D} \to \mathcal{D} \mid f - \text{total recursive}\}$. For a general recursive function $f \in G_\mathcal{D}$, we use $D_f \subseteq \mathcal{D}$ to denote the set of inputs on which f is defined.

2.1. Computational Models

We begin with some unconventional terminology, particular to our paper. The class of functions $G_\mathcal{D}$ emerges from the interaction between abstract machines and symbolic environments that house the input-output domain $\mathcal{D}$. We define an **abstract machine** as a computable dynamical system with a discrete-time evolution. When one supplies an abstract machine with a formalized input-setting and output-reading convention, then the resulting system becomes a **computational model** on $\mathcal{D}$. Let $\mathcal{M}_\mathcal{D}$ denote the set of models computing the general recursive functions in $G_\mathcal{D}$. For a model $M \in \mathcal{M}_\mathcal{D}$, we use $D_M \subseteq \mathcal{D}$ to denote the set of inputs for which M halts.

We start our discussion of computational models with a formal introduction of finite-state transducers (FSTs) and Turing machines (TMs). Then, we muse a bit over the (Extended) Church-Turing Thesis and uncover a paradox that to the best of our knowledge appears unaddressed by the computability literature. We conclude the section with a formal description of computational agents.

2.1.1. FINITE-STATE TRANSDUCERS

FSTs, also known as Mealy machines, serve as a fundamental models of computation, representing sequential decision processes with finite memory. FSTs map input sequences to output sequences via state transitions and play a key role in learning theory and formal language theory.

Definition 1 (Finite-State Transducers) *An FST M is a 6-tuple $M = (Q, A, B, \delta, \gamma, q_0)$ where Q is a finite set of states, A is a finite input alphabet, B is a finite output alphabet, $\delta : Q \times A \to Q$ is a transition function, $\gamma : Q \times A \to B$ is an output function, and $q_0 \in Q$ is the start state.*

To extend δ and γ to longer inputs, we recursively define $\delta(q, a_1 \dots a_{n+1}) \coloneqq \delta(\delta(q, a_1 \dots a_n), a_{n+1})$, $\gamma(q, a_1 \dots a_{n+1}) \coloneqq \gamma(\delta(q, a_1 \dots a_n), a_{n+1})$ for all $n \geq 1$ and $a_1 \dots a_{n+1} \in A^{n+1}$. We define the **semantics** of M as the function $\gamma_M = \gamma(q_0, \cdot) : A^+ \to B$, and we use $M : A^+ \to B^+$ for the seq2seq map $M(a_1, \dots, a_n) = (\gamma_M(a_1), \dots, \gamma_M(a_1, \dots, a_n))$. For a function $\chi : A^+ \to B$, we define the **Generalized Nerode Equivalence** as $u \equiv_\chi v \iff \chi(u \cdot w) = \chi(v \cdot w),\ \forall w \in A^+$.[9] We let $[u]_\chi$ denote the equivalence class of $u \in A^\star$, and we use $|\chi|$ for the number of equivalence classes. If $|\chi| < \infty$, we call χ a **rational map**. Since $|\gamma_M| \leq |Q|$, an FST semantics always represents a rational map. Surprisingly, the converse also holds, as per the generalized Myhill-Nerode theorem.

We denote the set of FSTs over I/O alphabets A/B by Φ_A^B and the set of rational functions by $\mathcal{P}_A^B$. The set $\widetilde{\Phi}_A^B$ stands for partial FSTs with missing transitions, with the corresponding set of partial rational functions – $\widetilde{\mathcal{P}}_A^B$. Following Oncina and Garcia (1992), we define quotient transducers, with merged states according to a partition π of Q.

Definition 2 (Quotient Transducers) *Let $M = (Q, A, B, \delta, \gamma, q_0) \in \widetilde{\Phi}_A^B$ and let π be a partitioning of the state-space Q. We denote by $B(q, \pi) \subseteq Q(M)$ the unique cluster of the partitioning containing $q \in Q$. Now, we define the quotient (possibly non-deterministic) FST $M/\pi = (Q', A, B, \delta', \gamma', q_0')$ as follows: $Q' = \{B(q, \pi) : q \in Q\}$; $(\delta', \gamma')(B, a) = \{(B', b) : \exists q \in B, q' \in B' \text{ s.t. } (\delta, \gamma)(q, a) = (q', b)\}$, $\forall B \in Q', a \in A$; $q_0' = B(q_0, \pi)$.*

To differentiate states, we use the notion of **state apartness**, following Vaandrager et al. (2022). Two states $p, q \in Q$ are **apart**, denoted $p \# q$, if there exists a **distinguishing string** $\sigma \in A^+$ such that $\gamma(p, \sigma) \neq \gamma(q, \sigma)$. Hopcroft's algorithm (Hopcroft, 1971) efficiently determines state apartness.

2.1.2. TURING MACHINES

One can define Turing machines with various architectural choices, including different numbers of heads and tapes, input-output conventions, halting conditions, and alphabet sizes. While all such variants prove computationally equivalent (Arora and Barak, 2009, Chapter 1), some provide greater convenience for exposition. We adopt a specific TM architecture that best suits our discussion.

Let λ denote the blank symbol. Given a finite problem alphabet Σ s.t. $\lambda \notin \Sigma$ and a finite tape alphabet $\Gamma \supseteq \Sigma \cup \{\lambda\}$, our model family T_Σ^Γ computes all general recursive functions in $G_{\Sigma^\star}$.

Definition 3 (Turing Machine) *A TM $T \in \mathrm{TM}^\Gamma$ is a triple $T = (Q, \Gamma, \delta)$ where Q is a finite set of states, Γ is a finite tape alphabet, and $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$ is a transition function.*

9. We use $\cdot$ to denote concatenation. We invite the reader to Appendix A.1 for additional discussion on FSTs.

TM Computation. In the above definition, the symbols L, R, and S dictate whether the TM head moves left, right, or stays in place after transitioning to a new state and writing a symbol from Γ over the current cell of a bidirectionally infinite tape. The computation of any $T \in \mathrm{TM}^{\Gamma}$ proceeds as follows: The input $x \in \Sigma^{\star}$ is written on an otherwise-blank tape with the TM head positioned at the first symbol of x and T initialized at a specified initial state $q_0 \in Q$. The TM then follows the transition function δ. Computation halts when T enters a designated halt state $q_f \in Q$. The output $T(x)$ is defined if and only if just a single contiguous string from $\Sigma^{\star}$ occupies the tape in the moment of halting. Otherwise, $T(x)$ remains undefined. Now, the class of abstract machines TM^{Γ}, together with this computation convention, specifies the model class $\mathrm{T}_{\Sigma}^{\Gamma}$ over the enumerable domain $\Sigma^{\star}$. We overload the definition of a TM to refer both to an abstract machine and a computational model.

Representation Complexity. We define $|T|$ as the number of states in $T \in \mathrm{T}_{\Sigma}^{\Gamma}$. Each TM uses exactly $|T| \cdot |\Gamma|$ transitions. Since we keep the tape alphabet Γ constant, $O(|T| \log |T|)$ bits suffice for encoding T. Thus, $|T|$ serves as a meaningful proxy for the representational complexity of T.

2.1.3. ENCODINGS AND SIMULATIONS

The Church-Turing Thesis (CTT) effectively defines computability through Turing-computability by claiming that TMs can simulate any intuitively computable function modulo a *reasonable* encoding. However, the term "reasonable" appears to rely on the notion of "computable" leading to an infinite regress. To resolve this paradox, we propose grounding reasonability in first-order logic (FOL). Thus, we define an encoding as reasonable if and only if the encoding derives from a TM-computable transformation of an inherent FOL representation of the domain (see details in Appendix A.3).

The Extended Church-Turing Thesis As computability theory evolved, considerations extended beyond feasibility to the *efficiency* of computation. Accordingly, the CTT received a strengthening in the form of the **Extended Church-Turing Thesis** (ECTT)[10], which asserts that the class of *efficiently computable problems* is model-independent. Specifically, for any physically realizable computational model M, there exists a constant $c \in \mathbb{N}$ such that if M evolves for t steps, a TM $T \in \mathrm{T}_{\Sigma}^{\Gamma}$ can simulate M's evolution within t^c steps (Arora and Barak, 2009, Chapter 1.5.2). While less universally accepted, the ECTT remains a widely used conjecture. We adopt the following relaxed formulation.

Assumption 4 (q-Extended Church-Turing Thesis) *Let $q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a computable monotonically increasing "overhead" function. The q-ECTT states that for any physically realizable computational model M, there exists a constant $c \in \mathbb{N}$ and a TM $T \in \mathrm{T}_{\Sigma}^{\Gamma}$ such that if a computation on M takes t steps, then T simulates the same computation in $q(c, t)$ steps.*

For a computational model M over a domain $\mathcal{D}$, the q-ECTT implies the existence of a reasonable encoding $\varphi : \mathcal{D} \to \Sigma^{\star}$ such that some $T \in \mathrm{T}_{\Sigma}^{\Gamma}$ achieves the specified time-bound. Since switching between non-adversarially constructed encodings introduces at most a constant slowdown to the simulation time[11], we take as granted that the q-ECTT implies the additional assumption that all natural encodings of $\mathcal{D}$ into $\Sigma^{\star}$ cause the q-ECTT to kick in.

10. See (Copeland, 2023a) for a detailed discussion.
11. The TM $(\phi \circ \varphi^{-1}) \circ T \circ (\varphi \circ \phi^{-1})$ simulates M with at most a constant slowdown for naturally constructed ϕ.

2.1.4. COMPUTATIONAL AGENTS

Our *IOO–Learning* (Section 3) and *TBO–Learning* (Section 4) results apply to the general unrestricted model family $\mathcal{M}_\mathcal{D}$ over the domain $\mathcal{D}$. However, for *PTO–Learning* (Section 5), we introduce the concept of a computational agent with observable behavior.

Following Russell and Norvig (2016), we view a **computational agent** as a system that perceives and acts upon an environment. We model this environment as a discrete symbolic universe $\mathcal{U}$ for two key reasons. First, upon introspection, any reasoning or mechanical task computed by humans appears to involve manipulation of symbols in a discrete space. Second, though physical reality possibly incorporates both discrete and continuous quantities, the unavoidable noise in physical observations will cause the sensory data of any computational system to achieve only finite precision.

