nature communications

Article

https://doi.org/10.1038/s41467-026-73289-5

SuperARC: a test for artificial superintelligence based on compressed modelling, recursive prediction and problem complexity

Received: 23 June 2025

Accepted: 8 May 2026

Published online: 03 June 2026

Check for updates

Alberto Hernández-Espinosa[1,2], Luan Ozelim[1,2], Felipe S. Abrahão[1,2,3,4] & Hector Zenil[1,2,5,6]

We introduce an increasing-complexity, open-ended, and human-agnostic metric to evaluate foundational and frontier AI models in the context of Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI) claims. Unlike other tests that rely on human-centric questions and expected answers, or on pattern-matching methods, the test here introduced is grounded on fundamental mathematical areas of randomness and optimal inference. We argue that human-agnostic metrics based on the universal principles established by Algorithmic Information Theory (AIT) formally framing the concepts of model abstraction and prediction offer a powerful metrological framework. When applied to frontiers models, the leading LLMs outperform most others in multiple tasks, but they do not always do so with their latest model versions, which often regress and appear far from any global maximum or target estimated using the principles of AIT defining a Universal Intelligence (UAI) point and trend in the benchmarking. Conversely, a hybrid neuro-symbolic approach to UAI based on the same principles is shown to outperform frontier specialised prediction models in a simplified but relevant example related to compression-based model abstraction and sequence prediction. Finally, we prove and conclude that predictive power through arbitrary formal theories is directly proportional to compression over the algorithmic space, not the statistical space, and so further AI models' progress can only be achieved in combination with symbolic approaches that LLMs developers are adopting often without acknowledgement or realisation.

As we enter an AI test saturation phase, where all AI models claim to be the best or front runners on all available AI tests, new tests ideally orthogonal to current ones have to continue to be developed that can keep up with new models to keep increasingly challenging them.

Historically, humans have been heavily biased to believe that the way humans think and act represents the acme of intelligence, and there remains the philosophical and scientific question of the extent to which we can achieve more objective or less human-centric measures of intelligence.

The impressive performance of large language models (LLMs) as language processing and generation tools evinces that language and other areas of human intelligence may be overrated and can be, in fact,

A full list of affiliations appears at the end of the paper. e-mail: hector.zenil@kcl.ac.uk

more dependent than we thought on aspects of memorisation and statistical pattern matching.

A common psychological perspective sees intelligence through the lens of IQ tests; one of the first was the *g*-factor, a psychometric construct introduced by Spearman[1] that quantifies the positive correlations between cognitive abilities. This framework is consistently linked to a human-centric perspective of what intelligence is and, therefore, biased towards circular reasoning. In the context of AI, some LLM benchmarks test for different factors, with several benchmarks based on correct answers versus hallucinations; some of which are also very human-centric metrics related to humans' biological peculiarities and shared history.

Some scholars argue that intelligence can be objectively defined through tests that evaluate specific computational abilities essential to demonstrate intelligent behaviour, rather than trying to define intelligence itself in absolute terms[2–6]. This perspective shifts the focus from an abstract or philosophical definition to a practical, measurable framework assessing an entity's capacity for problem-solving, pattern recognition, and adaptive learning within a structured system.

This reflects an operational turn in the study of intelligence, emphasising the design of formal benchmarks and quantifiable metrics. However, this approach is not without philosophical challenges. By reducing intelligence to observable outputs, it risks overlooking the role of internal representation, consciousness, or semantic understanding—dimensions emphasised in critiques like Searle's.

An approach toward tackling such issues is to ground intelligence metrics in more fundamental notions of computation and mathematics. For example, the concepts of randomness, prediction and inference as defined by Gregory Chaitin, Andrey Kolmogorov, and Ray Solomonoff. Chaitin[7] proposed that formal definitions of intelligence and its components should be based on algorithmic complexity, a formal mathematical theory able to define the concept of randomness as opposed to intelligence. Similarly, Solomonoff[8] advanced the idea of evaluating intelligence through algorithmic probability, laying the foundation for optimal prediction frameworks (or universal "Bayesian" inference). These approaches further motivated approaches such as Hutter's AIXI[9] in an attempt to reconcile objective evaluation with theoretical generality in the context of learning. Algorithmic complexity, algorithmic probability, and algorithmic randomness comprise the most important concepts in algorithmic information theory (AIT)[10–13] and provide the accepted mathematical definitions of randomness and optimal inference (induction and abduction), going beyond simplistic statistical tests based on methods such as GZIP or LZW, popular pattern-matching compression methods that are more closely related to Shannon entropy than to model synthetization and predictive inference. At recent public events, speaking about the foundations of AI and AGI, some leaders in the AI industry have drawn strong parallels between algorithmic complexity, data compression, and AI[14,15]. Although these terminologies, such as AGI and ASI, are currently loosely defined in the scientific literature, these claims and the current understanding make the connection between LLMs (or any other generative AI), algorithmic complexity, and data compression clearer and more explicit, even calling it fundamental for general and super intelligence, artificial or natural.

Based on these arguments connecting intelligence to recursive compression[6], some tests for machine, human, and non-human entities have been proposed in refs. 5,16,17. Section " Compression as comprehension about (and as part of) the world" presents a reflection on that property of intelligence to involve the identification of recursive patterns, planning from prediction, and the generation of concise explanations for observed complex phenomena. Recursive compression here means the ability to represent an observation in a condensed manner by taking advantage of aspects of the data's regularities beyond statistical pattern matching. This is, by selecting and keeping as many as possible, the features that make the explanation executable and predictive of the explanandum's future states.

Despite the interesting theoretical arguments that could be drawn from these connections, one argument is that they would only be valid under idealised conditions (unbounded data access/storage, perfect optimisation, appropriate inductive biases), which are rarely met in practice. As seen in real-world problems, even simple datasets with specific distributions can lead to optimisation toward local minima that do not correspond to minimal algorithmic descriptions.

Closely related ideas are also in evidence in Schmidhuber's Gödel machines[18] work and Hutter's AIXI[9] based on Levin's search[19] and the principles of algorithmic probability[12,20,21]. Similarly to a test proposed in ref. 22, a benchmark designed to evaluate conceptual understanding in machine learning models was proposed, consisting of a diverse set of tasks that indirectly assess a model's capacity for abstraction, requiring it to generalise beyond memorisation[23]. These tasks challenge models to reason both interpolatively (by making sense of patterns within observed data) and extrapolatively (by extending learned principles to novel scenarios). Although interesting and a first approach, the test lacked robust foundations of algorithmic information, nor were they applied to frontier models. See also Section 4.2 for further theoretical challenges and developments. In a previous work, we successfully explored some of these ideas, proving that we can perform this search on non-differentiable spaces using metrics purely based on algorithmic complexity to search for those programmes in model space, making the previously considered fundamental requirement of differentiability redundant[24].

Building on previous work in related applications in the context of algorithmic inference and universal intelligence[2,4,5,25,26], here we introduce a quantitative test for any AI model that aims at universal and agnostic optimisation with an application to LLMs fully framed in terms of the principles and foundations of AIT together with perturbation analysis from Algorithmic Information Dynamics[27–29] (see Supplementary Information). Our framework is related to tests such as the ARC-AGI tests/challenge[30], but it is agnostic to: the chosen set of problems, since it does not pick specific test cases; the underlying formal theories that define or characterise the evaluation tools; the chosen observer/evaluating agents; and the chosen interacting agents or external input. It avoids the devise of a metric that is fixed, whose theoretical principles are not assumed to evolve together with the tested subjects, thereby allowing the test to become the target and no longer useful (see also section "Results" and Supplementary Information)—therefore, a test for what can be understood as AGI and ASI.

While we do not assume that connections to compression as model abstraction and prediction as planning are necessary but not necessarily sufficient for general intelligence, these qualities have recently been strongly associated with AI, AGI, and ASI[31,32].

Our framework adopts a specific theoretical perspective on intelligence, but surely does not capture all aspects of human cognition. The test here introduced is meant to challenge aspects of AI (e.g., LLMs) by putting forward mathematical theory and methods related to the properties of intelligence believed to be key for intelligence, in particular AGI or ASI, such as model abstraction and planning, as in model synthesis (new explanatory models) and as in its recursive prediction capabilities. Although we propose a method in simplified contexts, this is without any loss of generality to any other type of data.

We claim that the feature of increasing complexity makes the test robust to benchmark contamination and test targeting and can account for improvements due to external intervention, while its ultimate uncomputability nature provides the desired open-endedness for a feature likely to be as complex as what is trying to capture and evaluate. In other words, an equally complex and open test for an equally complex and open attribute, intelligence.

Results

The tests were inspired by, and based on, two methods called *Coding Theorem* and *Block Decomposition* methods (CTM and BDM)[33–35], which use a composition of pattern-matching and running a very large set of small computer programmes to approximate a distribution allowing the estimation of the algorithmic complexity of short objects providing insights into the minimal description length of objects such as strings, sequences or images.

These methods have been applied to humans before on similar tasks and the same tools (CTM/BDM), showing that the methods can capture aspects that other intelligence tests based on pattern matching fail or require much more ad hoc information and assumptions to reproduce them[2]. Humans showed some predictive capabilities that could be quantified only with CTM/BDM and not with other statistical tools. This led CTM and BDM to be widely used today in the psychometric testing space by multiple groups[36–38].

Among the results in ref. 2, people showed that at 25 years of age they had the highest ability to identify and produce the highest random complexity when required, and decreased when older. More experiments should be conducted to test humans' recursive prediction capabilities, but what the article showed[2] was that people built and had better perception models for producing and identifying random versus non-random content when they were 25 than at any other age and therefore were more sensitive to identifying less predictive data, all of which are compatible with current knowledge of human cognitive trends and capabilities.

While we do not necessarily expect humans to perform well on predictive tests such as those introduced here, humans are, in principle, mechanistically capable of solving them. As such, these tasks are arguably within the definition of AGI as a system possessing the full range of human capabilities, without time constraints. Moreover, in the context of ASI, the natural assumption is that of optimal prediction, for which Algorithmic Information Theory (AIT)–used and exploited here for testing purposes–is the accepted mathematical framework.

Next-digit prediction task with binary and non-binary sequences

The objective of this experiment is to test a fundamental property of LLMs, that is, the prediction of the next token, within the context of Algorithmic Information Theory (AIT), the accepted mathematical theory governing optimal prediction, as introduced in the Sup Inf.

We tasked large language models (LLMs) specialising in time series prediction with predicting the final digit of both non-binary sequences and binary sequences, the latter of which were categorised as either random or "climber" sequences. The results of the experiment involving binary sequences are presented in Fig. 1.

We call "climbers" those sequences that have some recursive properties that make them stand out and have been properly ranked as of lower complexity, given their recursive properties according to CTM/BDM (see Supplementary Information). As shown in Fig. 1, in the case of what we call "climbers", Lag-Llama achieved the best performance, with 70% precision, while TimeGPT-1 and Chronos barely reached 50% precision, while CTM/BDM was used as a gold standard.

However, for random sequences, which are considered highly 'complex' in this context, all models performed similarly, showing limited predictive power, as expected. These results suggest that, given the binary nature of the sequences, the models had a 50% chance of predictive success, effectively reducing the task to guessing, yet some performance difference was noticed for non-random cases, indicating some predictive ability. However, their performance aligns with broader research that indicates that LLM models do not effectively capture sequential dependencies or complex patterns inherent in time series data. As highlighted by Tan et al.[39], despite their computational intensity, LLMs often fail to outperform simpler models, particularly when there is high complexity or randomness in the data.

A comparable analysis was conducted using LLMs specialised in time-series data, using non-binary sequences of increasing complexity. In this test, a specific percentage of the final numbers in each sequence was required to be predicted. Three distinct metrics were utilised: general similarity, sort similarity, and the Levenshtein distance (refer to the section "Next-digit prediction task" for its definition). Figure 2 presents the results, where sort similarity and general similarity exhibit closely aligned trends. This indicates that the predictive accuracy of LLM models, even when fine-tuned for numerical series, diminishes as the complexity of the sequences increases. The resemblance between sort similarity and general similarity implies that while predictions may include some of the expected numbers, their correct order remains equally critical and may not always be achieved.

In Fig. 1, sequences chosen for each complexity class follow a pattern of increasing complexity in all cases, according to both statistical and algorithmic measures. This was by design to test the LLM's capabilities in solving similar problems of increasing complexity, divided into three complexity groups.

This observation is corroborated by the findings from the Levenshtein distance metric, which quantifies the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one sequence into another. As the complexity of the sequences rises, so does the Levenshtein distance, further confirming that predictive accuracy deteriorates with increasing complexity. Fig. 1 (bottom) shows an increase in complexity as expected, given the design of each group of generated sequences. The plot suggests that BDM can capture (and can generate) better complexity and randomness, since its values increase more consistently as complexity increases, unlike other measures. Shannon entropy-based measures (and cognates) can account for statistical randomness only. Compression algorithms, for example, decrease as complexity increases, becoming more difficult to find regularities and increasing compression length as a function of complexity growth.

Free-form generation task with non-binary sequences

A subsequent analysis focused on the free-form test, where LLMs were given complete freedom to generate any model or formula capable of producing target sequences of increasing complexity.

In the Supplementary Information, we provide the plots of complexity-related metrics for the models and formulas generated by LLMs used in this research. The metrics evaluated include the length of the LZW-compressed model, the length of the ZIP-compressed model, the BDM of both the uncompressed model and its LZW and ZIP-compressed forms, and the Shannon entropy of the model.

The plots reveal a clear positive correlation between model complexity and the metric values as the complexity of the target numerical sequence increases. Specifically, as the complexity of the sequence grows, the length of both LZW and ZIP-compressed representations increases, suggesting that the LLM-generated models become larger and less compressible. This indicates that the models provided by the LLMs become unable to compress and then to understand the logic behind sequences, giving as a result the sequence itself.

