INTERESTINGNESS AS AN INDUCTIVE HEURISTIC FOR FUTURE COMPRESSION PROGRESS

Vincent Herrmann & Jürgen Schmidhuber
The Swiss AI Lab IDSIA/USI/SUPSI
Lugano, Switzerland
King Abdullah University of Science and Technology
Thuwal, Saudi-Arabia

ABSTRACT

One of the bottlenecks on the way towards recursively self-improving systems is the challenge of *interestingness*: the ability to prospectively identify which tasks or data hold the potential for future progress. We formalize interestingness as an inductive heuristic for future compression progress and investigate its predictability using tools from Kolmogorov Complexity and Algorithmic Statistics. By analyzing complexity-runtime profiles under various priors over computable objects, we demonstrate that the *inductive property of interestingness*—the capacity for past compression progress to signal future discovery—is theoretically viable and empirically supported. Expected future progress depends crucially on the recency of the last observed progress. However, this dependency is highly sensitive to the underlying distribution of objects.

1 INTRODUCTION

A common notion of how a general recursively self-improving open-ended intelligence might be achieved is, in very simple terms, as a cycle between two alternating phases (see Figure 1): In the learning phase, a system uses a learning algorithm—e.g., gradient descent or Reinforcement Learning (RL) techniques—to extract patterns and regularities in the available data, or to gain the skills necessary to solve available problems and tasks. In the generation phase, the insights from the learning phase are used to produce novel artifacts (synthetic data, new tasks, hitherto unsolved problems). From these, the system then can again learn new capabilities, ideally leading to never-ending progress. To achieve true open-endedness, this cycle must be fully autonomous. It cannot rely on human-in-the-loop filtering, hand-crafted curricula, or synthetic data created by researchers.

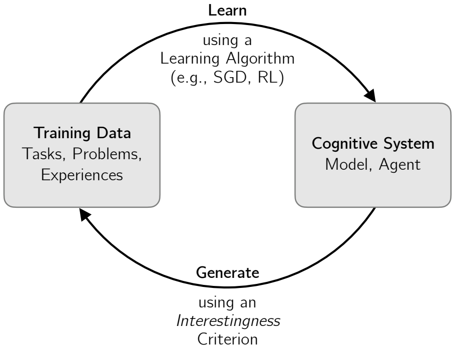

Figure 1: Minimalist depiction of a self-sustaining learning cycle. The Learning phase derives new skills or patterns from data, while the Generation phase creates novel artifacts. To sustain true open-endedness, the generation process must be guided by a criterion that distinguishes learnable structure from noise or already-acquired knowledge.

One example of such a system would be a Large Language Model (LLM) that is iteratively trained on self-generated data, see for example Wang et al. (2023); Zelikman et al. (2024). It is well known that a naive realization of such a setup leads to model collapse (Shumailov et al., 2024; Dohmatob et al., 2024). However, there exist efforts to modify the generation process, or filter the generated data in such a way that continued learning is possible (Lin et al., 2024; Herrmann et al., 2025b;a). Another example would be an RL agent that pursues its own goals in an environment. The goals are chosen according to some intrinsic motivation criterion (Schmidhuber, 1991c; Pathak et al., 2017; Colas et al., 2022).

With the success of large scale models trained with gradient descent and RL techniques, it can be argued that we have a good handle on the learning phase. Developing a generation phase that leads to continued progress, on the other hand, remains an open problem. Using the framing of Hughes et al. (2024), how can we generate artifacts that are both *novel and learnable*? What criterion allows us to select the samples, problems, tasks from which the system can learn something meaningful? How can we distinguish between what is unlearnable, already learned, and so-far unknown but learnable? How can we tell if such an artifact is *interesting*?

To lay out our arguments, this paper proceeds as follows: In Section 2, we review existing interestingness measures, and argue that they are largely post-hoc rather than prospective. This leads us to the question of the predictability of interestingness, and its inductive properties. Section 3 introduces Complexity vs. Runtime profiles. This framework allows us to investigate compression progress over time. In Section 4 we address the induction problem: Can we predict if an object is worth the effort? We analyze how different priors over possible objects—Length, Algorithmic, and Speed—affect our ability to extrapolate future progress. We conclude with a discussion of other kinds of priors that might lead to even more reliable prediction of learnability, and calling for a shift toward *introspective* models that can explicitly assess their own potential for future insight (Section 5).

2 Proposed Criteria of Interestingness

In the following, we refer to the artifacts a cognitive system (the *subject*) learns from as *objects*. This should highlight the fact that interestingness is not necessarily an intrinsic property of an object in isolation, but a relational property between the object's structure and the subject's current internal state.

Count-based, Space Coverage, and Maximum Entropy An object can be deemed interesting if it is *uncommon*, if it has been encountered fewer times than other objects (Sutton, 1990; Bellemare et al., 2016; Tang et al., 2017). In high-dimensional spaces, this requires a density model to quantify novelty via smoothing. If an agent seeks to maximize the entropy of its state-visitation distribution, it will, in the limit, converge toward a policy that encounters all reachable objects with equal probability (Hazan et al., 2019; Mutti, 2023). In certain restricted settings, this is a reasonable measure of interestingness: if it is possible for the cognitive system to encounter all possible objects multiple times, uncommon objects hold the highest surprise in the Shannon sense. But it does not clearly lead to open-ended learning: the system will converge towards the maximum entropy occupancy and then stop learning. If there are more possible objects than can ever be encountered, this measure of interestingness completely depends on some form of coarse-graining or smoothing. In other words, it depends on a *model*.

Prediction Error, Adversarial Criterion This notion moves beyond mere counting by introducing a predictive model with a learning algorithm. Here, interestingness is the inverse of the probability assigned by the model (the "adversarial" criterion). An object that is poorly predicted is considered interesting (Schmidhuber, 1990; 1991a; Pathak et al., 2017). However, this fails to distinguish between epistemic uncertainty (reducible through learning) and aleatoric uncertainty (irreducible randomness). This is the "Noisy TV" problem (Schmidhuber, 2010): a subject using pure prediction error as a reward will get stuck observing a source of pure noise (like a static TV) because it is always unpredictable, yet provides zero learnable structure.

Common approaches to tackle this problem share the insight that an object loses its interestingness once nothing new can be learned about it.

Information Gain, Bayesian Surprise If our model is set up in a probabilistic way, for example as a belief distribution over a set of hypotheses, then we can measure the information gain an object yields. Concretely, this can be quantified as the Kullback-Leibler divergence between the posterior distribution, given the object, and the prior distribution. In more grandiose terms, an object is interesting to the extent it leads to new insights and changes the model's "world view" (Storck et al., 1995; Itti & Baldi, 2005). Once nothing new can be learned from an object, the information gain subsides. Information gain can be measured not just over a set of fixed concrete hypotheses, but also for example over latent variables (Tishby & Zaslavsky, 2015; Herrmann et al., 2025b) or the predictions of existing data (Herrmann et al., 2025a).

Learning Progress, Competence Progress Closely related to gaining information from an object is the notion of learning progress (Schmidhuber, 1991b; Oudeyer et al., 2007; Stout & Barto, 2010). Instead of framing insight in terms of probabilistic inference and differences between posterior and prior, we can ask: Does the object lead to progress on some pre-defined objective function? An object is then interesting to the extent it improves the model's performance. Whether this is a reasonable measure of interestingness depends, of course, on the specific choice of objective function. Certainly not all tasks or objective functions are rich enough to allow open-ended learning. However, for many models—especially the ones we might want to use in the setups we mentioned, such as LLMs or world models (Schmidhuber, 2015; Ha & Schmidhuber, 2018; Bruce et al., 2024)—there is one dominant objective function: the negative log-likelihood (NLL). It can be argued that the success of the NLL as a training objective for models is due to the strong connection between learning and compression (Hutter, 2005; Delétang et al., 2023)

Compression Progress This leads us to the concept of interestingness as compression progress (Schmidhuber, 2009; 2006). An object might be deemed interesting if it allows the model to better losslessly compress all available data. For a full compression of the data, we must account for the size of the model itself: what we measure is the number of bits required to describe the model in addition to the bits required to encode the data given the model. This two-part encoding of objects is the core idea behind the Minimum Description Length (MDL) principle (Wallace & Boulton, 1968; Rissanen, 1978). The compression progress associated with an object is thus the reduction of the total encoding size—the sum of the model complexity and the data residual—once the model has incorporated the object. A rich description of compute-bounded and observer-dependent MDL models has recently been presented by Finzi et al. (2026).

