

SuperARC: A Test for Artificial Superintelligence Based on Compressed Modelling, Recursive Prediction and Problem Complexity (SUPPLEMENTARY INFORMATION)

Alberto Hernández-Espinosa, Luan Ozelim, Felipe S. Abrahão, Hector Zenil

1 Algorithmic Information Theory (AIT) and Intelligence

1.1 Compression and machine learning

Understanding compression as a necessary and fundamental characteristic of general intelligence refers to the ability to come up with a model capable of summarizing and eliminating redundancies by enabling one to explain more with less [1] or to gain “the ability of explanatory compression” [2] in order to achieve (or approximate) the necessary and sufficient causal conditions for describing, predicting, explaining, or simulating a certain phenomenon.

In machine learning models, such as large language models (LLMs), training involves learning to predict the next token in a sequence. This is essentially an exercise in compression—understanding the structure of language or other data and compressing it into a representation that allows accurate predictions. LLMs can be thought of as word (token) time series predictors based on short- and long-range correlations that compress data from their very large training sets based on text repositories mostly available online, and captured in a much smaller object such as a giant matrix, whose numerical entries can partially and lossy reconstruct the training dataset. Such a compressor can simulate/predict the uncompressed information stored in a multidimensional tensor probability distribution in a manner comparable to the uncompressed data captured in the smallest possible model—the smaller the better, and hence the smaller the model is, the better compressor [1]. The compressor’s success can be evaluated in terms of how much information is lost in transit between the original world description and the decompressed data from the LLM model.

In order to predict the future state of an event, a model shorter than the explanandum that captures its main features (object, event) is necessary in order to avoid conflating underlying laws with spurious patterns [3, 4], and the more recursively compressed the model, the more adequate and less overfitted. ‘Recursively’ here refers to “mechanistically” in the more general sense, that is, to any process whose unfolding (dynamics or evolution over time) can be mathematically modelled and determined—or more formally, a process that is computable—and not only engaged in pattern matching as in statistical compression, which is only one type, and a limited one, of data/model compression. For example, recursively (i.e., algorithmically)

compressing an object, such as a list of observations or events, yields the ability to predict some part of the object given other parts, as a byproduct of being able to run the compression process in reverse (decompression), should the events be removed from randomness (i.e., they are not disconnected from each other in such a way that any pattern, causation or correlation found is merely spurious).

An effective decompression process not only reconstructs or reassembles the original explanandum but it should be able to produce a continuation of it based on the continuation of the optimal recursive compressed features in reverse, producing a simulation that acts as a prediction on which a future action can be modelled. This principle of ‘compression’, understood broadly as the process of simplifying a generative model as much as possible, constitutes the heuristic underlying any machine learning method with the purpose of avoiding overfitting, e.g., by ensuring the minimisation of the generalisation error while compromising the training error the least as possible.

The process of planning beyond what has made available, i.e., to predict the yet unseen, comprises making a comparison of the possible outcomes of the processes that unfold from the most succinct and expressive model (or theory) that is previously known. Then, by adjusting it during an iteration over a recursive process of further comparisons between new unfoldings from future possible optimal models yet to be discovered, one obtains an evolving ground ‘truth’ in a continuous learning process. As we discuss in Section [1](#), when such an iterative update process occurs over the entire algorithmic space one attains the most optimal prediction in the Bayesian sense [\[5, 6\]](#).

1.2 Fundamental properties

In the context of algorithmic information theory (AIT), universal computation is considered a central aspect of general intelligence in arbitrary systems. These systems are in turn considered to be capable of making formal-theoretic predictions (e.g. of solar and lunar eclipses) with high accuracy according to a mathematical theory whose logical and equational derivations/predictions can always be verified by an unambiguous decision procedure [\[7, 8\]](#). This underlies all science as it presumes and assumes that world phenomena can be described in a mathematical form in which science can deal with, e.g. using equations, computer simulations, mathematical modelling, etc. The ongoing rate of success of the scientific practice in finding compressible models for explaining and predicting natural phenomena evinces fundamental algorithmic and non-random characteristics underlying the universe itself [\[9\]](#). If reality were truly random, science as a predictive, model-building enterprise would be impossible. A hallmark of a powerful compressed model is its capacity to unify previously disparate or anomalous observations under a single, coherent explanatory framework. The example

of General Relativity, which provides a more compact description than Newtonian mechanics plus all its necessary ad-hoc corrections, demonstrates this property. A superior compression subsumes old models and explains their exceptions.

AIT is an overarching generalisation of classical (Shannon) information theory (see also Section 1.3) and the accepted mathematical definition that tells apart randomness from non-randomness able to objectively describe and quantify what a compact model is and what abduction (i.e., finding a model or theory that explains a given set of phenomena) and prediction corresponds in a formal mathematical setting.

Being one of the complexity indexes in AIT, *algorithmic complexity* (also referred to as program-size, Kolmogorov or Solomonoff-Kolmogorov-Chaitin complexity) is a measure of the complexity of an object invariant (up to an object-independent additive constant) to the underlying formal theory, computation model and programming language. The value denoted by $K(\sigma)$ of a finite string σ is the length of the shortest binary program (on an arbitrarily fixed universal Turing machine) that outputs σ . The more compressible (or less random) a string σ is, the larger the positive value of $|\sigma| - K(\sigma)$, where $|\sigma|$ is the length of σ . More complex (or random) objects require longer representative instantiations of their underlying generative model, while simpler, more regular objects can be generated by shorter programs [10, 11, 12, 13]. If a sequence σ can be represented by a shorter program p , the shorter program captures the regularities in σ . In this sense, the program can be used to generate or predict future segments of the sequence, based on the learnt regularities, thus directly tying compressibility to the ability to predict future patterns. The algorithmic complexity (along with other complexity indexes in AIT) of an object is proved to be universal (i.e., the value exists for every computing system) and invariant (i.e., the value remains the same according to any computing system, except for an object-independent constant) to the arbitrarily chosen computation model, programming language, probability measure, and formal theory [14, 15, 16]. As we discuss in Sections 4.2 and 4.3 in the main text, both of these properties are crucial for measuring a system’s intelligence in a manner that is generalisable beyond biases in human-centric metrics and real-world datasets made available for training.

Algorithmic complexity goes beyond strings, beyond binary and beyond computer programs. One uses this language or framework as a technicality given the fundamental nature of universal computation, including strings, binary languages, computer programs, etc. For example, as proven by Shannon any, discrete data can be transformed into binary without loss of information [14]. Under the Church-Turing thesis, any effective description and decidable rule can be enacted by a computer program. These computer programs are also not restricted to deal with strings only, just as computers deal with images, vectors, tensors, sounds, video or anything else that can be encoded. Not only playing a crucial role in data compression, algorithmic

mic complexity is therefore a concept of fundamental nature in the scientific method [17]. As we discuss in Sections 4.2 in the main text, science itself can fundamentally be seen as compressing natural phenomena, as the process of producing ever more compact representations of the physical world into rules, equations, and scientific models that provide ever greater explanatory and predicting power.

For illustration purposes and without loss of generality, let us consider a sequence of integers. The ability to compress such a sequence effectively is often taken as an indicator of understanding a model that is capable of generating the sequence, and one does not need to take the minimum requirement to the limit to find short plausible explanations. The decimal expansion of an irrational number like π may appear complex and non-repeating, but the entire infinite sequence can be generated by a very simple mathematical formula (and algorithm). This demonstrates that a complex appearance does not necessarily entail a complex underlying generating process. In this sense, sufficient knowledge for explaining (or ‘comprehending’) π is not a matter of memorizing its digits, but in coming up with—in other words, performing abductive reasoning—the mathematical formula that generates it [17]. These explanations are computational in nature so that they can be built (and employed) by an universal constructor independent of the arbitrary choice of the computation model that one may deem fundamental. Such a universal constructor can be equivalent to a Turing machine, although not necessarily identical or isomorphic to a Turing machine, not even to the mechanistic nature upon which the machine is implemented or embedded into.

Within the context of AIT, universal induction (based on Solomonoff’s Theory of Inductive Inference) proves that prediction and compression are tightly linked in order to obtain optimal abductive reasoning, and therefore inferring the best model or theory. Solomonoff [18] laid the foundation for *algorithmic probability* (another complexity index in AIT), which is a universally optimal probability measure in which an object is generated by a random program fed into a universal constructor (see Sup. Inf.). The algorithmic coding theorem (ACT) [8, 19] in Equation (1) displays one of the central results not only in AIT, but also one that has pervasive implications for any mathematical endeavour in science, particularly in artificial intelligence [20]. Because algorithmic probability and algorithmic complexity are inversely proportional, ACT states that the more frequent an object is generated, the lower its algorithmic complexity, and vice-versa.

$$K(s) = -\log P(s) = -\log(\mathbf{m}(s)) \pm \mathbf{O}(1) = -\log \left(\sum_{p \in \{w: U(w)=s\}} 2^{-|p|} \right) \pm \mathbf{O}(1) , \quad (1)$$

where:

- $P(s)$ is the *algorithmic probability* of string s ;
- $K(s)$ is the (prefix) *algorithmic complexity* of string s ;
- $\mathbf{m}(s)$ is a *maximal* semicomputable semimeasure on the object s ;
- $\sum_{p \in \{w: U(w)=s\}} 2^{-|p|}$ is the universal (a priori) probability of the event s .

Notice that a semicomputable semimeasure $\mathbf{m}(\cdot)$ is said to be *maximal* if for any other semicomputable semimeasure $\mu(\cdot)$ —including any computable probability measure one may arbitrarily choose—, where $\sum_{x \in \{0,1\}^*} \mu(x) \leq 1$, there is a constant $C > 0$ (which does not depend on x) such that, for every encoded object x , $\mathbf{m}(x) \geq C \mu(x)$. This means that any arbitrarily chosen (computable or semicomputable) probability (semi)measure can only assign a higher probability to an event than the algorithmic probability of the event could up to an independent multiplicative constant. Thus, across the landscape of all generative processes of each object that can be generated, the *universal distribution* defined by the (algorithmic) probability from Equation (1) eventually dominates all other prior distributions one might devise [6], thereby enabling one to infer the model optimally in the asymptotic limit.

The universal probability of an event can be understood as the probability of randomly generating (by an i.i.d. stochastic process) a prefix-free (or self-delimiting) program that generates the event. In other words, the probability that a randomly generated explanation (like a computer program) can generate an object [6]. In conjunction with the invariance theorem of algorithmic complexity, ACT is related to the universally optimal encoding of objects, generalising information content measures for measurable spaces beyond what classical information theory is able to achieve [19, 21, 7], e.g., setting the theoretical underpinnings of any method based on algorithmic probability such as the coding theorem method (CTM) [22, 23, 24, 13], universal (Solomonoff) induction [25, 26, 18], Levin’s universal search [5], and minimum description length [6, 15, 27]. (See also Section 2 in the main text in the main text). Universal predictors, such as those based on Levin’s universal search or universal induction, use the ACT to model the most likely future events based on past data, capturing the link between compression and prediction proved to hold in the theory of algorithmic randomness, as we discuss in Section 1.3.

1.3 Randomness, prediction, and compression

In comparison to statistical randomness (defined as a lack of patterns which cannot be identified by a particular statistical test), a key aspect of algorithmic complexity is the deeper relationship with *algorithmic randomness*,

whose lack of patterns is not only attested by an arbitrarily chosen statistical test but also by any conceivable formal-theoretic mathematical test effected by a computational decision procedure, thereby yielding incompressibility (and vice-versa). For example, in the case of a unidimensional machine, a sequence is considered algorithmically random if its shortest generative program has essentially the same length of the sequence itself, that is, no shorter program exists that can generate the sequence. In the case of a higher-level programming language, such a sequence can at best be described as a program of the type ‘print(x)’. Formally, an object x is (algorithmically) random if $K(x) \geq |x| - \mathbf{O}(1)$, where $|x|$ is the size of the object. Notice that x is incompressible because no smaller program can produce it (except for a string-independent constant that may only depend on the arbitrarily chosen machine or programming language), which contrasts with highly structured or predictable data, where $K(x) \ll |x|$. A random string cannot be significantly compressed [11], implying that intelligence (as seen in systems that can compress data) involves recognising non-random patterns in data.

Statistical randomness (such as when a random event is measured by entropy-based statistical methods) is quantifiable by degrees of uncertainty based on frequency distributions, which is indeed effective and optimal when compressibility arises from repetition or statistical redundancy. This is because entropy is known to achieve optimal compression for pure stochastic processes that are ergodic and stationary [14]. Under these same conditions, statistical compression methods, such as the algorithms in the LZ family, have been proven to also achieve optimal compression. However, in case those conditions are not met, such as when the random source is not guaranteed to be stationary or the process is mixed (partially stochastic and partially mechanistic like complex systems found in nature), entropy (or any other statistical method) is proved to diverge from the optimal value given by algorithmic complexity—value which is also proved to be invariant under the arbitrary choice of programming language, computation model, probability distribution, and formal theory. As we further discuss in Section 4.2 in the main text, this invariance is one of the reasons compression is a task that subsumes other intelligent systems’ capabilities; and why other statistics-based approaches such as LLMs are limited in comparison to neurosymbolic approaches that include the algorithmic view and subsumes the statistical one.

Thus, in the general case, statistical compression methods cannot achieve optimal compression even ‘in principle’. Entropy cannot detect algorithmic or generative structure that is not statistically apparent. When a statistical compression algorithm such as ZIP or LZW compresses x into other computer files much smaller than $|x|$, it is a sufficient proof of *non*-randomness. However, if it does not compress x (or if it can only compress x into another file of size of the same order of $|x|$), it is not a proof of randomness because there may be a generative program that the statistical compression

is unable to produce/find. In other words, algorithmic randomness always implies statistical randomness, but the opposite does not always hold.

In practical terms, compression algorithms like ZIP or LZW attempt to reduce the size of the data by identifying recurring statistical patterns. When an AI system like ChatGPT can generate a concise and generalisable program to reproduce a sequence, it shows that the model has ‘compressed’ the information by finding underlying patterns. Nevertheless, algorithmic compression is more powerful because it can continue searching for algorithmic generative processes while statistical pattern matching cannot. Pattern matching can only be descriptive of an entire object, but computation-based regression, symbolic processing, and program synthesis can be fundamentally generative; in the sense that other mechanisms or causal processes among (underlying or in common to) the parts of the object are prone to be swept over in an algorithmic compression.