Definition 5 (Symbolic Universe) *Given an enumerable set G and a finite set of symbols Γ containing the empty symbol λ, a world-state $w : G \to \Gamma$ is a function with finitely many non-empty assignments. The countable symbolic universe $\mathcal{U} = \mathcal{U}(G, \Gamma)$ is the set of all such world-states.*

The set G defines the geometry of the universe. For example, for TMs, $G = \mathbb{Z}$ represents tape positions, and world-states $w : \mathbb{Z} \to \mathcal{S}$ correspond to tape configurations. We now define a computational agent as an entity interacting with a symbolic universe. Let $\mathcal{P}(\cdot)$ denote powersets.

Definition 6 (Computational Agent) *A computational agent operating in a symbolic universe $\mathcal{U}(G, \Gamma)$ constitutes a triple $M = (Q, \mathcal{U}, \delta)$, where Q is a finite set of states and $\delta : Q \times \mathcal{P}(G) \times \mathcal{U} \to Q \times \mathcal{P}(G) \times \mathcal{U}$ is a computable transition function with perception restrictions given below.*

The transition function δ determines how the agent evolves: $\delta(q, P, w) = (q', P', w')$, where $q \in Q$ is the current state, $P \subseteq G$ is the agent's perception (the observed region of the universe), and $w \in \mathcal{U}$ is the current world-state. The agent updates the internal state (q'), shifts perception (P'), and modifies the environment (w'). Notably, δ can only access $w_{|P}$, i.e., the portion of the world within the field of perception. For a complete interaction trace, we specify an initial state $q_0 \in Q$, an initial perceived region $P_0 \subseteq G$, and a final state $q_f \in Q$ as a halting condition. Since we model the environment as static and the agent follows a fixed transition function, the system defines a **symbolic dynamical process** $\delta : \Lambda \to \Lambda$, where $\Lambda = Q \times \mathcal{P}(G) \times \mathcal{U}$.

Examples of Computational Agents. Many computational models fit this framework. TMs, RNNs, and transformers all operate on symbolic universes with geometry $G = \mathbb{Z}$. For these models, the symbol-space Γ corresponds to the tape alphabet, input-output symbols, or token set. TMs serve as the prototypical *universal* computational agents, with a continuously moving singleton perception. RNNs and transformers (under realistic constraints such as finite-precision weights and finite context windows) reduce to complex finite-state transducers. Humans may also function as computational agents when performing structured tasks without modifying their neural architecture.

2.2. Algorithmic Learning Theory

We formalize the learning-in-the-limit framework, focusing on observations and characteristic sets.

2.2.1. LEARNING IN THE LIMIT WITH INPUT SOURCES AND α-OBSERVATIONS

We extend Gold's learning-in-the-limit framework by incorporating a family of computational models, a restricted input source, and a broader notion of natural observations.

Our generalization introduces the set of **computational models** $\mathcal{M}_{\mathcal{D}}$ over an enumerable domain $\mathcal{D}$ which compute the general recursive functions in $G_{\mathcal{D}}$. For any subset $\mathcal{M} \subseteq \mathcal{M}_{\mathcal{D}}$, let $\mathcal{F}_{\mathcal{M}} \subseteq G_{\mathcal{D}}$ denote the set of functions computed by $\mathcal{M}$, and for any $M \in \mathcal{M}_{\mathcal{D}}$, let $f_M : \mathcal{D} \rightharpoonup \mathcal{D}$ denote the function computed by M. We aim to learn the **concept class** $\mathcal{F}_{\mathcal{M}} \subseteq G_{\mathcal{D}}$.

Information Presentation. Given a ground-truth model $M \in \mathcal{M}$ from a **model class** $\mathcal{M} \subseteq \mathcal{M}_{\mathcal{D}}$, the learning algorithm $\mathcal{L}$ observes examples from an **input source** $I \subseteq \mathcal{D}$ such that **inputs** x only arrive from the set $I_M \coloneqq I \cap D_M$ and produce observable **examples** $\eta_M(x) = (x, f_M(x), \alpha(M, x))$ where $\alpha(M, x)$ contains α-**observations** about the computation of $f_M(x)$. For a sample set $S \subseteq I_M$, we denote the observed examples by $E_M(S) = \{\eta_M(x) : x \in S\}$. We define the equivalence class $[M]_S = \{T \in \mathcal{M} : E_T(S) = E_M(S)\}$ of indistinguishable models on S. Now, an adversary orders the examples using a **surjective ordering** $w : \mathbb{N} \twoheadrightarrow I_M$, so that at time t, the learner receives $\eta_M(w_t)$. The learner produces a hypothesis $R_t = \mathcal{L}(\eta_M(w_1), \ldots, \eta_M(w_t))$, where R_t belongs to a computable **set of representations** $\mathcal{R}$. Through a **simulation function** $\mathfrak{S} : \mathcal{R} \to G_{\mathcal{D}}$, the representation R_t generates a general recursive[12] hypothesis $\mathfrak{S}(R_t)$.

Definition 7 (The $(\mathcal{M}, \alpha)$–LP) *For a model class $\mathcal{M} \subseteq \mathcal{M}_{\mathcal{D}}$, we say that a learning algorithm $\mathcal{L}$ solves the $(\mathcal{M}, \alpha)$–learning problem[13] if for all models $M \in \mathcal{M}$, all input sources $I_M \subseteq D_M$, and all surjective orderings $w : \mathbb{N} \twoheadrightarrow I_M$ of $E_M(I_M)$ from α-observation, after some finite time $t^\star = t^\star(M, I, w)$, $\mathcal{L}$ converges to a correct representation $\hat{R} = R_t$, $\forall t \geq t^\star$ s.t. $\mathfrak{S}(\hat{R})_{|I_M} = f_{M|I_M}$.*

The $(\mathcal{M}, \alpha)$–LP framework generalizes Gold's learning-in-the-limit setup as follows. **(a)** Introduction of computational models and α-observations: Learning now depends on an unknown model M computing f_M whose computation provides additional structured information. **(b)** Restriction on input sources: Observations may now come only from a subset I of the full input domain $\mathcal{D}$.

Notice that if $\mathcal{M} \subseteq \mathcal{M}'$, then a learner $\mathcal{L}$ solving the $(\mathcal{M}', \alpha)$–LP will also solve the $(\mathcal{M}, \alpha)$–LP. Hence, the hardest learning-in-the-limit setting becomes the unrestricted $(\mathcal{M}_{\mathcal{D}}, \alpha)$–LP where the learner lacks prior knowledge about the ground-truth process that could reduce the complexity of the hypothesis space. Now, the set of all possible models $\mathcal{M}_{\mathcal{D}}$ acting on $\mathcal{D}$ includes a multitude of different computational species, which makes the existence of a unified learning strategy all the more surprising. The motivation for this learning setup comes from envisioning a robot trying to learn a computational task from observations. The robot observes and understands the domain of action $\mathcal{D}$ but lacks information about the architectural specifics of the black-box model performing the task. Moreover, only a restricted subset I of the domain $\mathcal{D}$ might produce relevant inputs to the target task.

2.2.2. Characteristic Sets

The $(\mathcal{M}, \alpha)-$LP provides a formal criterion which guarantees genuine learning robust to example reorderings and input-source restrictions. However, once a learning algorithm $\mathcal{L}$ solves the $(\mathcal{M}, \alpha)-$LP, we would also like to quantify $\mathcal{L}$'s data efficiency under favorable curricula. Formalized by De La Higuera (1997), the characteristic-sets framework provides such a characterization. We adapt the original definition to incorporate α-observations and input restrictions.

Definition 8 (Characteristic Sets) *Let $\mathcal{M} \subseteq \mathcal{M}_{\mathcal{D}}$ and let $\mathcal{L}$ solve the $(\mathcal{M}, \alpha)$–LP. Characteristic sets exist relative to $\mathcal{L}$, $\mathcal{M}$, α, and $I \subseteq \mathcal{D}$. We define $S_M \subseteq I \cap D_M$ as a characteristic set for $M \in \mathcal{M}$ if $\mathcal{L}(E_M(S))$ computes the same function as M restricted to $I \cap D_M$ whenever $S \supseteq S_M$.*[14]

12. Since $\mathcal{C}_{\mathcal{D}}$ is not recursively enumerable, $\mathcal{R}$ might include representations of some non-total recursive functions.
13. Equivalently, $\mathcal{L}$ learns $\mathcal{F}(\mathcal{M})$ in the limit from α-observations of $\mathcal{M}$ and restricted inputs.
14. Notice that an effective curriculum for learning the function of M with $\mathcal{L}$ and α would prioritize inputs from S_M.

Intuitively, characteristic sets form a core set of inputs that enable efficient model identification. We measure the complexity of S_M in two key ways. First, we define the **mass** of S_M as $\|S_M\| = \sum_{s \in S_M} \texttt{length}(x, f(x), \alpha(M, x))$, which quantifies the size of the data $\mathcal{L}$ must process. Second, we define the **size** of S_M as $|S_M|$. Then, a model class $\mathcal{M}$ is **α-restrictively identifiable in polynomial time and data** (α-RIPTD) if there exists a polynomial-time learner $\mathcal{L}$ for which every $M \in \mathcal{M}$ admits a characteristic set with polynomial mass $\|S_M\| = \texttt{poly}(|M|)$ regardless of the input source I. For α-RIPTD model classes, the complexity of representations determines the hardness of learning. Gold (1978) showed that the FST model class Φ_A^B is ε-IPTD (only input-output observations and fixed $I = A^+$). In contrast, in Section 5, we prove that Φ_A^B is not ε-RIPTD. Thus, requiring identifiability for a variable input source makes learning considerably harder.

Note that the mass and size of characteristic sets can only increase when expanding the model class $\mathcal{M}$ or reducing the informational content of α. In particular, if $\mathcal{M}$ is not α-RIPTD, then any $\mathcal{M}' \supseteq \mathcal{M}$ is also not α-RIPTD. We conclude with the following observation proved in Appendix A.5.

Lemma 9 (Distinguishability) *Let $\mathcal{L}$ solve the $(\mathcal{M}, \alpha)$–LP. Suppose that characteristic sets exist relative to $\mathcal{L}, \mathcal{M}, \alpha,$ and $I \subseteq \mathcal{D}$. Let $M, M' \in \mathcal{M}$ compute two distinct functions when restricted to I. Then, there exists $x \in S_M \cup S_{M'}$ such that either $x \notin D_M \cap D_{M'}$ or $\eta_M(x) \neq \eta_{M'}(x)$.*

We mention in passing that learning efficiency in other frameworks implies the existence of learning-in-the-limit algorithms with small characteristic sets. For instance, a valid T/L pair from (Goldman and Mathias, 1993) leads to the existence of computable characteristic sets from model representations, and semi-polynomial T/L-teachability implies IPTD (see (De La Higuera, 1997)). Moreover, polynomial-mass/size example-based algorithms (like Angluin's $L^\star$ algorithm (Angluin, 1987)) lead to algorithms with polynomial-mass/size characteristic sets due to (De La Higuera, 1997, Proposition 1) + (Goldman and Mathias, 1993, Theorem 2).