The criteria discussed—information gain, learning progress, and compression progress—are closely related in this context: they all measure shifts in the subject's uncertainty or description length. Information gain provides a probabilistic view of these shifts, learning progress (via NLL) provides an optimization-based view, and compression progress provides an algorithmic view. They provide an *introspective* account, subjective to the system, of how much insight has been gained. While this subjectivity seems at odds with the "objective" measures of Algorithmic Information Theory (AIT) we use later, we view AIT as the theoretical limit that these subjective models strive toward.

All of these criteria implicitly assume that the computational effort has already been spent. In an open-ended setting, this assumption is untenable: the generation phase must decide where to spend effort before the outcome is known. This means an interestingness criterion must be *prospective*. It is not enough to know how much has been learned; we need to know how much we can still learn. The criteria above are largely post-hoc: they quantify the progress made after the computational effort of training has been expended. This makes them insufficient for the "generation phase," where the system must select promising objects from a vast space of possibilities before committing significant resources to them.

In everyday speech, we often call a thing interesting because we have a feeling that it holds further secrets. This is the sense of "interesting" we advocate for: a prospective heuristic that predicts future learning progress based on past experience. This leads to a fundamental question: under what conditions is such a prediction even possible? To investigate this, we can formalize the relationship between past and future progress through the lens of complexity and runtime. It should be mentioned that many more conceptions of interestingness have been proposed. We discuss two further major directions in the Appendix: notions based on social or multi-agent interactions (A.1), and LLMs emulating a human-like judgement of interestingness (A.2).

2.1 Can We Learn How to Predict Learnability?

Before moving to our formal investigation, we must consider whether the prediction of learnability can be treated as just another learning problem. As described, information gain, learning progress, and compression progress measures require running the inference or learning algorithm to evaluate an object. This makes them not directly usable in an open-ended setting: we cannot simply "train on everything" to see what works. Doing so is not only computationally prohibitive but also risks model collapse, as we have mentioned before. Instead, we want to *predict* these measures before investing the effort to learn from them. What we usually do when we have "post-hoc" data and want to predict it is *learning* how to predict it. This is indeed what happens when compression progress is used as an intrinsic reward for an RL agent, such as in a Controller-Model setup (Schmidhuber, 2015; Kompella et al., 2017).

However, a meta-cognitive property like interestingness can be difficult to learn for several reasons: **Non-stationarity**—Interestingness is a moving target. As the subject learns, an object that was once insightful becomes boring, requiring the meta-model to constantly adapt to the subject's changing state. **Data Sparsity and Cost**—Ground-truth labels for progress are expensive to obtain, as they might require executing a full training phase just to evaluate a single data point or task. **Credit Assignment**—In the many batches of data required for training, it is profoundly difficult to identify which specific samples contributed to the reduction in loss. This credit assignment problem makes it hard for a predictor to associate specific object features with the resulting learning progress.

We have made informal arguments why it is difficult to predict future learnability. Essentially, we are asking about the *inductive property of interestingness*: can we infer how much there is yet to learn—and how easily accessible that content is—based on the trajectory of how much we have already learned? To answer this rigorously, we now turn to a formal setting where learning progress is mapped to algorithmic complexity and runtime.

3 Runtime and Complexity Profiles

To analyze interestingness with sufficient generality, we move to an abstract setting using the tools of Algorithmic Information Theory. We consider all data and artifacts as binary strings $x \in \{0,1\}^*$. Let $K(x)$ denote the standard prefix-free Kolmogorov complexity of x: the length of the shortest program p on a universal prefix Turing machine U that halts and outputs x. In the context of open-ended learning, we are not merely interested in the absolute compression limit $K(x)$, but in the compression progress made as a function of computational effort. Let $K^r(x)$ be the time-bounded Kolmogorov complexity: the length of the shortest program that computes x within r steps. While $K^r(x)$ is machine-dependent, it allows us to formalize the learning process as a trajectory of decreasing description lengths over time.

Consider a system that has observed data D and encounters a novel object o. We treat the concatenation $x = Do$ as a single string. If o contains learnable structure relative to D, then $K^r(Do)$ should decrease significantly as r increases, representing the discovery of algorithmic regularities. To study this discovery process, we use the complexity vs. runtime profile (Vereshchagin & Shen, 2016; Bauwens, 2010; Antunes et al., 2017):

$$D_x = \{(r,c) \mid K^r(x) \le c\}.$$

The boundary of this profile, $c(r) = \min\{c \mid (r,c) \in D_x\}$, tracks the shortest description of x available given a runtime budget r. An object is *logically deep* (Bennett, 1988) if this boundary continues to drop significantly even for very large r. Our central question is about the inductive properties of the profile: Does the shape of D_x for $r \le R$ allow us to predict the existence of further "drops" in complexity for $r > R$?

3.1 The Log-Size vs. Complexity Profile and Sophistication

To leverage results from algorithmic statistics, we introduce a related profile: the log-size vs. complexity profile P_x:

$$P_x = \{(i,j) \mid \exists A \text{ s.t. } x \in A, K(A) \le i, \log \#A \le j\},$$

where $A \subseteq \{0,1\}^*$ is a computable finite set of strings. The boundary of P_x represents, for a given set complexity, the optimal two-part description consisting of a model (the set A) and the data-to-model code (the index of x within A). This profile always has a negative slope of at most -1, reflecting the trivial exchange between halving the set size at the cost of adding one bit to the description length of the set. A "drop" in P_x occurs when the boundary falls below the line of slope -1. Such drops indicate that the corresponding model contains additional structural information about the string, beyond the random information of the index. The point where the boundary last meets the -1 slope line before following it indefinitely defines the *sophistication* of x (denoted m_x), representing the complexity of the "best" model for x (Koppel, 1987; Antunes & Fortnow, 2009).

3.2 The $P_x \leftrightarrow D_x$ Correspondence

The relationship between complexity vs. runtime and log-size vs. complexity profiles is bridged by the Busy Beaver function $\mathrm{BB}(k)$—the maximum number of steps a halting k-bit program can run. While BB is uncomputable, it serves as a universal timescale that abstracts away machine dependence. With this re-scaling, P_x and D_x are approximately affine transforms of each other:

$$(i, j) \mapsto (\mathrm{BB}(i), i + j), \tag{1}$$

within logarithmic precision $O(\log |x|)$ (see Vereshchagin & Shen (2016), Theorems 4 and 6). This correspondence is profound: it implies that every drop in P_x (a new non-trivial model with a higher complexity) corresponds to a drop in D_x (a progress in compression when the maximum runtime is increased). Figure 2 illustrates the connection between the two profiles.

3.3 Counting Strings with Specific Profiles

Any log-size vs. complexity profile P has three characteristic values: The log-size value of the leftmost boundary point $n_P = \min\{r|(0, r) \in P\}$ corresponds to the log length of a string with this profile. The complexity value of the rightmost boundary point $k_P = \min\{r|(r, 0) \in P\}$ corresponds to the Kolmogorov complexity of a string with this profile. And the complexity value of the point where the boundary meets the diagonal line leading to $(k_P, 0)$, namely $m_P = \min\{(r, k_P - r) \in P\}$, corresponds to the sophistication of a string following this profile.

We can quantify how many strings have a profile close to P, where P is a valid profile, meaning an upward-closed set whose boundary has a slope of at most -1, or formally, for which $(a, b + c) \in P \implies (a + b, c) \in P \forall a, b, c$. Vereshchagin & Shen (2016) (Theorem 19) limit the minimum number of strings close to P:

$$\#\{x|P_x \approx P\} \geq 2^{k_P - m_P + O(1)}, \tag{2}$$

where the closeness is of order $O(C(P) + \log n_P)$. Theorem 20 of the same work proves the maximum number of strings following this profile: There are at most

$$\#\{x|P_x \approx P\} \leq 2^{k_P - m_P(\epsilon) + 2\epsilon + O(\log n_p)} \tag{3}$$

strings that are ϵ close to profile P. The value of $m_p(\epsilon)$ is similar to m_p.