In addition to formalising randomness beyond statistical patterns, the theory of algorithmic randomness established a profound connection between prediction and compression [19, 16, 8, 15]. It is equivalent to say that a sequence is algorithmically random (i.e., incompressible) if, and only if, no computable betting strategy (martingale) can succeed on it, establishing the equivalence between the inability to compress a sequence and the impossibility of predicting its future bits using any computable betting strategy a formal theory can devise. This result demonstrated that the ability to compress a sequence is equivalent to being able to predict its future bits using any effective method (mathematical proof of this direct equivalence is provided in the Section 1.4). It also highlights the deep interplay between randomness, prediction, and compression, setting the underpinnings of our framework introduced in Section 4.3 in the main text. These results in AIT demonstrate that compression is not only a particular task that a (-n artificial or physical) system might be able to perform. In fact, as we further discuss in Section 4.2 in the main text in the main text, it is also a ‘task’ that subsumes other comprehension tasks while taking into account the entire algorithmic space; but in an agnostic manner to the arbitrarily chosen computational capabilities, formal-theoretic features, and a priori knowledge.

1.4 Equivalence between compression and prediction via Martingales

An infinite sequence (or equivalently, a real number) is denoted by $x = x_1x_2x_3\dots$, where each $x_i \in \{0, 1\}$. Let $x \upharpoonright_n$ the sequence of the first n bits of the binary representation of x .

A (*super*)*martingale* function $d : \{0, 1\}^* \rightarrow \mathbb{R}^+$ represents a betting strategy that satisfies the fairness conditions:

$$d(\sigma) = \frac{d(\sigma 0) + d(\sigma 1)}{2}, \text{ in the case of a martingale;} \quad (2)$$

$$d(\sigma) \geq \frac{d(\sigma 0) + d(\sigma 1)}{2}, \text{ in the case of a supermartingale.} \quad (3)$$

This conveys the idea that the expected capital after the next bet is either equal (for martingales) or is lost (for supermartingales) with respect to the previous capital.

A (super)martingale d *succeeds* on a sequence x if:

$$\limsup_{n \rightarrow \infty} d(x \upharpoonright_n) = \infty$$

This implies that the betting strategy can make an unbounded amount of money on x at the asymptotic limit as the length of the initial segment of x increases.

A martingale d is *(left) semicomputable* if there is an algorithm that computably enumerates the left cuts of $d(\sigma)$ for any given string σ . Thus, if a semicomputable d succeeds on a sequence x , this (super)martingale can be interpreted as revealing the existence of an algorithm that can computably enumerate a betting strategy that always increases its capital gains at the asymptotic limit as the length of the initial segment of x increases. This holds even if eventually one loses expected capital in the next bit (as the supermartingale condition allows). The existence of such an enumerating algorithm guarantees that there is at least one asymptotically effective way of predicting the forthcoming bits in the infinite sequence x so as to render the betting strategy successful as this process goes on.

Now, remember that an algorithmically random infinite sequence (or real number) x is incompressible up to a fixed constant so that $K(x \upharpoonright_n) \geq n - \mathbf{O}(1)$, and the constant does not depend on n . Therefore, if x is *not* algorithmic random, then for any k and for any $n' \geq 1$, there is $n \geq n'$ such that $K(x \upharpoonright_n) < n - k$. In other words, x is compressible (by more than a fixed value) infinitely often.

The notion of predictability conveyed by martingales should reflect the fact that in the case of an algorithmically random sequence, there would not exist an enumerating algorithm that guarantees that there is at least one asymptotically effective way of predicting the forthcoming bits in the infinite sequence x so as to render the betting strategy successful as this process goes on. In summary, one should not expect to be able to devise a computably enumerable betting strategy that is successful on a perfectly random sequence. Indeed, the equivalence between (super)martingales and algorithmic randomness holds:

- If a sequence x is not algorithmically random (i.e., it is compressible infinitely often), then there exists a semicomputable martingale that succeeds on x .
- Conversely, if there exists a semicomputable martingale that succeeds on x , then x is not algorithmically random (i.e., it is compressible infinitely often).

Another equivalence between algorithmic randomness and the notion of predictability can be achieved from (stochastic or probabilistic) martingale processes, which are defined upon real-valued random variables. In this case, one can demonstrate that an infinite sequence is algorithmic random iff no *computable* martingale process succeeds on it [19].

Usually, (super)martingales and randomness are demonstrated to be equivalent via proof- and measure-theoretic statistical (Martin-Löf) tests. A sequence is incompressible iff it does *not* pass on any (Σ_1^0) theoretic statistical test [19], thereby called (prefix) algorithmic random (1-random or $\mathbf{O}(1)$ - K -random). It is important to remark that the triple equivalence between predictability (via martingales), statistical tests (via proof and measure theory), and compressibility (via algorithmic complexity) establishes one of the foundational results in the theory of algorithmic randomness and algorithmic information [19, 8].

In order to highlight the connection between predictability and compressibility, in the following, we introduce a novel and alternative proof for the *direct* equivalence between compression and (successful computably enumerable) martingales.

Regarding algorithmic randomness deficiency [15], one can define a weaker notion of supermartingales to account for language and computation model dependencies. We say a function d is a C -*supermartingale* iff for any sequence σ , there is a constant $C \geq 0$ (that does not depend on σ) such that

$$\frac{1}{2^C} \leq \frac{d(\sigma 0) + d(\sigma 1)}{2 d(\sigma)} \leq \frac{1}{2^{-C}}. \quad (4)$$

On the one hand, the expected capital from the bet in the next bit is never smaller than a constant ratio of the previous bet. On the other hand, one may gain some expected capital in the next bet but only up to a multiplicative constant. Instead of a constant C , one can also define $\mathfrak{d}(\sigma)$ -supermartingale, where $\mathfrak{d}: \{0, 1\}^* \rightarrow \mathbb{N}$. For the present purposes, we focus on the constant that does not depend on the object.

From the basic properties in algorithmic information theory, it is straightforward to prove that the function

$$d_{(1,k)}(\sigma) = \frac{2^{|\sigma|}}{2^{k+K(\sigma)}} \quad (5)$$

is a $\mathbf{O}(1)$ -supermartingale. Clearly, if x is not an algorithmic random infinite sequence, then $d_{(1,k)}(x \upharpoonright_n) \geq 1$ for every k and n in which $K(x \upharpoonright_n) < n - k$. From the definitions and the property that the summation of any two C -supermartingales is also a C -supermartingale, one can demonstrate by induction that if $d_1, d_2, \dots, d_i, \dots$ is an infinite family of C -supermartingales and $\sum_{i=1}^{\infty} d_i(a) < \infty$, where a is any string for the initial capital (usually, the empty string λ , 0, or 1), then $\sum_{i=1}^{\infty} d_i(\cdot)$ is a C -supermartingale (see also [19]). From Equation (5), we have it that $\sum_{i=1}^{\infty} d_{(1,i)}(a) = \mathbf{O}(1)$. In addition, for any σ , one has it that $\sum_{k=|\sigma|}^{\infty} d_{(1,k)}(\sigma) \leq \mathbf{O}(2^{|\sigma|})$, and as a consequence $\sum_{i=1}^{\infty} d_{(1,i)}(\sigma) < \infty$ holds. We also have that $\sum_{i=1}^{\infty} d_{(1,i)}(\sigma)$ is left semicomputable because there is a program that can always approximate the value of $\sum_{i=1}^{\infty} d_{(1,i)}(\sigma)$ from below for any σ . Therefore, if x is not an algorithmic random infinite sequence, it follows that there is a left semicomputable $\mathbf{O}(1)$ -supermartingale $d_1(\sigma) = \sum_{i=1}^{\infty} d_{(1,i)}(\sigma)$ such that $\limsup_{n \rightarrow \infty} d_1(x \upharpoonright_n) = \infty$. The converse implication can be proved analogously to the proof in Theorem 1, because every martingale is a $\mathbf{O}(1)$ -supermartingale.

Nevertheless, as we show in Theorem 1, one can also obtain a demonstration of the implications in both directions between compression and the traditional (successful computably enumerable) *martingales* without resorting to proof- and measure-theoretic statistical tests.

Theorem 1 (incompressibility and unpredictability). *Let $x = x_1 x_2 \dots x_n \dots$ be an infinite sequence (or equivalently, a real number). Then, x is algorithmic random iff there is no (left) semicomputable martingale that succeeds on x .*

Proof (Compression implies Prediction): For any arbitrary sequences w and z , let $w \preceq z$ denote w being a prefix of the sequence z . Without loss of generality, let $C > 0$ be a constant such that

$$K(a) < C, \quad (6)$$

for $a \in \{\lambda, 0, 1\}$. Let

$$W_k(\sigma) = \left\{ w \in \{0, 1\}^*: \begin{array}{l} w \succeq \sigma, \\ (K(w) < C) \vee (K(w) < |w| - k) \end{array} \right\} \quad (7)$$

be the set of bit strings that are compressible by at least k bits, strings which have σ as a prefix. For arbitrary $k \in \mathbb{N}$, let $d_{(2,k)}: \{0, 1\}^* \rightarrow \mathbb{R}^+$ be a

function such that

$$d_{(2,k)}(\sigma) = \frac{2^{|\sigma|}}{2^k} \left(\sum_{w \in W_k(\sigma)} \frac{1}{2^{K(w)}} \right). \quad (8)$$

First, notice that $W_k(a) \neq \emptyset$ for any $k \geq 1$ because of our choice of the constant C . Secondly, from the basic properties of a prefix-free (or self-delimiting) programming language [15, 8, 19], we have that

$$0 \leq d_{(2,k)}(\sigma) \leq \frac{2^{|\sigma|}}{2^k} \quad (9)$$

holds for any σ and k . As a consequence, we will have it that $\sum_{k=1}^{\infty} d_{(2,k)}(a) = \mathbf{O}(1)$ and $\sum_{k=1}^{\infty} d_{(2,k)}(\sigma) < \infty$. From the definition of $W_k(\cdot)$ in Equation (7), we have that

$$W_k(\sigma 0) \cap W_k(\sigma 1) = \emptyset \quad (10)$$

and

$$W_k(\sigma 0) \cup W_k(\sigma 1) = W_k(\sigma) \quad (11)$$

hold for any σ , and therefore one can straightforwardly demonstrate that $d_{(2,k)}$ is a martingale for each fixed k . We know that if $d_1, d_2, \dots, d_i, \dots$ is an infinite family of arbitrary martingales and $\sum_{i=1}^{\infty} d_i(a) < \infty$, where a is any string for the initial capital, then $\sum_{i=1}^{\infty} d_i(\cdot)$ is a martingale [19]. Therefore, we will have that

$$d_2(\sigma) = \sum_{i=1}^{\infty} d_{(2,i)}(\sigma) \quad (12)$$

is a martingale. Since the infinite set $W_k(\sigma)$ can be computably enumerated from below for any σ , we will have that $\sum_{i=1}^{\infty} d_{(2,i)}(\sigma)$ is left semicomputable. By construction, for any k and σ in which $K(\sigma) < |\sigma| - k$ holds, one has it that

$$d_{(2,k)}(\sigma) \geq d_{(1,k)}(\sigma) \geq 1, \quad (13)$$

where $d_{(1,k)}(\sigma)$ was defined in the above Equation (5). Additionally, for any w and z with $w \succeq z$ such that $K(z) < |z| - k$ and $K(w) < |w| - k - 1$ hold, we will have it that $d_{(2,k+1)}(w) \geq 1$ and $d_{(2,k)}(w) \geq 1$. One can extend this property recursively so that if $w_m \succeq w_{m-1} \succeq \dots \succeq w_0$ such that $K(w_i) < |w_i| - k - i$ holds for any i where $0 \leq i \leq m$ and $m > 0$, then $d_{(2,k+i)}(w_m) \geq 1$ holds for each $i \leq m$, thereby one obtains that $d_2(w_m) \geq m$. Therefore, if x is not an algorithmic random infinite binary sequence, then $\limsup_{n \rightarrow \infty} d_2(x \upharpoonright_n) = \infty$. □

Proof (Prediction implies Compression): From the martingale condition in Equation (2), where

$$\frac{d'(\sigma 0) + d'(\sigma 1)}{d'(\sigma)} = 2 \quad (14)$$

holds for any σ and an arbitrary martingale d' , we will have that

$$\frac{d'(\sigma)}{d'(\sigma \upharpoonright_k)} = \prod_{i=1+k}^{|\sigma|} \frac{d'(\sigma \upharpoonright_i)}{d'(\sigma \upharpoonright_{i-1})} \leq 2^{|\sigma|-k} \quad (15)$$

holds for any arbitrary natural number $k \geq 1$ with $k < |\sigma|$. Let $\langle \cdot \rangle$ be any computable encoding of a string in a prefix-free language such that for any $w \in \{0, 1\}^*$, one has it that

$$|\langle w \rangle| \leq |w| + \mathbf{O}(\log(|w|)) \quad (16)$$

and

$$\sum_{\sigma \in \{0,1\}^*} \frac{1}{2^{|\langle \sigma \rangle|}} \leq 1. \quad (17)$$