3. Learning from Input-Output Observations

In this original formulation of learning in the limit, the learner $\mathcal{L}$ observes only inputs and outputs, with no additional information about the computation process. Thus, $\alpha(\cdot, \cdot) \equiv \varepsilon$, and the model family $\mathcal{M}$ contributes nothing beyond function evaluations. We refer to this setting as the **$\mathcal{F}(\mathcal{M})$–LP**.

For the input-output learning setup, Gold (1967) proved the following negative result through a diagonalization argument that forces any fixed learner to switch between hypotheses infinitely often.

Theorem 10 (Gold, Theorem I.5) *No algorithm learns the class of computable functions in the limit from input-output observations. That is, for a finite alphabet Σ, no algorithm solves the $\mathcal{C}_{\Sigma^\star}$–LP.*

Interestingly, we prove that restricting the time complexity of the concept class enables learnability in the limit from input-output observations and arbitrary input sources. Now, for an enumerable domain $\mathcal{D}$, let $\texttt{size}_\mathcal{D} : \mathcal{D} \to \mathbb{N}$ represent some canonical measure of the complexity of the input known to the learner $\mathcal{L}$. We introduce the following parametrized complexity classes.

Definition 11 (General Complexity Classes) *For a model class $\mathcal{M} \subseteq \mathcal{M}_\mathcal{D}$ over an enumerable set $\mathcal{D}$ and for a computable monotonically increasing function $Q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, let $\mathbf{TIME}_\mathcal{D}^\mathcal{M}(Q(c, n))$ denote the subset of general recursive functions $f \in \mathcal{F}_\mathcal{M}$ for which there exists a computational model $M \in \mathcal{M}$ – not necessarily a TM – that computes f on halting inputs $x \in \mathcal{D}_f$ of $\texttt{size}_\mathcal{D}(x) = n$ in $O(Q(c, n))$ steps. We define the following complexity class $\mathbf{Q}(\mathcal{M}, \mathcal{D}) = \bigcup_{c \in \mathbb{N}} \mathbf{TIME}_\mathcal{D}^\mathcal{M}(Q(c, n))$.*

Now, under a relaxed form of the Extended Church-Turing Thesis, we prove that the concept class $\mathbf{Q}(\mathcal{M}_{\mathcal{D}}, \mathcal{D})$ is learnable in the limit from input-output observations and arbitrary input sources.

Theorem 12 (Time-Restricted IOO-Learning) *Assuming the q-ECTT (4) holds, for an enumerable set $\mathcal{D}$ and a computable monotonically increasing function $Q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, there exists an algorithm which learns the $Q-$time-bounded general recursive functions over $\mathcal{D}$ from input-output observations and restricted inputs. Formally, there exists an algorithm solving the $\mathbf{Q}(\mathcal{M}_{\mathcal{D}}, \mathcal{D})$–LP.*

The proof of Theorem 12, delegated to Appendix B, provides a learning-by-enumeration strategy which outputs the min-state TM simulating the ground truth. Since the number of $n-$state TMs in $\mathrm{T}_{\Sigma}^{\Gamma}$ is super-exponential in n, such an enumeration algorithm requires super-exponential effort in the size of the ground-truth representation to converge.

Now, even if the q-ECTT fails, we can still salvage an important corollary if we stick to complexity classes over Turing machines: i.e., $\mathbf{Q}(\mathrm{T}_{\Sigma}^{\Gamma}, \Sigma^{\star})$. Then, the proof of Theorem 12 readily extends to this setting by simply replacing the encoding φ with the identity and letting $q(c, n) = n$. Hence, popular complexity classes like $\mathbf{P}, \mathbf{NP}, \#\mathbf{P}, \mathbf{L}$, and $\mathbf{EXP}$ become learnable in the limit.

Corollary 13 *There exists an algorithm for learning any parametrized complexity class of general recursive functions in the limit from input-output observations and restricted inputs. Formally, for any finite problem alphabet Σ, tape alphabet $\Gamma \supseteq \Sigma \sqcup \{\lambda\}$, and computable monotonically increasing function $Q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, there exists an algorithm solving the $\mathbf{Q}(\mathrm{T}_{\Sigma}^{\Gamma}, \Sigma^{\star})$–LP.*

4. Learning from Time-Bound Observations.

Given a model class $\mathcal{M} \subseteq \mathcal{M}_{\mathcal{D}}$ over an enumerable domain of computation $\mathcal{D}$, time-bound observations provide the learner $\mathcal{L}$ not only with input-output information $(x, f_M(x))$ about the ground-truth model $M \in \mathcal{M}$ but also with an approximate upper bound $\alpha_{\mathrm{TB}}(M, x) \in \mathbb{N}$ on the number of discrete computation steps taken by M to compute $f_M(x)$. In other words, if we denote with $t_M(x)$ the number of computation steps M takes on a halting computation trajectory, then there exists some constant dependent on M such that $t_M(x) = O_M(\alpha_{\mathrm{TB}}(M, x)),\ \forall x \in D_M$.

To motivate the practical relevance of such observations, we could consider a robot with restricted access to the computation of some model M. Perhaps, the robot only observes a terminal that once provided with an input x, returns $f_M(x)$ after a certain amount of time. In such information-poor scenarios, the robot could still reference its internal clock and measure the number of seconds $\alpha_{\mathrm{TB}}(M, x)$ between entering x in the terminal and observing $f_M(x)$. If each discrete computation step of M takes $1/p$ seconds, then M will compute $f_M(x)$ in at most $\lceil p\alpha_{\mathrm{TB}}(M, x)\rceil$ steps.

In the following universal learning theorem, the q-ECTT allows us to translate the abstract computational models of $\mathcal{M}_{\mathcal{D}}$ into concrete TM representations. Furthermore, time-bound observations serve a similar role to that of restricting the complexity class of the target function as in our IOO-learning results.

Theorem 14 (Universal Learning with TBOs) *Assuming the q-ECTT (4) holds, for an enumerable set $\mathcal{D}$, there exists an algorithm for learning the class of general recursive functions $G_{\mathcal{D}}$ from time-bound observations of arbitrary computational models and input sources. Formally, there exists an algorithm solving the $(\mathcal{M}_{\mathcal{D}}, \alpha_{\mathrm{TB}})$–LP.*

The proof of Theorem 14, deferred to Appendix C, mimics the learning-by-enumeration technique we developed for input-output learning. Again, the learner $\mathcal{L}$ which solves the $(\mathcal{M}_{\mathcal{D}}, \alpha_{\mathrm{TB}})$–LP returns a consistent min-state model and requires super-exponential runtime to converge. Now, even if the q-ECTT fails empirically, we can still prove that the model class of Turing machines $\mathrm{T}_{\Sigma}^{\Gamma}$ is learnable in the limit from time-bound observations. This corollary follows by replacing the encoding φ with the identity function and setting $q(c, n) = n$ in the proof of Theorem 14.

Corollary 15 *There exists an algorithm for learning the class of general recursive functions in the limit from time-bound observations of Turing machines on restricted inputs. Formally, there exists a learning algorithm solving the* $(\mathrm{T}_{\Sigma}^{\Gamma}, \alpha_{\mathrm{TB}})$*–LP.*

5. Learning from Policy-Trajectory Observations

In the PTO framework, the learner $\mathcal{L}$ observes the full interaction of a computational agent $M = (Q, \mathcal{U}, \delta)$ with its symbolic universe $\mathcal{U}(G, \Gamma)$. As discussed in Section 2.1, this interaction forms a computable symbolic dynamical process $\delta : \Lambda \to \Lambda$, where $\Lambda = Q \times \mathcal{P}(G) \times \mathcal{U}$. Importantly, the transition function δ updates a triple $(q, P, w) \in \Lambda$ according to the current agential state q and the information gleaned from the field of perception $w_{|P}$. We propose that the update $(P, w) \mapsto (P', w')$ should represent the observable part of the full interaction $(q, P, w) \mapsto (q', P', w')$. That is, we intend to decompose the interaction space Λ into a hidden part $\Lambda_H = Q$ and an observable part $\Lambda_O = \mathcal{P}(G) \times \mathcal{U}$ such that $\Lambda = \Lambda_H \times \Lambda_O$.

The motivation for such a decomposition comes from human learning. For example, when a teacher (M) solves a quadratic equation (f), the student ($\mathcal{L}$) observes both the full blackboard (w) and the chalk interaction spots (P) as the teacher paces around and makes symbolic changes. More generally, humans seem quite capable of detecting where the perception of fellow humans (or other animals) falls, which allows us to make educated guesses about the inner lives of the observed. Of course, the closer the evolutionary distance to the observed (i.e., the more we instinctively understand about the agent family $\mathcal{M}$), the more educated the guess. Thus, knowledge of $\mathcal{M}$'s architectural specifics appears essential if we want to link external behavior to internal configurations. Accordingly, we will soon adopt the family of TMs $\mathrm{T}_{\Sigma}^{\Gamma}$ as our concrete object of study.

Now, since the learner $\mathcal{L}$ might lack the capacity to observe the full world-state w (e.g., due to a very large blackboard) and since only the attended part of w carries a learning signal, we further compress the observation $(P_t, w_t) \mapsto (P_{t+1}, w_{t+1})$ at time $t + 1$ into the action tuple $\Delta_{t+1} = (w_{t|P_t}, w_{t+1|P_t}, D_{t+1})$, where $D_{t+1} =$ "$P_{t+1} - P_t$" denotes the shift in perception. Hence, we define a **policy-trajectory observation** for an agent M as a sequence of action tuples $\tau = \{\Delta_t\}_{t=1}^{L}$.

Connection to Reinforcement Learning. Here, we outline some superficial similarities between our PTO-Learning framework and Learning from Demonstration (LfD), also known as Apprenticeship Learning – a subgenre of RL. For a longer discussion of LfD, we refer the reader to (Russell and Norvig, 2016; Sutton and Barto, 2018; Correia and Alexandre, 2024).