The first important observation is that for any given valid profile P, there exist strings which have a profile close to it. The term $k_P - m_P$ represents the "random" part of the string—the bits that cannot be further compressed into a structural model. Crucially, the number of strings with a given profile is dominated by this residual randomness. This suggests that while structure is rare, strings with the

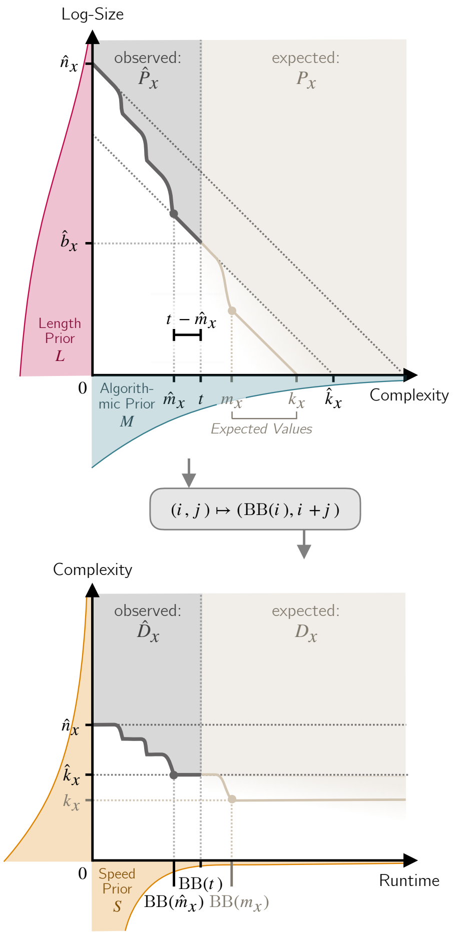

Figure 2: Complexity and Profile Dynamics. (Top) The log-size vs. complexity profile (P_x) of a string x, with the partial profile $\hat{P}_x$ observed up to complexity t. The observed characteristic values are $\hat{m}_x$ (complexity of the last drop) and $\hat{b}_x$ (log-size of the smallest set at complexity t), which determines the current estimated complexity $\hat{k}_x$. We evaluate the expected continuation of $\hat{P}_x$ under various priors; the expected values $\mathbb{E}[m_x]$ and $\mathbb{E}[k_x]$ are functions of the gap $t - \hat{m}_x$, representing the effort expended since the last discovery. (Bottom) The complexity vs. runtime profile (D_x) derived from P_x via the transformation in Eq. 1. Priors are indicated on their relevant axes: the **Length Prior** (string length at complexity 0), the **Algorithmic Prior** (program complexity), and the **Speed Prior** (the product of distributions over complexity and log-runtime, as sketched on the lower axes).

same structural profile are plentiful, differing only in their structureless noise. The value n_P, i.e. the log-length associated with profile P, contributes only logarithmically, compared to k_P and m_P, to the upper bound of the number of strings. The exact shape of the profile—beyond the values n_P, k_P and m_P—only plays a role via its complexity $C(P)$ in the closeness to the lower bound of strings. In the next section, we use these bounds to determine whether observing a partial profile $\hat{P}_x$ up to complexity t allows us to extrapolate the full profile.

4 Quantifying the Inductivity of Interestingness

We now address the core of our position: if we observe a *partial profile* $\hat{P}_x$ up to complexity t, what can we infer about its continuation? Operationally, the boundary of $\hat{P}_x$ corresponds to the best compression achieved so far under a fixed compute budget (via $P_x \leftrightarrow D_x$), not to full access to the profile. Specifically, we look at the expectations for the last drop (m_x) and the ultimate complexity (k_x). To formulate these expectations, we must assume a prior distribution over strings, representing the "world" our learning system inhabits. The concepts from this section are summarized in Figure 2.

Length Prior The simplest prior considers only string length, effectively asking for the probability that a monkey randomly typing bits generates x. To ensure this is a valid semi-measure, we use a prefix-free encoding (e.g., duplicating bits and ending with `01`),

$$L(x) = 2^{-(2|x|+2)}. \tag{4}$$

Using the bounds from Equations (2) and (3), we find that for strings sampled from L, the observed partial profile is highly predictive:

Proposition 4.1. *Let $\hat{P}_x$ be a partial log-size vs. complexity profile, up to complexity t, of a string x sampled from L. Let $\hat{b}_x = \min\{r|(t,r) \in \hat{P}_x\}$ and $\hat{m}_x = \min\{r|(r, \hat{b}_x - r) \in \hat{P}_x\}$. The expected last drop of the complete profile is*

$$\mathbb{E}_{x\sim L}[m_x|\hat{P}_x] \approx \hat{m}_x + (t - \hat{m})2^{-(t-\hat{m}_x-1)},$$

which converges to $\hat{m}$ exponentially fast from above as $t - \hat{m}$ increases.

Let $\hat{k}_x = \min\{j|(t,j) \in \hat{P}_x\} + t$. The expected shortest program computing x has length

$$\mathbb{E}_{x\sim L}[k_x|\hat{P}_x] \approx \hat{k}_x - 2^{-(t-\hat{m}_x-1)},$$

which approaches $\hat{k}$ exponentially fast from below as $t - \hat{m}_x$ increases.

The proof can be found in Appendix C.1. Before we discuss the implications, we look at two more reasonable priors over strings.

Algorithmic Prior We can explain the Algorithmic or Solomonoff prior (Solomonoff, 1964) in similar terms as the length prior. But instead of the monkey typing the string directly, we now take the probability of the monkey typing a program that, when run on a prefix Turing machine U, outputs x. The Algorithmic prior is

$$M(x) = \sum_{p:U(p)=x} 2^{-|p|}. \tag{5}$$

This prior prefers algorithmically simple strings (ones with a short description length) as opposed to the Length Prior, which prefers short strings.

Proposition 4.2. *For $x \sim M$, the expected last drop is*

$$\mathbb{E}_{x\sim M}[m_x|\hat{P}_x] \approx \hat{m}_x + t(\hat{k} - t)2^{-(t-\hat{m}_x+1)}.$$

The expected shortest program computing x has length

$$\mathbb{E}_{x\sim M}[k_x|\hat{P}_x] \approx \hat{k}_x - (\hat{k}_x - \frac{\hat{k}+t}{2})(\hat{k} - t)2^{-(t-\hat{m}_x+1)}.$$

For the proof, please see Appendix C.2. The convergence here is slightly slower than the Length Prior. Because M values simplicity, it allows for the possibility that a significantly more compressed program exists just beyond our current computational horizon, especially if recent progress was made.

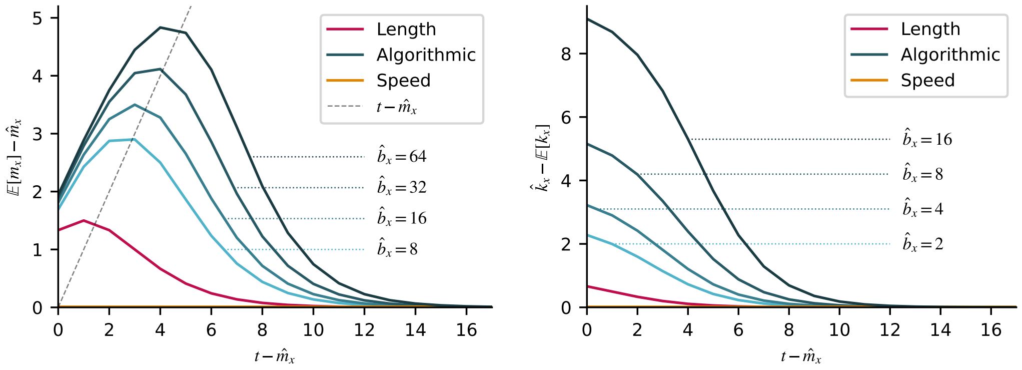

Figure 3: The expected difference between the complexity of the last drop m_x and the last observed drop $\hat{m}_x$ (left), and the expected future compression progress (right). Both are plotted as a function of the difference $t - \hat{m}$, for the Length, Algorithmic, and Speed Prior. The exact curves plotted are Equations (11)&(12), (24)&(25), and (39)&(40) from the Appendix. Only for the Algorithmic Prior, the expectation depends on the value $\hat{b}$: as it increases, so does the expectation of a future drop in description length. Under the Speed Prior, no future drop is expected.