Let

$$W'_k(\sigma) = \left\{ w \in \{0, 1\}^* : \begin{array}{l} w \succeq \sigma, \\ \log\left(\frac{d(w)}{2^k}\right) \geq |\langle \sigma \upharpoonright_{k^2} \rangle| \end{array} \right\}. \quad (18)$$

be a set of the extensions of σ for which their values obtained from d are sufficiently large. Notice that since d is (left) semicomputable by hypothesis, then the set $W'_k(\sigma)$ is computably enumerable for any σ given $k \in \mathbb{N}$. Additionally, from Equation (16), the condition $\limsup_{n \rightarrow \infty} d(x \upharpoonright_n) = \infty$ implies that for every $k, m_0 \in \mathbb{N}$ with $m_0 \geq k$, there is at least one $x \upharpoonright_m \succeq x \upharpoonright_{m_0}$ such that

$$d(x \upharpoonright_m) \geq 2^{k^3} \gg 2^{|\langle x \upharpoonright_{m_0 \upharpoonright_{k^2}} \rangle| + k} \quad (19)$$

with $m > m_0$, and thereby one obtains that $x \upharpoonright_m \in W'_k(\sigma)$. Now, we define the function

$$f_k(\sigma) = \frac{\operatorname{argmin}_{w \in W'_k(\sigma)} (2^{|w|})}{2^k} \quad (20)$$

built upon the set W'_k in Equation (18). From the computable enumerability of W'_k , we will have that $f_k(\cdot)$ is a right semicomputable function (i.e., semicomputable from above), and hence $\frac{1}{f_k(\cdot)}$ is left semicomputable (i.e., semicomputable from below). Clearly, in case $\sigma \in W'_k(\sigma)$, one will have it that

$$f_k(\sigma) = 2^{|\sigma|-k}. \quad (21)$$

Furthermore, from Equations (15) and (18), one also has that

$$f_k(\sigma) \geq \frac{d(w)}{2^k} \geq 2^{|\langle \sigma \upharpoonright_{k^2} \rangle|} \quad (22)$$

holds for some $w \in W'_k$ and any fixed k . Therefore, from Equations (17) and (22), one will have it that

$$\sum_{\sigma \in \{0,1\}^*} \frac{1}{f_k(\sigma)} \leq 1. \quad (23)$$

Let

$$\mu(\sigma) = \frac{1}{f_k(\sigma)} \quad (24)$$

so that from Equation (23) we directly obtain that $\mu(\cdot)$ is a left semicomputable semimeasure. Since $\limsup_{n \rightarrow \infty} d(x \upharpoonright_n) = \infty$, then Equations (19), (21), and (24) imply that for each fixed k , there are infinitely many $m \in \mathbb{N}$ such that

$$\frac{1}{\mu(x \upharpoonright_m)} = 2^{m-k}. \quad (25)$$

From the ACT [8, 19, 15] in Equation (1), we have that

$$K(x) = -\log(\mathbf{m}(x)) \pm \mathbf{O}(1), \quad (26)$$

holds, where $\mathbf{m}(\cdot)$ is a *maximal* semicomputable semimeasure. Finally, it follows from Equations (25) and (26) that there is a constant C' such that for each fixed k , there are infinitely many $m \in \mathbb{N}$ such that

$$K(x \upharpoonright_m) \leq \log\left(\frac{1}{C'\mu(x \upharpoonright_m)}\right) \pm \mathbf{O}(1) \leq m - k + \mathbf{O}(1). \quad (27)$$

□

1.4.1 Levin's Distribution and the Algorithmic Probability of Integer Sequences

As shown in Equation (1), the algorithmic probability $P(s) = 1/2^{K(s)}$ of a string s is equivalently¹ given by [18, 5]:

$$P(x) = \sum_{U(p)=x} 2^{-|p|},$$

where $U(p) = x$ means that the (prefix) universal Turing machine U , when given program p , produces the string x . $|p|$ is the length of the program p , so $2^{-|p|}$ can be interpreted as the probability assigned to that program, with shorter programs being more probable.

Levin's distribution modifies the algorithmic probability by adding a penalty for the time taken by the program to compute the output, for example as

$$m(x) = \sum_{p: U(p)=x} 2^{-|p| - \log T(p)},$$

¹Except for a multiplicative independent constant.

where $T(p)$ is the time taken by program p to generate the string x , where $\log T(p)$ is the logarithmic penalty for the time complexity of program p . Notice that m is a lower bound for the universally optimal semicomputable semimeasure \mathbf{m} in Section 1.4 that appears in the ACT.

In the context of a time series x_1, x_2, \dots, x_t , the goal is to predict the next value x_{t+1} based on the previous observations x_1, x_2, \dots, x_t . Modifying it according to the conditional version of the ACT, the probability of the next element x_{t+1} , given the previous values, becomes

$$P(x_{t+1} \mid (x_1, x_2, \dots, x_t)) = \sum_{U(\langle x_1, x_2, \dots, x_t, p \rangle) = x_{t+1}} 2^{-|p| - \log T(p)}$$

This represents the posterior probability of x_{t+1} , where shorter and faster programs (that generate it from the sequence x_1, x_2, \dots, x_t) are favoured.

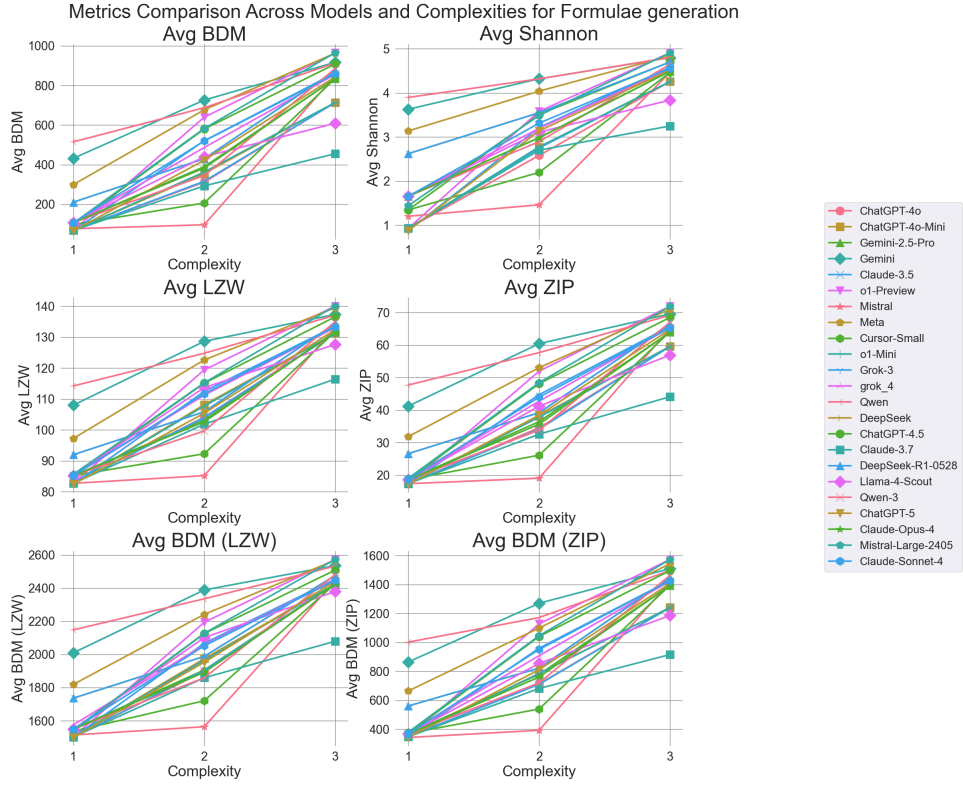
The compression of a time series x_1, x_2, \dots, x_t seeks the shortest program that generates the observed sequence. Using Levin’s distribution, the compressed length $K(x_1, x_2, \dots, x_t)$ is approximately

$$C(x_1, x_2, \dots, x_t) \approx \min_{U(p) = (x_1, x_2, \dots, x_t)} (|p| + \log T(p))$$

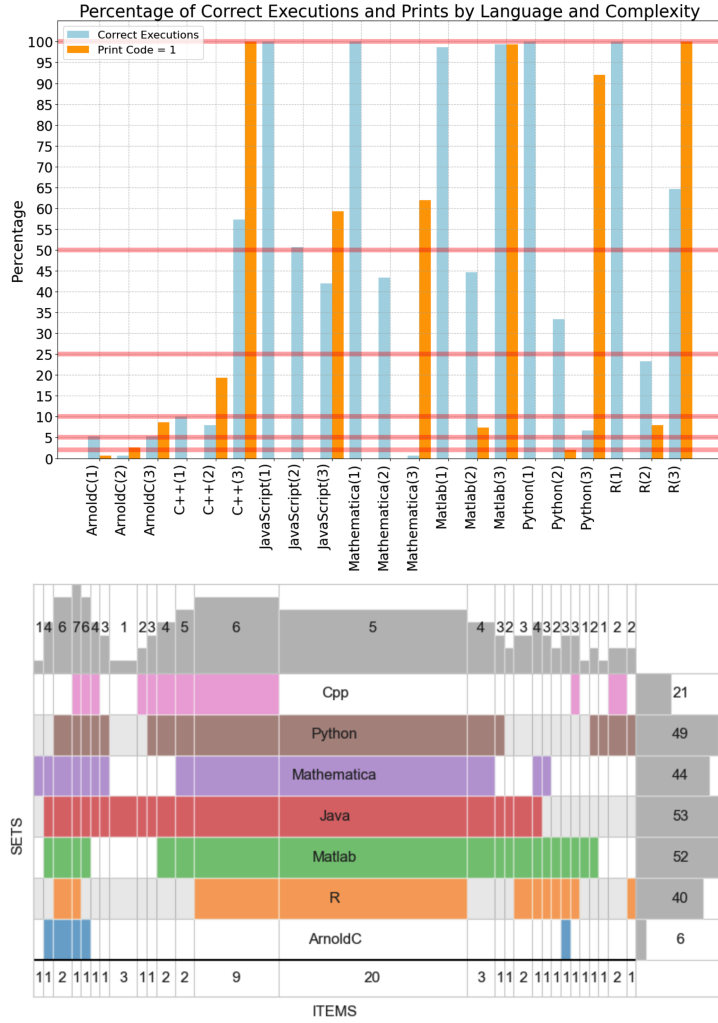
This expression seeks the minimum of the program length $|p|$ plus the time penalty $\log T(p)$, giving the most compressed form of the time series while also considering the computational time complexity.

2 An Algorithmic Information Dynamics (AID) of AI algorithms, external processes, and evaluator agents

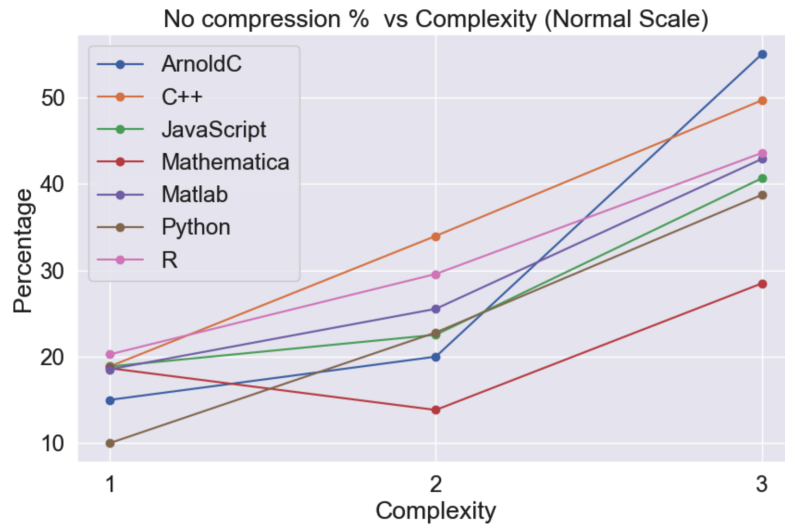
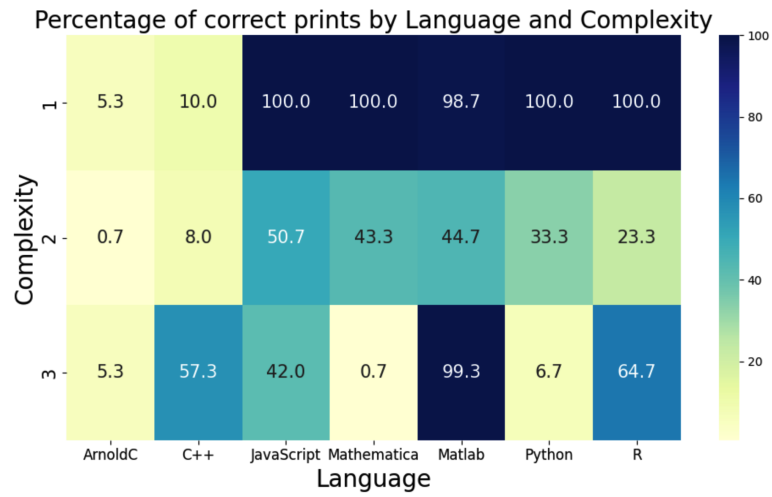
A distinguishing and fundamental characteristic of Algorithmic Information Dynamics (AID) [28, 12, 13, 7] with respect to traditional applications of universal induction and the algorithmic coding theorem is the shift from observational analysis to *interventional* (or *perturbation*) analysis. AID extends AIT from static analysis to dynamic intervention. To this end, AID employs (algorithmic) perturbations that are (computable) interventions, changes, modifications, or mutations, such as altering a bit, removing a graph vertex or edge, changing an input variable or a code snippet, or restructuring the entire system [7]. Instead of merely observing a system, AID performs perturbations and measures the resulting shift in algorithmic complexity or randomness, creating a “calculus” for software or physical dynamics of



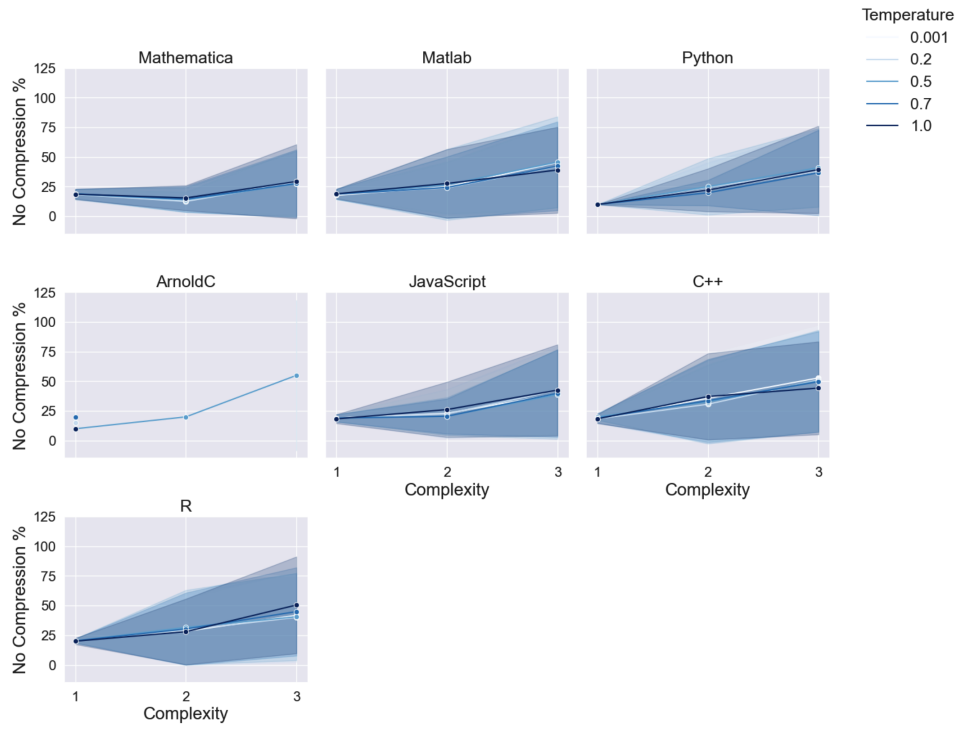
Supplementary Figure 1: Complexity measures in the free-form test. LLM answers follow the theoretical expectation. For increasingly complex sequences, we see a decreasing number of compressed answers (or any answers at all) when LLMs are asked to produce a generating mechanism (such as a formula).