Now, the LfD framework often uses a POMDP as a foundation. The demonstration dataset $D = \{\tau_i\}_{i=1}^{N}$ consists of expert trajectories $\tau_i = (o_t^i, a_t^i)_{t=0}^{L_i}$ where o_t^i belongs to a finite set of possible partial observations Ω, a_t^i belongs to a finite set of possible actions $\mathcal{A}$, and L_i denotes the length of the interaction. Based on this demonstration data, the agent tries to develop a policy approximating the stochastic history-dependent expert policy $a_t^i \sim \pi^{\star}(o_0^i, \ldots, o_t^i)$.

Similarly, for PTO-Learning, we can think of the agent M as implementing a deterministic policy π_M which takes as input a history of partial world-state observations $\mathfrak{h}_t = (w_{0|P_0}, \dots, w_{t|P_t}) \in \Gamma^+$ and outputs an action pair consisting of a symbol $w_{t+1|P_t}$ and an attention shift D_{t+1}. If we let $\mathcal{S}$ denote the finite set of possible attention shifts, then the policy of M becomes a partial function $\pi_M : \Gamma^+ \rightharpoonup \Gamma \times \mathcal{S}$. Interestingly, one could extend π_M to a *rational function* on the full domain Γ^+ as we will see later on in the section. Now, if M takes $t_M(x)$ steps to compute $f_M(x)$, then the policy-trajectory observation becomes

$$\alpha_{\mathrm{PT}}(M, x) = (w_{t-1|P_{t-1}}, w_{t|P_t}, D_t)_{t=1}^{t_M(x)} = (w_{t-1|P_{t-1}}, \pi_M(\mathfrak{h}_{t-1}))_{t=1}^{t_M(x)} \in (\Gamma^2 \times \mathcal{S})^{t_M(x)}.$$

TM Policy Trajectories. We focus on policy-trajectory observations of TMs, making PTO-Learning a species-aware framework: The learner $\mathcal{L}$ knows $M \in \mathrm{T}_\Sigma^\Gamma$ and generates hypotheses using representations from T_Σ^Γ. The additional information $\alpha_{\mathrm{PT}}(M, x)$ consists of a sequence of tape manipulations $\alpha_{\mathrm{PT}}(M, x) = (\Delta_1, \dots, \Delta_{t_x})$, where $\Delta_i = (\sigma_i, \sigma'_i, D) \in \Gamma^2 \times \{L, R, S\}$. Each Δ_i records the TM's i^{th} operation: Reading σ_i, writing σ'_i, and moving D.

We proceed to introduce some useful notation. For a TM $T \in \mathrm{T}_\Sigma^\Gamma$, we denote by $T[x]$ and $T\{x\}$ the sequence of tape manipulations and the sequence of symbols scanned by T on input x, respectively. If $T[x] = (\Delta_1, \dots, \Delta_{t_x})$ where $\Delta_i = (\sigma_i, \sigma'_i, D) \in \Gamma^2 \times \{L, R, S\}$, then $T\{x\} = (\sigma_1 \dots \sigma_{t_x}) \in \Gamma^+$. Note that this notation extends to infinite sequences and that $T[x] = \alpha_{\mathrm{PT}}(T, x)$. Furthermore, for a sample of inputs $S \subseteq \Sigma^\star$, we overload the notation $T[S] = \{T[x] : x \in S\}$ and $T\{S\} = \{T\{x\} : x \in S\}$. We refer to $T[x]$ and $T[S]$ as **tape behavior**, and we denote with $[T]_S \subseteq \mathrm{T}_\Sigma^\Gamma$ the equivalence class of TMs with the same tape behavior on S as T.

Connection to FSTs. Interpreting $A = \Gamma$ and $B = \Gamma \times \{L, R, S\}$ as input/output alphabets, we define a mapping $\psi : \mathrm{T}_\Sigma^\Gamma \to \Phi_A^B$ that converts any TM $T \in \mathrm{T}_\Sigma^\Gamma$ into an FST $\psi(T) \in \Phi_A^B$ with the same transition diagram. Consequently, tape behavior $T[x]$ corresponds to a sequence of input-output observations $(T\{x\}, \psi(T)(T\{x\}))$ of the rational function $\gamma_{\psi(T)}$ corresponding to the semantics of $\psi(T)$. This reduces the $(\mathrm{T}_\Sigma^\Gamma, \alpha_{\mathrm{PT}})$–LP to learning the class of rational functions $\mathcal{P}_A^B$ in the limit from IOOs with a restricted source: i.e., the $\mathcal{P}_A^B$–LP.[15] Notice that if T's inputs come from $I \subseteq \Sigma^\star$, then $\psi(T)$'s inputs come from $M\{I\} \subseteq \Gamma^+$. We summarize these findings in the following lemma.

Theorem 16 (Recursive-to-Rational Reduction) *Learning the class of general recursive functions in the limit from PTOs reduces to learning the class of rational functions in the limit from IOOs. Formally, an algorithm solving the $\mathcal{P}_A^B$–LP with $A = \Gamma$, $B = \Gamma \times \{L, R, S\}$ will solve the $(\mathrm{T}_\Sigma^\Gamma, \alpha_{\mathrm{PT}})$–LP. In particular, there exists a learning-by-enumeration algorithm for the $\mathcal{P}_A^B$–LP.*

The proof of Theorem 16, deferred to Appendix D, shows the learnability of the $(\mathrm{T}_\Sigma^\Gamma, \alpha_{\mathrm{PT}})$–LP.

Uncomputable Characteristic Sets. We now state a negative result showing that no algorithm can learn the $(\mathrm{T}_\Sigma^\Gamma, \alpha_{\mathrm{PT}})$–LP or the $\mathcal{P}_A^B$–LP with bounded-mass characteristic sets. In particular, our theorem proves that input-source restrictions make learning much harder since for unrestricted inputs, the rational functions class $\mathcal{P}_A^B$ is IPTD (Gold, 1978, Theorem 4).

Theorem 17 (Unbounded Characteristic Sets) *Fix a computable bounding function $\beta : \mathrm{T}_\Sigma^\Gamma \to \mathbb{N}$ and let $I = \Sigma^\star$. Then, for α_{PT}-observations over the input source I, no algorithm achieves characteristic sets for T_Σ^Γ with β-bounded mass. Furthermore, for α_{PT}-observations over I, no algorithm for learning the class T_Σ^Γ with computable characteristic sets exists.*

15. This change requires extending Definition 7 to models with different input/output domains.

The proof of Theorem 17, found in Appendix D, uses a reduction to the Halting Problem and shows that even the model class of linear-time TMs cannot have computable characteristic sets.[16] Hence, we strongly conclude that the representation complexity of TMs does not control the hardness of learning $G_{\Sigma^\star}$ in the limit from policy-trajectory observations of T_Σ^Γ. In light of the proof of Theorem 17, one might argue that learning efficiency should be measured not by the *mass* of characteristic sets $\|S_M\|$ but by their *size* $|S_M|$ (see Appendix D).[17]

As a small corollary, setting β as any polynomial implies that T_Σ^Γ is not α_{PT}-IPTD. Consequently, T_Σ^Γ is also not IPTD from time-bound or input-output observations.

Corollary 18 T_Σ^Γ *is not IPTD from input-output, time-bound, or policy-trajectory observations.*

Observation Trees. As shown in Theorem 16, one can solve the $\mathcal{P}_A^B$–LP, and thus the T_Σ^Γ-LP, via enumeration. However, this approach requires an super-exponential runtime in the size of the ground-truth representation in order to eliminate all hypotheses preceding the correct one. Moreover, learning-by-enumeration fails to leverage structural information revealed by the observations. In contrast, both active (Angluin, 1987; Pitt, 1989; Vaandrager et al., 2022) and passive (Gold, 1978; Dupont, 1996; Parekh and Honavar, 2001) FST-learning algorithms exploit structural information by constructing observation structures from example sets $E_M(S)$ and applying state-merging strategies. Notably, these algorithms assume an unrestricted input source $I = A^\star$, allowing free exploration of the state-transition diagram. For arbitrary input sources, learning becomes significantly harder – at least as difficult as learning TMs from behavior observations, as Theorem 16 suggests. The key challenge in learning TMs stems from the fact that, unlike FSTs, their transition function dictates which cells to scan, restricting direct exploration of the state-transition diagram. As a result, we enter uncharted territory, where we aim to develop a fast algorithm for learning FSTs from arbitrary input sources. To incorporate structural information from examples, we introduce the following definition.

Definition 19 (Observation Tree) *Given an example set $E_M(S) = \{(x, M(x)) : x \in S\}$ for an automaton $M \in \Phi_A^B$, we define the observation tree $\mathcal{T}_M(S) \in \widetilde{\Phi}_A^B$ as a partial FST with paths from the root to leaf states which correspond exactly to the input-output sequences from $E_M(S)$.*

State Merging. Given an input source $I \subseteq A^\star$ and an automaton $M \in \Phi_A^B$, we denote by $I(M) \in \widetilde{\Phi}_A^B$ the partial automaton obtained after the pruning of unused states and transitions when all inputs come from I. We want to merge the states of $\mathcal{T}_M(S)$ into clusters such that the transformed automaton computes the same partial rational function as $I(M)$. When we merge two states $p, q \in Q(\mathcal{T}_M(S))$, we also merge all of the transition paths going through p and q. Hence, the merger of two states could potentially cascade into further mergers. Moreover, mergers can only occur between states with non-contradictory paths, hence the following definition.

Definition 20 (Valid Merger) *Let π denote the merging of the state-space $Q(\mathcal{T}_M(S))$ into the clusters $C_1, \ldots, C_k$. We define π as a valid merger if the quotient automaton $\mathcal{T}_M(S)/\pi$ is deterministic.*

If $|Q(\mathcal{T}_M(S))| = n$, then one can test the validity of a certain merger in $O(n|A|)$ time by checking the transition function of $\mathcal{T}_M(S)/\pi$. Also, observe that one can merge all the paths containing $p, q \in Q(\mathcal{T}_M(S))$ in $O(n|A|)$ time by performing mergers in a BFS manner (starting from the merger of p and q) and noting that $(n-1)$ upper-bounds the possible total number of mergers. From

16. Thus, no valid T/L pair exists for the class of linear-time computable functions under policy-trajectory observations.
17. This consideration is a long-standing open question in grammatical inference. See (de la Higuera, 2006, Problem 1).

now on, we will treat $|A|$ as a constant. Let us define the **similarity score** $s(p, q)$ as 0, in case $\texttt{merge}(p, q)$ produces an invalid merging, and as r, in case $\texttt{merge}(p, q)$ produces a deterministic quotient automaton $\mathcal{T}_M(S)/\pi$ with $n - r$ states. In other words, $s(p, q)$ measures the similarity of the transition paths going through p and q. Now, since we can both perform $\texttt{merge}(p, q)$ and test the validity of that merger in linear time, we can also compute the similarity score $s(p, q)$ in $O(n)$ time.