Speed Prior To address the uncomputability of M, the Speed Prior $S(x)$ (Schmidhuber, 2002b) penalizes both program length and runtime. We write $p \to_i x$ iff it takes $2^{i-|p|}$ steps for program p to compute output x. The speed prior the is defined as

$$S(x) = \sum_{i=1}^{\infty} \sum_{p \to_i x} 2^{-(i+|p|)}. \tag{6}$$

It is closely related to Levin Complexity $Kt(x) = \min\{|p| + \log \mathrm{time}(p) : U(p) = x\}$ (Levin, 1984).

Proposition 4.3. *For $x \sim S$ and sufficiently large t, $\mathbb{E}_{x\sim S}[m_x|\hat{P}_x] = \hat{m}_x$ and $\mathbb{E}_{x\sim S}[k_x|\hat{P}_x] = \hat{k}_x$.*

The proof can be found in Appendix C.3. The Speed Prior is the most "conservative". Because it heavily penalizes runtime, it assumes that if a faster way to compress the string existed, it would have been found already. This prior essentially treats the current frontier of compression as the final one.

4.1 Discussion: The Inductive Property of Interestingness

With Equation (1), it is possible to transform the above results from P_x profiles to the compression against effort profile D_x, since the strings—and hence also the numbers of strings—with a certain P profile shapes are the same the ones with the corresponding D profile shapes. Thus, we see that under standard priors, the "Inductive Property of Interestingness" holds a specific mathematical form: past compression progress can be an indicator of future progress, but only if that progress is recent relative to the computational effort expended. As shown in Figure 3, if the gap $t - \hat{m}$ is small, there is an expectation for future drops, at least for the Length and Algorithmic Prior. This may justify investing resources into objects that have recently yielded insight. Conversely, if a large computational gap exists without progress, the probability of a future "aha!" moment vanishes exponentially. Note, however, that this does not mean that it is impossible: as mentioned in Section 3.3, all valid profile shapes are closely followed by some strings. They are just rare under the priors we investigated. But in general, all else being equal, the most promising object for further engagement is the one which showed the most recent progress.

A critical caveat of the $P_x \leftrightarrow D_x$ correspondence is its use of the Busy Beaver function (BB). Because BB grows faster than any computable function, the runtimes discussed here are physically unrealizable. However, BB serves a vital theoretical role: it abstracts away machine dependence, ensuring these results reflect the *intrinsic* algorithmic nature of the objects rather than the specifics of a reference computer. This machine independence is important, since there cannot be any particular "most basic" or "most universal" computer (Müller, 2010). While these findings are theoretical, they provide some conceptual grounding. They suggest that "interestingness" can be a principled heuristic for navigating the space of computable objects. To corroborate our findings, we test the inductive property of interestingness experimentally in three different computationally universal paradigms: Tag machines, cellular automata and a minimalistic programming language (Appendix B). We see that our theoretical results about recent compression progress implying future compression progress robustly hold in these much more practical empirical settings.

5 Conclusion & Future Work

We have argued that *interestingness* is a necessary algorithmic heuristic for autonomous open-ended intelligence. By formalizing interestingness as a prospective assessment of future compression progress, we have grounded the concept in the rigorous framework of Algorithmic Information Theory. Our analysis of priors over computable objects—Length, Algorithmic, and Speed—reveals that the "inductive property of interestingness" is theoretically and empirically viable: past compression progress can indeed signal the potential for future discovery. However, this signal is highly sensitive to the nature of the underlying prior and the recency of the progress.

Especially this reliance on specific priors warrants further investigation. An inversion of our inquiry could lead to further insights: if we take the existence of inductive interestingness—the property that past learning progress reliably predicts future insight—as a given, what must be true about the underlying distribution of objects? This is a fundamental question for recursive self-improvement: how must a "world" be structured to enable never-ending progress? Our previous analysis of universal priors suggests a somewhat limited memory; the probability of future progress was largely determined by the recency of the last drop. However, in structured domains, we intuitively expect that the *entire history* of progress matters. We do not expect long, steady sequences of discovery to terminate abruptly. This property of sustained, multi-level learnability can be characterized as *scale-free* (Barabási & Albert, 1999) emergence. In algorithmic terms, drops in the P_x profile represent transitions between different levels of description or emergence (Bédard & Bergeron, 2022). If an object exhibits scale-free emergence, its profile contains a steady continuation of such drops across many orders of magnitude of complexity. Natural phenomena, such as biological systems or weather and climate, appear to possess this property: they provide an almost endless well of regularities where each discovery uncovers a new layer of puzzles. Also synthetic artifacts with similar characteristics exist: for example certain fractals like the Mandelbrot set, or cellular automata like Conway's Game of Life. For such objects, we would expect via the $P_x \leftrightarrow D_x$ correspondence that past compression progress is a robust indicator of future learnability. While Jansma & Hoel (2025) have shown that such objects exhibiting causal contributions which are spread out across many levels of coarse-graining can be engineered, it remains a challenge to formally define the conditions required for a prior to prefer these scale-free structures.

Furthermore, we believe that moving beyond human-in-the-loop systems and towards truly self-improving open-ended intelligence, *introspective assessment* of our models is necessary. Current machine learning objectives focus almost exclusively on minimizing post-hoc loss. To achieve autonomy, we must develop introspective models capable of predicting their own learning progress. This requires architectures that do not just compress data, but explicitly model their own "compression frontier"—identifying where an increase in computational effort is likely to yield the highest gain. In the context of modern LLMs, this frontier is increasingly defined by test-time compute and Chain-of-Thought reasoning steps. An autonomous agent must have the ability to decide whether an object is boring (meaning additional runtime will not yield further compression) or interesting (meaning it holds the potential for a complexity drop given more reasoning tokens). Crucially, the exact nature of this computational effort requires a more rigorous taxonomy. Whether defined by longer recurrence, increased thinking steps, additional gradient descent iterations, or expanded model capacity, the underlying algorithmic relationship between these diverse resources remains a vital open question for future research.

By shifting our focus from pure learning to the principled selection of *what* to learn, we can begin to build systems that do not merely solve the tasks we give them, but autonomously seek to discover the richness of the universe they inhabit.

References

Antunes, L. and Fortnow, L. Sophistication revisited. *Theory of Computing Systems*, 45:150–161, 2009.

Antunes, L., Bauwens, B., Souto, A., and Teixeira, A. Sophistication vs logical depth. *Theory of Computing Systems*, 60(2):280–298, 2017.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 286(5439): 509–512, 1999.

Bauwens, B. *Computability in statistical hypotheses testing, and characterizations of independence and directed influences in time series using Kolmogorov complexity*. PhD thesis, Ghent University, 2010.

Bédard, C. A. and Bergeron, G. An algorithmic approach to emergence. *Entropy*, 24(7):985, 2022.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Bennett, C. H. Logical depth and physical complexity. In *The Universal Turing Machine: A Half Century Survey*, volume 1, pp. 227–258. Oxford University Press, Oxford and Kammerer & Unverzagt, Hamburg, 1988.

Brant, J. C. and Stanley, K. O. Minimal criterion coevolution: a new approach to open-ended search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 67–74, 2017.

Bruce, J., Dennis, M. D., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Cocke, J. and Minsky, M. Universality of tag systems with p= 2. *Journal of the ACM (JACM)*, 11 (1):15–20, 1964.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

Cook, M. et al. Universality in elementary cellular automata. *Complex systems*, 15(1):1–40, 2004.

Delétang, G., Ruoss, A., Duquenne, P.-A., Catt, E., Genewein, T., Mattern, C., Grau-Moya, J., Wenliang, L. K., Aitchison, M., Orseau, L., et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

Dohmatob, E., Feng, Y., and Kempe, J. Model collapse demystified: The case of regression. *Advances in Neural Information Processing Systems*, 37:46979–47013, 2024.