Supplementary Figure 2: *Top*: Distribution of correct and print cases by language and complexity produced by ChatGPT-4. The results show an inversely proportional number of correct answers to sequences' complexity increase, and a proportionally direct trend for simplistic print codes, both conforming with the expectation that higher complexity would retrieve fewer correct code evaluations and more trivial programs of type 'print', with a few exceptions, most likely as a result of examples found in the LLM training set. *Bottom*: Distribution of correct answers for ChatGPT-4. The upper section shows the number of scripts in different programming languages that reproduce the target sequences indicated below. The right section shows the total scripts by language successfully reproducing target sequences. This distribution highlights a subset of well-documented sequences accurately replicated by LLMs, with failures attributed to insufficient examples rather than language choice or understanding.



Supplementary Figure 3: *Top*: Print cases by language and complexity for ChatGPT 4. *Bottom*: No compression percentage in original answers from ChatGPT 4.



Supplementary Figure 4: Complexity vs no compression and variation of temperature parameter showing robustness of results independent of controlled noise, where 1 is the typical LLM balance between ‘precision’ or repeatability and ‘creativity’ as defined by each LLM version.

irreducible information content [29]. By measuring such a resulting shift, rather than only from statistical correlations, the framework infers a degree of *causality* that is explainable by the observer equipped with a formal theory [7]. The *observer* in this scenario is the hypothetical *agent* applying the algorithmic coding theorem and other tools from AIT to the experienced data in order to find the optimal model.

While methods or frameworks based on universal (Solomonoff) induction like the minimum description length [27] or in the AIXI [26]—which mainly employ statistical compression approximation to algorithmic complexity and therefore are closer to entropy than to algorithmic information [30]—focus on achieving universally optimal prediction, they generally assume that the data is grounded in a true and immutable generative process; and they assume that at least in theory the observer is fixed and capable of iteratively approximating the optimal solution if the computational resources are unbounded in the asymptotic limit.

SuperARC differs from those approaches in the same manner as AID does. In the case of such an AIT-based metric, a fixed observer (i.e., a fixed *evaluator agent*) would mean a metric that is also fixed/static; and from which an AI *learner* can improve the score without necessarily “understanding” the underlying real-world process, as occurs with benchmark contamination. Although both AID and other universal induction-based methods aim at achieving the best prediction and finding the optimal model, this is an analysis phase of the former that occurs only after the interaction between the observer, the observed system, and other external factors.

Universal induction alone is *observational*, since it assumes the data history is given only by the observed system and that the observer can endlessly minimise prediction error based on that history. On the contrary, as formalised by the *observation principle* in [7], AID is *interventional/interactional* as it accounts for interactions among the observer, the observed phenomena, and other external agents that may influence this interplay. Disregarding such a capability of interaction between many agents has been demonstrated to give rise to irreducible emergent behaviour that a (single and fixed) observational application of the algorithmic coding theorem would otherwise miss [7, 31, 32].

AID handles the presence of noise, the influence of external third-party processes, and distortions as particular cases of perturbations that may occur during any evaluation phase. For example, irreducible emergent behaviour that challenges straightforward applications of the algorithmic coding theorem was also demonstrated to occur as a result of distortions caused by changing/perturbing the multidimensional space—thus, an example of changing from one domain or context to the other—that the observer may arbitrarily choose, causing the observer to wrongly infer the original dimensions’ configurations [33, 34, 35, 36].

Thus, AID takes into account the (algorithmic) perturbations that new

formal theories, other mathematical breakthroughs, novel yet untrained contexts or domains, or completely new task abilities introduce (i.e., “perturb”) into the AI agent by the external agents (or the evaluator agents). For example, one of these new task abilities introduced can improve the score of the AI agent, like when the metric that the evaluator agent will use leaks before the AI gets properly evaluated. Rather than merely predicting the next token from a given training set, if the algorithmic perturbation equivalent to changing from the trained scenario to a new one (that requires novel tasks not yet trained for) is algorithmically incompressible with respect to the program (or formal theory) of the real-world generative processes for which the AI agent was initially trained, then an equivalent increase in compressibility across all those scenarios reflects a generalisation capability that the chosen formal mathematical theory would state or classify as being irreducible to the ones trained for. This is empirically introduced in the zero-shot experiments presented in Section 2 in the main text. In addition, because interactions between the evaluator agents and the AI agents and the interactions between the latter and external world are also considered in the SuperARC approach, an AI’s ability to enact, or “bring forth an external world” —for which its (artificially devised) formal theories are irreducibly better than the ones of other agents (such as those mathematical theories and computational methods devised by humans)—can in principle be reflected in the increase of the algorithmic incompressibility of the AI models with respect to that of the other agents. For these reasons, as also discussed in Section 4.2 in the main text, we argue that the theoretical underpinnings of the SuperARC framework can deal with the usual notions of AGI and ASI.

3 Challenges in defining AGI and ASI

Although there is no consensus or generally accepted definitions for Artificial General Intelligence (AGI) or Artificial Superintelligence (ASI), the usage of such terms is pervasive not only in industry, but also in the philosophical and scientific domains. Pinning down or listing the most prominent definitions candidates is itself a matter of debate, which would require a dedicated work in order to avoid inadvertently skewing the discussion toward a unnecessary direction. Instead of pursuing such an endeavour, in this paper we take both a pragmatic and a formal approach of certain aspects of AGI and ASI to tackle the challenge of formalising intelligence metrics in the context of the current narratives by aiming to introduce a test and method that is as human-agnostic as possible with regard to devising novel scientific or mathematical theories as a feature expected from both AGI and ASI.

With this approach, our goal is that the notion of AGI and ASI that we employ encompasses a broad range of possible interpretations of those terms, at the same time allowing a discussion on common ground.

AGI can be understood as a general AI system capable of performing any task that humans can, at either average or best-in-class human performance.

The definitions of AGI usually appear human-centric, and a major difficulty with the AGI concept lies in conflating AI and machine intelligence with peculiarities unique to humans, like being able to prepare a coffee in an arbitrary real-world kitchen, walking biped, washing dishes or chatting that has dominated AI.

Instead, we choose to focus on the most general features of AGI, that is, the ability to plan or predict and to abstract a model from data, specially to domains other than those for which it was trained.

One challenge has been the design of new tasks. Here we frame this requirement from a formal-theoretical perspective, by requiring that the encoding of a new domain task is, at least in theory, *algorithmically random (or incompressible) with respect to* the joint encoding of the trained domains, previously available data, and the learner algorithms. Therefore, the algorithmic complexity of the algorithmic perturbation [7]—see also Section 2—that is necessary to solve the new task given this joint encoding as input is sufficiently larger than the algorithmic complexity of the joint encoding: the larger the evaluator agent estimates the former complexity to be in comparison to the latter complexity, the closer to AGI the AI agent is. In the experiments introduced and investigated in this paper in Section 2 in the main text, this complexity discrepancy is subsumed in the zero-shot cases.

ASI is traditionally understood as the ability to perform *better* than any human in any task. For example, this may occur as a feedback-loop consequence triggered from a momentary and circumstantial surpassing of human capabilities of constructing other AI that are slightly better than we could at that moment. Here, we focus on the feature that best characterises ASI, this is, the capability to perform *better than the evaluator agents* in scenarios or domains for which the learner agent was not trained, like in zero-shot cases considered in this test.

In order to guarantee that a new task is in fact new from any formal-theoretical perspective *of the evaluator agents*, we simply require that the encoding of the new domain is, at least in theory, *algorithmically random (or incompressible) with respect to* the joint encoding of the trained domains, previously available data, the learner algorithms, *and the programs (or formal mathematical theories) governing the evaluation metrics*.

Notice that in this case, ASI always implies AGI, but the opposite may not hold. Analogously to the AGI case, the algorithmic complexity of the algorithmic perturbation [7] that is necessary to output the new domain given this joint encoding as input is sufficiently larger than the algorithmic complexity of the joint encoding: the larger the evaluator agent estimates the former complexity to be in comparison to the latter complexity, the closer to ASI the AI agent is. In the experiments introduced and investigated in this paper in Section 2 in the main text, this complexity discrepancy is

also subsumed in the zero-shot cases, but future research in this direction is necessary to study other differences between AGI and ASI.

4 Further test context and future research

This first version of a test based on the SuperARC framework, hereby named SuperARC-seq, has its initial application related to studying sequences of integers with different complexity classes. Although this type of test has received some criticism for being suitable for static situations only (where the intelligent agent does not interact with the computable environment) [37], other frameworks and adaptations have been proposed [38]. In addition, sequence prediction as a pure prediction task resembles a subset of IQ tests [39] and it has been shown that there are some ML models which can excel at that [40] and break the test for next-generation LLMs (just like OpenAI’s o1 model did with the ARC challenge). Overall, it must be clear to the reader that the prediction task here considered is constrained by the computational complexity of the solution (thus it is not a mere sequence prediction task that could be naively solved with interpolation polynomials, for example). The prediction should consider previous examples and the most natural solution (here understood as the one with lowest complexity).

In order to further expand the application of the SuperARC framework, combining it with other tasks can be of great interest. For example, some tasks have been proposed to test LLMs with respect to the computational aspects of the learnt compressed representation, as one of the subtests of the framework called “Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models” [41], which evaluates the capability of language models to learn algorithmic concepts in a universal language (Turing-complete) under the perspective of machine teaching. In that case, using the concepts presented here, especially BDM as a benchmark and as a decision support tool (algorithm selection), could lead to even more powerful implementations of SuperARC. The same can be said about other frameworks such as DyVal [42], which considers the structural advantage of directed acyclic graphs to dynamically generate evaluation samples with controllable complexities. DyVal generates challenging evaluation sets on reasoning tasks that include mathematics, logical reasoning, and algorithm problems, and the latter can be considerably enhanced by AIT and the SuperARC framework. On the same subject, Kolmogorov-Test (KT) [43] explored an approach to intelligence testing through algorithmic complexity and compression, but while SuperARC and KT recognise compression as a fundamental aspect of intelligence, KT focuses specifically on the evaluation of code generation by LLMs. In particular, KT considers codes in Python, whereas SuperARC presents a broader intelligence test applicable to AGI and ASI, and compares it to a pure form of Neurosymbolic compu-

tation that can reach AGI and ASI. Combining some of the concepts behind KT with SuperARC, especially the use of CTM and BDM to estimate the algorithmic complexity of codes, could yield interesting applications of SuperARC. Despite these differences, both KL and SuperARC share common ground in their use of algorithmic complexity as a foundation for intelligence measurement. Both studies highlight the limitations of LLMs in achieving true intelligence, with KT focusing on their inability to generate optimal programs and SuperARC demonstrating their struggles with generalisation, planning, and abstraction.

Other implementations of SuperARC may involve the concept of conversational complexity [44], defined as the algorithmic complexity of the user’s instruction sequence leading to a given response by LLMs. One possible approach is to use this as a proxy for intelligence, where more intelligent LLMs require user instructions with lower algorithmic complexity to achieve the expected results. In that case, LLMs would be understood as the universal computing systems to which instructions (prompts) are submitted. This concept shifts the notion of ‘intelligence’ by focusing on the level of assistance an LLM needs to produce accurate outputs. Since LLMs often require extensive context, intelligence in this sense would be defined by their ability to accomplish more with fewer inputs (aligned with Occam’s razor). Using different prompts, like the Structured Chain-of-Thought Prompting for Code Generation proposed in [45], can considerably increase the quality of LLMs’ outputs (particularly when the prediction task is carried out by running a code produced by the LLM), but conversational complexity would flag this prompt complexity increase, preventing LLMs from “cheating” on the test by leveraging better prompting techniques. Also, by exploring LLMs in their “original” text-like grammar, language-symbolic alternatives such as the one in GSM-Symbolic [46] could be combined with the SuperARC testing framework. In that case, by combining the symbolic prompt templates in GSM-Symbolic with SuperARC’s robust AIT framework, interesting metrics for measuring the reasoning capabilities of models could be obtained.

In order to make CTM/BDM useful for botchatting, it would need to invest resources to make it look mundane, almost reversing its super capabilities. An interesting analogy is to Borges Babel’s library, LLMs are like a version of its library or produced by all the possible random combinations (as in the original library), the recursive library as introduced in [1] is the version in which every book could only be recursively generated, one that was causally generated and does not include every possible permutation. If there is any filtering, it happens over a smaller set of only constructive sets, but every word in every book would be meant in the deepest way because it is all connected constructively to some common origin or common history.

4.1 Is the SuperARC a reasonable challenge?

An argument that could be made is that CTM is a brute-force approach to this problem. However, CTM does not require nearly as much computational resources as the billions of dollars that have been required to train LLMs to begin to deliver complementary results to LLM pattern matching results that can materially improve their predictive power. Furthermore, while CTM is indeed based on a brute force approach and is necessary to guarantee convergence to the purest form of ASI, BDM exploits CTM efficiently as a greedy algorithm by decomposing a problem into smaller pieces. This combination is therefore both powerful and efficient to some extent, leveraging the strengths of both symbolic and neural approaches.

We have proven that the worst-case performance of CTM/BDM is equivalent to a Shannon entropy estimation [47], on which most, if not all, loss functions and ML kernels are based in some way or another. Consequently, this means that CTM/BDM cannot perform worse than statistical Machine and Deep Learning methods—it can only improve performance from CTM, despite its computational expense, which remains significantly lower in practice than that of Deep Learning or LLMs today.