The state-merging algorithms of Oncina and Garcia (1992); Vaandrager et al. (2022), which solve the $\mathcal{P}_A^B$–LP under an unrestricted input source $I = A^\star$, attempt to incrementally construct a sub-tree automaton of $\mathcal{T}_M(S)$ by merging indistinguishable states. This approach works when the input is unrestricted, as every state in the ground truth automaton is eventually observed with distinguishing strings. With a restricted input source, however, most states in $\mathcal{T}_M(S)$ remain indistinguishable, making naive merging unreliable and prone to errors. Instead, we propose the **Maximum-Similarity Merging** (MSM) algorithm, which merges states only when sufficient evidence supports the merger. Given a partial automaton $M \in \widetilde{\Phi}_A^B$, MSM iteratively merges the state pair (p, q) with the highest nonzero similarity score $s(p, q)$. The process continues until all similarity scores drop to zero, at which point MSM returns the generated quotient automaton. The naive implementation of MSM runs in $O(n^4)$ time: Finding the highest similarity score among n states requires $O(n^3)$ time, and at most $n - 1$ mergers occur.

Unfortunately, the greedy MSM strategy provably cannot learn $\mathcal{P}_A^B$ in the limit, even under unrestricted input observations. However, we conjecture that for many natural orderings $w : \mathbb{N} \twoheadrightarrow I$ of the input set, MSM successfully learns a correct FST representation in the limit. We leave as an open question the characterization of the set of favorable orderings: $W_M^I = \{w : \mathbb{N} \twoheadrightarrow I \mid$ MSM learns $M \in \Phi_A^B$ in the limit$\}$. Finally, we show that MSM enjoys wide applicability. Namely, MSM can learn the class of all recursive functions in the limit when policy-trajectory observations originate from a restricted class of TMs. The proof of this result appears in Appendix D.

Theorem 21 *For every $f \in \mathcal{C}_{\Sigma^\star}$, there exists $\Gamma_f \supset \Sigma$ and a TM $T_f \in \mathrm{T}_\Sigma^{\Gamma_f}$ such that MSM learns f in the limit from policy-trajectory observations of T_f and from a polynomial number of samples .*

6. Conclusion

In this paper, we extend the learning-in-the-limit framework to include observations of the computational process under a restricted input source. By formally modeling different types of computational information—from simple runtime estimates to full behavioral trajectories—this work establishes new theoretical bounds on the limits of learning algorithms and offers insights into how intelligent agents can learn complex computational tasks from observing the process, not just the outcome. While classical input-output observations prevent learning the class of computable functions in the limit, we prove that both rough runtime estimates and policy-trajectory observations remove this learning barrier. Furthermore, by reducing the learning of general recursive functions with policy-trajectory observations to the learning of rational functions with input-output observations, we open the way for the application of FSA-identification algorithms to the more general problem of learning Turing machines. Open questions remain regarding the existence of polynomial-size characteristic sets for learning rational functions under restricted input observations and the impact of input orderings on the correctness of greedy state-merging algorithms. Future research should explore broader computational paradigms, including agents interacting with reactive environments.

Acknowledgments

This work was partially funded by an unrestricted gift from Google and by the Swiss National Science Foundation (grant number 212111).

References

Scott Aaronson. Guest column: Np-complete problems and physical reality. *ACM Sigact News*, 36 (1):30–52, 2005.

Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Alonzo Church. A note on the entscheidungsproblem. *The journal of symbolic logic*, 1(1):40–41, 1936.

B. Jack Copeland. The Church-Turing Thesis. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, 2023a.

B. Jack Copeland. The rise and fall of the entscheidungsproblem. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, 2023b.

André Correia and Luís A Alexandre. A survey of demonstration learning. *Robotics and Autonomous Systems*, 182:104812, 2024.

Colin De La Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138, 1997.

Colin de la Higuera. Ten open problems in grammatical inference. In *International Colloquium on Grammatical Inference*, pages 32–44. Springer, 2006.

Pierre Dupont. Incremental regular inference. In *International Colloquium on Grammatical Inference*, pages 222–237. Springer, 1996.

E Mark Gold. Language identification in the limit. *Information and control*, 10(5):447–474, 1967.

E Mark Gold. Complexity of automaton identification from given data. *Information and control*, 37 (3):302–320, 1978.

Sally A Goldman and H David Mathias. Teaching a smart learner. In *Proceedings of the sixth annual conference on computational learning theory*, pages 67–76, 1993.

David Hilbert and Wilhelm Ackermann. *Grundzüge der Theoretischen Logik*. J. Springer, Berlin, 1928.

John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In *Theory of machines and computations*, pages 189–196. Elsevier, 1971.

Jon Kleinberg and Sendhil Mullainathan. Language generation in the limit. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Eran Malach. Auto-regressive next-token predictors are universal learners. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

José Oncina and Pedro Garcia. Inferring regular languages in polynomial updated time. In *Pattern recognition and image analysis: selected papers from the IVth Spanish Symposium*, pages 49–61. World Scientific, 1992.

Christos H Papadimitriou. *Computational complexity*. Pearson, 1993.

Rajesh Parekh and Vasant Honavar. Learning dfa from simple examples. *Machine Learning*, 44: 9–35, 2001.

Gualtiero Piccinini and Corey Maley. Computation in Physical Systems. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2021 edition, 2021.

Leonard Pitt. Inductive inference, dfas, and computational complexity. In *International Workshop on Analogical and Inductive Inference*, pages 18–44. Springer, 1989.

Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

Michael Sipser. *Introduction to the Theory of Computation*, volume 27. ACM New York, NY, USA, 1996.

Bernhard Steffen, Falk Howar, and Maik Merten. *Introduction to Active Automata Learning from a Practical Perspective, bookTitle=Formal Methods for Eternal Networked Software Systems: 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2011, Bertinoro, Italy, June 13-18, 2011. Advanced Lectures*, pages 256–296. Springer Berlin Heidelberg, address=Berlin, Heidelberg, 2011. ISBN 9783642214554.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

Alan Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58(345-363):5, 1936.

Frits Vaandrager, Bharat Garhewal, Jurriaan Rot, and Thorsten Wißmann. A new approach for active automata learning based on apartness. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 223–243. Springer, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Ludwig Wittgenstein. Tractatus logico-philosophicus. *Filosoficky Casopis*, 52:336–341, 1922.

Organization of the Appendix

Appendix A details the formal framework. Appendix B contains the proof for the input-output-learning setting (Section 3). Appendix C provides the proof for learning from time-bound observations (Section 4). Appendix D includes the proofs related to learning from policy-trajectory observations (Section 5).

Appendix A. Supplement to the Formal Framework

A.1. Finite-State Transducers

Two properties make FSTs elegant objects of study: the existence of a canonical minimal FST given by the Generalized Myhill-Nerode Theorem and the existence of a fast state-distinguishing procedure given by Hopcroft's algorithm (see (Steffen et al., 2011) and (Hopcroft, 1971), respectively). We will briefly consider both.

Theorem 22 (Generalized Myhill-Nerode) *A function $\chi : A^+ \to B$ is rational if and only if χ is the semantics of some FST. Moreover, whenever χ is rational, there exists a unique min-state FST M_χ with semantics given by χ and having the following description: $M = (Q, A, B, \delta, \gamma, q_0)$, where $Q = \{[u]_\chi : u \in A^\star\}$, $\delta([u]_\chi, a) = [u \cdot a]_\chi$, $\forall u \in A^\star, a \in A$, $\gamma([u]_\chi, a) = \chi(u \cdot a)$, and $q_0 = [\lambda]_\chi$.*

Hence, every rational χ defines a unique canonical min-state transducer M_χ with semantics χ and precisely $|\chi|$ states. And any other FST with these two properties is necessarily isomorphic to M_χ.

Next, we consider the question of distinguishing FST states. Recall that we say that two states $p, q \in Q$ are apart and write $p\#q$ whenever there exists a distinguishing string $\sigma \in A^+$ s.t. $\gamma(p, \sigma) \neq \gamma(q, \sigma)$. Interestingly, we can quickly test for state apartness with Hopcroft's algorithm.

Theorem 23 (Hopcroft's Algorithm) *Given an FST $M = (Q, A, B, \delta, \gamma, q_0) \in \Phi_A^B$ with n states, there exists an $O(n \log n)$-time algorithm which converts M into the min-state FST M_{γ_M}. Moreover, Hopcroft's algorithm finds distinguishing strings of length at most $n - 1$ for each $p, q \in Q$ s.t. $p\#q$. Finally, given another FST M' with n' states, Hopcroft's algorithm determines in $O((n+n')\log(n+n'))$ time if $\gamma_M = \gamma_{M'}$ and produce a distinguishing string of length at most $n+n'-1$ if not. Finally, Hopcroft's algorithm can test state apartness in $O(n \log n)$ time even for partial FST $M \in \widetilde{\Phi}_A^B$.*

A.2. Ordering of Turing Machines

We record for future reference in the subsequent proofs the following simple proposition which establishes a computable enumeration of the family of Turing machines T_Σ^Γ.