Faldor, M., Zhang, J., Cully, A., and Clune, J. Omni-epic: Open-endedness via models of human notions of interestingness with environments programmed in code. *arXiv preprint arXiv:2405.15568*, 2024.

Finzi, M., Qiu, S., Jiang, Y., Izmailov, P., Kolter, J. Z., and Wilson, A. G. From entropy to epiplexity: Rethinking information for computationally bounded intelligence. *arXiv preprint arXiv:2601.03220*, 2026.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Herrmann, V., Alcaide, E., Wand, M., and Schmidhuber, J. Multiple token divergence: Measuring and steering in-context computation density. *arXiv preprint arXiv:2512.22944*, 2025a.

Herrmann, V., Csordás, R., and Schmidhuber, J. Measuring in-context computation complexity via hidden state prediction. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 13–19 Jul 2025b.

Hughes, E., Dennis, M., Parker-Holder, J., Behbahani, F., Mavalankar, A., Shi, Y., Schaul, T., and Rocktäschel, T. Position: open-endedness is essential for artificial superhuman intelligence. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 20597–20616, 2024.

Hutter, M. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media, 2005.

Itti, L. and Baldi, P. F. Bayesian surprise attracts human attention. In *Advances in Neural Information Processing Systems (NIPS) 19*, pp. 547–554. MIT Press, Cambridge, MA, 2005.

Jansma, A. and Hoel, E. Engineering emergence. *arXiv preprint arXiv:2510.02649*, 2025.

Kompella, V. R., Stollenga, M., Luciw, M., and Schmidhuber, J. Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. *Artificial Intelligence*, 247: 313–335, 2017.

Koppel, M. Complexity, depth, and sophistication. *Complex Systems*, 1(6):1087–1091, 1987.

Lehman, J. and Stanley, K. O. Beyond open-endedness: Quantifying impressiveness. In *Artificial Life Conference Proceedings*, pp. 75–82. MIT Press, 2012.

Levin, L. A. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

Lin, Z., Gou, Z., Gong, Y., Liu, X., Xu, R., Lin, C., Yang, Y., Jiao, J., Duan, N., Chen, W., et al. Not all tokens are what you need for pretraining. *Advances in Neural Information Processing Systems*, 37:29029–29063, 2024.

Meulemans, A., Nasser, R., Wołczyk, M., Weis, M. A., Kobayashi, S., Richards, B., Lajoie, G., Steger, A., Hutter, M., Manyika, J., et al. Embedded universal predictive intelligence: a coherent framework for multi-agent learning. *arXiv preprint arXiv:2511.22226*, 2025.

Müller, M. Stationary algorithmic probability. *Theoretical Computer Science*, 411(1):113–130, 2010.

Müller, U. Brainfuck–an eight-instruction turing-complete programming language. *Available at the Internet address http://en. wikipedia. org/wiki/Brainfuck*, 1993.

Mutti, M. Unsupervised reinforcement learning via state entropy maximization. 2023.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. F. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, 2007.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Pathak, D., Gandhi, D., and Gupta, A. Self-supervised exploration via disagreement. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5062–5071. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/pathak19a.html`.

Post, E. L. Formal reductions of the general combinatorial decision problem. *American journal of mathematics*, 65(2):197–215, 1943.

Pugh, J. K., Soros, L. B., and Stanley, K. O. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:202845, 2016.

Rissanen, J. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Sancaktar, C., Blaes, S., and Martius, G. Curious exploration via structured world models yields zero-shot object manipulation. *Advances in Neural Information Processing Systems*, 35:24170–24183, 2022.

Schmidhuber, J. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90, Institut für Informatik, Technische Universität München, February 1990. (In November there appeared a revised and extended version.).

Schmidhuber, J. A possibility for implementing curiosity and boredom in model-building neural controllers. In Meyer, J. A. and Wilson, S. W. (eds.), *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pp. 222–227. MIT Press/Bradford Books, 1991a.

Schmidhuber, J. Curious model-building control systems. In *Proceedings of the International Joint Conference on Neural Networks, Singapore*, volume 2, pp. 1458–1463. IEEE press, 1991b.

Schmidhuber, J. A possibility for implementing curiosity and boredom in model-building neural controllers. 1991c.

Schmidhuber, J. What's interesting? Technical Report IDSIA-35-97, IDSIA, 1997. ftp://ftp.idsia.ch/pub/juergen/interest.ps.gz; extended abstract in Proc. Snowbird'98, Utah, 1998; see also Schmidhuber (2002a).

Schmidhuber, J. Exploring the predictable. In Ghosh, A. and Tsuitsui, S. (eds.), *Advances in Evolutionary Computing*, pp. 579–612. Springer, 2002a.

Schmidhuber, J. The speed prior: a new simplicity measure yielding near-optimal computable predictions. In *International conference on computational learning theory*, pp. 216–228. Springer, 2002b.

Schmidhuber, J. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

Schmidhuber, J. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In Pezzulo, G., Butz, M. V., Sigaud, O., and Baldassarre, G. (eds.), *Anticipatory Behavior in Adaptive Learning Systems. From Psychological Theories to Artificial Cognitive Systems*, volume 5499 of *LNCS*, pp. 48–76. Springer, 2009.

Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010. ISSN 1943-0604. doi: 10.1109/TAMD.2010.2056368.

Schmidhuber, J. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *Preprint arXiv:1511.09249*, 2015.

Shumailov, I., Shumaylov, Z., Zhao, Y., Papernot, N., Anderson, R., and Gal, Y. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

Shyam, P., Jaśkowski, W., and Gomez, F. Model-based active exploration. In *International conference on machine learning*, pp. 5779–5788. PMLR, 2019.

Solomonoff, R. J. A formal theory of inductive inference. part i. *Information and control*, 7(1): 1–22, 1964.

Storck, J., Hochreiter, S., and Schmidhuber, J. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks, Paris*, volume 2, pp. 159–164. EC2 & Cie, 1995.

Stout, A. and Barto, A. G. Competence progress intrinsic motivation. In *2010 IEEE 9th International Conference on Development and Learning*, pp. 257–262. IEEE, 2010.

Sutton, R. S. Integrated modeling and control based on reinforcement learning and dynamic programming. In *Neural Information Processing Systems*, 1990. URL `https://api.semanticscholar.org/CorpusID:62620110`.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. *2015 IEEE Information Theory Workshop (ITW)*, 2015.

Vereshchagin, N. and Shen, A. Algorithmic statistics: forty years later. In *Computability and Complexity: Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, pp. 669–737. Springer, 2016.

Wallace, C. S. and Boulton, D. M. An information measure for classification. *The Computer Journal*, 11(2):185–194, 1968.

Wang, H. Tag systems and lag systems. *Mathematische Annalen*, 152(1):65–74, 1963.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

Wolfram, S. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601, 1983.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. Star: Self-taught reasoner bootstrapping reasoning with reasoning. In *Proc. the 36th International Conference on Neural Information Processing Systems*, volume 1126, 2024.

Zhang, J., Lehman, J., Stanley, K. O., and Clune, J. Omni: Open-endedness via models of human notions of interestingness. *ArXiv*, abs/2306.01711, 2023. URL `https://api.semanticscholar.org/CorpusID:259064135`.

A Additional Discussion

A.1 Open-endedness emerges from social multi-agent interactions

A compelling alternative is that interestingness is not an intrinsic property of objects, but an emergent phenomenon of social interaction. Such settings span from competitive two-player games (Schmidhuber, 1997) to complex multi-agent ecosystems involving theory-of-mind and embedded agency (Meulemans et al., 2025). There, learning potential is often quantified through disagreement: if a population of models exhibits high variance in their predictions of an object, that object is deemed to contain unresolved, learnable structure (Pathak et al., 2019; Shyam et al., 2019; Sancaktar et al., 2022). Similarly, the notion of impressiveness (Lehman & Stanley, 2012) suggests that an artifact is interesting if it serves as a "proof of work"—a hard-to-reach state that signals computational effort to other agents. While we recognize the power of these social dynamics, we argue they do not bypass the fundamental problem of interestingness, they merely decentralize it. Whether an agent is filtering synthetic data or evaluating a peer's "impressive" achievement, it is still performing a prospective assessment of future progress. Even a co-evolutionary ecosystem (Pugh et al., 2016; Brant & Stanley, 2017) can be viewed as a single macro-agent in a self-generating loop. Therefore, we believe that the inductive properties of learnability we analyze in the following sections remain relevant to these social systems.