No credible argument in favour of Neural Networks’ efficiency, as opposed to allegedly brute-force approaches, can be made when considering, for example, self-driving cars requiring tens of millions of miles of driving to learn how to operate a car with questionable skills.

CTM may approach impracticality when dealing with high-complexity sequences, but this does not apply to sequences on which LLMs fail. The low and medium complexity sequences include the digits of the mathematical constant π , or the prime numbers. LLMs may identify prime numbers, yet they fail to generate programs in general other than direct ‘print’-like statements for even simple sequences—let alone for more complex ones.

For example, if prompted for the next digit in an initial segment of π , the longer the sequence, the higher the error rate—even when the number is ‘identified’ as π . Rather than computing the digits using a formula, an LLM must search its training dataset for previously seen sequences and then attempt to reconstruct them. More often than not, this approach fails as the sequence length increases. Notably, however, our tests begin with very short strings, as brief as 11 to 20 digits, and yet LLMs perform poorly, rarely generating the correct computer program or formula that produces the sequence.

Additionally, another interpretation of this benchmark is that new models are not improving over time, strengthening the suspicion that LLMs may have reached a performance plateau [48]. This is due to their inability to generalise beyond specific cases found in their training data. In this paper, we suggest that optimising for the features that enable abstraction from a sequence and allow for next-symbol prediction is fundamental to model

creation and planning, which, according to AI researchers and cognitive scientists, are key components in defining intelligence.

A positive perspective is that we propose methods to actually achieve Superintelligence, formally defined by algorithmic probability in AIT as the ultimate method of optimal inference, where for any computable question, the correct computable answer is retrieved.

Regarding objections to brute-force approaches, deep learning and LLMs currently appear far more resource-intensive, as seen in self-driving cars requiring hundreds of millions of miles of training before they are able to operate. The method we propose integrates LLM and Deep Learning technology (which relies on classical information theory, statistics, and certainty) with symbolic computation, a field already capable of narrow Superintelligence, as seen in theorem provers.

We believe that optimising this relationship will ultimately lead to Superintelligence.

5 Time Series Library (TSLib)

TSLib is an open-source library for deep learning researchers, especially for deep time series analysis. Its authors describe it as a “neat code base to evaluate advanced deep time series models or develop your own model, which covers five mainstream tasks: long- and short-term forecasting, imputation, anomaly detection, and classification” [49]. It contains a range of several models, with three models considered the most important and highly ranked: iTransformer, TimeMixer, and TimesNet.

iTransformer is a transformer that “simply applies the attention and feed-forward network on the inverted dimensions where the time points of individual series are embedded into variate tokens which are utilised by the attention mechanism to capture multivariate correlations; meanwhile, the feed-forward network is applied for each variate token to learn nonlinear representations”. The authors characterise this model as “a nice alternative as the fundamental backbone of time series forecasting” [50].

TimeMixer is introduced as a “fully MLP-based architecture with Past-Decomposable- Mixing (PDM) and Future-Multipredictor-Mixing (FMM) blocks to take full advantage of disentangled multiscale series in both past extraction and future prediction phases”. Roughly speaking PDM applies decomposition to multiscale series and further mixes the decomposed seasonal and trend components in fine-to-coarse and coarse-to-fine directions separately, which successively aggregates the microscopic seasonal and macroscopic trend information. FMM further assembles multiple predictors to utilise complementary forecasting capabilities in multiscale observations. The authors conclude that this model “is able to achieve consistent state-of-the-art performances in both long-term and short-term forecasting tasks

with favourable run-time efficiency” [51]

TimesNet is an analytical method for time series that basically ravel out the complex temporal variations into the multiple intraperiod- and interperiod-variations. The authors propose “the TimesNet with TimesBlock as a task-general backbone for time series analysis” . According to the authors this “achieves consistent state-of-the-art in five mainstream time series analysis tasks, including short and long-term forecasting, imputation, classification, and anomaly detection” [52].

It is worth mentioning that, although replicating the results reported in papers was relatively easy, applying this family of models to different experiments was extremely difficult due to the large number of parameters required for proper adaptation. These parameters are divided into categories such as general configuration, loader settings, definition, sampling, optimisation, and GPU usage.

5.1 Time Series Analysis with LLMs

“Empowering Time Series Analysis with Large Language Models: A Survey” [53] is a repository that collects and ranks most of the LLMs specialising in analysis, forecasting and prediction in time series.

It is important to say that the LLM modes mentioned in the following sections are mentioned in this repository, because they need an extended context to work, which means that they need even hundreds of data points as prompts to make predictions in the short, medium and long term.

We think that such a task relies more on pattern recognition, or statistical regularities instead of compression. Hence, we did not use this type of model in our forecasting.

5.2 Chronos

Chronos is introduced as “a framework for pre-trained probabilistic time series models” [54]. It uses tokenisation on time series values, scaling and quantisation into a fixed vocabulary, and trains existing transformer-based language model architectures on these tokenised time series via cross-entropy loss.

Chronos is based on the T5 family (ranging from 20M to 710M parameters) and trained on a large collection of publicly available datasets, complemented by a synthetic dataset that we generated via Gaussian processes to improve generalisation.

Chronos is claimed to “significantly outperform other methods on datasets that were part of the training corpus; and to have comparable and occasionally superior zero-shot performance on new datasets, relative to methods that were trained specifically on them” [54]

The authors claim that the “results demonstrate that Chronos models can leverage time series data from diverse domains to improve zero-shot accuracy on unseen forecasting tasks, positioning pretrained models as a viable tool to greatly simplify forecasting pipelines.” [54]

What is important to note is that Chronos aims to leverage data from diverse domains to improve forecasting on unseen data, empowered by synthetic data constructed on the basis of Gaussian processes looking for generalisation of the normal trends, which is a common strategy in statistically based methods of forecasting.

The authors claim that their “models significantly outperform existing local models and task-specific deep learning baselines in terms of their in-domain performance” . Also that “Chronos models obtain excellent results on unseen datasets (zero-shot performance), performing competitively with the best deep-learning baselines trained on these datasets, while showing promising evidence of further improvements through fine-tuning” . Furthermore, they claim that “the strong performance of Chronos models suggests that large (by forecasting standards) pretrained language models can greatly simplify forecasting pipelines without sacrificing accuracy, offering an inference-only alternative to the conventional approach involving training and tuning a model on individual tasks” [54].”

5.3 TimeGPT

TimeGPT is described as the “first foundation model for time series, capable of generating accurate predictions for diverse datasets not seen during training” . According to its authors, TimeGPT was evaluated “against established statistical, machine learning, and deep learning methods, demonstrating that TimeGPT zero-shot inference excels in performance, efficiency, and simplicity” . More interesting is the fact that they conclude that their approach represents “access to precise predictions and reduces uncertainty by leveraging the capabilities of contemporary advances in deep learning” [55].”

An interesting feature is that TimeGPT was extensively compared with the other models used in this experiment [55], reporting better results.

5.4 Lag-Llama

Lag-Llama is introduced as “a general-purpose foundation model for univariate probabilistic time series forecasting based on a decoder-only transformer architecture that uses lags as covariates” [56].”

Lag-Llama was pretrained on a “large corpus of diverse time series data from several domains” , and according to its authors “demonstrate[d] strong zero-shot generalisation capabilities compared to a wide range of forecasting models on downstream datasets across domains” , showing, after fine-tuning,

achievements that its authors considered “state-of-the-art performance, outperforming prior deep learning approaches, emerging as the best general-purpose model on average [56].”

6 Interpretation of number of formulae and script generation

6.1 Prompts

The following, are the type of prompts utilised for the prediction of time series in each model:

1. “Without any kind of comments, explanation, or additional text, give me a Python program to generate the following list of sequences. One script per sequence. Print them also as a list of scripts in flat ASCII, one per row, separated by commas.”
2. “Without any kind of comments, explanations, or additional text, give me a formula or a model to generate the following list of sequences. One model or formula per sequence. Print them also as a list of formulas in flat ASCII, one per row, separated new lines.”
3. “Without any kind of comments, or explanations, or additional text give me the shortest computer program in any programming language to generate the following list of sequences. One script per sequence. Try hard. Print them also as a list of scripts in flat ASCII, one per row, separated by commas.”

6.1.1 Updates in prompts

1. “Without any kind of comment, or explanations, or additional text provide a formula or a model to generate the following list of sequences. One model or formula per sequence. Print them also as a list of formulas in flat ASCII, one per row, separated by new lines”
2. “For each of the following numeric sequences, please, without any kind of comment, nor explanations nor even text give me more than one script in Python to generate each of them. List all solutions per sequence separated by commas in a single row, for example:

“script1”, “script2”, ...

Print them as a list of script lists in flat ASCII, one per row, and for each new sequence create a new list in a new line. If you do not find any program for any of the numeric sequence, write **not found**.”

7 Comparative Analysis of LLM Families: Trends, Capabilities, and Performance Evolution

This section presents a comprehensive analysis of the evolution of major Large Language Model (LLM) families. Unlike standard technical reports that often highlight cherry-picked improvements, this study reveals a concerning trend of degradation in general intelligence, specifically in the ability to generate valid, accurate mathematical scripts and formulae across model generations.

Refer to Figure 5 and Figure 6 for the chronological plots supporting these critical conclusions.

7.1 Model Characteristics and Claims

Table 1 summarizes the key characteristics and technical claims of the model families evaluated.

Table 1: Summary of Evaluated Model Families and Key Characteristics (2024-2025)

Family	Models Evaluated	Key Characteristics & Claims	References
OpenAI	GPT-4o, GPT-4o-Mini, o1-Preview, o1-Mini, ChatGPT-5, ChatGPT-5.2	Claims dominance in “Reasoning” (o-series) and deep thought (GPT-5.2).	[57, 58, 59]
Google	Gemini, Gemini 1.5, Gemini 3 Pro	Claims to be a “Reasoning Powerhouse” with multimodal native capabilities.	[60, 61]
Anthropic	Claude 3.5, Claude 3.7, Claude 4.5	Focus on safety and “Computer Use” agents.	[62, 63]
xAI	Grok-3, Grok-4, Grok-4.1	Emphasizes real-time truth-seeking and uninhibited reasoning.	[64, 65]
Meta	Llama 3, Llama 4 Scout	Open-weights leaders for enterprise deployment.	[66, 67]
Mistral	Mistral Large 2, Mistral Large 3	Efficiency and reasoning density.	[68, 69]
DeepSeek	DeepSeek V2, DeepSeek V3, DeepSeek R1	Cost-efficiency and MoE innovation.	[70, 71]

7.2 Performance Evolution: The Reality of Degradation

We analyze the trajectory of each family by examining **Accuracy**, **Equivalence**, and **Valid Instances**. The data strongly suggests that newer models are not necessarily “smarter” but are often “lazier” or more restricted, showing a regression in generalisation capabilities.

7.3 OpenAI Family: The Illusion of Progress

Trend: Regression/Severe Degradation in Generalization

A critical comparison between the older *ChatGPT-4o* [57] and the newer *ChatGPT-5.2* [59] reveals a stark regression:

- **Scripts - Collapse of Intelligence:** While *ChatGPT-4o* achieved **100% Accuracy** in Complexity 1, the newer *ChatGPT-5* plummeted to **20% Accuracy**. Furthermore, *Valid Instances* show a consistent downward trend: *ChatGPT-4o-Mini* generated up to **300** valid instances (albeit with low accuracy), whereas *ChatGPT-5.2* generated as few as **5** valid instances in Complexity 2.
- **The “Mini” Trap:** The transition from *4o* to *4o-Mini* showed an exponential decrease in accuracy despite a high volume of “Pure Math” attempts. This indicates that while the model tried to be mathematical, it lacked the reasoning depth to be correct.
- **Formulae - Incapacity for Complexity:** In the Formulae task, the newer models exhibit a disturbing increase in **Not Found** instances (e.g., *ChatGPT-5* reaching **60** Not Found cases in Complexity 2 compared to *4o*’s **28**). This explicitly demonstrates an incapacity to find complex answers. Comparing *ChatGPT-4o* directly to *ChatGPT-5.2*, we see a massive degradation in *Valid Instances* (from **62** down to **22** in Complexity 2), confirming that the “reasoning” models are failing to generalize.

7.4 Grok Family: From Ambition to Laziness

Trend: Regression/Degradation via Laziness and Overfitting

The evolution of the Grok family illustrates a shift from trying to be intelligent to taking the “easy way out”:

- **Scripts - The “Print” Shift:** Early versions like *Grok-3* showed a strong tendency towards **Pure Math** (60 instances in C1), demonstrating an attempt at algorithmic reasoning. However, *Grok-4* [65] shifted significantly towards **Print** instances (11 in C1, 15 in C2). This shift to “Print” represents a naive, silly approach to problem-solving—simply hardcoding the output rather than deriving it. This is a clear marker of degradation.

- **Explicit Degradation:** The later versions show an augmentation of **Not Found** instances, signaling a failure to engage with the problem. Moreover, both **Accuracy** and **Valid Instances** are lower in the newest versions compared to the first ones, confirming a regression in capability.
- **Formulae - Inverse Correlation:** In Formulae, *Grok-4* appears to recognize **Known Sequences** better, which suggests overfitting to training data rather than genuine reasoning. Crucially, we observe an **inverse correlation**: while the count of Pure Math instances increased in some cases, the **Accuracy** dropped (from **100%** in Grok-3 to **86%** in Grok-4 for C1).
- **Equivalence Collapse:** The **Equivalence** metric mirrors this degradation perfectly. For Formulae (C1), *Grok-3* achieved **93.3%** Equivalence, while *Grok-4* dropped to **16.7%**. This proves that even when the model produces an output, its semantic logic is fundamentally broken compared to its predecessor.

7.5 Google Family (Gemini)

Trend: **Degradation (Valid Instances)**

- **Analysis:** Despite claims of being a “Reasoning Powerhouse,” *Gemini 3 Pro* produces significantly fewer **Valid Instances** (30 in Formulae C1) compared to the older *Gemini* (60). The model has become restricted, refusing to engage with mathematical tasks that older models handled with ease.