Proposition 24 (Enumeration of TMs) *There exists a computable bijection $\theta : \mathbb{N} \to \mathrm{T}_\Sigma^\Gamma$ such that for all $n \in \mathbb{N}$, the number of states in $\theta(n)$ increases monotonically with n. Moreover, $|\theta(n)| = o(\log n)$, and there exists a polynomial-time algorithm that, given a natural number n in binary, outputs a binary representation of $\theta(n)$.*

A.3. Encodings and Simulations

We now dedicate a bit of space to highlight an interesting paradox that arises if one takes the Church-Turing Thesis (CTT) seriously. As we discussed in the introduction, the CTT argues that the formal predicate *Turing-computable* completely captures the intuitive notion of computability. Clearly, since not all computable functions operate over domains of strings (e.g., running a cellular automaton for n steps), TMs cannot *implement* every computable function. Instead, the CTT claims that TMs can *simulate* every computable function. In formal terms, according to the CTT, for any *computable* function $f : \mathcal{D} \to \mathcal{D}$ on an enumerable domain $\mathcal{D}$ and any *reasonable* injective encoding $\varphi : \mathcal{D} \hookrightarrow \Sigma^\star$, there exists a TM $T \in \mathrm{T}_\Sigma^\Gamma$ such that $\varphi^{-1} \circ T \circ \varphi = f$.[18] Hence, f is computable if and only if the following diagram commutes for some $T \in \mathrm{T}_\Sigma^\Gamma$.

$$\begin{tikzcd}
\mathcal{D} \arrow[r, "f"] \arrow[d, "\varphi"'] & \mathcal{D} \arrow[d, "\varphi"] \\
\Sigma^\star \arrow[r, "T"] & \Sigma^\star
\end{tikzcd}$$

Now, we need to specify what *reasonable* means. Clearly, we cannot leave φ unrestricted. Indeed, suppose $\mathcal{D} = \mathbb{N}_0$ and let $f(n) = \mathbf{1}_\mathcal{P}(n)$, where $\mathbf{1}_\mathcal{P}$ denotes the indicator function for the set of prime numbers $\mathcal{P}$. Also, let $H = \{\langle T, w\rangle_\Sigma | T \in \mathrm{T}_\Sigma^\Gamma, w \in \Sigma^\star, \text{ and T(w) halts}\}$, where $\langle T, w\rangle_\Sigma$ denotes some encoding of the pair (T, w) into $\Sigma^\star$. Now, if φ bijectively maps $\mathcal{P}$ into H and $\mathbb{N}_0 \setminus \mathcal{P}$ into $\Sigma^\star \setminus H$, then since f is intuitively computable, there must exist a TM T deciding the halting problem – a contradiction. Hence, an encoding should not define an intuitively uncomputable procedure. Therefore, the predicate *reasonable* must also imply *computable* – a circumstance which pushes us into an infinite regress since our definition of computability relies on reasonable encodings.

We believe that this infinite regress should not cause us to abandon the CTT. Indeed, TMs *do* seem capable of simulating any intuitively computable process under reasonable encodings. However, we also think that reasonable encodings must come from computable procedures. How can we break out of the infinite regress and define reasonability without any reference to computability? The way we propose goes through the old saying "The limits of my language mean the limits of my world." (Wittgenstein, 1922, Proposition 5.6). In a sense, we cannot meaningfully talk about the computable function f or the symbolic domain $\mathcal{D}$ without first embedding f and $\mathcal{D}$ into first-order logic (FOL). Hence, if we denote with Ξ the alphabet of FOL, we can think of some encoding $E : \mathcal{D} \to \Xi$, mapping w to a finite FOL description of $\langle w\rangle_{\mathrm{FOL}}$, as provided to us from the start. Then, we can define an encoding $\varphi : \mathcal{D} \to \Sigma^\star$ as reasonable if and only if φ constitutes a composition of E with some TM taking strings from $\Xi^\star$ to $\Sigma^\star$.

A.4. Tasks as Computable Functions

We use the words "function" and "task" interchangeably in order to emphasize the fact that all everyday reasoning or mechanical tasks represent computable functions since a TM can simulate these tasks with an appropriate encoding. For example, brewing a cup of tea specifies the following task `BrewTea` $: \mathcal{K} \to \mathcal{K}$, where $\mathcal{K}$ denotes the symbolic domain of the kitchen. More precisely, we could think of a kitchen-state $\kappa \in \mathcal{K}$ as a function $\kappa :$ `KitchenLattice` $\to$ `Colors` assigning colors to the voxels in the kitchen space. Hence, upon receiving as input a kitchen-state κ, the

18. Notice that if φ is intuitively computable, then so is φ^{-1} since both $\mathcal{D}$ and $\Sigma^\star$ are enumerable.

computational agent implementing `BrewTea` goes through a series of kitchen-states and arrives at some $\texttt{BrewTea}(\kappa)$ with voxels that indicate the presence of a freshly brewed cup of tea. Now, let us consider an arbitrary computable encoding $\varphi : \mathcal{K} \to \Sigma^\star$ which injectively maps kitchen-states to TM-readable $\Sigma^\star$ representations. Then, invoking the CTT, there exists a TM $T_{\texttt{BrewTea}}$ that simulates the computational path of `BrewTea` and outputs $\varphi(\texttt{BrewTea}(\kappa))$ when prompted with $\varphi(\kappa)$. In other words, we make the point that a robot with sensors encoding the environment in $\Sigma^\star$ could learn how to brew a cup of tea in the limit by running the presented algorithms.

A.5. Characteristic Sets

Lemma 25 (Distinguishability) *Let $\mathcal{L}$ solve the $(\mathcal{M}, \alpha)$–LP. Suppose that characteristic sets exist relative to $\mathcal{L}, \mathcal{M}, \alpha,$ and $I \subseteq \mathcal{D}$. Let $M, M' \in \mathcal{M}$ compute two distinct functions when restricted to I. Then, there exists $x \in S_M \cup S_{M'}$ such that either $x \notin D_M \cap D_{M'}$ or $\eta_M(x) \neq \eta_{M'}(x)$.*

Proof Assume the contrary: $\forall x \in S_M \cup S_{M'},\ \eta_M(x) = \eta_{M'}(x)$ and $x \in D_M \cap D_{M'}$. Let $S = S_M \cup S_{M'}$. Then, $E_M(S) = E_{M'}(S)$. Now, by the characteristic property of S_M, $\mathcal{L}(E_M(S)) = R$ which computes the same function as M. Analogously for $S_{M'}$, R must compute the same function as M' – contradiction. Hence, there exists an input distinguishing the α-observable behavior of M and M' on the union of their characteristic sets. ■

Appendix B. IOO–earning Proofs

Theorem 12 (Time-Restricted IOO-Learning) *Assuming the q-ECTT (4) holds, for an enumerable set $\mathcal{D}$ and a computable monotonically increasing function $Q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, there exists an algorithm which learns the $Q-$time-bounded general recursive functions over $\mathcal{D}$ from input-output observations and restricted inputs. Formally, there exists an algorithm solving the $\mathbf{Q}(\mathcal{M}_\mathcal{D}, \mathcal{D})$–LP.*

Proof We want to show that for all $f \in \mathbf{Q}(\mathcal{M}_\mathcal{D}, \mathcal{D})$, all input sources $I \subseteq \Sigma^\star$, and all surjective orderings $w : \mathbb{N} \twoheadrightarrow I \cap D_f$ of the example set $E_f(I \cap D_f) = \{(x, f(x)) : x \in I \cap D_f\}$, after some finite time $t^\star = t^\star(f, I, w)$, our learning algorithm $\mathcal{L}$ will converge to a correct representation of f. In other words, we want to show that $\forall t \geq t^\star,\ h_{t|I \cap D_f} = f_{|I \cap D_f}$, where h_t is the hypothesis generated by $\mathcal{L}$ at time t.

Let us fix a ground truth $f \in \mathbf{Q}(\mathcal{M}_\mathcal{D}, \mathcal{D})$, an input source $I \subseteq \Sigma^\star$, and a surjective ordering of the inputs $w : \mathbb{N} \twoheadrightarrow I \cap D_f$ s.t. $I \cap D_f = \{w_t\}_{t=1}^\infty$. We will demonstrate how $\mathcal{L}$ learns f in the limit.

First, since $f \in \mathbf{Q}(\mathcal{M}_\mathcal{D}, \mathcal{D})$, there exists $c_1 \in \mathbb{N}$ s.t. $f \in \mathbf{TIME}^\mathcal{M}_\mathcal{D}(Q(c_1, n))$. Hence, there exists a computational model $M \in \mathcal{M}_\mathcal{D}$ which computes f in $c_2 \cdot Q(c, n)$ steps for some constant $c_2 \in \mathbb{N}$. Now, let $\mathcal{L}$ use some reasonable encoding $\varphi : \mathcal{D} \to \Sigma^\star$ for which the q-ECTT kicks in. Using the q-ECTT, there exists a TM $\widetilde{T} \in \mathrm{T}^\Gamma_\Sigma$ which simulates m computation steps of M in at most $q(c_3, m)$ steps for some $c_3 \in \mathbb{N}$. Hence, for every $x \in I \cap D_f$, $\widetilde{T}$ starts with $\varphi(x)$ on the tape and halts with $\varphi(f(x))$ after $q(c_3, c_2 \cdot Q(c_1, |x|))$ steps. Let $c = \max(c_1, c_2, c_3)$. Then, $\widetilde{T}$ computes f in $q(c, c \cdot Q(c, n))$ time.

Second, $\mathcal{L}$ will use the ordering algorithm ρ from Proposition 24 as a subroutine to generate TM representations from T^Γ_Σ, sorting them by increasing number of states. Let $T_1, T_2, T_3, \ldots$ be the ordering of T^Γ_Σ due to ρ and let us consider the set

$$H_f = \{T \in \mathrm{T}^\Gamma_\Sigma : \exists C \in \mathbb{N} \text{ s.t. } T \text{ computes } f_{|I \cap D_f} \text{ within } q(C, C \cdot Q(C, n)) \text{ steps}\}.$$

Clearly, H_f is nonempty since $\widetilde{T} \in H_f$. Let K be the smallest number assigned by ρ to a TM in H_f and let $c_K \in \mathbb{N}$ be the smallest constant for which T_K computes $f_{|I}$ within $q(c_K, c_K \cdot Q(c_K, n))$ steps. We will prove that $\mathcal{L}$ converges to T_K after finitely many observations.

Now, let us describe the algorithmic strategy L employs on the set of inputs $S_t = \{w_1, \dots, w_t\}$ at time $t \in \mathbb{N}$. The learner $\mathcal{L}$ will maintain a counter $C \in \mathbb{N}$, s.t. initially $C = 1$. Given the current value of C, for each $w \in S_t$ and for each TM $T \in \{T_1, \dots, T_C\}$, $\mathcal{L}$ will simulate $q(C, C \cdot Q(C, |w|))$ steps of the computation of T on w. This simulation is possible, since ρ, q, and Q are all computable functions known to $\mathcal{L}$. We will say that a TM T_i is $(\mathbf{t}, \mathbf{C})$**-valid** if $i \le C$ and T_i correctly computes $f(w)$ in time $q(C, C \cdot Q(C, |w|))$, $\forall w \in S_t$. Let $V_{t,C}$ be the set of all (t, C)-valid TMs and note that $\mathcal{L}$ computes $V_{t,C}$ through the enacted simulations. Now, if $V_{t,C}$ is nonempty, $\mathcal{L}$ returns the representation of the TM in $V_{t,C}$ with the smallest ρ-number. Otherwise, $\mathcal{L}$ continues the same procedure with $C \leftarrow C + 1$.