A.2 We can predict human-distilled Interestingness via LLMs

A compelling alternative to explicit algorithmic measures is the use of LLMs as a "Model of Interestingness". The OMNI framework (Zhang et al., 2023; Faldor et al., 2024) argues that because LLMs are trained on large amounts of human-generated data, they have already internalized nuanced human notions of what is worthwhile and novel. By prompting an LLM to generate tasks that a human would find interesting, these systems can successfully navigate near-infinite task spaces while avoiding uninspiring or repetitive data.

However, we must consider the long-term limits of this approach. OMNI fundamentally emulates human capacity for nuanced judgment; it effectively asks: "What would a human want to learn next?" While LLMs possess rich commonsense priors, what yields learning progress for a human is not necessarily identical to what yields progress for an artificial subject. Humans and our current AI systems possess vastly different learning architectures and existing knowledge bases. Perhaps, eventually we will converge to a shared conception of interestingness among general intelligences—whether artificial or human. But if the goal is to reach general intelligence through an open-ended process, we cannot assume that the current state of LLMs is already sufficiently close. LLMs currently have no privileged insight into their own internal compression frontiers—they are effectively selecting data based on an external social heuristic rather than an introspective assessment of their own potential for insight. While it is conceivable that a model eventually bootstraps a sophisticated self-understanding, true autonomous open-endedness likely requires the ability to identify potential progress in domains that human intuition hasn't yet charted.

A.3 Inductive interestingness is limited, we should care about content over curves

A valid critique of the inductive property of interestingness is its apparent disregard for *content*. One might argue that a system should not predict future progress based on the shape of a complexity profile, but rather on the semantic nature of the object itself. From this perspective, an agent can recognize a task as interesting because it identifies familiar motifs—physics-like patterns, linguistic structures, or causal hierarchies—which tend to yield to further analysis.

We can make the argument, however, that accounting for the content is analogous to choosing what kind of prior to use. To an agent with no domain knowledge, the curve is the only universal signal available. But as a system matures, it develops "meta-compression" models—essentially domain-specific priors. These models allow the agent to classify strings into different "structural families." For instance, a string known to contain large amounts of random noise can prompt a prior algorithmically favoring objects that plateau early, while a string representing a mathematical proof might elicit a prior that favors deep objects and steady progress over many drops.

B EMPIRICAL RESULTS

The theoretical results from Section 4 hold for the busy beaver regime, which involves runtimes that are not physically realizable. While they provide insight into the fundamental algorithmic properties of objects and their time-bounded compressibility, it is not immediately obvious how relevant these results are to real-world scenarios. To address this, we conduct empirical experiments that mirror our theoretical results. We choose three fundamentally different, yet still universal and practical, computational paradigms that represent a wide variety of possible universal computers: 2-Tag systems, elementary cellular automata with Rule 110, and the Brainfuck (BF) language.

In each of these systems, we run all programs up to a certain length with generous runtime limits. This allows us to construct real-world runtime vs. complexity profiles for all objects (i.e., output strings) computed by multiple different programs. From these profiles, we can compute the relationship between steps since the last observed progress and either the empirically achieved further compression progress or the relative time of the last further compression progress. We re-weight objects based on their prior probabilities and plot the curves in Figures 4, 5, and 6. These are the empirical equivalents of the theoretical expectations shown in Figure 3.

We observe that in all three computational paradigms, the fundamental trends clearly hold: Recent compression progress implies further compression progress in the future, especially under the assumption of the Algorithmic Prior. The most significant difference from the theoretical results is with respect to the Speed Prior: in the heavily constrained empirical setting, the logarithmic bias against long runtimes is less pronounced. This is especially the case for Rule 110 cellular automata, for which—due to their construction—no programs with very short runtimes exist. The exact experimental setups for the three systems are detailed below.

B.1 2-TAG

In this system, we run 2-tag systems (Post, 1943) with the alphabet $\{a, b, c, H\}$, where H is the halting symbol. These kinds of systems have been shown to be Turing complete (Wang, 1963; Cocke & Minsky, 1964). We enumerate all possible production rules up to a combined length of 13, with H appearing only at the end of a single rule. The starting word is always 'aaa'. This leads to approximately 120 million programs which we run for up to 100k steps. If no output is produced within this limit, we consider the program non-halting. For the results shown in Figure 4, we filter out all objects that show no drops in the runtime vs. complexity profile, as they do not meaningfully contribute to the analysis.

B.2 RULE 110

Rule 110 is a Turing-complete elementary cellular automaton (Wolfram, 1983; Cook et al., 2004). We use a state of size 512 with a cyclic boundary condition[1]. The programs are defined as the leftmost bits of the tape, up to the last value of 1. We enumerate all programs up to length 25, resulting in approximately 34 million simulations. A program halts as soon as any particular state appears for a second time; this state is then the output of the program. If no state repeats within 100k steps, we consider the program non-halting.

B.3 BF

BF (Müller, 1993) is a highly compact Turing-complete programming language. Since we are not providing any external inputs, we do not use the *read* instruction '`,`'. We enumerate all programs up to length 11 consisting of the remaining 7 instructions, leading to approximately 2.3 billion different programs, which we run for up to 100k steps.

[1] We do this purely for practical reasons and are aware that this technically reduces the automaton to a finite state machine.

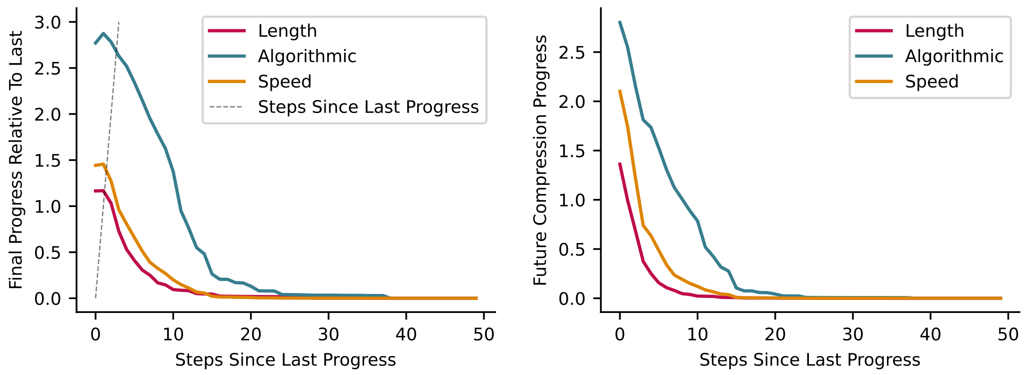

Figure 4: 2-Tag Systems. The average step of the final compression progress (left), and the average remaining compression progress (right), relative to the most recent progress. Plotted as a function of the steps since this most recent progress. As the number of these steps increases, the average remaining compression progress vanishes. Curves shown for outputs weighted with the Length, Algorithmic and Speed Prior. These results mirror the theoretical findings from Section 4.

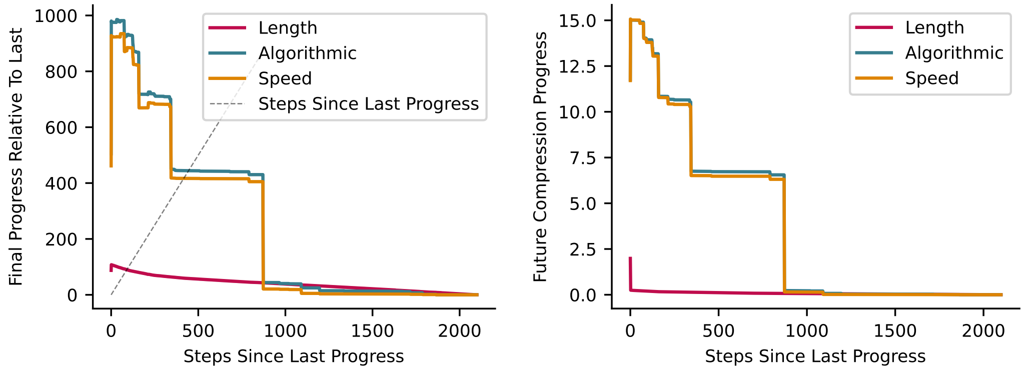

Figure 5: Rule 110. Analogous to Figure 4. Due to there not being any programs with very short runtimes, the difference between Algorithmic Prior and Speed Prior is less pronounced.