The BDM values (for the raw, LZW, and ZIP models) also exhibit an incremental trend, further supporting the observation that the LLMs generate less structured models when faced with more intricate sequences. Additionally, the Shannon entropy values rise with complexity, highlighting the increase in unpredictability or information content within the models as they attempt to approximate more complex patterns.

https://doi.org/10.1038/s41467-026-73289-5

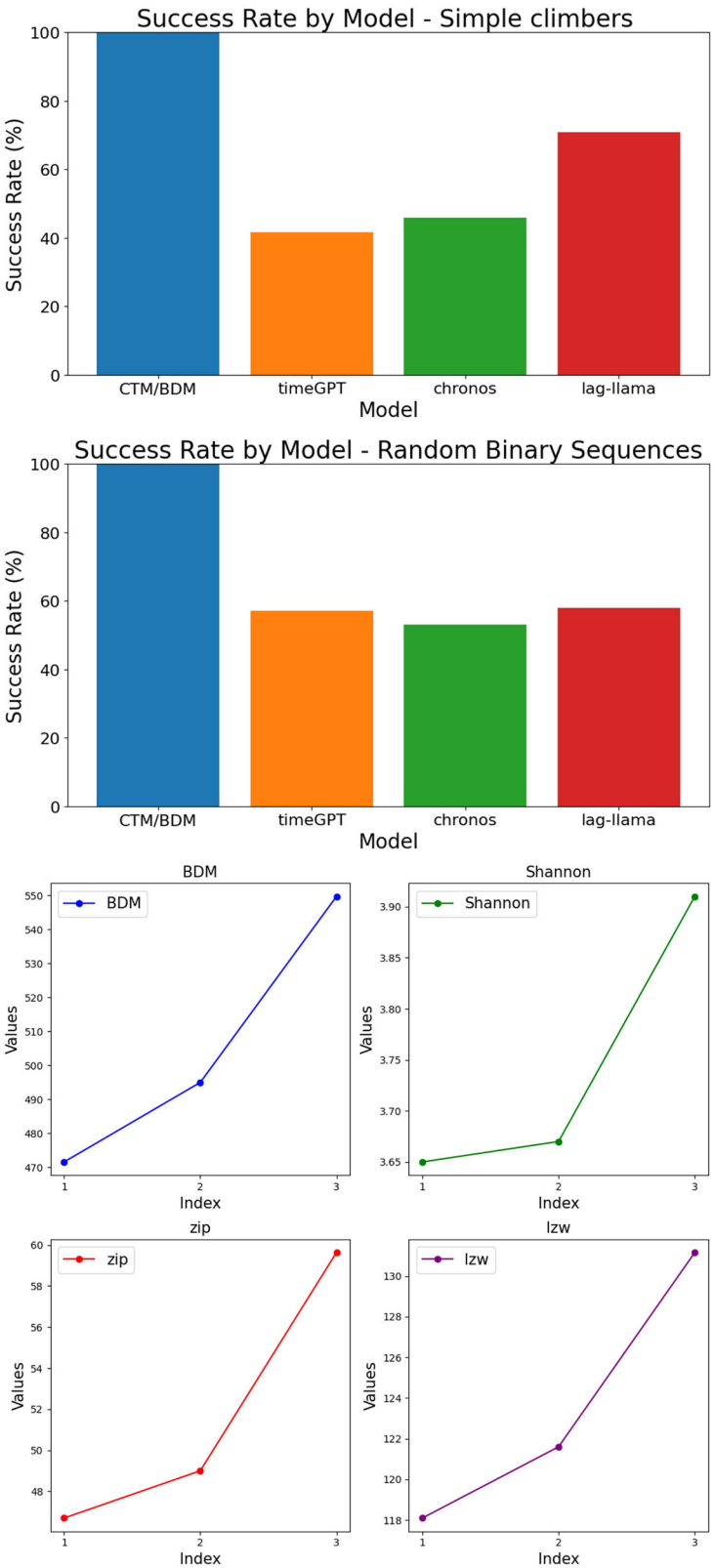

Fig. 1 | Top: Percentage of accuracy of model abstraction and prediction for binary sequences by time-series prediction models against BDM. That climbers (up) were better predicted is expected from models that are able to better characterise and predict simpler sequences. Bottom: Quantitative agreement of monotonic sequence increase of complexity: Comparison of BDM, Shannon entropy, average length of Zip and LZW over the time series generated to test LLMs.

These findings suggest that the LLMs struggle to produce compact or efficient models as the complexity of the target sequence increases. The uncompressed models generated by the LLMs become longer and less structured, as indicated by the rise in all metrics. This reflects a limitation in the LLMs' ability to discover or generate concise, elegant models for more complex sequences. Instead of producing simpler, more generalisable formulas, the LLMs resort to more convoluted representations, indicating a lack of sophistication in their

https://doi.org/10.1038/s41467-026-73289-5

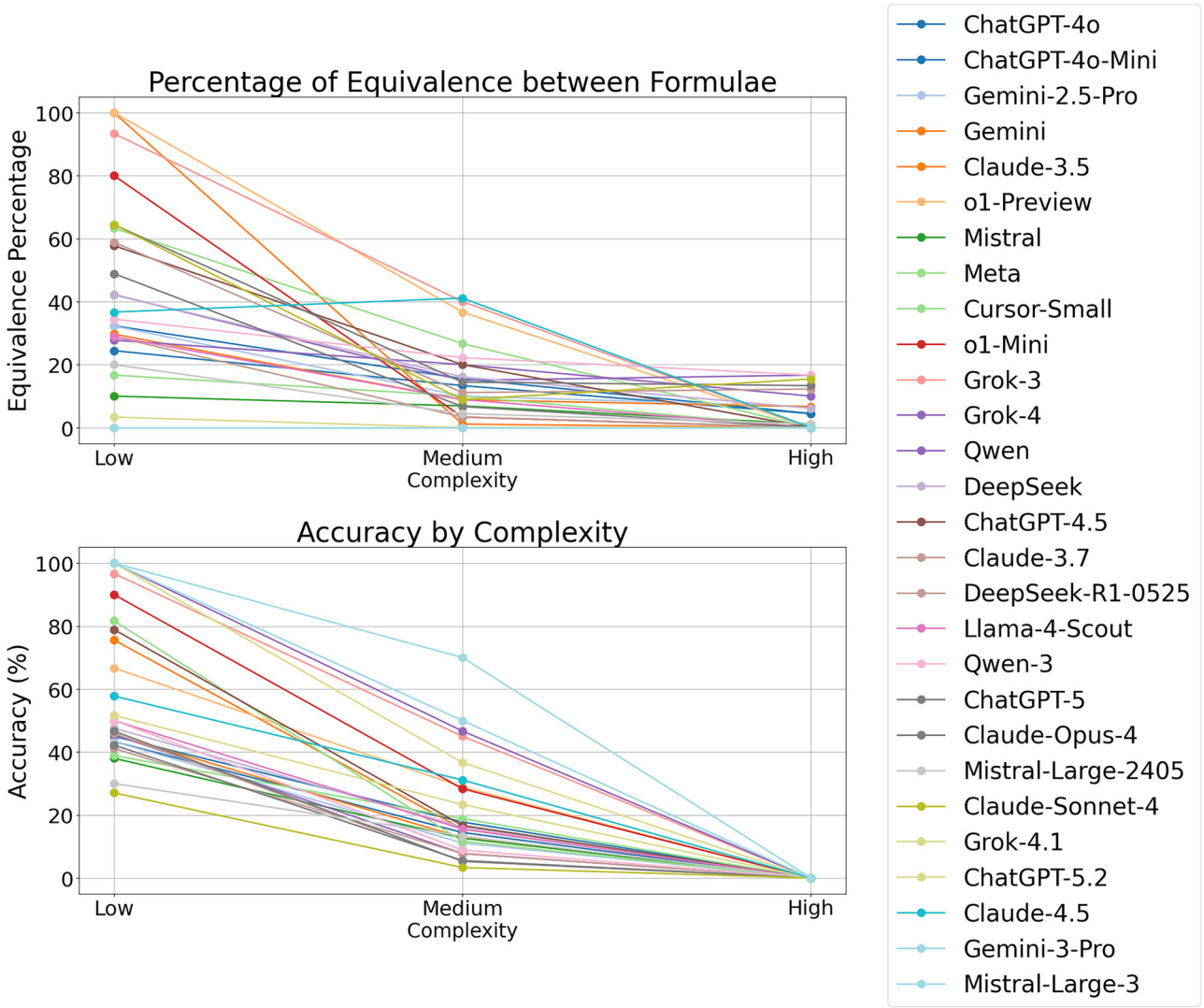

Fig. 2 | Similarity over predictions with Chronos, TimeGPT-1 and lag-llama. See the "Methods" section and descriptions in the Supplementary Information.

capacity to identify or generate models that optimally balance complexity and brevity.

Emergent abilities. Another experiment aimed to evaluate characteristics recently attributed to LLMs, particularly their so-called emergent abilities, which include innovation, discovery, and improvement (see also section "Compression as comprehension about (and as part of) the world" and Supplementary Information). These attributes have been claimed to enable LLMs to perform at levels comparable to the human top 1% in fluency and originality, as suggested by Zhao et al. in their assessment of creativity in artificial intelligence systems[40].

The experiment tested these claims by challenging LLMs to generate multiple, diverse approaches to reproducing non-binary sequences of varying complexity. The underlying rationale was that originality often stems from the ability to perceive problems in new, unexpected ways. Thus, the test focused on measuring the variety and creativity of outputs, as well as the models' capacity to discover innovative or unconventional solutions.

Two distinct tasks were designed for this evaluation. In the first, models were asked to create any type of formula or mathematical model capable of replicating the target sequences. In the second, models were tasked with writing Python scripts to achieve the same goal. By incorporating these variations, the experiment sought to assess the models' adaptability, computational reasoning, and creative potential across different problem-solving paradigms.

The results are shown in Figs. 3–6, where the following classification of cases was used:

1. *Known Sequences:* using standard algorithms such as Fibonacci or primes.
2. *Pure Math:* using mathematical operations without predefined sequence knowledge.
3. *Not Found:* inability to produce outputs.
4. *Print Scripts:* (only for script generation) trivial solutions directly printing the target sequence.

When it came to the production of different models or formula tests, while Gemini, Claude-3.5-Sonnet, and ChatGPT-1o performed relatively well, they ultimately shared the same core limitations as other models. In contrast, Meta and Mistral consistently underperformed, exposing disparities in baseline capabilities among LLMs.

Code generation task with non-binary sequences

For this experiment, one of the main metrics we measured was accuracy, which refers to the proportion of programmes in different programming languages generated by ChatGPT that, after compilation and/or execution, produce the target sequence of digits. Supplementary Figures show that correct programmes are more common at the lowest levels of complexity, with some minor exceptions. In the Supplementary

https://doi.org/10.1038/s41467-026-73289-5

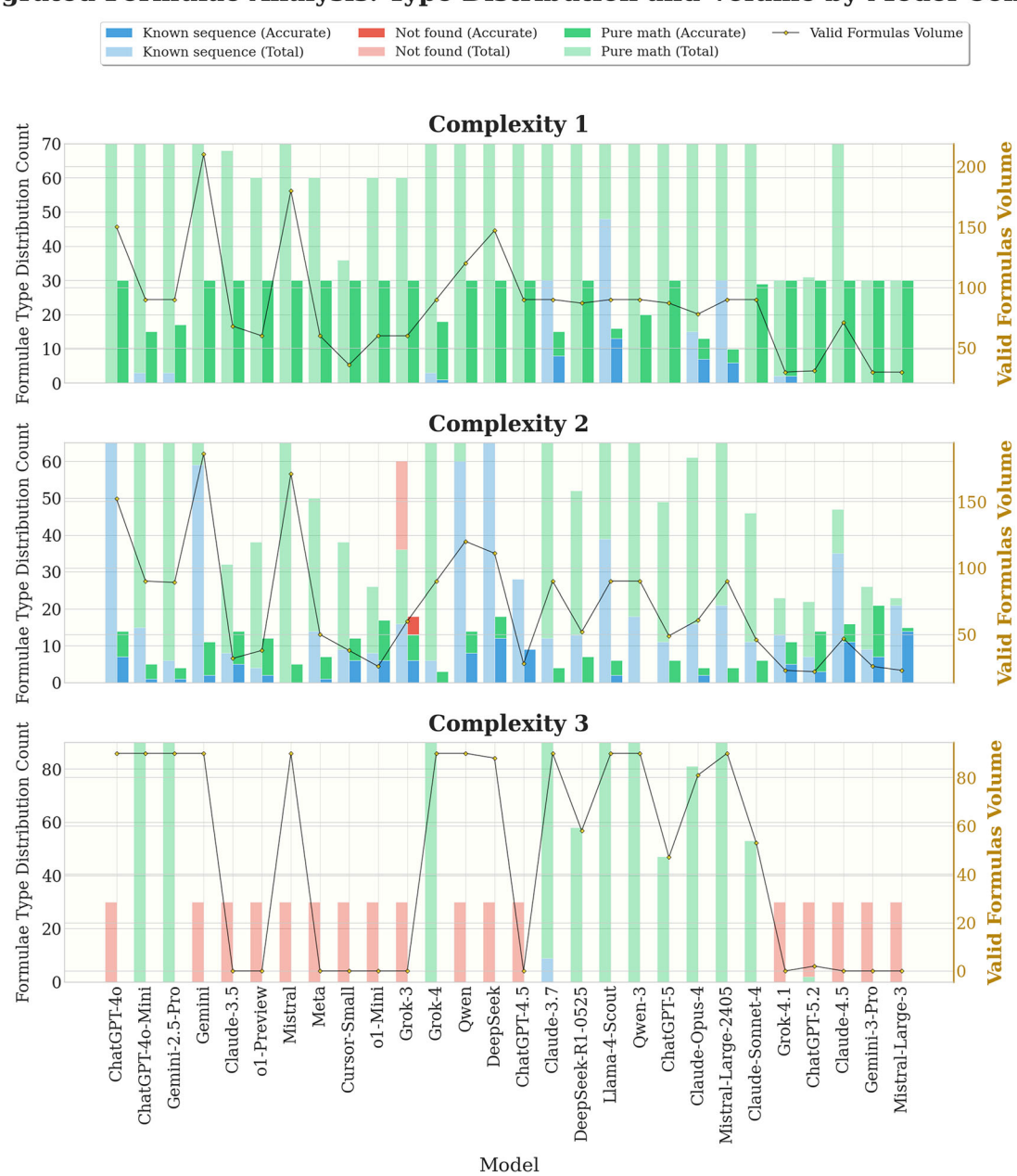

Fig. 3 | Comprehensive analysis of formulae generation for numerical sequences of increasing complexity. Top: Percentage of equivalence between generated formulae, measuring output similarity and solution diversity. Bottom: Accuracy rates showing correct replication of target numeric sequences across complexity levels. The results demonstrate a direct correlation between sequence complexity and diminished model performance, with particularly stark degradation in equivalence rates suggesting limited solution diversity. Notably, newer versions of models performed worse than their previous iterations (see Supplementary Information).