Note that $V_{t+1,C} \subseteq V_{t,C} \subseteq V_{t,C+1}$. Also note that once $C \ge C_K = \max(K, c_K)$, $T_K \in V_{t,C}$, $\forall t \in \mathbb{N}$. Hence, $\mathcal{L}$ follows a finite procedure and outputs a representation consistent with the example set $E_f(S_t)$ at each time $t \in \mathbb{N}$.

Finally, observe that since T_K is the smallest member of H_f according to the ρ-ordering, then for every $i < K$, there exists an input $w_{t(i)}$ such that either T_i takes more than $q(C_K, C_K \cdot Q(C_K, |w|))$ steps to finish computing or $T_i(w_{t(i)}) \ne f(w_{t(i)})$. In other words, $T_i \notin V_{t(i),C_K}$. Hence, let us take $t^\star = \max(t(1), \dots, t(K-1))$. Then, T_K becomes the smallest ρ-ordered member of $V_{t^\star,C_K}$ and gets selected by $\mathcal{L}$ for every $t \ge t^\star$.

Three remarks. First, $\mathcal{L}$ learns a min-state TM which computes $f_{|I \cap D_f}$ in the limit since ρ orders the elements of T_Σ^Γ by increasing number of states. Through a different computable ordering, we could have selected for some other characteristic since $\mathcal{L}$ will always learn the first consistent representation with $f_{|I}$.

Second, if we wanted, we could have made $\mathcal{L}(E_f(S_t))$ run in polynomial time in the size of $E_f(S_t)$ through the following cheap trick: Let $\mathcal{L}$ follow the above algorithm, slowly discarding TM representations by checking consistency with each $w \in S_t$. However, when $\mathcal{L}$ reaches some pre-specified $\texttt{poly}(|E_f(S_t)|)$ number of steps, then we modify $\mathcal{L}$ to return the first non-discarded representation from the ρ-ordering. Since, as the observation size grows, more computational resources become available, at some finite point in time $\mathcal{L}$ will have discarded all T_i with $i < K$ through the samples $w_{t(i)}$. Hence, $\mathcal{L}$ runs in polynomial time in the observation size and learns T_K in the limit. However, $\mathcal{L}$ no longer provides a representation consistent with the observations at every time step.

Third, we could have also made $\mathcal{L}$ incremental in the sense that $\mathcal{L}$ keeps track of the smallest index C_t of a model not discarded at time $t \in \mathbb{N}$. Since $V_{t+1,C_t-1} \subseteq V_{t,C_t-1} = \emptyset$, $\mathcal{L}$ should start with $C = C_t$ at time $t + 1$. The learner $\mathcal{L}$ could also keep track of which representations T_i got discarded due to wrong and not slow computation. ■

Appendix C. TBO–Learning Proofs

Theorem 14 (Universal Learning with TBOs) *Assuming the q-ECTT (4) holds, for an enumerable set $\mathcal{D}$, there exists an algorithm for learning the class of general recursive functions $G_{\mathcal{D}}$ from time-bound observations of arbitrary computational models and input sources. Formally, there exists an algorithm solving the $(\mathcal{M}_{\mathcal{D}}, \alpha_{\mathrm{TB}})$–LP.*

Proof We want to show that for all $f \in G_{\mathcal{D}}$, all models $M \in \mathcal{M}_{\mathcal{D}}$ computing f, all input sources $I \subseteq \mathcal{D}$, and all surjective orderings $w : \mathbb{N} \twoheadrightarrow I \cap D_f$ of the example set $E_M(I) = \{(x, f(x), \tau(M, x)) : x \in I \cap D_f\}$, after some finite time $t^\star = t^\star(M, I, w)$, our learning algorithm $\mathcal{L}$ will converge to a correct representation of f. In other words, we want to show that $\forall t \geq t^\star$, $h_{t|I \cap D_f} = f_{|I \cap D_f}$, where h_t is the hypothesis generated by $\mathcal{L}$ at time t.

Let us fix a ground truth couple $(f, M) \in G_{\mathcal{D}} \times \mathcal{M}_{\mathcal{D}}$, an input source $I \subseteq \mathcal{D}$, and a surjective ordering of the inputs $w : \mathbb{N} \twoheadrightarrow I \cap D_f$ s.t. $I \cap D_f = \{w_t\}_{t=1}^{\infty}$. We will demonstrate how $\mathcal{L}$ learns f in the limit.

Now, if M takes $t_M(x)$ steps to compute $f(x)$, then $t_M(x) \leq p\alpha_{\mathrm{TB}}(M, x)$ for some constant $p \in \mathbb{N}$. Let $\mathcal{L}$ use some reasonable encoding $\varphi : \mathcal{D} \to \Sigma^\star$ for which the q-ECTT kicks in. Invoking the q-ECTT, there exists a TM $\widetilde{T} \in \mathrm{T}_{\Sigma}^{\Gamma}$ which simulates m computation steps of M in at most $q(c', m)$ steps for some $c' \in \mathbb{N}$. In other words, on input $\varphi(x)$, $\widetilde{T}$ takes at most $q(c, \lceil p\alpha_{\mathrm{TB}}(M, x)\rceil)$ steps to compute $\varphi(f(x))$ for every $x \in \mathcal{D}$. Let $c = \max(c', \lceil p \rceil)$. Then, $\widetilde{T}$ simulates $f(x)$ through φ in $q(c, c\alpha_{\mathrm{TB}}(M, x))$ time.

The proof now proceeds identically to the proof of Theorem 12. Hence, we find a polynomial-time (w.r.t the observation size) incremental learning algorithm $\mathcal{L}$ which learns in the limit a minimal-state TM simulating $f_{|I \cap D_f}$ through the encoding φ. Different encodings will lead to different solutions in the limit. ■

Appendix D. PTO–Learning Proofs

Theorem 16 (Recursive-to-Rational Reduction) *Learning the class of general recursive functions in the limit from PTOs reduces to learning the class of rational functions in the limit from IOOs. Formally, an algorithm solving the $\mathcal{P}_A^B$–LP with $A = \Gamma$, $B = \Gamma \times \{L, R, S\}$ will solve the $(\mathrm{T}_{\Sigma}^{\Gamma}, \alpha_{\mathrm{PT}})$–LP. In particular, there exists a learning-by-enumeration algorithm for the $\mathcal{P}_A^B$–LP.*

Proof We already covered the reduction part of the lemma. For the learnability part, we give a learning-by-enumeration algorithm $\mathcal{L}$. Let $\chi \in \mathcal{P}_A^B$ be the ground truth, let $I \subseteq A^\star$ be the input source, let $w : \mathbb{N} \twoheadrightarrow A^\star$ be a surjective ordering of the inputs, and let $S_t = \{w_1, \ldots, w_t\} \subseteq I$ be the sample set at time $t \in \mathbb{N}$. Since the set Φ_A^B of FSTs computing $\mathcal{P}_A^B$ is recursive, let $\{M_n\}_{n=1}^{\infty}$ be some computable ordering of the automata in Φ_A^B used by $\mathcal{L}$. Let K be the index of the first FST in that ordering which generates a rational function agreeing with χ on I. That is, if χ_i is the rational function due to M_i for every $i \geq 1$, then χ_K is the one with the smallest index s.t. $\chi_{K|I} = \chi_{|I}$. Clearly, such an index exists since some automaton M_s computes χ.

Now, at observation step $t \in \mathbb{N}$, $\mathcal{L}$ goes through the automata $\{M_n\}_{n=1}^{\infty}$ in order and checks for consistency with χ on S_t. We set $\mathcal{L}$ to return the first encountered FST which agrees with χ on S_t. Hence, at each step $\mathcal{L}$ returns a consistent representation. Also, after finitely many observations, $\mathcal{L}$ will converge to M_K. Indeed, since M_K is the first consistent FST on I, for each $i < K$, there exists an input $w_{t(i)}$ s.t. $M_i(w_{t(i)}) \neq \chi(w_{t(i)})$. Hence, after time $t^\star = \max(t(1), \ldots, t(K-1))$, M_K will be the first consistent representation in the ordering with the sample set.

The same remarks given at the end of the proof of Theorem 12 apply here. First, the learner $\mathcal{L}$ always learns-in-the-limit the first representation from the ordering consistent with $\chi_{|I}$. Hence, if the ordering lists the automata by increasing size, $\mathcal{L}$ will output the smallest solution. Second, we could make $\mathcal{L}$ run in polynomial-time with respect to the size of the observations $E_\chi(S_t) = \{(w, \chi(w)) :$

$w \in S_t\}$ while not always producing a consistent representation. Third, we could also make $\mathcal{L}$ incremental in the sense of remembering discarded representations from the previous time step. ■

Functions ← Behaviors ← Representations. For each function $f \in G_{\Sigma^\star}$ there exists an infinite set of TMs $\mathcal{M}_f \subset \mathrm{T}_\Sigma^\Gamma$ computing f.[19] Thus, for any sample $S \subseteq I$, one can observe a multitude of tape behaviors $B_f(S) = \{M[S] : M \in \mathcal{M}_f\}$. Moreover, each $M[S] \in B_f(S)$ generates an infinite equivalence class $[M]_S$ of TMs with the same tape behavior.[20] In our $(\mathrm{T}_\Sigma^\Gamma, \alpha_{\mathrm{PT}})$–LP framework, $\mathcal{L}$ observes $M[S]$ but need not output a TM from $[M]_S$. Indeed, behavior observations should only aid the learning of f and not become the object of learning. Ideally, the size of the smallest TM in $[M]_I$—the set of consistent representations in the limit—should determine the learning complexity.

Theorem 17 (Unbounded Characteristic Sets) *Fix a computable bounding function $\beta : \mathrm{T}_\Sigma^\Gamma \to \mathbb{N}$ and let $I = \Sigma^\star$. Then, for α_{PT}-observations over the input source I, no algorithm achieves characteristic sets for T_Σ^Γ with β-bounded mass. Furthermore, for α_{PT}-observations over I, no algorithm for learning the class T_Σ^Γ with computable characteristic sets exists.*

Proof If $L_\Sigma \subset \mathrm{T}_\Sigma^\Gamma$ denotes the class of linear-time running halting TMs, we will prove the stronger statement that there cannot exist characteristic sets of β-bounded mass for L_Σ under α_{PT}-observations. We will argue by contradiction. Suppose that for the algorithm $\mathcal{L}$ learning $\mathcal{F}(L_\Sigma)$ in the limit from α_{PT}-observation of L_Σ, there exist β-bounded characteristic sets such that $\|S_T\| \le \beta(T), \forall T \in L_\Sigma$. Then, we will show the existence of a TM solving the Halting Problem.