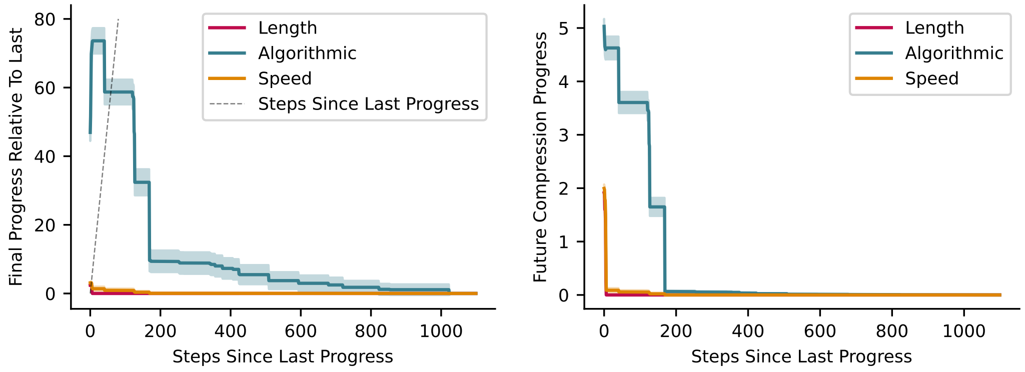

Figure 6: BF. Analogous to Figures 4 and 5. In all three figures, results are plotted with the standard error. It is noticeable only here.

C Mathematical Proofs

From Equations (2) and (3), we learn that the number of strings with a profile close to the given profile P is

$$2^{k_P - m_P + O(\log n_P)}. \tag{7}$$

Here, we set ϵ to be a small constant value, which also justifies replacing $m_P(\epsilon)$, as defined in Vereshchagin & Shen (2016) for technicalities of their proof, with m_P.

C.1 Proof of Proposition 4.1 (Length Prior)

Proof. Equation 7 allows us to quantify the number of strings with a partial profile $\hat{P}$ up to complexity t characterized by

$$\begin{aligned} \hat{b} &= \min\{r \mid (t, r) \in \hat{P}\}, \\ \hat{m} &= \min\{r \mid (r, \hat{b} - r) \in \hat{P}\}, \\ \hat{k} &= \min\{j \mid (t, j) \in \hat{P}\} + t, \text{ and} \\ \hat{n} &= \min\{t \mid (0, t) \in \hat{P}\}, \end{aligned}$$

with $0 < \hat{m} \leq t \leq \hat{k} \leq \hat{n}$. There are

$$N_L = 2^{\hat{k} - \hat{m} + O(\log \hat{n})} + \sum_{m=t+1}^{\hat{k}} \sum_{k=m}^{\hat{k}} 2^{k - m + O(\log \hat{n})} \tag{8}$$

such strings. The first term accounts for the number of strings with a profile where the observed drop at $\hat{m}$ remains the last. The two nested sums account for all strings where the last drop is after the observed partial profile $\hat{P}$. The outer sum enumerates the position of the drops along the complexity axis, the inner sum the size of the drop along the log-size axis.

Since the length $\hat{n}$ is fixed by the partial profile $\hat{P}$, the prior Length Prior $L(x)$ from Equation 4 is constant for all strings following $\hat{P}$. To compute the expected values, we calculate the sum of the characteristic values m and k weighted by the counts of strings possessing those values:

$$M_L = \hat{m} 2^{\hat{k} - \hat{m} + O(\log \hat{n})} + \sum_{m=t+1}^{\hat{k}} \sum_{k=m}^{\hat{k}} m 2^{k - m + O(\log \hat{n})} \tag{9}$$

$$K_L = \hat{k} 2^{\hat{k} - \hat{m} + O(\log \hat{n})} + \sum_{m=t+1}^{\hat{k}} \sum_{k=m}^{\hat{k}} k 2^{k - m + O(\log \hat{n})} \tag{10}$$

The expected values are defined as the ratios of these weighted sums to the total count N:

$$\mathbb{E}_{x \sim L}[m_x \mid \hat{P}_x] = \frac{M_L}{N_L}, \tag{11}$$

$$\mathbb{E}_{x \sim L}[k_x \mid \hat{P}_x] = \frac{K_L}{N_L}. \tag{12}$$

To analyze the behavior of these expectations, we first simplify the nested sums. Let us simplify the notation by treating the logarithmic term $O(\log \hat{n})$ as a bounded multiplicative factor $C \approx \hat{n}^{O(1)}$, noting that it effectively cancels in the ratio of expectations for large t. The asymptotic behavior is driven by the exponential terms.

We evaluate the inner summation over k as a geometric series. Let $S_{inner} = \sum_{k=m}^{\hat{k}} 2^{k-m}$. By substituting $j = k - m$, we obtain:

$$S_{inner} = \sum_{j=0}^{\hat{k}-m} 2^j = 2^{\hat{k} - m + 1} - 1. \tag{13}$$

Substituting this back into the expression for N_L, the total count is:

$$N_L \approx 2^{\hat{k}-\hat{m}} + \sum_{m=t+1}^{\hat{k}} \left(2^{\hat{k}-m+1} - 1\right). \tag{14}$$

The sum $\sum_{m=t+1}^{\hat{k}} 2^{\hat{k}-m+1}$ is a geometric series dominated by its first term (where $m = t + 1$). Thus, for $t < \hat{k}$, the total count scales as:

$$N_L \approx 2^{\hat{k}-\hat{m}} + 2^{\hat{k}-t+1}. \tag{15}$$

We observe two regimes. If the partial profile has already identified a significant drop such that $\hat{m} \ll t$, the first term $2^{\hat{k}-\hat{m}}$ dominates. This represents the strings that follow the observed drop at $\hat{m}$. The second term represents the "tail" of strings that might have a drop later than t.

We now derive the numerator M_L. Splitting the sum similarly gives:

$$M_L \approx \hat{m}2^{\hat{k}-\hat{m}} + \sum_{m=t+1}^{\hat{k}} m2^{\hat{k}-m+1}. \tag{16}$$

The sum in the numerator is dominated by the term at $m = t + 1$, contributing approximately $(t+1)2^{\hat{k}-t+1}$. The expectation is the ratio M_L/N_L:

$$\mathbb{E}[m] \approx \frac{\hat{m}2^{\hat{k}-\hat{m}} + t2^{\hat{k}-t+1}}{2^{\hat{k}-\hat{m}} + 2^{\hat{k}-t+1}} = \frac{\hat{m} + t2^{\hat{m}-t+1}}{1 + 2^{\hat{m}-t+1}}. \tag{17}$$

Using the approximation $\frac{A+B}{1+C} \approx A + B - AC$ for small C, and assuming $t > \hat{m}$:

$$\mathbb{E}[m] \approx \hat{m} + (t - \hat{m})2^{-(t-\hat{m}-1)}. \tag{18}$$

As t increases, the term $(t - \hat{m})2^{-(t-\hat{m})}$ vanishes exponentially. Since $t > \hat{m}$, the residual term is positive. Thus, $\mathbb{E}[m]$ converges to $\hat{m}$ from above.