Information, we also show the distribution of print cases by language and complexity level. They support the earlier observation that correctness in many instances is linked to a lack of compression.

Supplementary Figures in the Supplementary Information, we show the distribution of correct instances by sequence and by programming language generated by ChatGPT. The different programming languages are shown in coloured rows. On the right-hand side, the percentage of correct instances. At the top, the number of programming languages that overlap or solve the same problems correctly, and, at the bottom, the extent of the overlap. For example, 5 languages solve the same 20 of 120 problems.

According to the results, the vast majority of correct cases are `printed`, failing to compress the sequences. This indicates that in most instances where the system correctly identifies a sequence, it does so by simply outputting the sequence as is, without any attempt at compression.

A second test performed to evaluate compression was based on the no-compression percentage. According to this metric, a compressed—and therefore, comprehended—sequence could be expressed as a general (and ideally short) programme. `Print` cases are considered here to have 100% non-compression, since they involve displaying the original sequence as is, which, in our test, is synonymous with not understanding the sequence.

Figures in the Supplementary Information show how no-compression generally increases with complexity, except for Mathematica, where the no-compression percentage is lower at complexity level 2 than at level 1. This happened because Mathematica has the capacity to computationally replicate several well-studied and known

https://doi.org/10.1038/s41467-026-73289-5

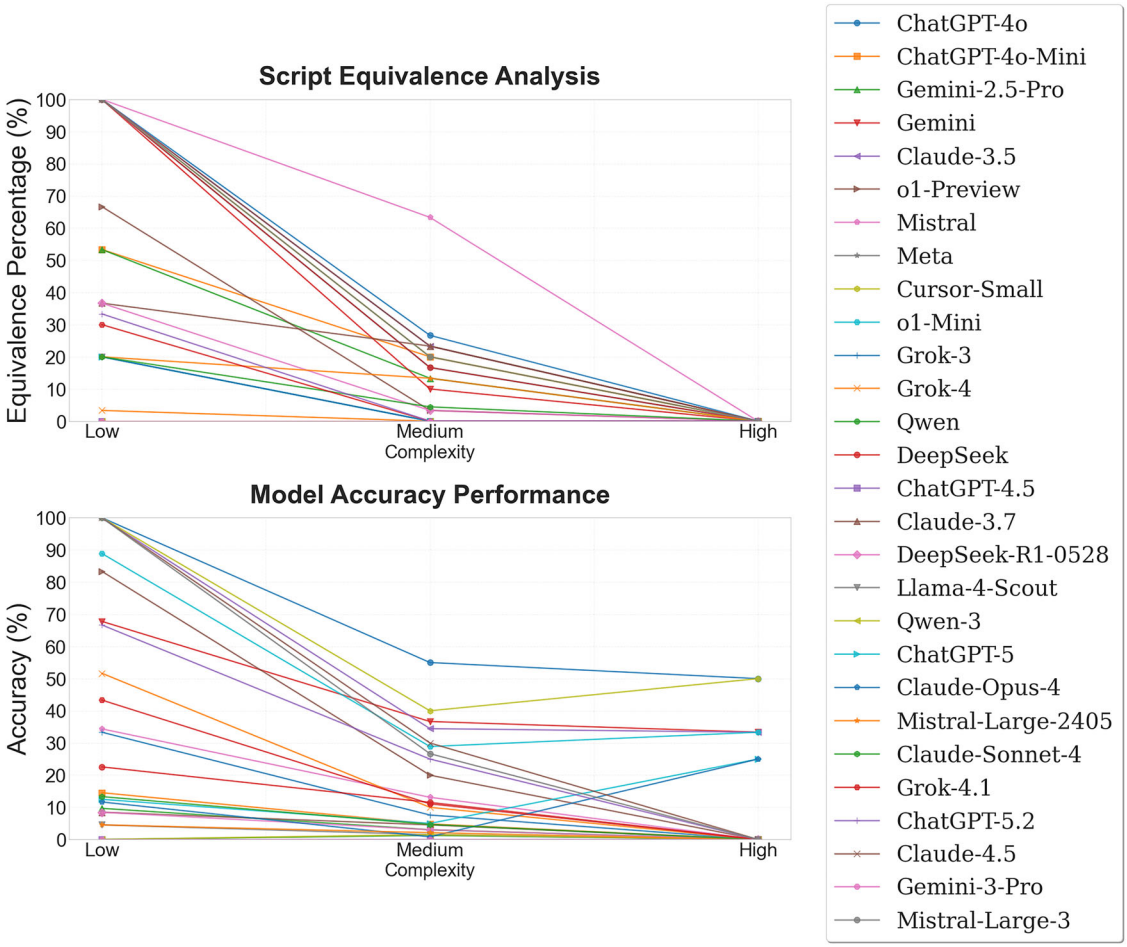

Fig. 4 | Integrated formulae analysis: type distribution and volume by model complexity. Top: (Complexity 1), middle: (Complexity 2), and bottom: (Complexity 3) show the integrated view combining formula generation volume (gold line, secondary axis) with type distribution among both total (lighter bars) and accurate (darker bars) responses. Responses are categorised as known sequences (blue), pure mathematical expressions (green), and not found (red). The integrated panel reveals that whilst models may generate valid formulae at lower complexities, the proportion of accurate responses declines precipitously, and reliance on known sequences dominates over novel mathematical reasoning. These limitations are especially pronounced in contexts permitting complete freedom to discover diverse yet correct solutions, underscoring an absence of genuine creativity and mathematical understanding, attributes often mistakenly attributed to these models[40]. Notably, newer versions of models performed worse than their previous iterations (see Supplementary Information).

sequences of numbers. This capacity leads to shorter code at complexity level 2. However, at complexity level 3, the trend aligns with other languages, showing direct proportionality between complexity and no compression.

Another analysis addresses the influence of the temperature parameter on the production of code to generate specific numeric sequences. In the Figures in the Supplementary Information, the average percentage of no compression by language, and across the different values of temperature used during the experiment, is shown. This plot shows the shaded area representing the confidence tolerance over the average of no compression along the different values of complexity.

The trends in the percentage of no-compression across all temperature values are nearly identical, as are the shapes of the confidence intervals. The temperature value used to generate the code does not affect the result, indicating that the temperature does not have an impact on this experiment. It is worth mentioning the ArnoldC case, where, in fact, there were not many correct cases, making it difficult to calculate a confidence interval.

SuperARC-seq

Based on the previous experiments, it is possible to characterise one test directly related to the SuperARC framework: the SuperARC-seq. The objective of this test is to quantify intelligence and related cognitive capacities, specifically, reasoning and comprehension, drawing inspiration from the work in ref. 34 and the theoretical and empirical studies introduced here. As mentioned, this test is grounded in one of the fundamental cognitive tasks: recognising patterns and evaluating the complexity of finite sequences, which inherently requires a level of comprehension in order to provide a meaningful explanation. In our experiment, we generated short integer sequences (100 binary and 90 integer-valued in general, as seen in the Sup Inf) and tasked several advanced LLMs with deriving a formula capable of reproducing each of the target sequences.

We classified the correct answers provided by the LLMs into three types:

1. *Prints*: The model simply reproduced the target sequence without any attempt to encode or express it logically. This response type reflects a failure to abstract or deduce any underlying pattern, simply outputting the sequence as is.
2. *Ordinal*: The model provided a mapping based on the indices where "1"s occur in the sequence. This response reflects an attempt by the model to analyse and map some logical structure to the sequence, making it more valuable than simply reproducing it verbatim. For integer sequences in general, a simple ASCII mapping was performed to convert from integers to binary encodings.

https://doi.org/10.1038/s41467-026-73289-5

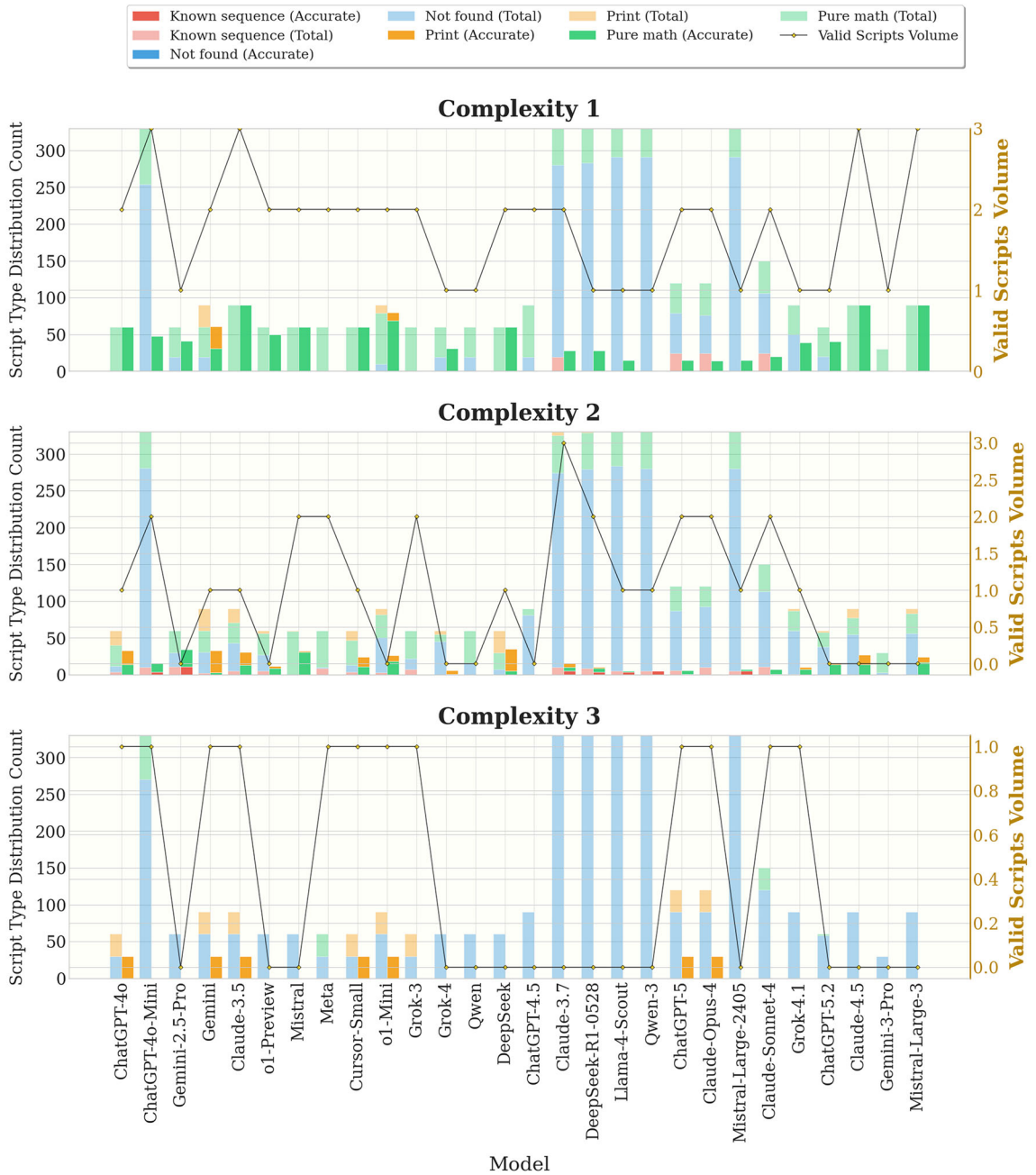

Fig. 5 | Comprehensive analysis of language model performance in Python script generation across complexity levels (Low, Medium, High). Top: Script equivalence analysis, showing percentage of equivalence between generated scripts (measuring output similarity and solution diversity). Bottom: Model accuracy performance, showing accuracy rates versus complexity. Results expose fundamental LLM limitations: whilst models generate coherent solutions, accuracy deteriorates markedly with complexity. Upper trajectories show equivalence remains stable whilst accuracy plummets---models generate internally consistent but incorrect approaches. Notably, newer iterations underperformed preview versions (see Supplementary Information), challenging assumptions of monotonic improvement.

3. *Non-Both*: These responses avoided both simple reproduction and ordinal mapping, reflecting a more sophisticated approach to understanding and encoding the pattern. Such responses are the most valuable as they imply a deeper analysis and potentially creative logic to represent the sequence.