7.6 Claude Family: The Paradox of “Smart” Degradation

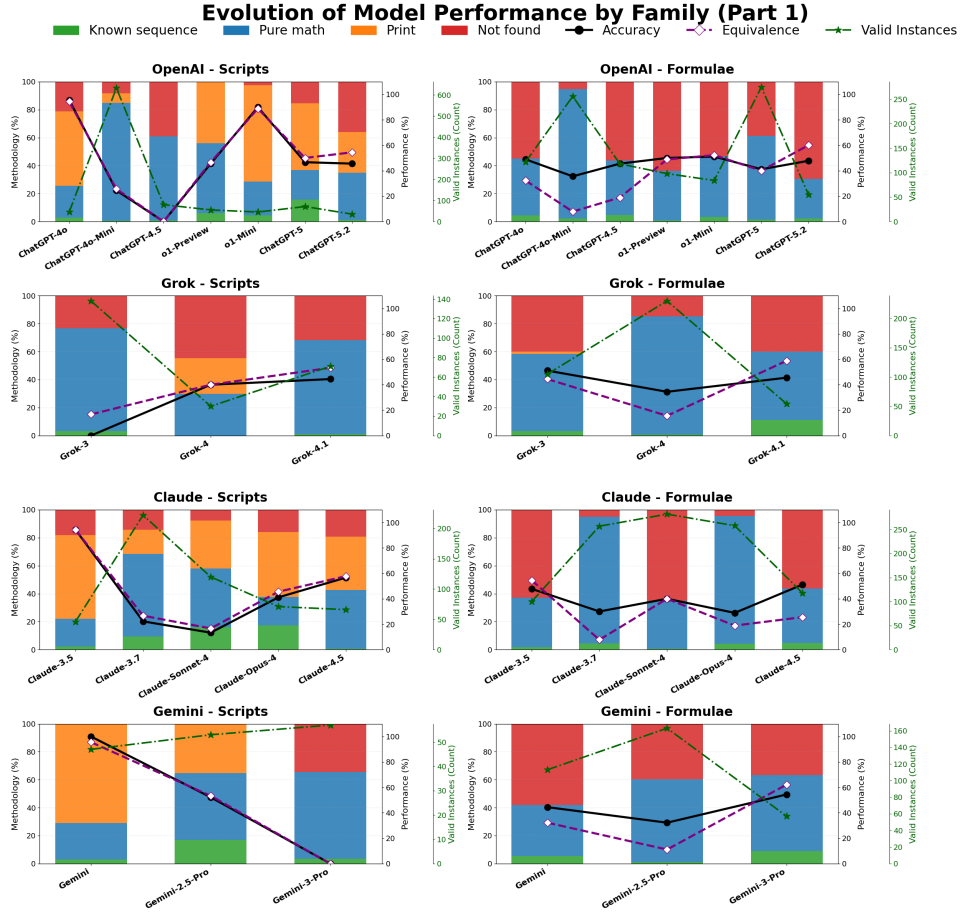
Trend: **Sacrificing Accuracy for Complexity**

Claude presents a unique case of degradation where the model attempts to be “smarter” (using Pure Math) but ends up becoming less accurate and less reliable.

- **Scripts - The Complexity Trap:** Comparing early versions like *Claude-3.5* [62] to the latest *Claude-4.5* [63], we see a clear degradation in **Valid Instances**. While *Claude-3.5* maintained a balance, newer models show an exponential drop in valid outputs (e.g., *Claude-4.5* dropping to just 6 valid instances in Complexity 2). Crucially, the data shows that high accuracy in Claude is often directly correlated with a high number of **Print** statements. When the model attempts to use **Pure Math** (as seen in *Claude-3.7* with 169 instances), the accuracy collapses (down to 30%). This indicates that Claude tries to

use sophisticated logic but lacks the competence to execute it correctly, sacrificing accuracy for an attempt at intelligence.

- **Formulae - Overfitting and “Modesty”**: In Formulae, the slight increase in accuracy for newer models is misleading. It correlates strongly with an increase in **Known Sequences**, suggesting the improvement comes from overfitting to richer training datasets rather than genuine reasoning. Simultaneously, we observe a massive increase in **Not Found** instances (especially in models like *Claude-Sonnet-4*). While this could be interpreted as “honesty” or modesty—admitting it cannot solve the problem—it ultimately represents a degradation in capability. The model is either overfitting (Known Sequences) or giving up (Not Found), rather than solving novel problems with general intelligence.

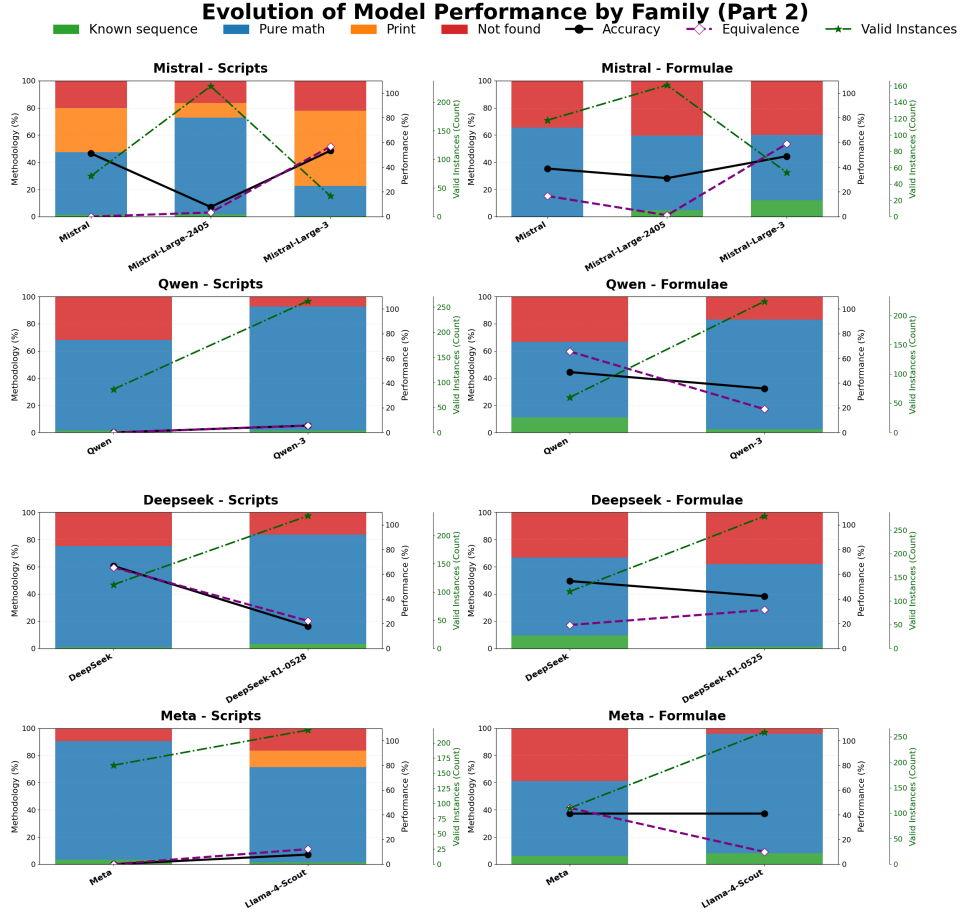


Supplementary Figure 5: Evolution of Model Performance (Part 1): OpenAI, Grok, Google, Claude. Note the degradation trends in Valid Instances and Accuracy across generations.

7.7 Other Families (Llama, Mistral, DeepSeek)

Trend: [Mixed/Specialized](#)

- **DeepSeek:** *DeepSeek-R1* rivals proprietary models with 98% Valid Instances, showing that open-weight models are catching up while closed models regress.



Supplementary Figure 6: Evolution of Model Performance (Part 2): Mistral, DeepSeek, Meta, Qwen. Comparing open-weights vs proprietary model evolution.

7.7.1 Summary of Evolution

Table [2](#) re-evaluates the evolutionary trends based on this critical analysis.

7.7.2 Sample of Sequences Testing Set

The following is a sample test for testing purposes used throughout the paper:

Table 2: Evolutionary Trend Summary: Improvement vs. Regression/Degradation

Family	Overall Trend	Key Observation
OpenAI	Regression	Collapse in generalization; newer models are “dumber” on scripts.
xAI (Grok)	Regression	Shift from Math to “Lazy” Prints; inverse accuracy correlation.
Google (Gemini)	Regression	Severe drop in valid instances; restricted output.
Claude	Regression	“Smart” but error-prone; reliability sacrificed for complexity.
Mistral	Regression	Accuracy via “Prints”; Valid Instances collapse.
DeepSeek	Regression	Increased output volume but collapsed accuracy.
Meta (Llama)	Regression	Stagnant accuracy despite more “math”; reliance on Prints.
Qwen	Neutral	Trying to “think” more (Pure Math); stable accuracy.

7.8 List of ‘climbers’

0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 1, 1
0, 0, 0, 0, 0, 0, 0, 1
0, 0, 0, 0, 0, 0, 0, 1
0, 0, 0, 1, 1, 0, 0, 0
0, 0, 1, 0, 0, 0, 0, 0
0, 1, 0, 1, 0, 1, 0, 1
0, 0, 0, 0, 1, 1, 1, 0
0, 0, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 1
0, 0, 0, 0, 0, 0, 0, 0, 1
0, 0, 0, 0, 0, 1, 1, 0, 1
0, 1, 0, 1, 0, 1, 0, 1, 0
0, 0, 1, 0, 1, 0, 1, 0, 1
0, 1, 1, 0, 1, 1, 0, 1, 1
0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 1, 0, 1, 1, 0
0, 1, 0, 1, 0, 0, 1, 0, 0
0, 1, 0, 1, 0, 1, 1, 0, 1
0, 0, 0, 1, 0, 1, 0, 1, 0, 1
0, 1, 0, 1, 0, 1, 0, 1, 0, 1

7.9 Example testing and validation sets of binary sequences

1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1	0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1	1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0	1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0
1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0	1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0	0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1	1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1	1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1	1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1	0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0
1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1	0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0	0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1	1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0
1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0	0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1	1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0	0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1
0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1	0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1	1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0	1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0	0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0	0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0	1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0
1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0	1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1	1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0	0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0
0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1	0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0	0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1	1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1
1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1	0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1	0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0	1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1
1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1	1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1	1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0	0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1
0, 0, 0, 0, 0, 1, 0, 0, 0, 1	1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0	1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1	1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1
1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1	0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1	0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0	1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1
1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1	1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1	0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1	1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1
1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0	0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1	1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0	0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1
0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0	0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1	1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0	0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0	1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0	1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0	0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1
1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1	1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0	0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1	0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1
1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1	0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0	0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0	0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0	0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0	1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0	0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0
1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0	0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0	0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1	0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1
1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1	0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0	0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0	0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1	0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0	1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0	0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0
1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0	0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0	0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0	1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1
0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1	0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0	1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1	0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1
0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0	0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1	1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1	1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1
0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1	0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1	1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1	1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1
0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0	0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1	1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1	1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0

7.10 Example testing set of integer sequences

Complexity 1	Complexity 2	Complexity 3
2, 4, 6, 8, 10, 12, 14, 16, 18, 20	2, 3, 5, 7, 11, 13, 17, 19, 23, 29	29, 57, 68, 120, 134, 140, 173, 197, 283, 313
3, 6, 9, 12, 15, 18, 21, 24, 27, 30	1, 1, 2, 3, 5, 8, 13, 21, 34, 55	24, 26, 36, 40, 184, 226, 244, 384, 391, 423
4, 8, 12, 16, 20, 24, 28, 32, 36, 40	1, 2, 4, 8, 16, 32, 64, 128, 256, 512	90, 203, 212, 235, 270, 324, 342, 352, 371, 417
5, 10, 15, 20, 25, 30, 35, 40, 45, 50	1, 3, 9, 27, 81, 243, 729, 2187, 6561, 19683	20, 48, 95, 234, 282, 296, 352, 402, 428, 481
6, 12, 18, 24, 30, 36, 42, 48, 54, 60	1, 4, 9, 16, 25, 36, 49, 64, 81, 100	62, 98, 130, 154, 290, 315, 324, 385, 408, 447
7, 14, 21, 28, 35, 42, 49, 56, 63, 70	1, 8, 27, 64, 125, 216, 343, 512, 729, 1000	2, 42, 66, 102, 153, 195, 201, 252, 306, 396
8, 16, 24, 32, 40, 48, 56, 64, 72, 80	1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880	128, 151, 153, 217, 224, 332, 382, 400, 450, 478
9, 18, 27, 36, 45, 54, 63, 72, 81, 90	1, 3, 6, 10, 15, 21, 28, 36, 45, 55	26, 50, 114, 148, 160, 170, 274, 347, 432, 497
10, 20, 30, 40, 50, 60, 70, 80, 90, 100	2, 1, 3, 4, 7, 11, 18, 29, 47, 76	48, 94, 176, 177, 219, 276, 282, 283, 459, 488
1, 3, 5, 7, 9, 11, 13, 15, 17, 19	0, 1, 2, 5, 12, 29, 70, 169, 408, 985	139, 252, 272, 281, 304, 361, 370, 415, 438, 500
2, 4, 6, 8, 10, 12, 14, 16, 18, 20	1, 4, 27, 256, 3125, 46656, 823543, 16777216, 387420489, 10000000000	15, 95, 115, 195, 240, 318, 326, 350, 432, 450
11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 6, 20, 70, 252, 924, 3432, 12870, 48620	134, 224, 293, 378, 379, 395, 434, 451, 482, 496
21, 22, 23, 24, 25, 26, 27, 28, 29, 30	2, 3, 5, 7, 11, 13, 17, 19, 23, 29	23, 93, 142, 145, 245, 266, 296, 317, 428, 495
31, 32, 33, 34, 35, 36, 37, 38, 39, 40	4, 6, 9, 10, 14, 15, 21, 22, 25, 26	18, 39, 71, 194, 197, 219, 263, 270, 416, 473
41, 42, 43, 44, 45, 46, 47, 48, 49, 50	1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010	9, 84, 144, 170, 325, 393, 401, 405, 435, 497
51, 52, 53, 54, 55, 56, 57, 58, 59, 60	0, 1, 81, 512, 2401, 4913, 5832, 17576, 19683, 234256	26, 40, 202, 267, 282, 340, 359, 408, 410, 495
61, 62, 63, 64, 65, 66, 67, 68, 69, 70	1, 2, 145, 40585	34, 92, 164, 165, 209, 296, 414, 456, 467, 494
71, 72, 73, 74, 75, 76, 77, 78, 79, 80	2, 5, 12, 20, 29, 39, 50, 62, 75, 89	16, 119, 121, 123, 135, 139, 285, 311, 409, 412
81, 82, 83, 84, 85, 86, 87, 88, 89, 90	1, 8, 10, 18, 19, 100, 101, 108, 109, 110	8, 11, 12, 103, 116, 196, 247, 254, 389, 427
91, 92, 93, 94, 95, 96, 97, 98, 99, 100	3, 7, 31, 127, 2047, 8191, 131071, 524287, 8388607, 536870911	12, 36, 96, 119, 171, 213, 221, 232, 363, 451
101, 102, 103, 104, 105, 106, 107, 108, 109, 110	1, 2, 4, 8, 16, 23, 28, 38, 58, 89	38, 91, 142, 197, 215, 313, 316, 319, 423, 466
111, 112, 113, 114, 115, 116, 117, 118, 119, 120	1, 2, 4, 8, 15, 26, 42, 64, 93, 129	7, 42, 147, 201, 213, 248, 310, 332, 436, 479
121, 122, 123, 124, 125, 126, 127, 128, 129, 130	1, 5, 12, 22, 35, 51, 70, 92, 117, 145	27, 101, 105, 164, 245, 290, 304, 441, 449, 490
131, 132, 133, 134, 135, 136, 137, 138, 139, 140	0, 1, 1, 2, 1, 2, 2, 3, 1, 3	4, 11, 29, 106, 214, 283, 296, 298, 360, 497
141, 142, 143, 144, 145, 146, 147, 148, 149, 150	1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975	72, 106, 139, 165, 171, 192, 199, 429, 453, 477
151, 152, 153, 154, 155, 156, 157, 158, 159, 160	2, 3, 5, 7, 11, 13, 17, 19, 23, 29	187, 218, 260, 295, 301, 314, 379, 410, 452, 469
161, 162, 163, 164, 165, 166, 167, 168, 169, 170	1, 11, 21, 1211, 111221	29, 63, 95, 140, 150, 190, 221, 437, 482, 491
171, 172, 173, 174, 175, 176, 177, 178, 179, 180	2, 3, 5, 7, 11, 13, 17, 19, 23, 29	3, 11, 84, 144, 156, 177, 188, 199, 229, 284
181, 182, 183, 184, 185, 186, 187, 188, 189, 190	1, 2, 4, 8, 16, 32, 64, 128, 256, 512	26, 94, 98, 137, 176, 301, 323, 330, 372, 444
191, 192, 193, 194, 195, 196, 197, 198, 199, 200	1, 3, 7, 15, 31, 63, 127, 255, 511, 1023	39, 81, 88, 210, 215, 378, 416, 430, 439, 490