For the complexity expectation $\mathbb{E}[k]$, we evaluate the weighted inner sum $\sum_{k=m}^{\hat{k}} k2^{k-m}$. Using the identity $\sum_{j=0}^{X}(m+j)2^j = (m + X - 1)2^{X+1} + 2 - m$, with $X = \hat{k} - m$, the dominant term is roughly $(\hat{k} - 1)2^{\hat{k}-m+1}$. Summing this over m from $t + 1$ yields a tail contribution proportional to $(\hat{k} - 1)2^{\hat{k}-t+1}$. The expectation becomes:

$$\mathbb{E}[k] \approx \frac{\hat{k}2^{\hat{k}-\hat{m}} + (\hat{k}-1)2^{\hat{k}-t+1}}{2^{\hat{k}-\hat{m}} + 2^{\hat{k}-t+1}}. \tag{19}$$

Simplifying the fraction by dividing by $2^{\hat{k}-\hat{m}}$:

$$\mathbb{E}[k] \approx \frac{\hat{k} + (\hat{k}-1)2^{\hat{m}-t+1}}{1 + 2^{\hat{m}-t+1}} \approx \hat{k} - 2^{-(t-\hat{m}-1)}. \tag{20}$$

The expectation $\mathbb{E}[k]$ converges to $\hat{k}$ exponentially fast as t grows, approaching from below.

□

C.2 Proof of Proposition 4.2 (Algorithmic Prior)

Proof. For the Algorithmic Prior $M(x)$, as defined in Equation 5, the probability of a string depends on its complexity k instead of its length. That means instead of simply counting strings, we need a normalization constant where we multiply the number of strings per profile with the probability associated with the profile:

$$N_M = 2^{-\hat{k}}2^{\hat{k}-\hat{m}+O(\log \hat{n})} + \sum_{m=t+1}^{\hat{k}} \sum_{k=m}^{\hat{k}} 2^{-k}2^{k-m+O(\log \hat{n})} = 2^{-\hat{m}+O(\log \hat{n})} + \sum_{m=t+1}^{\hat{k}} \sum_{k=m}^{\hat{k}} 2^{-m+O(\log \hat{n})} \tag{21}$$

The corresponding weighted sums are

$$M_M = \hat{m}2^{-\hat{m}+O(\log \hat{n})} + \sum_{m=t+1}^{\hat{k}} \sum_{k=m}^{\hat{k}} m2^{-m+O(\log \hat{n})} \tag{22}$$

$$K_M = \hat{k}2^{-\hat{m}+O(\log \hat{n})} + \sum_{m=t+1}^{\hat{k}} \sum_{k=m}^{\hat{k}} k2^{-m+O(\log \hat{n})}. \tag{23}$$

We can calculate the expectations the same way:

$$\mathbb{E}_{x\sim M}[m_x \mid \hat{P}_x] = \frac{M_M}{N_M}, \tag{24}$$

$$\mathbb{E}_{x\sim M}[k_x \mid \hat{P}_x] = \frac{K_M}{N_M}. \tag{25}$$

We proceed by simplifying the nested sums. As before, we treat $O(\log \hat{n})$ as a negligible multiplicative factor in the limit.

A key difference from the Length Prior is the behavior of the inner sums. The probability weight 2^{-k} cancels the volume factor 2^k, transforming the inner geometric series into arithmetic sums.

For the normalization constant N_M, the inner sum is simply a count of the integers k in the interval $[m, \hat{k}]$:

$$S_N = \sum_{k=m}^{\hat{k}} 1 = \hat{k} - m + 1. \tag{26}$$

Substituting this into the expression for N_M:

$$N_M \approx 2^{-\hat{m}} + \sum_{m=t+1}^{\hat{k}} (\hat{k} - m + 1)2^{-m}. \tag{27}$$

The sum over m is of the form $\sum P(m)2^{-m}$, where $P(m)$ is linear. This series is dominated by the first term where $m = t + 1$. Thus, the "tail" contribution scales approximately as $(\hat{k} - t)2^{-(t+1)}$. Comparing the main term (where the drop is at $\hat{m}$) to the tail:

$$N_M \approx 2^{-\hat{m}} + (\hat{k} - t)2^{-(t+1)}. \tag{28}$$

Next, we derive the expectation for the drop size m. The numerator M_M involves the same inner sum S_N, but weighted by m:

$$M_M \approx \hat{m}2^{-\hat{m}} + \sum_{m=t+1}^{\hat{k}} m(\hat{k} - m + 1)2^{-m}. \tag{29}$$

The tail sum is again dominated by the term at $m = t+1$, contributing roughly $(t+1)(\hat{k}-t)2^{-(t+1)}$. The expectation is:

$$\mathbb{E}[m] \approx \frac{\hat{m}2^{-\hat{m}} + t(\hat{k} - t)2^{-(t+1)}}{2^{-\hat{m}} + (\hat{k} - t)2^{-(t+1)}}. \tag{30}$$

For $t > \hat{m}$, the $2^{-\hat{m}}$ term dominates. Factorizing $2^{-\hat{m}}$ reveals the convergence:

$$\mathbb{E}[m] \approx \hat{m} + t(\hat{k} - t)2^{-(t-\hat{m}+1)}. \tag{31}$$

Since the tail term adds positive contributions (where $m > t > \hat{m}$), $\mathbb{E}[m]$ converges to $\hat{m}$ from above with exponential speed $O(2^{-(t-\hat{m})})$.

Finally, for the complexity $\mathbb{E}[k]$, we evaluate the inner sum of the numerator K_M:

$$S_K = \sum_{k=m}^{\hat{k}} k = \frac{(\hat{k}+m)(\hat{k}-m+1)}{2}. \tag{32}$$

This is the sum of an arithmetic progression. For the tail terms where $m \approx t$, the average value of k is approximately $\frac{\hat{k}+t}{2}$. The numerator K_M becomes:

$$K_M \approx \hat{k}2^{-\hat{m}} + \sum_{m=t+1}^{\hat{k}} \frac{(\hat{k}+m)(\hat{k}-m+1)}{2} 2^{-m}. \tag{33}$$

The tail contribution is dominated by $m = t+1$, scaling as $\frac{\hat{k}+t}{2}(\hat{k}-t)2^{-(t+1)}$. The expectation is:

$$\mathbb{E}[k] \approx \frac{\hat{k}2^{-\hat{m}} + \frac{\hat{k}+t}{2}(\hat{k}-t)2^{-(t+1)}}{2^{-\hat{m}} + (\hat{k}-t)2^{-(t+1)}}. \tag{34}$$

This can be viewed as a weighted average between the main profile (value $\hat{k}$) and the tail profiles (average value $\approx \frac{\hat{k}+t}{2}$). Since $t < \hat{k}$, the tail value $\frac{\hat{k}+t}{2}$ is strictly less than $\hat{k}$. Therefore, the tail drags the average down.

$$\mathbb{E}[k] \approx \hat{k} - (\hat{k} - \frac{\hat{k}+t}{2})(\hat{k}-t)2^{-(t-\hat{m}+1)}. \tag{35}$$

Thus, $\mathbb{E}[k]$ converges to $\hat{k}$ exponentially fast from below.

□

C.3 Proof of Proposition 4.3 (Speed Prior)

Proof. The Speed Prior $S(x)$ from Equation 6 assigns long running programs a lower probability. The outer sum enumerates computational phases with exponentially growing computational budgets. Programs that compute x in a later phase, i.e. after a longer runtime, have exponentially lower probability.

The transform described by Equation (1) allows us to convert set complexity values m to program runtimes via the busy beaver function BB(x). Due to the extremely fast growing nature of BB(x), according to the speed prior, the probabilities of descriptions with lower set complexity (left on the P_x profile) completely dominate description with higher set complexity (right on the P_x profile). This means we can ignore tail sums and our normalization constants and weighted sums are

$$N_S = 2^{\hat{k}-\hat{m}+O(\log \hat{n})} \tag{36}$$
$$M_S = \hat{m}2^{\hat{k}-\hat{m}+O(\log \hat{n})} \tag{37}$$
$$K_S = \hat{k}2^{\hat{k}-\hat{m}+O(\log \hat{n})}, \tag{38}$$

leading directly to the expectations

$$\mathbb{E}[m] = \frac{M_S}{N_S} = \hat{m} \text{ and} \tag{39}$$
$$\mathbb{E}[k] = \frac{J_S}{N_S} = \hat{k}. \tag{40}$$

□

Declaration of LLM Usage

We used Gemini 3.0 to proofread and occasionally to improve sentence flow and wording.