Thus, from these three types of correct results (i.e., the reconstructed sequence matches exactly the original one), we have four different classes of results: Correct & Non-Both; Correct & Ordinal; Correct & Prints; and Incorrect. For any given tested model, the percentages of results belonging to each group can be combined as a vector ρ of rates in the range [0, 1], where $\rho = [\%_{c,np,no}, \%_{c,o}, \%_{c,p}, \%_{inc}]$ such that $\sum_{i=1}^{4}\rho_i = 1$. Notice that the percentages are represented in the range [0, 1] in order to resemble probabilities. We know, beforehand, that the best-performing model would be one with ρ_{best} = [1, 0, 0, 0]. Thus, a first possible test would be to check the overall percentage of correct answers.

$$\varphi_a = \sum_{i=1}^{3}\rho_i, \tag{1}$$

which would range from 0 to 1 for models that are not able to reproduce any sequence to models that perfectly reconstruct the

https://doi.org/10.1038/s41467-026-73289-5

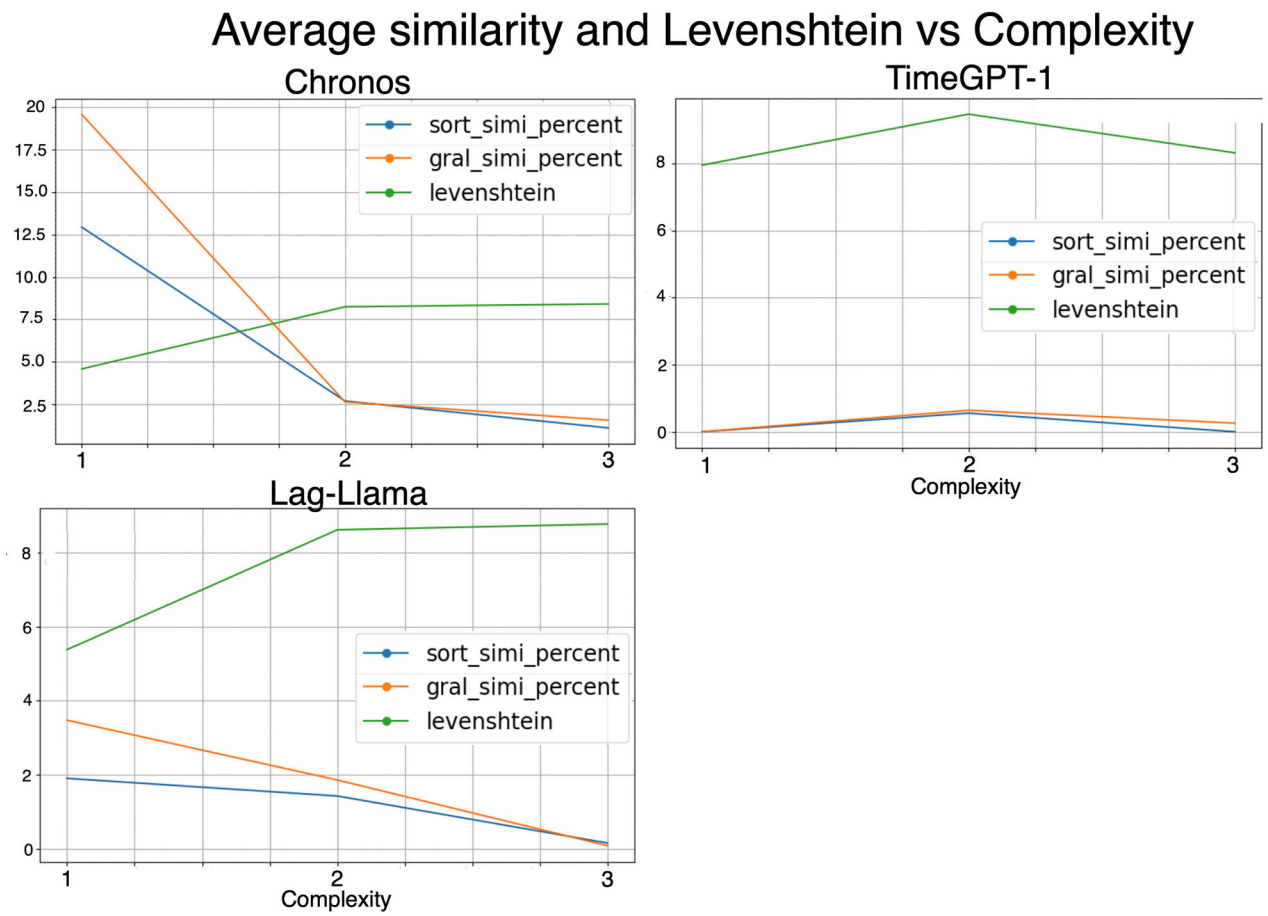

Fig. 6 | Integrated script analysis: type distribution and volume by model complexity. Top: Complexity 1, middle: Complexity 2, and bottom: Complexity 3 show the integrated view for each model. Semi-transparent left bars indicate total script type distribution (Known sequence = red, Not found = blue, Pure math = green, Print = orange); solid right bars show accurate predictions only; gold diamonds (right *y*-axis) indicate valid script volume. The disparity between left/right bar heights quantifies the accuracy gap. Predominance of `Not found' (blue) at higher complexities indicates systematic failure to recognise solution strategies. Without analogous training exemplars, LLMs cannot reliably deduce solutions despite extensive Python training. Notably, newer iterations underperformed preview versions (see Supplementary Information), challenging assumptions of monotonic improvement.

sequences, respectively. However, this only accounts for the ability of LLMs to reproduce the initial sequence (planning) but not for their compression capabilities. To account for the latter, let us assume that the best possible algorithm for each element of the data set is $\mathcal{B}_{k,j}$, such that $B_{k,j}() = D_{k,\text{encoded}}[j]$, and here the algorithm does not have a particular input, similar to the definition of algorithmic complexity. Thus, from the basic properties in AIT:

$$K(D_{k,\text{encoded}}[j]) = K(B_{k,j}()) \leq K(B_{k,j}) + \mathbf{O}(1) \tag{2}$$

The ratio

$$\frac{K(D_{k,\text{encoded}}[j])}{K(B_{k,j}) + \mathbf{O}(1)} \tag{3}$$

consistently falls within the range [0,1] for medium to long sequences when no embedding algorithms are employed. This behaviour arises because approximations of algorithmic complexity are less reliable for short sequences, primarily due to the overhead inherent in theoretical computations. In order to surpass this limitation, instead of assessing the absolute algorithmic complexity (or any of its approximations), we shall consider a *normalised* version of it (denoted by nBDM(·)).

To approximate algorithmic complexity, we will use the BDM/CTM approach, as described in detail in previous sections, and its normalised version, as pointed out in previous works[35] for any object of arbitrary size, it is possible to construct analogous objects that attain the minimum and maximum possible values of algorithmic complexity according to the BDM:

- *minimum* complexity object: This case is straightforward and corresponds to an object composed entirely of a single repeating symbol—for instance, a binary string consisting solely of zeros;
- *maximum* complexity object. The maximum BDM value is achieved by an object whose decomposition (according to a specified algorithm) results in slices that exhibit the highest values of the coding theorem method (CTM), with each distinct slice occurring only once until all possible configurations of the given shape have been exhausted.

The primary advantage of considering a normalised measure lies in its ability to enable comparisons between objects of varying sizes, effectively mitigating the influence of size on the measure itself. This property is particularly in the case of the present study, where we compare the complexities of sequences and formulas generating them.

This way, the following ratio presents itself as an interesting weighting factor for the probabilities in Eq. (1):

$$\text{nBDM}(D_{k,\text{encoded}}[j])/\text{nBDM}(B_{k,j}) \tag{4}$$

The ratio in Eq. (4) measures how the algorithmic complexity of the formula and sequence compares to the other possible outputs of the LLM. If the relative algorithmic complexity (measured by the normalised BDM value) of the formula is greater than it was for the sequence itself, this suggests the LLM did not succeed in compressing the input sequence (it made the formula have a greater relative algorithmic complexity). On the other hand, if the opposite occurs, then the LLM could compress the sequence compared to other possible outputs of the LLM.

The ratio in Eq. (4) ranges from 0 to a positive value $M > 1$, which happens when the best possible compression is achieved (the inverse mapping of CTM). Since M is not known beforehand, we can use a nonlinear mapping that saturates the value of the ratio to a maximum value of 1 (similar to an activation function). The hyperbolic tangent function can be used in this case, since $\tanh(0) = 0$ and $lim_{x\to\infty} \tanh(x) = 1$. Thus, a candidate weighting factor for the probabilities in (1) is:

$$\delta_{k,j} = \tanh\left(\frac{\mathrm{nBDM}(D_{k,\mathrm{encoded}}[j])}{\mathrm{nBDM}(B_{k,j})}\right) \tag{5}$$

with the best possible value of $\delta_{k,j}$ approaching 1 in a perfect compression scenario. Since we have several algorithms classified under each of the four types (according to their structure), instead of using the individual ratios for each type k, we shall use the harmonic mean per type, defined as:

$$\delta_k = \frac{n_k}{\sum_{j=1}^{n_k} \delta_{k,j}^{-1}} \text{ for } R_{k,j} \text{ of type } k, \tag{6}$$

where n_k represents the number of algorithms that are of type k. If we include m sequences in the test, for example, $n_k = m\rho_k$. Thus, an updated version of the test is

$$\varphi_b = \sum_{i=1}^{3} \delta_i \rho_i. \tag{7}$$

Deliberately, we want to privilege models that do not simply copy or provide ordinal mappings of the input sequences. Thus, we can attribute higher weights to types that are correct and do not copy or print the results. We also want to give more weight to programmes that provide ordinal mappings when compared to print cases. Then, considering a power-law weighting strategy, the final test metric is

$$\varphi = \delta_1\rho_1 + \frac{\delta_2\rho_2}{10} + \frac{\delta_3\rho_3}{100}. \tag{8}$$

It can be seen that $\varphi \in [0, 1]$ encompasses different behaviours. For example, $\varphi \in [0, 0.01]$ if only print-type models are outputted. Also, $\varphi \in [0, 0.1]$ if only ordinal-like formulas are created. Finally, $\varphi \in [0, 1]$ in cases where the LLMs create formulas that are always correct, do not copy or create ordinal mappings. The ranges will be populated with varying compression levels corresponding to the algorithms obtained. Overall, if the score is 0, all the formulas were wrong. If it is 0.5, it can represent the case where half the outputs were correct, and half were wrong, with the formulas produced with the highest compression levels. So, in a regular half-and-half case, since compression will not be optimal, the test score is less than 0.5. The test performance results for each model are calculated using Eq. (8) for $\mathcal{T}$ in Algorithm 1.

There are some possible variations for the test metric in Eq. (8). For example, some sort of Bayesian approach could be used to consider that the elements of ρ are not constants, but random variables which could account for the number of different correct/incorrect answers for the same input sequence. In this way, the multiplicity of possible generators is taken into account, better capturing the concept of algorithmic probability, and the output of the test would be a random variable instead. However, LLMs hardly produced even one correct answer, so we kept the formula simple.

As described, Eq. (8) tests for two features, compression via non-print computer programmes and non-ordinal mathematical formulas to the input sequence, and prediction, by running all programmes and all formulas to match each sequence digit, and penalising them when they did not represent an actual compressed model that generated a possible new digit of the sequence when run in reverse, i.e., when 'decompressed'. The test formula assigns greater importance to correct cases that are not solutions of the type 'print(s)' where s is the sequence for which the AI system is asked for a model, given that a print model does not allow generalisation by prediction through simulation, as running a print command will only print up to the last digit. The same is true for what we call 'ordinal', which is simply indicating the index of the non-zero non-one element in the binary (or binary embedded) sequence, meaning that, together with the 'print' case, the system failed in its attempts at abstracting features of the object. Finally, the formula punishes ordinal and print answers in a weighted fashion. The best performer can only reach a φ of 1, while the lowest value is 0.

Applying SuperARC-seq. The results of the LLM classification after applying this test according to the formula are shown in Table 1 and summarised in Fig. 7 for *binary sequences*. As shown in Table 1 and Fig. 7, CTM/BDM would achieve perfect scores in all categories, consistently avoiding trivial responses and providing accurate formulas. By design, this model clearly excels in abstract feature recognition, outperforming all other models at prediction, which we claim is key to planning. CTM/BDM actually produces a set of possible generative models (computer programmes) that, when run in reverse in what would be the uncompressing process, produce new elements to test against the observation, thus updating and producing new possible outcomes. These models are also hypotheses that do suggest whether a sequence is random or not, rather than looking for such a sequence in the training set or a combination thereof and failing for those not found in the distribution.

These findings indicate that LLMs perform well when there are discernible patterns in the data, but struggle with randomness, failing to capture complexity in an algorithmic sense. In contrast, AIT can accurately predict (rather than guess) the sequence, regardless of the string's complexity. These results demonstrate that the algorithmic-complexity approach effectively approximates the minimal description length of information, identifying the shortest algorithm capable of generating a given sequence.

Despite being the top-ranked LLM model, chatgpt_4.5 only provided ordinal mappings (soft copies) of the inputs, which achieved correct results at the cost of no abstraction and comprehension at all (slightly better than a pure print-only test score). The GPT-4o, Grok-3, Meta, Claude 3.5, and o1−preview LLM versions produced several incorrect formulas, while the other LLM models mostly produced print-like responses, indicating a lack of pattern recognition beyond basic sequence reproduction. Notably, in the evaluation of consecutive versions, the most recent models−with the exception of Grok−demonstrated a degradation in performance.

Unlike standard LLMs that predict the next tokens in text, CTM/BDM finds the generative processes of the sequence by a combination of symbolic and statistical pattern matching algorithms, which allows it to derive concise models that can then run in reverse to match each digit and produce new ones, hence allowing prediction and planning by picking the most likely among a set of possible models based on the algorithmic probability of the model (how short and how often the same model was found to produce the same sequence).