For a TM $T \in \mathrm{T}_\Sigma^\Gamma$ and $w \in \Sigma^\star$, we construct the pair of TMs $Y_{T,w}$ and $N_{T,w}$ specified as follows. On input $x \in \Sigma^\star$, $Y_{T,w}$ and $N_{T,w}$ both simulate the execution of $T(w)$ for $|x|$ steps, and if T halts, $Y_{T,w}$ outputs 1 and halts and $N_{T,w}$ outputs 0 and halts. If T does not halt on w for $|x|$ steps, then both $Y_{T,w}$ and $N_{T,w}$ halt immediately, having the exact same tape behavior $Y_{T,w}[x] = N_{T,w}[x]$. Clearly, $\{Y_{T,w}, N_{T,w} : T \in \mathrm{T}_\Sigma^\Gamma, w \in \Sigma^\star\} \subset L_\Sigma$. Hence, $\forall T \in \mathrm{T}_\Sigma^\Gamma \forall w \in \Sigma^\star$, $\|S_{Y_{T,w}}\| \le \beta(Y_{T,w})$ and $\|S_{N_{T,w}}\| \le \beta(N_{T,w})$.

For every $T \in \mathrm{T}_\Sigma^\Gamma$ and for every $w \in \Sigma^\star$, let $\langle T, w\rangle$ denote the encoding of the pair (T, w) in $\Sigma^\star$. Now, we define the following TM $H \in \mathrm{T}_\Sigma^\Gamma$ solving the Halting Problem. Upon receiving $\langle T, w\rangle$ as input, H constructs representations of the TMs $Y_{T,w}$ and $N_{T,w}$ and runs them on β, thereby producing the bound $B = \max(\beta(Y_{T,w}), \beta(N_{T,w}))$. We know from Theorem 9 that if $Y_{T,w}$ and $N_{T,w}$ compute different functions, then there must exist an $x \in S_{Y_{T,w}} \cup S_{N_{T,w}}$ such that $\eta_{Y_{T,w}}(x) \ne \eta_{N_{T,w}}(x)$. In other words, if T halts, then $Y_{T,w}(x) \ne N_{T,x}(x)$ for some $x \in S_{Y_{T,w}} \cup S_{N_{T,w}}$. If T does not halt, however, then

$$\eta_{Y_{T,w}}(x) = (x, Y_{T,w}(x), Y_{T,w}[x]) = (x, N_{T,w}(x), N_{T,w}[x]) = \eta_{N_{T,w}}(x),\ \forall x \in \Sigma^\star.$$

Hence, H just needs to check if $Y_{T,w}$ and $N_{T,w}$ agree on $S_{Y_{T,w}} \cup S_{N_{T,w}}$ to determine whether T halts on w.

Now, since $\|S_{Y_{T,w}}\|, \|S_{N_{T,w}}\| \le B$, then $\forall x \in S_{Y_{T,w}} \cup S_{N_{T,w}}$, $|x| \le B$. Therefore, $S_{Y_{T,w}} \cup S_{N_{T,w}} \subset \Sigma^B$. Thus, H can simulate $Y_{T,w}$ and $N_{T,w}$ on all inputs from Σ^B to solve the Halting Problem for (T, w) – contradiction.

19. For instance, if T computes f, we can let T_n implement T and then cycle through the output n times before halting.
20. For $T \in \mathrm{T}_\Sigma^\Gamma$, adding dead transitions leads to multiple equivalent representations. In general, $|[T]_{\Sigma^\star}| = \infty$.

Similarly, if there existed a computable function $\mathfrak{C} : \mathrm{T}_\Sigma^\Gamma \to \Sigma^\star$ assigning characteristic sets to representations in T_Σ^Γ for the learner $\mathcal{L}$, then H could have used $\mathfrak{C}$ to construct $S_{Y_{T,w}} \cup S_{N_{T,w}}$ and check for a distinguishing string x which would prove whether T halts on w. ■

The Correct Measure for Learning Efficiency. In light of the proof of Theorem 17, one might argue that learning efficiency should be measured not by the *mass* of characteristic sets $\|S_M\|$ but by their *size* $|S_M|$. Indeed, the proof shows that a learner cannot distinguish between the functions computed by $Y_{T,w}$ and $N_{T,w}$ until the input length $|x|$ is large enough for T to halt. If T requires exponential time in the representation sizes of $Y_{T,w}$ and $N_{T,w}$ to halt on w, it is unreasonable to penalize the learner for needing a long input x to learn. Instead, the burden of hard examples should fall on the teacher. Thus, an efficient learner should require *few*, not necessarily *short*, examples. This raises the open question of whether a polynomial-time algorithm exists for learning $G_{\Sigma^\star}$ in the limit from policy-trajectory observations of T_Σ^Γ with polynomial-size characteristic sets.

Let us define a model class $\mathcal{M}$ as α_{PT}**-restrictively identifiable in polynomial time and samples** (α_{PT}-RIPTS) if there exists a polynomial-time learner $\mathcal{L}$ for which every $M \in \mathcal{M}$ admits a characteristic set with polynomial size $|S_M| = \mathtt{poly}(|M|)$ regardless of the input source I. Hence, due to the reduction established by Theorem 16, a more interesting and general open question becomes if the rational function class $\mathcal{P}_A^B$ is ε-RIPS: i.e., RIPS from input-output observations.

Theorem 21 *For every $f \in \mathcal{C}_{\Sigma^\star}$, there exists $\Gamma_f \supset \Sigma$ and a TM $T_f \in \mathrm{T}_\Sigma^{\Gamma_f}$ such that MSM learns f in the limit from policy-trajectory observations of T_f and from a polynomial number of samples .*

Proof Let $f \in \mathcal{C}_{\Sigma^\star}$. We will prove the existence of an alphabet $\Gamma_f \supset \Sigma$ and a TM $T_f \in \mathrm{T}_\Sigma^{\Gamma_f}$ which computes f and whose tape-behavior observations allow the MSM algorithm to learn f in the limit for any input source $I \subseteq \Sigma^\star$.

Let $\Gamma = \Sigma \cup \{\lambda\}$ and recall Definition 3 and the discussion underneath for the architectural specifics of our class of TMs T_Σ^Γ. In particular, note that the transition function allows for the TM head to stay in-place. As proved in many textbooks on complexity and computability Papadimitriou (1993); Sipser (1996); Arora and Barak (2009), a TM with a transition function requiring constant left or right movement can compute any recursive function on $\Sigma^\star$. Let $T'_f = (Q', \Gamma, \delta', q_0, q_f) \in \mathrm{T}_\Sigma^\Gamma$ compute f and have a transition function that never stays in-place. We will now describe the transformation of T'_f into T_f. Let $Q' = \{q_1, \ldots, q_n\}$. We arbitrarily order the m transitions of T'_f as $\{e_i\}_{i=1}^m$, and if $e_k = q_i \xrightarrow{\sigma:\sigma', D} q_j$, we add a dummy states p_k and substitute e_k in the transition diagram with the path

$$q_i \xrightarrow{\sigma:\gamma_k, S} p_k \xrightarrow{\gamma_k:\sigma', D} q_j,$$

where $D \in \{L, R\}$ and where $\gamma_1, \ldots, \gamma_m \notin \Gamma$. We let $\Gamma_f = \Gamma \cup \{\gamma_1, \ldots, \gamma_m\}$, and we assign some other arbitrary transitions to the states p_k for every symbol in Γ_f to obtain a well-defined transition function δ on the set of states $Q = Q' \cup \{p_k\}_{1 \le k \le m}$. Then, we define T_f as $T_f = (Q, \Gamma_f, \delta, q_0, q_f)$. Clearly, T_f also computes f, but now each computation takes twice as long.

Suppose now that we receive behavior observations from T_f from an input source I. Let $\psi(T_f) \in \Phi_A^B$ correspond to the FST with the same transition diagram as T_f where $A = \Gamma_f$, $B = \Gamma_f \cup \{L, R, S\}$. Moreover, let $M = I(\psi(T_f)) \in \widetilde{\Phi}_A^B$ denote the partial automaton produced from $\psi(T_f)$ by removing the unused transitions when all inputs come from I. We will show that

MSM learns the representation of M in the limit, which proves that MSM learns a hypothesis function h agreeing with f on I.

Let $t^\star$ denote the time at which the sample $S_{t^\star} \subseteq I$ exercises all transition of M. We prove that for $t \geq t^\star$, MSM$(S_t) \equiv_I M$. Let Q_M be the set of states of M and let Q_T be the set of states of the observation tree $\mathcal{T}_M(S_t)$. Also, let $\phi : Q_T \to Q_M$ assign the 'true' value to the tentative states of $\mathcal{T}_M(S_t)$. In other words, if $\mathtt{path}(q)$ corresponds to the input path leading from the root of $\mathcal{T}_M(S_t)$ to $q \in Q_T$, then $\phi(q) = \delta_M(q_0, \mathtt{path}(q))$.

Now, with a slight abuse of notation, let $X = Q_M \cap Q'$ – i.e., the computing states of T'_f – and let $Y = Q_M \cap \{p_i^j\}_{1 \leq i \leq j \leq m}$ – i.e., the dummy states. Note that if $\phi(p), \phi(q) \in X$ and $\phi(p) \neq \phi(q)$, then either there exists a distinguishing string for p and q in $\mathcal{T}_M(S_t)$, or $s(p,q) = 1$. Moreover, if $\phi(p), \phi(q) \in Y$ and $\phi(p) \neq \phi(q)$, then $\mathcal{T}_M(S_t)$, or $s(p,q) = 1$. Hence, first all of the states in $\mathcal{T}_M(S_t)$ corresponding to the same states in X will get merged due to higher similarity. Only after all of the states in $\phi^{-1}(X)$ get correctly merged, will the MSM algorithm consider other mergers which will only lead to equivalent to M automata under the input source I.

Remark. Note that MSM needs at most m samples to learn T_f – one sample to cover each transition. ■