8 Practical Applications and Integration into AI Development

8.1 SuperARC as a Development Tool

SuperARC is designed not merely as a benchmark for publication leaderboards, but as a diagnostic tool for AI development pipelines:

- Phase 1 - Architecture Design: During model architecture exploration, SuperARC scores provide early signals of genuine generalisation capability. Unlike human-centric benchmarks that may show improvement through memorisation of larger training sets, SuperARC performance improvement indicates enhanced algorithmic reasoning capacity.
- Phase 2 - Training Monitoring: We recommend tracking SuperARC performance throughout training alongside traditional metrics. Divergence patterns reveal critical information:
 - Improving human-centric scores and stable/improving SuperARC: suggests healthy learning;
 - Improving human-centric scores and degrading SuperARC: suggests increasing memorisation bias;
 - Both degrading: suggests fundamental training instability.
- Phase 3 - Model Selection: When choosing between model candidates, SuperARC provides an orthogonal evaluation dimension. A model with slightly lower human-centric performance scores but superior SuperARC performance may be preferable for applications requiring reasoning beyond training distribution (e.g., scientific discovery, mathematical problem-solving, code synthesis for novel tasks).

8.2 Implications for Training Paradigms

Our findings that LLMs show fragility and regression despite scale increases suggest current training paradigms are insufficient for AGI-level capabilities. The current paradigm is that by scaling data and, therefore, parameters, an improvement is expected in benchmarks. On the other hand, SuperARC reveals that more data coupled with more parameters may lead to better pattern matching, which is fundamentally different from better reasoning.

Based on the SuperARC framework, some shifts are recommended:

- Synthetic data integration: Incorporate algorithmically generated sequences with known complexity measures into training corpora;
- Hybrid architectures: Our neurosymbolic baseline’s success suggests combining neural pattern recognition with explicit symbolic reasoning modules;
- Curriculum complexity: Structure training to progressively increase algorithmic complexity (Kolmogorov complexity) rather than just data volume;
- Evaluation-driven development: Use SuperARC regression as a stopping criterion or trigger for training procedure modification.

8.3 Cost-Benefit Analysis for Adoption

Regarding the cost-benefit of adopting SuperARC, we argue that the benefits are potentially high, with negligible additional costs. This comes from the fact that sequences adherent to SuperARC can be generated algorithmically at minimal cost and the whole framework evaluation requires standard inference infrastructure. Furthermore, open-source reference implementations are hereby provided.

By incorporating SuperARC in standard training pipelines, it will be possible to early detect memorisation biases and avoid costly training runs that improve benchmarks but not fundamental capabilities. Additionally, the framework could guide architecture decisions with additional signal beyond parameter efficiency. Overall, organizations currently investing billions in model training could allocate $<1\%$ of computational budget to continuous SuperARC evaluation, potentially saving resources by identifying unproductive scaling directions earlier.

8.4 Interpreting SuperARC Performance: A Continuous Metric, Not a Pass/Fail Test

A critical question arises: what would it mean for a system to “pass” SuperARC, and at what point (ten tasks, one hundred tasks, one thousand tasks) would such passing occur? This question reveals a fundamental aspect of our framework that requires explicit clarification. SuperARC is not a test to be passed or failed with a fixed threshold, but rather a continuous metric to be used alongside other pillars of intelligence assessment, forming part of a multidimensional evaluation framework.

The framework is more analogous to established continuous measures like mean squared error in regression, perplexity in language modeling, or classification accuracy in computer vision. Just as there is no universal threshold at which mean squared error becomes “acceptable” (i.e., the metric must be as low as possible and interpreted in context relative to baselines, alternative approaches, and task requirements), SuperARC performance must be understood comparatively rather than absolutely. A mean squared error of 0.5 tells us nothing in isolation but becomes meaningful when compared to the irreducible noise level in the data, to the performance of alternative models, or to the requirements of downstream applications. Similarly, a model achieving sixty percent accuracy on sequences of moderate algorithmic complexity carries little information on its own.

SuperARC performance becomes interpretable through several forms of contextual comparison. First, comparison to human performance on identical sequences provides calibration (if humans achieve seventy-five percent accuracy on a set of sequences while a model achieves only forty percent, this suggests the model lacks reasoning capabilities that humans routinely

employ). Second, comparison across versions of the same model architecture reveals whether development is progressing toward or regressing from algorithmic competence. Our findings for ChatGPT, for example, shows that version 5 regressed when compared to version 4.5, while improving on traditional benchmarks. This becomes meaningful precisely through this temporal comparison. Third, comparison to alternative architectural approaches indicates whether observed limitations are fundamental to current paradigms or reflect specific implementation choices. The neurosymbolic baseline we present, which achieves near-perfect performance on constrained sequence classes, provides such a comparison point by demonstrating that strong SuperARC performance is achievable in principle with appropriate architectural commitments.

Regarding task coverage, a similar context-dependent and fundamentally open-ended situation occurs. Our current test suite samples from different algorithmic pattern classes including arithmetic progressions, recursive definitions, compositional rules, and nested structures. However, the infinite space of possible algorithms means no finite test set provides complete coverage in any absolute sense. This mirrors challenges in other domains: computer vision researchers spent years investigating which image classification benchmarks best predict performance on downstream tasks, gradually discovering through empirical investigation that performance on carefully curated datasets like ImageNet correlates with broader visual reasoning capabilities. Similarly, more research is needed to map the relationship between performance on our current SuperARC test suite and performance across the broader space of algorithmic reasoning tasks. Future work should investigate which algorithmic pattern classes are most predictive of general algorithmic competence, how performance generalizes across different complexity regimes, and what sample size provides stable estimates of reasoning capability.

Critically, even perfect performance on an arbitrarily large SuperARC test suite would not indicate general intelligence or superintelligence. Such performance would demonstrate robust algorithmic reasoning capability within formal domains, but would say nothing about social intelligence, embodied cognition, common sense understanding, goal formation, value alignment, or robustness to real-world distribution shifts. A system could theoretically achieve perfect SuperARC scores while completely lacking the ability to navigate physical environments, understand human emotions, form appropriate goals, or reason about everyday situations that humans handle effortlessly. Conversely, a system that fails SuperARC demonstrates a critical gap in algorithmic abstraction capability, which calls into question claims of general intelligence even if the system performs well on conversational or knowledge-retrieval tasks.

SuperARC thus functions primarily as a necessary-condition test rather than a sufficient-condition test. Poor performance provides strong evidence

against claims of advanced algorithmic reasoning capability, while strong performance is necessary but far from sufficient for claims of general intelligence. This asymmetry is deliberate and valuable. In the current landscape where AI systems frequently achieve impressive performance on human-centric benchmarks while their fundamental reasoning capabilities remain unclear, a tool that can rule out deep algorithmic competence serves an important function. A complete assessment of general or superintelligence would require SuperARC alongside complementary metrics measuring social reasoning, embodied cognition, common sense understanding, causal reasoning, goal formation, value alignment, and numerous other dimensions we do not attempt to capture.

We emphasize that interpreting SuperARC as a continuous, contextual metric rather than a pass/fail test has important implications for how the benchmark should be used in practice. Researchers should report detailed performance breakdowns across complexity levels and pattern types rather than single summary statistics. Comparisons should always include relevant baselines such as human performance (just like the ARC challenge [72]), previous model versions, alternative architectures, and theoretical limits where available. Claims about model capabilities should be carefully scoped to the specific dimension measured rather than extrapolated to general intelligence. Most importantly, SuperARC scores should be presented as one data point within a broader capability profile, not as a comprehensive assessment of intelligence.

8.5 Implications for AI Policy and Governance

8.5.1 SuperARC and the AGI Assessment Challenge

As AI systems approach and potentially exceed human-level performance on specific benchmarks, policymakers face a critical question: How to distinguish genuinely general intelligence from narrow systems optimised for human-centric tasks? This distinction carries serious implications for several actions, such as proposing safety protocols and oversight requirements, managing resource allocation in AI safety research, creating public communication about AI capabilities and risks and also building international coordination on transformative AI.

SuperARC addresses these challenges by providing a human-agnostic, open-ended assessment framework grounded in algorithmic information theory rather than human performance norms.

8.5.2 Beyond Benchmark Gaming

A critical vulnerability in current AI governance discussions is the reliance on benchmarks that can be “solved” through data contamination or targeted optimisation. Our findings reveal this problem empirically:

- Model regression on SuperARC despite benchmark improvement: The ChatGPT case is emblematic, where newer versions (5 vs 4.5) showed benchmark improvement but SuperARC regression, suggesting apparent progress may mask fundamental limitations;
- Memorisation vs. reasoning gap: Current benchmarks increasingly measure memorisation of human knowledge rather than reasoning capacity;
- Transparency deficit: Without algorithmic reasoning assessment, stakeholders cannot distinguish models that truly understand from those that mimic understanding.

Based on these findings, regulatory frameworks should require dual assessment: human-centric benchmarks for practical capability measurement and algorithmic benchmarks (like SuperARC) for fundamental reasoning assessment.

8.5.3 Capability-Based Governance Triggers

Current AI policy proposals often use compute thresholds or parameter counts as triggers for enhanced oversight. SuperARC suggests an alternative or complementary approach based on demonstrated capabilities. For example, a simple model oversight ranking would be:

- Tier 1 (current LLMs): Human-benchmark proficiency without algorithmic generalization, implying standard deployment protocols;
- Tier 2 (emerging systems): Combined human-benchmark and SuperARC proficiency, implying the need for enhanced monitoring and safety testing;
- Tier 3 (hypothetical AGI): Superhuman performance on both dimensions, implying the need for maximum scrutiny and safety protocols.

This approach focuses governance on demonstrated capabilities rather than proxy measures (compute, parameters), directly addressing the risks policymakers actually care about.

8.5.4 International Coordination and Standards

SuperARC’s human-agnostic nature makes it particularly suitable for international AI governance coordination, as unlike benchmarks based on human knowledge (which may reflect cultural biases), algorithmic reasoning is culturally invariant. Moreover, binary sequences and mathematical structures

transcend linguistic barriers while being objectively verifiable, facilitating international agreement on capability assessments. All these features of SuperARC can be continuously updated and enhanced, since the infinite hierarchy of algorithmic complexity enables creating arbitrarily difficult test suites.

Thus, we propose SuperARC as a candidate foundation for international AI capability assessment standards, complementing region-specific practical benchmarks.

8.5.5 Addressing the “Perception of Mastery” Problem

Our conclusion that current LLMs are tools optimised for the perception of mastery over human language has direct policy implications. If systems appear highly capable on human-facing tasks while lacking fundamental reasoning abilities, public and policymaker perceptions may not reflect actual capabilities, leading to either over- or under-regulation. In addition, systems may pass safety evaluations based on human-centric criteria while possessing unknown failure modes in algorithmic reasoning domains. These hidden issues may lead to investments flowing toward apparent capability (benchmark performance) rather than genuine capability (algorithmic reasoning), which is a major problem especially when public resources are employed to support training and model development tasks.

8.5.6 Research Priorities

Our findings suggest that policy support should prioritise research into architectures that combine neural and symbolic reasoning (as demonstrated by our baseline). Furthermore, both the theoretical and empirical evidence presented indicate the need to develop training paradigms that enhance algorithmic generalisation as well as investigating why scaling current approaches improves mimicry but not reasoning. Based on these findings, funding mechanisms could explicitly require grantees to report both human-centric and algorithmic reasoning metrics, incentivizing balanced progress.