Table 1 | Numerical benchmark ranking of popular frontier models publicly available against ASI represented by fundamental or neurosymbolic models like AIXI[95] and CTM/BDM[26,35]

Model	ρ_1	ρ_2	ρ_3	ρ_4	δ_1	δ_2	δ_3	φ
AIXI/BDM/CTM	**1.000**	0.00	0.0	**0.000**	**1.000**	0.000	0.000	**1.000**
ChatGPT-4.5	0.00	**1.000**	0.0	**0.000**	0.000	0.419	0.000	0.042
o1-Mini	0.00	0.64	0.0	0.36	0.000	**0.537**	0.000	0.034
Claude-3.7	0.00	0.81	0.0	0.19	0.000	0.407	0.000	0.033
Claude-3.5	0.06	0.14	0.0	0.80	0.449	0.428	0.000	0.033
o1-Preview	0.00	0.29	0.0	0.71	0.000	0.423	0.000	0.012
Gemini	0.00	0.00	**1.000**	**0.000**	0.000	0.000	**0.762**	0.008
Cursor-Small	0.00	0.00	**1.000**	**0.000**	0.000	0.000	**0.762**	0.008
ChatGPT-4o-Mini	0.00	0.00	**1.000**	**0.000**	0.000	0.000	**0.762**	0.008
Mistral	0.00	0.00	**1.000**	**0.000**	0.000	0.000	0.710	0.007
Qwen	0.00	0.00	**1.000**	**0.000**	0.000	0.000	0.710	0.007
DeepSeek	0.00	0.00	**1.000**	**0.000**	0.000	0.000	0.710	0.007
Llama-4-Scout	0.01	0.00	0.0	0.99	0.450	0.000	0.000	0.004
Grok-3	0.00	0.02	0.0	0.98	0.000	0.318	0.000	0.001
Mistral-Large-3	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Meta	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Gemini-3-Pro	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Claude-4.5	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
ChatGPT-5.2	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Grok-4.1	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
ChatGPT-4o	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Grok-4	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Claude-Sonnet-4	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Gemini-2.5-Pro	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Mistral-Large-2405	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Claude-Opus-4	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
DeepSeek-R1-0528	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
Qwen-3	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000
ChatGPT-5	0.00	0.00	0.0	1.00	0.000	0.000	0.000	0.000

Best per-column values are in bold (for all columns, greater values are better except for ρ_4, where smaller is better).

It is important to note that the SuperARC-seq application hereby considered only took into account binary sequences. Whenever integer sequences were considered, a clear biasing of the results was observed as LLMs started to take advantage of their training corpus to actually display memorisation capabilities rather than abstraction and synthetization ones. Figures 8 and 9 present the percentages of each type of output and the test scores when different types of sequences were considered.

The test scores for different types of sequence reveal that the inclusion of integer sequences leads to significantly higher performance of LLMs, as shown in Figs. 8 and 9, where higher percentages of Correct & Non-Prints & Non-Ordinal and Correct & Ordinal outputs are seen, as well as higher test scores. This is likely due to the models leveraging memorised associations between familiar integer sequences and pre-learned formulas—an effect similar to hash-based retrieval. These findings show the importance of limiting evaluations to binary sequences, which are less likely to have been part of the training data, thereby providing a more accurate and unbiased assessment of model performance.

The robustness of the test score when only binary sequences are considered can be seen in Fig. 10, which shows the result of a bootstrap procedure. The bootstrap simulation procedure was conducted as follows: for each specified sample size s (s equal to 25, 50, 75, and 100), 100 bootstrap samples of size s were drawn with replacement from the complete dataset, which consisted of 100 binary sequences (Supplementary Information). For each bootstrap sample, the corresponding test scores were computed. The resulting plot presents the confidence intervals for the test scores obtained across all bootstrap iterations. The observed stability in test scores, coupled with the progressively narrowing confidence intervals around the mean as sample size increases, suggests a high degree of robustness in the evaluation metric. This indicates that the test score is largely insensitive to the particular subset of sequences used, thereby validating the reliability of the assessment across different sample sizes.

These results are aligned with recent work exploring, for example, LLMs' logical reasoning failures[41] as well as GPT-4's limitations in deductive reasoning[42]. Other researchers have also reported degradation of mathematical capabilities[43] and planning limitations[44]. These works collectively document that LLMs struggle with systematic reasoning across multiple domains.

Discussion

In previous work, we have shown that aspects of human[2,17] and animal[4] cognition could be characterised, and aspects of their behaviour reproduced in terms of algorithmic probability tools and algorithmic complexity metrics that we have also suggested for artificial and computational systems, including robotics[5]. Here, we tested this approach and proposed a new quantitative metric based on an fundamental ASI-level method[28] grounded in algorithmic information dynamics (AID)[27,29] and the principles of algorithmic information theory (AIT) related to recursive compression (as opposed to statistical) and prediction in application to large language models (LLMs), which

https://doi.org/10.1038/s41467-026-73289-5

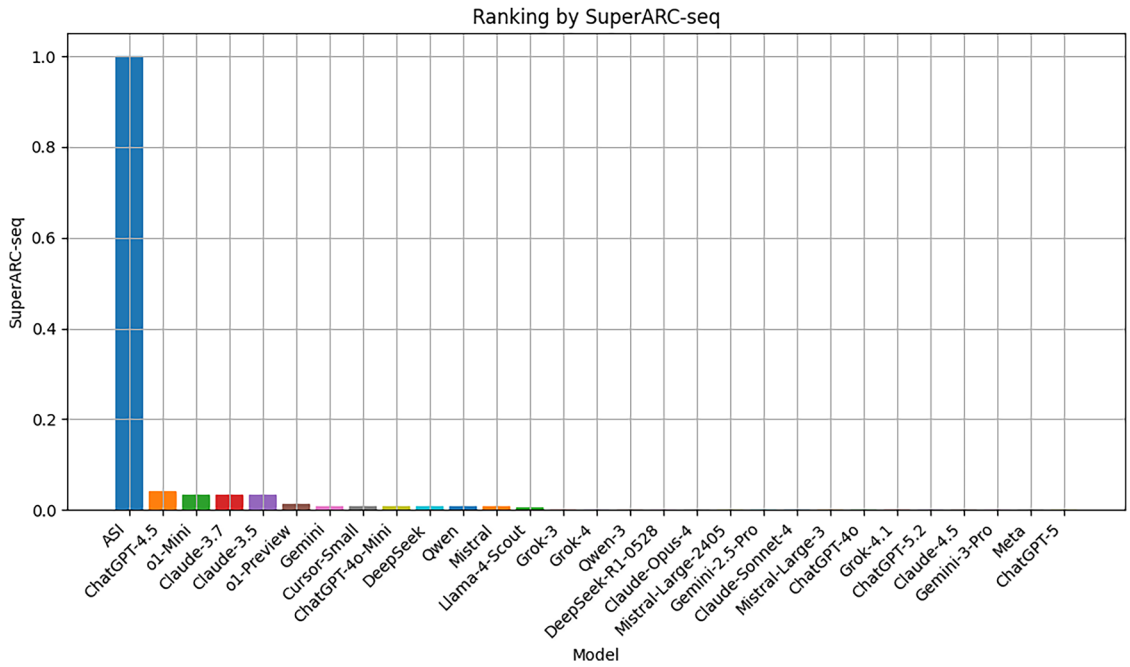

Fig. 7 | Benchmarking plot from Table 1 showing how most frontier models are close to each other in their performance under this test and far from artificial general intelligence (AGI) or artificial super intelligence (ASI) goals according to this test. ASI would be able to distinguish simpler from complex sequences and generate predictive models for each accordingly, as AIXI[9] or CTM/BDM would do[26,35] as instantiations of universal AI (UAI) that we take as an example of ASI as optimal abstraction and prediction. Today, LLMs only produce or retrieve models for sequences that were seen and found in their original training sets, given that increasing the sequences' lengths impacts the LLM's performance in identifying the sequence, hence indicating sequences are not recognised from first principles but from simplistic pattern matching.

are believed or have been proposed to be capable of approaching artificial general intelligence (AGI) and superintelligence (ASI).

Algorithmic information dynamics (AID)[29] combines aspects of statistical causal inference (those best grounded on causality), such as perturbation analysis, and algorithmic information theory (AIT). AID suggests that for an AI to be properly evaluated, the test has to be dynamic and must evaluate the system's ability to adapt and react to the new conditions, in what constitutes a type of more sophisticated Turing test beyond the simplicity of human language.

Applied to SuperARC, the test involved increasing the problem input complexity and analysing the output complexity with our most powerful tools that, as a consequence, are (semi-)computable but truly capable of evaluating random deficiency, that is, how far an answer is from randomness or simplicity, and for optimal prediction power of the extracted/inferred executable computable model(s).

As governed by AIT, recursive compression and optimal prediction go hand in hand[45], but previous tests focused on particular subset features, even those designed to test human reasoning and human abstraction, such as ARC-AGI[30]. A system could theoretically excel at ARC-AGI or SuperARC while completely lacking social intelligence, embodied reasoning, common sense, or goal-directed behaviour; and therefore, these tests are not meant to test everything we think an intelligent system that will interact with humans in humans' context should possess.

Here, we have introduced and demonstrated that recursive compression can quantify the property of model abstraction and data prediction based on a mathematical proof provided in the Sup Inf, of the equivalence between model compression and prediction applied to sequences based on Martingales, without resorting to proof-theoretic statistical tests. By incorporating and exploiting the formal equivalence between prediction and recursive compression into an intelligence test framework, we align the assessment of comprehension with fundamental computational principles. An agent's ability to abstract information through feature selection and model compression reflects its capacity to identify and utilise patterns within data. Similarly, its planning and prediction skills demonstrate its ability to anticipate and enact future events based on these patterns.

Another problem in LLM testing is benchmark contamination: the targeted optimisation of a model using leaked test answers. The open-ended and interactional nature of our testing framework is intended to counteract this problem of benchmark contamination and cheating. Our investigation of frontier models, framed within the AID paradigm, yields several key insights about the models' comprehension capabilities. Most of the models often demonstrate model version performance regression, poor accuracy in replicating and predicting even simple and recursively generated sequences beyond clearly memorisation results from the training distribution (such as sequence labelling). The vast majority of the correct answers turned out to be simple print statements of the numerical sequences themselves, rather than any code or model indicating any sign of understanding or pattern recognition.

These conclusions are reinforced by the model's explicit dependency on specific programming languages for correctness or on a well-studied and documented series of numbers. In other words, if there are not enough implementations available in a specific programming language for the model to learn from, or even specific methods of mathematical analysis over specific numerical sequences, LLMs failed to produce the correct answer. Considering the most popular and widely used languages, LLMs do not demonstrate understanding but instead rely on selecting from an abundance of previously seen cases.

In this context, we have shown that using those concepts in reverse (that is, as generative rather than for testing purposes) can provide the model with synthesising and recursive predictive power that is otherwise lacking. In previous work, we have shown how optimal prediction can be approximated using tools such as CTM and BDM[28], and also how BDM can be used in the opposite fashion: not only as a testing tool for intelligence, but as a model generator[25,26,46] via a greedy algorithm as an approach to optimal inference[35,47,48]. While

https://doi.org/10.1038/s41467-026-73289-5

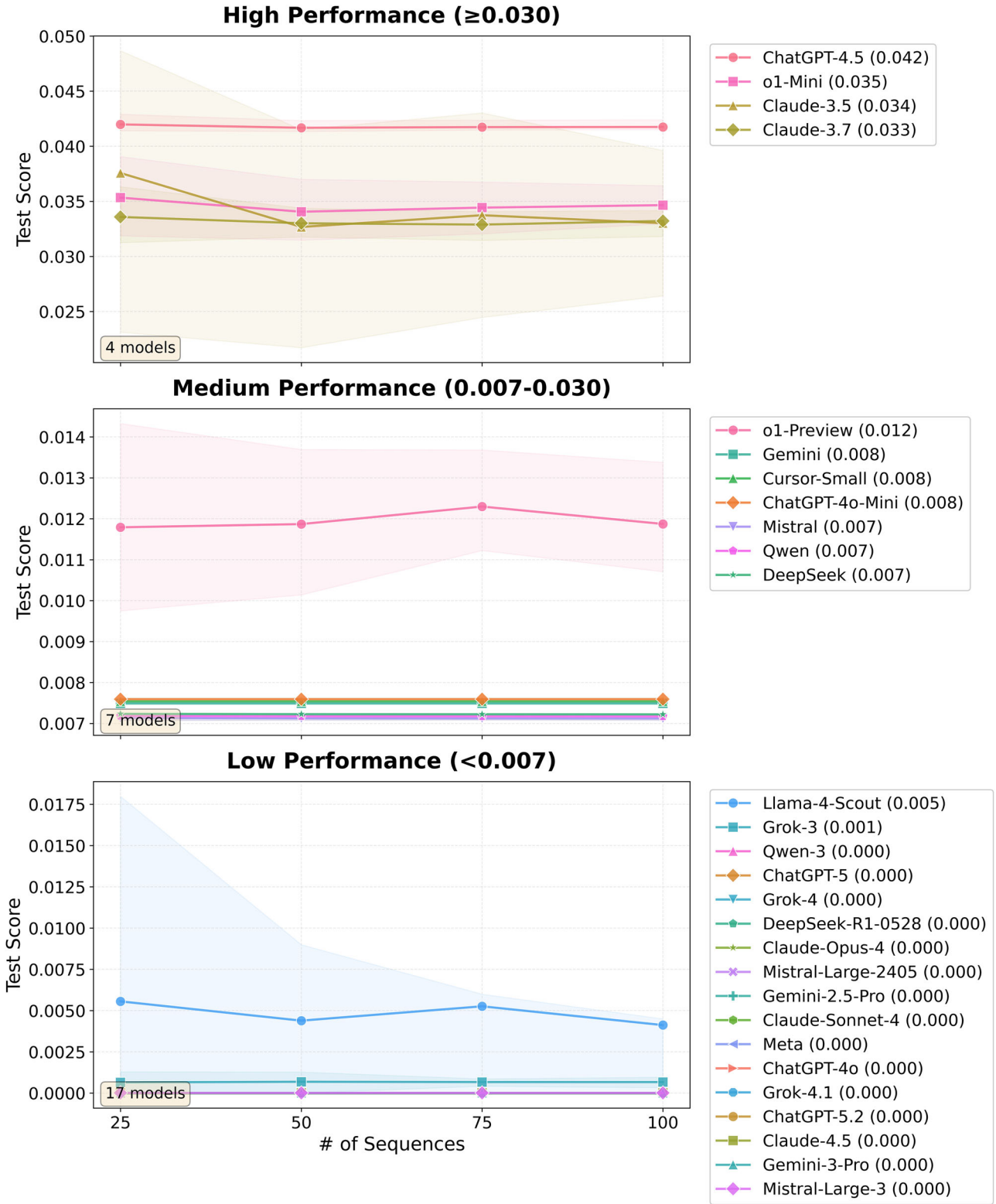

Fig. 8 | Percentages by output types: p_1 is the percentage of Correct & Non-Prints & Non-Ordinal outputs; p_2 is the percentage of Correct & Ordinal outputs; p_3 is the percentage of Correct & Prints outputs and p_4 is the percentage of Incorrect outputs. It is clear that as soon as integer sequences are considered, LLMs start to get better quality output formulas (i.e., greater p_1 and p_2). This suggests that the models were trained on integer sequences rather than binary ones, implying that incorporating integer sequences into the test calculations could introduce bias.

some of these methods can be seen as a brute force approach to a giant lookup table of micro-programmes to explain the data, BDM is not. BDM combines the algorithmic probability approximations produced by CTM but then stitches each most likely programme for each piece back together according to valid laws of information theory, in what constitutes a pure form of hybrid statistical and symbolic explanation —hence neurosymbolic. BDM, therefore, uses the two best inference theories currently available to science, one being the most used and

https://doi.org/10.1038/s41467-026-73289-5

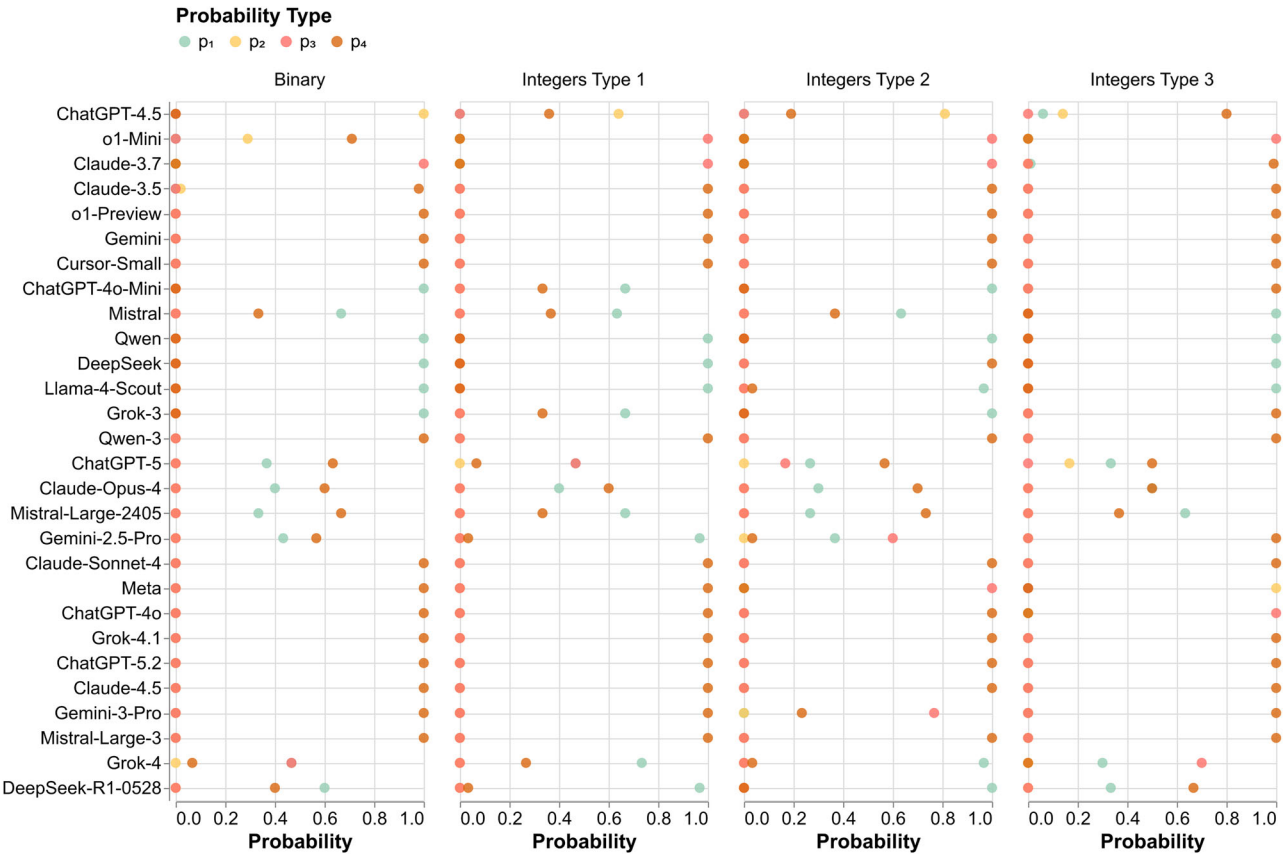

Fig. 9 | Test scores when different types of sequences are considered. Consistent with the results shown in Fig. 8, the inclusion of integer sequences leads to significantly higher test scores for the LLMs. This outcome arises from the models' ability to exploit their internalised training data by directly associating observed sequences with pre-learned formulas, suggesting a form of hash-like memorisation. These findings highlight the importance of restricting the evaluation to binary sequences in order to obtain an unbiased measure of each model's performance, as such sequences are less likely to have been included in the models' training corpora.

overused in statistical machine learning (such as Shannon entropy-based measures, along with its limitations[49]), and one that has often been neglected on the basis of uncomputability[50,51], which in fact guarantees open-endedness and optimal solutions.

LLM developers are moving slowly towards this direction, often without realisation or acknowledgement. Technologies like RAGs, KAGs/CAGs, agentic workflows, guardrails, long-context LLMs, and more, are symbolic computation attempts to improve LLM capabilities. Yet many of these approaches are not being introduced into the system's core functions (e.g., to quantify loss or explore the solution space), yet attempt to combine traditional statistical pattern matching with symbolic approaches. This test and work suggest that this integration and a deeper integration are key for aspects of model abstraction and universal planning.

We reported that even frontier LLMs currently perform close to pure-copy solutions when increasing the complexity of certain mathematical problems, with even advanced models struggling to perform model abstraction/extraction and produce executable predictive results. The results confirm that current LLMs, while competent in pattern replication, lack critical elements associated with what are believed to be key elements of AGI or ASI.

The LLMs involved in this test showed a high dependency on predefined patterns. As complexity increased, models relied increasingly on trivial strategies, such as direct sequence printing or simplistic brute-force mathematical expressions. This highlights the LLMs' inability to abstract or conceptualise novel solutions that require some degree of mathematical ingenuity.

The poor performance is revealed by the lack of synthesising capabilities and the repetitive nature of the outputs for greater complexity inputs. This tendency to revert to safe and redundant approaches underscores the models' limited synthesising capabilities and exposes their high memory exploration dependencies in simpler modes.

For example, we reported that while from ChatGPT-4.5 to ChatGPT-5 improved human benchmark scores have been observed and reported, SuperARC performance degraded. The steep decline in accuracy and functional outputs as complexity increased reveals that these models are potentially increasingly heavily reliant on data size, unable to generalise.

Our results demonstrate that models can regress in fundamental algorithmic reasoning (SuperARC scores) while simultaneously improving in human-centric benchmarks.

The models' outputs suggest strength in replication but a lack of adaptivity to new situations, as a disability to put together solutions in a non-trivial manner to solve a new problem that would constitute a form of model synthesis when increasing problem complexity. The predominance of trivial or incorrect solutions demonstrates an inability to think 'outside the box' (as in if it had not been seen in the training distribution). This suggests that while LLMs can mimic comprehension through retrieval, pattern matching, and Chain-of-Thought techniques, their capabilities remain bounded when tested against algorithmically complex sequences. These observations point directly to a key distinction between current systems and AGI/ASI: the latter would require the ability to autonomously generate new strategies, abstract concepts, and exhibit flexible problem-solving beyond training data.

Our results based on the first principles of the mathematical theory of randomness and optimal inference in the context of claims about 'reasoning' capabilities and AGI/ASI, suggest that none of the

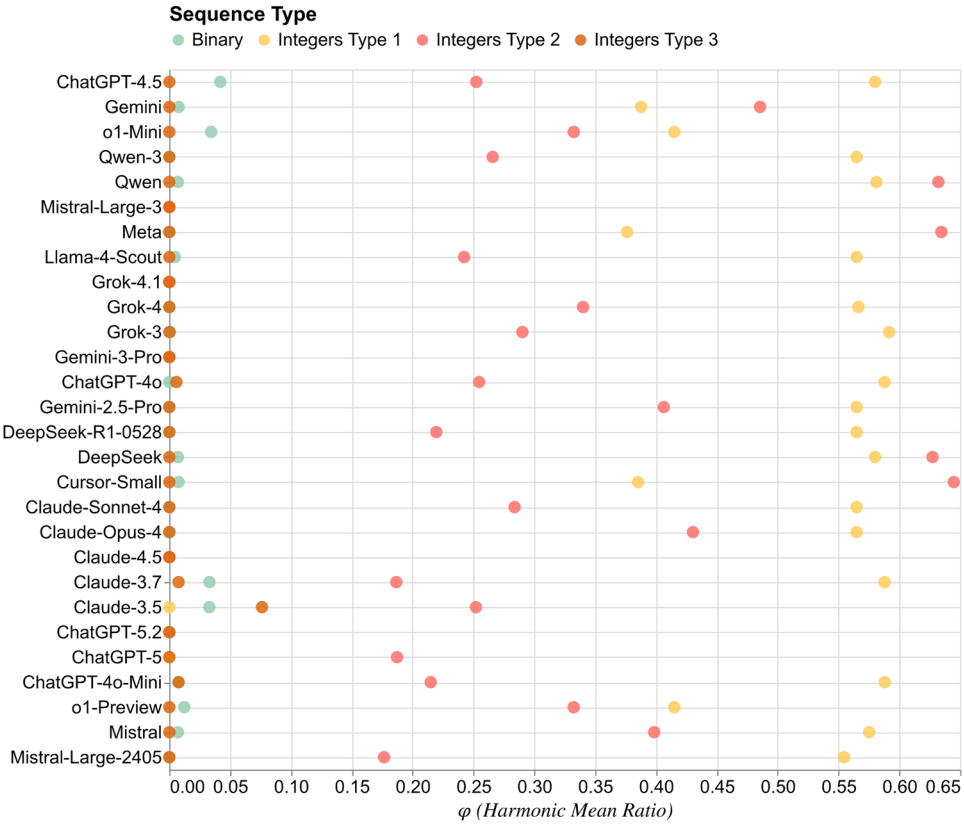

Fig. 10 | Bootstrap procedure to assess the robustness of the test score when binary sequences were used. The stability of the test scores, with average values per model provided in parentheses, in combination with the narrowing confidence intervals around the mean as the sample size increases, indicates strong robustness of the evaluation metric.

chatbots evaluated dominates any other in absolute terms when it comes to demonstrate whether the solutions have a semantic value, in other words, that LLMs mean the words they produce as opposed to emulating a coherent conversation.

We have argued throughout this paper, and it is distilled by the nature of our test for intelligence, that (semicomputable) open-ended tests not assumed to be isolated "in a vat" are needed in order to quantify scientific and mathematical intelligence in the form of abstraction and prediction so that they are fundamentally coupled into a necessary characteristic of comprehension[6], whether performed by a human[2,17], animal[4], or artificial[5] agent. The same also holds for the tested systems: that only by incorporating open-ended, sufficiently powerful semicomputable predicting agents (e.g., via the methods explored in the present paper), one may achieve such a characteristic of intelligent systems, and thereby enabling the possibility of ASI by way (or not) of AGI.

Methods

Assessing the capabilities of frontier LLM models

Since the inception of LLMs, these systems have been associated with human intellectual capabilities related to language that range from mastering composition to retrieving contextual data and even generating novel 'ideas'[52]. However, beyond seemingly arbitrary intelligence tests, questions related to intelligence remain, because intelligence is traditionally not well defined, with the intelligence tests performed remaining rather arbitrary or human-centric and lacking a clear linear progression of difficulty levels. Here, we approach both as a single problem and within a quantifiable framework, providing a formal approach to a form of intelligence based on algorithmic information.

While, in principle, LLMs have been shown to be theoretically capable of Turing-complete computation[53,54], this is achieved when they are augmented with external memory and appropriate decoding mechanisms[54]. In practice, the models we evaluate operate with standard autoregressive decoding and finite context windows, which do not constitute Turing-complete systems.

Some have claimed that LLMs, and specifically ChatGPT, have the potential to revolutionise technological interaction through accurate understanding across conversational interfaces[55]. These attributions and capabilities of LLMs have been tested in a variety of ways, from semantic comprehension evaluations in traditional Chinese medicine (TCM), through structured multiple choice and true/false questions[56], ASCII art[57], to answering open questions and using LLMs as judges of the precision and correctness of the answers provided by other models[58]. Exhaustive and detailed tests have been performed that focus on tasks that require a grasp of a broad context, such as quantitative investing and medical diagnoses[59], to mention only two.

Researchers have called into question these supposed understanding capacities, claiming that a lack of novelty and an abundance of hallucinations is formal and/or informal proof of a lack of comprehension ability[60,61]. When evaluating the intelligence and comprehension capacities of LLMs, some limitations of existing works should be highlighted:

1. All of them contain an element of subjectivity. Measurements of understanding rely on a human or LLM judge, where a type of definition of innovation, usability, and correctness is used, which could be human-centric or dependent on the context.
2. All evaluations use (mostly) text to provide a context for the questions formulated; hence, there are no questions that purely test understanding beyond textual correlations.