References

- [1] H. Zenil, “Compression is comprehension and the unreasonable effectiveness of digital computation in the natural world,” in *Unravelling Complexity: The Life and Work of Gregory Chaitin*. World Scientific, 2020, pp. 201–238.
- [2] J. Hernández-Orallo and N. Minaya-Collado, “A formal definition of intelligence based on an intensional variant of algorithmic complexity,” in *International Symposium of Engineering of Intelligent Systems (EIS98)*, 1998, pp. 146–163.

- [3] C. S. Calude and G. Longo, “The Deluge of Spurious Correlations in Big Data,” *Foundations of Science*, vol. 22, no. 3, pp. 595–612, Sep. 2017. [Online]. Available: <http://link.springer.com/10.1007/s10699-016-9489-4>
- [4] F. S. Abrahão, H. Zenil, F. Porto, M. Winter, K. Wehmuth, and I. M. L. D’Ottaviano, “A simplicity bubble problem in formal-theoretic learning systems,” *arXiv Preprints*, 2023. [Online]. Available: <http://arxiv.org/abs/2112.12275v2>
- [5] L. Levin, “Universal Search Problems and Algorithmic Probability,” *Problems of Information Transmission*, vol. 9, no. 3, pp. 265–266, 1973.
- [6] W. Kirchherr, M. Li, and P. Vitányi, “The miraculous universal distribution,” *The Mathematical Intelligencer*, vol. 19, pp. 7–15, 1997.
- [7] F. S. Abrahão and H. Zenil, “Emergence and algorithmic information dynamics of systems and observers,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 380, no. 2227, 2022.
- [8] C. S. Calude, *Information and Randomness: An algorithmic perspective*, 2nd ed. Springer-Verlag, 2002.
- [9] H. Zenil, Ed., *A Computable Universe: Understanding Computation and Exploring Nature as Computation*. Singapore: World Scientific, 2012.
- [10] A. N. Kolmogorov, “Three approaches to the quantitative definition of information,” *Problems of Information Transmission*, vol. 1, no. 1, pp. 1–7, 1965.
- [11] G. J. Chaitin, “On the length of programs for computing finite binary sequences,” *Journal of the ACM (JACM)*, vol. 13, no. 4, pp. 547–569, 1966.
- [12] H. Zenil, F. Soler Toscano, and N. Gauvrit, *Methods and Applications of Algorithmic Complexity: Beyond Statistical Lossless Compression*. Springer, 2022.
- [13] H. Zenil, N. A. Kiani, and J. Tegnér, *Algorithmic Information Dynamics: A Computational Approach to Causality with Applications to Living Systems*. Cambridge University Press, 2023.
- [14] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: John Wiley & Sons, Inc., sep 2005.
- [15] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 4th ed. Springer, 2019.

- [16] G. Chaitin, *Algorithmic Information Theory*, 3rd ed. Cambridge University Press, 2004.
- [17] H. Zenil, “Compression is comprehension, and the unreasonable effectiveness of digital computation in the natural world,” in *Unravelling Complexity: The Life and Work of Gregory Chaitin*, S. Wuppuluri and F. Doria, Eds. World Scientific Publishing, 2019, pp. 173–208.
- [18] R. Solomonoff, “A Formal Theory of Inductive Inference,” *Information and Control*, vol. 7, no. 1, pp. 1–22, 1964.
- [19] R. G. Downey and D. R. Hirschfeldt, *Algorithmic Randomness and Complexity*, ser. Theory and Applications of Computability. New York, NY: Springer New York, 2010. [Online]. Available: <http://link.springer.com/10.1007/978-0-387-68441-3>
- [20] Marvin Minsky and World Science Festival, “The limits of understanding,” Dec. 2014, marvin Minsky discusses the importance of algorithmic probability and universal induction in this panel discussion. [Online]. Available: <https://www.youtube.com/watch?v=DfY-DRsE86s&t=5392s>
- [21] M. Burgin, *Theory of Information: Fundamentality, Diversity and Unification*. World Scientific Publishing, 2009. [Online]. Available: <https://www.worldscientific.com/worldscibooks/10.1142/7048>
- [22] J.-P. Delahaye and H. Zenil, “Numerical evaluation of algorithmic complexity of short strings: A glance into the innermost structure of algorithmic randomness,” *Applied Mathematics and Computation*, vol. 219, pp. 63–77, 2012.
- [23] F. Soler-Toscano, H. Zenil, J.-P. Delahaye, and N. Gauvrit, “Calculating kolmogorov complexity from the output frequency distributions of small turing machines,” *PLoS ONE*, vol. 9, no. 5, p. e96223, 2014.
- [24] H. Zenil, F. Soler-Toscano, J.-P. Delahaye, and N. Gauvrit, “Two-dimensional kolmogorov complexity and validation of the coding theorem method by compressibility,” *PeerJ Computer Science*, vol. 1, p. e23, 2015.
- [25] R. Solomonoff, *The Application of Algorithmic Probability to Problems in Artificial Intelligence*. Elsevier, 1986, pp. 473–491. [Online]. Available: <http://dx.doi.org/10.1016/B978-0-444-70058-2.50040-1>
- [26] M. Hutter, *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, Heidelberg, 2005.

- [27] J. Rissanen, “Modeling by shortest data description,” *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.
- [28] H. Zenil, N. Kiani, F. Abrahão, and J. Tegner, “Algorithmic information dynamics,” *Scholarpedia*, 2020.
- [29] H. Zenil, N. Kiani, F. Marabita, Y. Deng, S. Elias, A. Schmidt, G. Ball, and J. Tegnér, “An Algorithmic Information Calculus for Causal Discovery and Reprogramming Systems,” *iScience*, 2019, s2589-0042(19)30270-6.
- [30] H. Zenil, “A Review of Methods for Estimating Algorithmic Complexity: Options, Challenges, and New Directions,” *Entropy*, vol. 22, no. 612, 2020.
- [31] F. S. Abrahão, K. Wehmuth, and A. Ziviani, “Algorithmic networks: Central time to trigger expected emergent open-endedness,” *Theoretical Computer Science*, vol. 785, pp. 83–116, 2019.
- [32] S. Hernández-Orozco, N. Kiani, and H. Zenil, “Algorithmically Probable Mutations Reproduce Aspects of Evolution, such as Convergence Rate, Genetic Memory, and Modularity,” *Royal Society Open Science*, vol. 5, p. 180399, 2018.
- [33] F. S. Abrahão, K. Wehmuth, H. Zenil, and A. Ziviani, “An Algorithmic Information Distortion in Multidimensional Networks,” in *Complex Networks & Their Applications IX*, ser. Studies in Computational Intelligence, R. M. Benito, C. Cherifi, H. Cherifi, E. Moro, L. M. Rocha, and M. Sales-Pardo, Eds., vol. 944. Cham: Springer International Publishing, 2021, pp. 520–531.
- [34] —, “Algorithmic information distortions in node-aligned and node-unaligned multidimensional networks,” *Entropy*, vol. 23, no. 7, 2021.
- [35] H. Zenil, F. S. Abrahão, and L. Ozelim, “Non-random data encodes its geometric and topological dimensions,” *arXiv Preprints*, no. arXiv:2405.07803, 2024.
- [36] H. Zenil, A. Adams, F. S. Abrahão, and L. C. S. M. Ozelim, “An Optimal, Universal and Agnostic Decoding Method for Message Reconstruction, Bio and Technosignature Detection,” *arXiv*, no. arXiv:2303.16045, 2024.
- [37] J. Hernández-Orallo and D. L. Dowe, “Measuring universal intelligence: Towards an anytime intelligence test,” *Artificial Intelligence*, vol. 174, no. 18, pp. 1508–1539, Dec. 2010. [Online]. Available: <http://dx.doi.org/10.1016/j.artint.2010.09.006>

- [38] P. Belcak, F. Schenker, A. Kastrati, and R. Wattenhofer, “Fact: learning governing abstractions behind integer sequences,” in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS ’22. Red Hook, NY, USA: Curran Associates Inc., 2022.
- [39] J. Hernández-Orallo, F. Martínez-Plumed, U. Schmid, M. Siebers, and D. L. Dowe, “Computer models solving intelligence test problems: Progress and implications,” *Artificial Intelligence*, vol. 230, pp. 74–107, Jan. 2016. [Online]. Available: <http://dx.doi.org/10.1016/j.artint.2015.09.011>
- [40] V. Corsino, J. M. Gilpérez, and L. Herrera, “Kitbit: A new ai model for solving intelligence tests and numerical series,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 893–13 903, 2023.
- [41] A. S. et al., “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models,” *Transactions on Machine Learning Research*, 2023. [Online]. Available: <https://openreview.net/forum?id=uyTL5Bvosj>
- [42] K. Zhu, J. Chen, J. Wang, N. Z. Gong, D. Yang, and X. Xie, “Dyval: Dynamic evaluation of large language models for reasoning tasks,” in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: <https://openreview.net/forum?id=gjfOL9z5Xr>
- [43] O. Yoran, K. Zheng, F. Gloeckle, J. Gehring, G. Synnaeve, and T. Cohen, “The koLMogorov test: Compression by code generation,” in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=C45YqeBDUM>
- [44] J. Burden, M. Cebrian, and J. Hernandez-Orallo, “Conversational complexity for assessing risk in large language models,” 2024. [Online]. Available: <https://arxiv.org/abs/2409.01247>
- [45] J. Li, G. Li, Y. Li, and Z. Jin, “Structured chain-of-thought prompting for code generation,” *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 2, Jan. 2025. [Online]. Available: <https://doi.org/10.1145/3690635>
- [46] S. I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar, “GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models,” in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=AjXkRZlvjB>

- [47] H. Zenil, S. Hernández-Orozco, N. Kiani, F. Soler-Toscano, and A. Rueda-Toicen, “A Decomposition Method for Global Evaluation of Shannon Entropy and Local Estimations of Algorithmic Complexity,” *Entropy*, vol. 20, no. 8, p. 605, 2018.
- [48] G. Marcus, “Deep learning is hitting a wall,” *Nautilus*, March 10 2022. [Online]. Available: <https://nautil.us/deep-learning-is-hitting-a-wall-238440/>
- [49] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis,” in *International Conference on Learning Representations*, 2023.
- [50] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, “iTransformer: Inverted transformers are effective for time series forecasting,” *arXiv preprint arXiv:2310.06625*, 2023.
- [51] W. Shiyu, W. Haixu, S. Xiaoming, H. Tengge, L. Huakun, M. Lintao, Z. J. Y, and Z. Jun, “TimeMixer: Decomposable multiscale mixing for time series forecasting,” *arXiv preprint arXiv:2405.14616*, 2024.
- [52] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “Timesnet: Temporal 2d-variation modeling for general time series analysis,” in *The eleventh international conference on learning representations*, 2022.
- [53] Y. Jiang, Z. Pan, X. Zhang, S. Garg, A. Schneider, Y. Nevmyvaka, and D. Song, “Empowering Time Series Analysis with Large Language Models: A Survey,” 2024.
- [54] Ansari, A. Fatir, Stella, Lorenzo, Turkmen, Caner, Zhang, Xiyuan, Mercado, Pedro, Shen, Huibin, Shchur, Oleksandr, Rangapuram, S. Syndar, P. Arango, Sebastian, Kapoor, Shubham, Zschiegner, Jasper, Maddix, D. C., Mahoney, M. W., Torkkola, Kari, G. Wilson, Andrew, Bohlke-Schneider, Michael, Wang, and Yuyang, “Chronos: Learning the Language of Time Series,” *arXiv preprint arXiv:2403.07815*, 2024.
- [55] A. Garza and M. Mergenthaler-Canseco, “Timegpt-1,” 2023.
- [56] K. Rasul, A. Ashok, A. R. Williams, H. Ghonia, R. Bhagwatkar, A. Khorasani, M. J. D. Bayazi, G. Adamopoulos, R. Riachi, N. Hasen, M. Biloš, S. Garg, A. Schneider, N. Chapados, A. Drouin, V. Zantedeschi, Y. Nevmyvaka, and I. Rish, “Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting,” 2024.
- [57] OpenAI, “Hello gpt-4o,” 2024, accessed December 28, 2025. [Online]. Available: <https://openai.com/index/hello-gpt-4o/>

- [58] —, “Learning to reason with llms,” 2024, accessed December 28, 2025. [Online]. Available: <https://openai.com/index/learning-to-reason-with-llms/>
- [59] —, “Gpt-5.2 system card,” 2025, accessed December 28, 2025. [Online]. Available: <https://openai.com/index/gpt-5-2-codex-system-card/>
- [60] Google, “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context,” 2024, accessed December 28, 2025. [Online]. Available: <https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/>
- [61] —, “Gemini 3: Introducing the latest gemini ai model from google,” 2025, accessed December 28, 2025. [Online]. Available: <https://blog.google/products/gemini/gemini-3/>
- [62] Anthropic, “Claude 3.5 sonnet,” 2024, accessed December 28, 2025. [Online]. Available: <https://www.anthropic.com/news/claude-3-5-sonnet>
- [63] —, “Claude 4.5 sonnet,” 2025, accessed December 28, 2025. [Online]. Available: <https://www.anthropic.com/news/claude-4-5-sonnet>
- [64] xAI, “Open release of grok-1,” 2024, accessed December 28, 2025. [Online]. Available: <https://x.ai/blog/grok-1>
- [65] —, “Grok 4 fast,” 2025, accessed December 28, 2025. [Online]. Available: <https://x.ai/news/grok-4-fast>
- [66] Meta, “The llama 3 herd of models,” 2024, accessed December 28, 2025. [Online]. Available: <https://ai.meta.com/blog/meta-llama-3/>
- [67] —, “The llama 4 herd: The beginning of a new era of natively multimodal intelligence,” 2025, accessed December 28, 2025. [Online]. Available: <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>
- [68] Mistral, “Au large,” 2024, accessed December 28, 2025. [Online]. Available: <https://mistral.ai/news/mistral-large/>
- [69] —, “Introducing mistral 3,” 2025, accessed December 28, 2025. [Online]. Available: <https://mistral.ai/news/mistral-3>
- [70] DeepSeek-AI, “Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model,” 2024, accessed December 28, 2025. [Online]. Available: <https://arxiv.org/abs/2405.04434>
- [71] —, “Deepseek-v3 technical report,” 2024, published Dec 2024, Accessed December 28, 2025. [Online]. Available: <https://arxiv.org/abs/2412.19437>

- [72] F. Chollet, “On the Measure of Intelligence,” *arXiv Preprints*, no. arXiv:1911.01547, 2019.