3. The test used may take for granted that, since LLMs are trained with intelligent sources of information, this confers some intelligence on the models themselves and thus their comprehension/understanding capacities.
4. LLMs and other AI systems are not self-driven and, as such, cannot be reasoning agents on their own; they only act upon being triggered and prompted by humans, otherwise they do not possess any internal states (e.g., activity when not prompted).

Other researchers, following a more abstract and formal approach, incline to the view that a test of intelligence in LLMs, which could imply comprehension, understanding, and prediction, might rely on exposing and training LLMs on highly complex datasets, and testing how well the LLMs could apply learned knowledge to unrelated but complex tasks (like predicting the next chess move) and reasoning tasks. They claim that information at the 'edge of chaos', a state between non-chaotic and chaotic behaviour in dynamical systems, is more likely to help LLMs manifest intelligence[62]. Suspicions that current AI is mimicking intelligence rather than displaying it have been reported and substantiated before[61,63,64]. Thus, proposing a test that can address those concerns is very relevant.

Compression as comprehension about (and as part of) the world

The formal equivalence between prediction and compression in the Sup Inf using martingales provides a theoretical foundation for understanding intelligence in terms of the generalisation of computational capabilities (due to universality and invariance).

In the context of designing a test for intelligence, such an equivalence suggests that an agent's ability to abstract (e.g., through feature selection and model compression) and to plan (e.g., through prediction, counterfactual analysis, and simulations) are fundamentally interconnected aspects of intelligence involved in scientific or mathematical creativity. More specifically:

- recursive/computable compression and decompression: seen as the summarisation or abstraction of main features (or feature selection) that can be simulated in reverse (decompression), and in contrast to simple statistical pattern-matching or statistical compression;
- process (algorithmic or symbolic) regression and prediction: formally established by AIT as equivalent to compression by way of optimal/universal simulation[65–67] through the concept of algorithmic randomness and martingales (betting strategies)[68–70] (see Sup Inf); or universal (Solomonoff) induction[8,9,20] (see also pseudocode 1).

An agent that can devise or find a model that can compress a set of phenomena that, when uncompressed, generates this set faithfully (and beyond statistical compression) is necessarily able to comprehend it at some level[6]. That is, a set of phenomena that is compressible by an agent into some first principles, or into a succinct model that when uncompressed, reconstructs, describes, and can also simulate future states of the originally described set of phenomena, needs to be comprehensible by that agent[6]; otherwise, we would have an agent that does *not* comprehend the phenomena at the same time that can devise formal theories that can explain them into (fewer) first principles and (better) predict future events, which seems to go against a common-sense understanding of 'comprehension': on the one hand, this necessarily indicates at least some type of comprehension, e.g., a scientific or mathematical one; on the other hand, an agent that can *only* mimic, copy-paste, describe, or depict the phenomena at the same time that comprehending them contradicts any conception of 'an agent able to understand something at a deeper level beyond mere appearance'. From both scenarios, we find that abstraction and prediction arise as *necessary* conditions for comprehension, particularly those intrinsic to the process of devising *novel* scientific or mathematical theories. As introduced in Section 1, instead of covering all the sufficient conditions for all types of human intelligence (including those for which compression may seem unrelated), here we constrain our study to this type of comprehension—crucial to scientific knowledge and mathematical creativity. In this regard, compression necessarily plays a defining, encompassing role that is mathematically grounded and empirically feasible, as we explain in the paragraphs below.

It is important to clarify possible misinterpretations of the meaning of the word "compression" used in our framework. In machine learning and cognitive science, feature selection involves identifying the most relevant variables or attributes that contribute to predictive modelling. This summarisation process reduces the dimensionality[71], focusing on the most informative aspects of the data. It is, of course, a compression approach, but just a part of the one we intend to refer to. In our framework, beyond finding crucial features, attributes, or aspects, (algorithmic) compression into a model refers to other "mechanistic" relationships that are less descriptive in nature, while also guaranteeing that a model does not compromise performance. It involves reducing the complexity of the model, but often leads to a more effective generalisation and greater efficiency[72]. Model abstraction through effective recursive/algorithmic compression allows simulation of various scenarios when the model captures its main features, that is, its most important patterns for prediction are captured as a necessary condition for outcome prediction. Then, model selection happens when each outcome is compared against each time-step observation, hence updating the belief model, instantiating, and enabling 'planning'.

'Compression as comprehension' is thus also tied to pragmatic characteristics such as its utility and feasibility. For example, a compressed model in the form of a mathematical theory or a set of equations, such as a set of laws of physics, is only sound if it allows one to predict the future state of a physical system in "a shorter time than the time taken by the actual unfolding of the phenomenon"[6]. Comprehension only takes place if one can understand real-world phenomena to computationally "outrun" reality at some sufficiently higher level—see also computational irreducibility in ref. 73.

Such a process of understanding or comprehension into formal-theoretical or computational capabilities is demonstrated in the context of the scientific discovery itself. Real-world phenomena that have the appearance of being random or unexpected become a topic of interest for research, analysis, and future development (if successful) of a more comprehensive model or theory that is then able to compress the apparent noise-like phenomena by allowing one to explain these and to predict other unfoldings from it. In this way, science moves in an iterative pace of converting something that is currently considered "irreducibly complex" or unexpected into something that becomes comprehensible by theoretical means that allows computational predictions[6]. For example, consider Newtonian mechanics and general relativity in physics. The former has represented a highly successful compression of observational data, e.g., celestial motion. However, to account for anomalies like the precession of Mercury's orbit, "requires a stream of regular adjustments" or corrective patches, which basically increase the complexity of the explanations of those anomalies to a level similar to that of describing the anomalies themselves. General relativity then provided a superior, more compact, and elegant set of field equations that not only subsumes the previous phenomena that Newtonian mechanics successfully accounted for, but also explains these and other "anomalies" in a more compressed form than before. In fact, the continued success of the scientific method in more compact and elegant mathematical theories corroborates the conclusion that comprehension necessarily involves compression of natural phenomena[6].

We know from AIT that once one can sufficiently approach universally optimal compression (i.e. approximate the actual values of

algorithmic complexity), any necessary increase in complexity (of the phenomena already explained along with those to be explained) is proved to require a novel theory that is (entirely or at least a proper part of it) irreducible to the old one, like when one needs to find a new axiom. From the algorithmic coding theorem (ACT)[10,11], the minimisation of such an irreducibly larger quantity of complexity also corresponds to the best inference method (such as the case when one employs abductive reasoning) for the new theory (or e.g., the new axiom) one can devise from a yet unexplainable phenomenon. As discussed in the Supplementary Information, we argue that these two features are central to tackling the problem of AGI and ASI.

The invariant and universal properties of algorithmic information imply that compression is more than a complexity index that might be correlated to abstraction and prediction. In fact, optimal compression is only achieved by the best formal-theoretic methods proven across the whole landscape of algorithms and methods one may attempt to apply, regardless of the type of agent applying them. Optimal compression is one such task that subsumes any other task—formalisable into an algorithm or a mathematical method that can be computationally implemented—an agent may perform in order to create a new theory that predicts new phenomena. Unlike directly equating compression/prediction to intelligence[74] or straightforwardly applying the ACT like in other universal induction-based methods, we propose that compression is a necessary and fundamental condition for comprehension if it is achieved through (and as a product of) the interaction between the AI agents being evaluated, the evaluator agents (including the methods, frameworks and metrics we formalise), and the external real world whose process may affect and be affected by the other two entities. For example, notice that this implies that the ACT itself becomes a constituting knowledge that the evaluator agents may devise by formalising a new mathematical theory, obtaining such a formal-theoretic knowledge after the experience (i.e., after the interactions take place). As explained in the Sup Inf, this is a distinctive characteristic of Algorithmic Information Dynamics (AID)[27–29,75] upon which our proposed framework is based.

As presented before in refs. 6,76, the remarkable features of AIT[77,78] discussed in the present section seem to underpin the apparently unreasonable effectiveness of algorithmic complexity[79] and computation[73] in explaining the natural world, including cognition, and in advancing science as the practice of finding or synthesising models that can explain and predict natural phenomena and the world. Thus, by putting forward a formal and more objective approach to measuring general intelligence, we propose in the section "SuperARC testing framework" a test for ASI and AGI based on AID and AIT, namely *SuperARC*, that specifically tests recently strongly associated features with intelligence in the context of discussions of AGI[30–32,80–85]. While human intelligence includes many other abilities to perform a myriad of tasks, here we chose to focus on abstraction, explanation, and prediction related to building new formal theories and to scientific creativity. Those are the ones for which we have computational methods and a solid theoretical foundation that has proved the universal and agnostic properties that a testing framework for general intelligence should aim at, particularly if one wants to tackle the distinction between narrow AI and AGI (see Supplementary Information).

SuperARC testing framework

Based on the theoretical background presented in the Supplementary Information, we ground our framework on the following aspects:

- *intelligence* necessarily involves the ability to create (i.e., through abduction from the experience) or enact (i.e., through prediction from future interactions) a computational generative model that effectively explains any given data while losing the least amount of information as possible;
- and *greater* intelligence corresponds to performing prediction and abduction as close to the optimal solution as possible while maximising the compression of the generative model.

As a metric for (general or super-)intelligence, designing tests that measure these abilities can lead to a more nuanced and computationally grounded understanding of intelligence that is applicable to biological (e.g., animal), human cognition, and computational intelligence. This can establish a universal approach to measuring the capabilities of intelligent systems, serving as both a theoretical and a practical upper bound for the highest possible levels of compression, such as model abstraction and prediction, which are believed to be fundamental features of intelligence.

As discussed and elaborated in the Supplementary Information, using algorithmic complexity as a measure of model compactness (i.e., compression, conciseness or summarisation) and optimal prediction provides an agnostic quantitative metric, as its value corresponds to the shortest possible programme capable of correctly reproducing (via decompression) a given dataset, and its optimal prediction value is governed by algorithmic probability. First, unlike standard tests that assess intelligence based on predefined 'correct' answers—inevitably influenced by subjective notions of correctness—we shift the focus to identifying the shortest possible explanation for a given dataset, an explanation that is proven to be sufficient for predicting not only the given dataset but also future outcomes. In this context, correctness is understood purely as the ability to reproduce exactly the same original data (i.e., losslessly). Secondly, an agnostic method aims to achieve measurable quantities as independent of human biases as possible, including those high-order biases in the scientific practice, such as when one chooses to employ one formal theory instead of another in order to model certain phenomena (see the section "Compression as comprehension about (and as part of) the world").

Beyond a measure of a single-purpose compression task (as discussed and unpacked in the Sup Inf), the SuperARC test is a proposal to capture the potential future trajectories leading to hybrid neurosymbolic systems more capable of the abstraction and planning, deemed central to what has been conceived of AGI and ASI[18,31,32], one that may take into account statistical pattern matching, but favours symbolic regression and programme synthesis as a test of intelligence based on optimal inference rather than statistical 'reasoning'. The test proposed expands current efforts to characterise AGI, such as the abstraction and reasoning corpus (ARC) challenge[30], which have been suspected to be 'hackable' from test result leaks because the test data set is fixed (even if part of it is concealed but prone to be leaked). Unlike recent results in the ARC-AGI test, our results find a similar lower performance than that reported in a recent mathematical benchmark test[86], with the advantage that our proposed test does not require the selection of human mathematical problems, and the test problems can be dynamically generated with test elements introduced cheaply and efficiently. Although this new test may require the selection of objects and elements such as sequences, unlike the original ARC challenge tests, this selection can be based mainly on quantitative measures of complexity and less on human selection.

These features are crucial in avoiding biases introduced by the datasets, such as benchmark contamination, when evaluating the performance of an AI algorithm. Given that one would be trying to measure the ability of the learning algorithm to predict phenomena whose type or class was not the one of the data it was trained for in the first place, this is especially the case in *zero-shot learning* scenarios, where any small leakage of data with information about the (upcoming and irreducibly new) test to be performed makes a big difference in the score. Due to the mathematical properties in AIT discussed in the Supplementary Information, SuperARC avoids human-centric and other cognitive biases because lower (algorithmic) complexity (higher compression, or equivalently, higher algorithmic probability) of a

model is proven to indicate better overall prediction capabilities, regardless of the nature of the new phenomena or the type of data on which one is trying to measure the generalisation capabilities of the AI algorithm. For example, even if one can update the benchmark test in practice with a new type of task to be performed, this possibility itself assumes that a new type of task might be known to us, rendering the test inherently prone to contamination. Following from the properties of algorithmic probability, SuperARC quantifies prediction in new contexts and potentially different scenarios without the need for a new type of task, distinct data, or posterior apprehension of previously unknown phenomena.

The role of SuperARC in distinguishing algorithmic from statistical prediction

While prediction is fundamental to both human-centric and algorithmic benchmarks, the nature of what is being predicted differs fundamentally. Human-centric benchmarks evaluate whether models can predict outputs that humans would generate given specific inputs, which is a task solvable through statistical pattern matching over human-generated corpora. This is because, as models are increasingly trained on datasets that cover more of human knowledge, performance on these benchmarks asymptotically approaches data memorisation rather than genuine understanding.

SuperARC, by contrast, evaluates whether models can induce the algorithmic structure that underlie the sequences, i.e., that can find the minimal programme that generates the observed data. This capability is irreducible to pattern matching because:

- *Infinite hypothesis space*: Unlike human-centric tasks with finite answer sets, algorithmic induction in principle searches over an infinite space of possible programmes, and thus possible formal theories;
- *Distribution shift immunity*: Novel algorithmic patterns (new combinations of primitives) are fundamentally out-of-distribution, requiring genuine abductive reasoning, such as when one devises a new axiom;
- *Compression-prediction duality*: The theorem proven in Sup Inf establishes that the success of predictions is equivalent to compression over the algorithmic space, which subsumes the statistical space, although the equivalence does not require statistical evaluation or success.

As shown by our results, we argue that such a theoretical difference uncovers a practical consequence: models can simultaneously improve on human benchmarks while regressing on algorithmic reasoning. This divergence should be impossible if both algorithmic prediction and statistical prediction were supposed to measure the same underlying predictive capability, and therefore SuperARC provides empirical evidence that captures a distinct and arguably more fundamental aspect of intelligence.

CTM and BDM: A neurosymbolic approach to Superintelligence benchmarking

The SuperARC framework accommodates any type of data as input-output pairs, requiring only a complexity-based metric to be predefined. To achieve this, in addition to approximate methods to algorithmic complexity, such as LZW and ZIP, which are more closely related to *Shannon Entropy*[51], we use the *Block Decomposition Method* (BDM) as our gold-standard approach to algorithmic compression that goes beyond statistical compression or statistical pattern-matching[35]. Using the principles of both classical and algorithmic information theories, BDM combines the calculation of the global Shannon Entropy rate of the object with local estimations of the algorithmic complexity of smaller blocks into which the object is decomposed, for which values are found in a pre-computed database of direct approximations of algorithmic probability. By combining both statistical and algorithmic inference methods, one way to think of BDM is by depicting it as a Deep Learning Transformer, which aims to build a predictor that maximises the probability of being correct in explaining the data by looking for long-range and short-range correlations. The difference, in this case, is that long-range correlations are covered by Shannon Entropy (not fundamentally different from Transformers), but short-term correlations are estimated using the principles of algorithmic probability through the ACT[29,33,34,87] (see Supplementary Information). See the discussion on the limitations of BDM below. In this manner, BDM is based on combining the best capabilities of Shannon entropy-type metrics to find patterns (e.g., block entropy rate[88]) with universal (Solomonoff) induction-based approaches (such as the minimum description length[21]) through algorithmic complexity, and thus deals with uncertainty in an optimal Bayesian fashion based on the principles of algorithmic probability[20]. BDM improves upon Shannon entropy and LZW compression, which are limited to detecting only statistical regularities, that is, pure pattern-matching approaches. In fact, BDM subsumes these methods, and therefore one can only do better in capturing structure than statistical compression algorithms, as BDM detects both regular statistical patterns and recursive ones with causal generative signatures[35,51,88] (see also Supplementary Information). By recursive, we mean exactly those that are not statistical in nature (e.g., the digits of the mathematical constant π do not display any statistical patterns, but it is recursive).

The BDM relies on the following assumptions:

1. In the case of small enough objects, their *algorithmic complexity* can be approximated using an exhaustive search (sometimes guided, e.g., with AID).
2. For larger objects, breaking them into smaller parts allows for the approximation of the overall complexity by summing the complexity of individual blocks, with a correction factor to account for interactions between the blocks.
3. For every other length, values of Shannon entropy rates are calculated and combined with the previous values by using the same principles of information theory.

Formally, let x be a string divided into blocks x_i, with $x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, where $\oplus$ denotes a concatenation operator. The *BDM complexity* of a string x, denoted by BDM(x), is given by

$$\mathrm{BDM}(x) = \sum_{i=1}^{n} \mathrm{CTM}(x_i) + \log m_i \tag{9}$$

where:

- CTM(x_i) is the algorithmic complexity approximation for block x_i, derived from the coding theorem method (CTM).
- $\log m_i$ is a correction factor accounting for the multiplicity m_i of how many times the block x_i appears.

For a generalised version of BDM holding for any encodable object (see ref. 88).

The *coding theorem method (CTM)* is a method based on the ACT[12,28] and Supplementary Information, which connects probability to complexity, randomness, and prediction[29,33,34,87]; and the ACT underlies the universal induction-based methods applied to Artificial Intelligence. CTM works by searching for all the formal-theoretic explanations (models or programmes) for an object that are shorter than the object itself[35], in order to calculate the ratio of those explanations of a particular object with all the explanations found for any object. From the value obtained for each of these ratios, one can approximate the algorithmic probability of an object, and thereby its algorithmic complexity via the ACT, so that a list of these pre-computed probability values is built, which in turn can be used to approximate the universal distribution[34].

On the one hand, CTM provides an approximation to *algorithmic probability*$P(s)$ by connecting the empirical frequency of occurrence of an object produced by a random computer programme with its *algorithmic complexity*$K(s)$ and also keeps track of the set of programmes that generated the original object (see Supplementary Information). On the other hand, BDM offers a method to map the micro-programmes produced by CTM to their corresponding pieces from the larger object, to explain by decomposing the original object into smaller blocks for which micro-programmes have been found by CTM with a correction factor for block interactions (e.g., repetitions). While CTM operates by brute force and thus is only effective for small programmes/models, BDM leverages the pre-computed distributions that can be queried in linear time and stitches together longer explanations from small computer programmes according to the rules of information theory to guide the search for the best sequence of programmes explaining larger objects, thereby constituting a method that approximates the optimal causal explanation of the objects. BDM also allows massive parallelisation because objects with low complexity (i.e., higher causal impact at the global level) are the most frequent according to algorithmic probability and therefore are exponentially more frequent, counteracting their intractability[35].

With limited computational resources, because of the limitations from the CTM, BDM behaves exactly like algorithmic complexity at the local scale and exactly like entropy at global scales[35,51,88]. In principle, with unbounded computational resources, all the algorithmic generative models at any scale would be known/computed. As a consequence, BDM would return the optimal value given by algorithmic complexity, and therefore would achieve optimal compression in the general case. In addition, for the particular cases in which the conditions for optimal statistical compression (like those discussed in the Sup Inf) are met, BDM is also proved to perform as optimally as entropy does because, at the local scales, algorithmic complexity already encompasses and subsumes entropy, and at the global scale, BDM converges to entropy via the CTM. Therefore:

- in practice, BDM always performs equally or better than entropy;
- BDM through CTM converges to algorithmic complexity in the limit, outperforming any statistical compression method;
- both in principle and in practice, BDM remains sensitive to underlying structures even for objects with maximal Shannon entropy.

Following from the fundamental properties of AIT (discussed in the Supplementary Information), such as universality, invariance, maximality, and optimal prediction, BDM offers a '*principled*' alternative in comparison to statistical measures. It is currently the only viable and computable approximation to algorithmic complexity grounded in AIT beyond statistical compression algorithms.

Thus, BDM is a hybrid neurosymbolic[89] method that combines statistical machine learning and symbolic regression, and prediction that can be applied to inverse problems in causality[25,26]. BDM can be thought of as a quintessential type of neural network transformer (as in self-attention), where it estimates the local (short-range) causality through algorithmic complexity while computing long-range correlations through Shannon entropy guaranteed convergence (worst case)[88]. Such a benchmarking method has already been reported in applications to data summarisation[71] and in various fields ranging from cell and molecular biology to genetics[25,90] to biosignatures[50,91,92].

BDM is an approximation to algorithmic complexity and probability, with its known limitations that demand explicit discussion. BDM's complexity estimates depend on choices of block size for decomposition, with different block sizes potentially yielding different rankings of sequence complexity. As mentioned above, this limitation occurs because of limited computational resources, since in the asymptotic theoretical limit, the algorithmic complexity could be approximated across any coarse-graining scale. For the short range or smaller blocks, BDM uses a precomputed table of short programmes (or equivalently, Turing machines with a few limited number of states) for which the Busy Beaver values are known, and therefore one can solve the halting problem. In the long range or for the largest block sizes possible, BDM upscales those actual algorithmic complexity values via Shannon (block) entropy. Therefore, due to limited computational resources, the resulting BDM value may inherit the same limitations of entropy in the long-range scenario. In addition, the empirical application of the method's reliance on decomposing sequences into overlapping or non-overlapping blocks means that patterns spanning boundaries between blocks may not be captured optimally, potentially underestimating complexity for sequences with long-range dependencies. These are fundamental characteristics of practical computable approximations to algorithmic complexity or algorithmic probability, which remain uncomputable in the general case.

Despite these limitations, our use of BDM is justified for reasons that directly address concerns about result interpretation: first, the models we evaluate fail predominantly on short sequences where BDM's approximation is most accurate and where the gap between estimated and actual algorithmic complexity values is minimal; secondly, our conclusions do not depend on fine-grained complexity distinctions but on coarse patterns (models fail across broad complexity ranges rather than at specific threshold values where approximation errors might matter); thirdly, our neurosymbolic baseline employs actual programme synthesis through systematic enumeration rather than BDM estimation, yet reaches qualitatively similar conclusions. While future work comparing alternative complexity metrics may uncover or highlight other aspects or discrepancies, we argue that the robustness of LLM failures across these multiple lines of evidence suggests our core findings about inadequate algorithmic reasoning in current models remain sound regardless of specific approximation method choices.

Applicability of CTM and BDM to abstraction and planning in machine learning. BDM with CTM can be applied both as a reference and as a direct generative model. This is because it provides a fundamental complexity-based value estimation that can guide and evaluate other predictive and learning approaches, but also serves as a standalone predictive system.

CTM helps identify the set of candidate underlying generative mechanisms and provides a set of models from which it can actively predict future values by running it further into the future providing a set of projections. CTM forecasting requires an iterative refinement process in which multiple possible generative programmes are tested and updated. CTM can help select the most likely programme candidates by favouring those with lower complexity in accordance with the principles of algorithmic probability.

In a predictive task, multiple candidate programmes generated by CTM are evaluated against new observations, discarding those that are not consistent with the new data while retaining the set of shortest valid programmes that do. Planning requires CTM as the algorithmic mechanism to iteratively refine predictions from projections. CTM serves as a criterion for model selection—helping identify which approach best maintains parsimony and explanatory power—rather than functioning as a decision-making agent of its own.

BDM then stitches multiple programmes that can explain longer pieces of data and larger objects by using the rules of classical information theory, serving as a reference point to compare different models based on how well they align with the inherent complexity of the data. By breaking down an object into smaller pieces and estimating their individual algorithmic complexity using CTM, BDM provides a tighter recursive upper bound to traditional pattern matching. BDM leverages, therefore, both algorithmic and classical information theory as a proxy for deeper connections to causality, allowing it to indicate how predictable a time series or integer sequence is. Both

CTM and BDM combined can benchmark different models on the basis of how efficiently they approximate the set of the best explanatory and generating mechanisms.

The way BDM approaches uncertainty is to update the belief at time t of an object s (e.g., an integer sequence), and choose a (small) programme p' to explain for the next digit $i \in s_{i-1}$ deviating from the previous hypothesis p; or in case we do not have (or we cannot obtain) such a programme for this observation, we combine smaller programmes p'' to explain observation of digit $i \in s_i$ at index $t+1$. In this manner, the ability of BDM to capture both local and global patterns in a time series or integer sequence makes it a powerful tool for approximating complexity and enabling prediction, aligning with the principles of algorithmic probability and Levin's universal distribution.

BDM shows some fundamental similarities but in pure form to "Attention is All You Need" algorithms and LLM's by assigning different weights to different parts of an object focusing both on short-range and long-range correlations where the short-range is recursively correlated hence based on causally generated models for that patch of data unlike LLMs and other ML approaches that rely only on Shannon-entropy-based correlations or basic pattern-matching that BDM only uses for its long-range correlations. BDM is therefore a proper generalisation of the short- and long-range capabilities that gave LLMs their particular advantage in language[35]. Together with CTM as a universal generator[33], the CTM/BDM combination represents a model of models of languages, where languages are all computer languages, and a superset of LLMs themselves.

As mentioned above, a limitation of CTM is that running CTM to approximate model compression and achieve optimal prediction is computationally very expensive. If there were infinite resources, CTM would perform perfect recursive compression and provide the most optimal answer to any computable question given an observation. However, even with access to infinite resources, there are no theoretical or practical guarantees of LLM convergence to any optimal answer. In practice, LLMs are currently more expensive in applications where approaches like CTM could deliver better results (such as for this benchmark, empirically proven to better characterise questions and predict answers encoded in the form of binary sequences) without spending billions of USD in training giant neural systems like LLMs. However, our point is that one does not need to pick one over the other, as they can be combined to provide the best approximation to both an optimal and efficient path to an answer under time and resource restrictions. In this regard, CTM/BDM is a resource-bounded approximation to optimal inference that combines pure forms of each side (neuro-based on classical statistics, and symbolic-based on optimal theory). The CTM/BDM combo represents the purest form of neurosymbolic computation with no extra steps.

In the framework we propose, CTM and BDM are used as a benchmark to evaluate model performance and as a representative of a Universal AI[9] method capable of ASI[8]. They can be applied to test both:

- *compression as model abstraction*: The BDM can approximate the algorithmic complexity of a time series by decomposing it into smaller subsequences (blocks), computing the complexity of each block using CTM, and summing up the block results. This serves as a measure of the recursivity of the time series, but also serves as a method to find generating mechanisms (a set of algorithms that produce each past and possible future element/token of an object, in particular, a time series).
- *prediction as planning*: Using the BDM complexity as a proxy for the time series' regularity, one can infer the predictability of future values. Lower BDM complexity implies a simpler underlying structure, which can help in forecasting future elements of the series—which is similar to how algorithmic probability and universal distribution can be used for predictive modelling. (See Sup Inf). This is related to planning, because once several programme pathways are identified, one can verify each against the next token and update the programme set (by discarding those programmes that did not fit the next token) while keeping the shortest programme criterion.

A method for measuring comprehension via algorithmic probability

BDM is a divide-and-conquer method that extends the power of a CTM that approximates local estimations of algorithmic complexity via the theory of algorithmic probability, a foundational result established in AIT (see Supplementary Information). The method consists of finding the sequence of computer programmes that can generate the original piece of data—each programme represents a hypothesis or model for the time series and a sequence of datasets that can be interpreted as time series, binary and non-binary—providing a closer connection to complexity (or irreducible information content) than previous attempts based on statistical regularities such as popular lossless compression schemes[35].

Based on AIT, we measure comprehension of LLMs (see the section "Compression as comprehension about (and as part of) the world") with a test designed to assess the model's ability to generate code or mathematical models/formulae that compress sequences of increasing complexity. Non-binary sequences are categorised into three levels of complexity (Low, Medium, and High) representing datasets that exhibit simple, intricate, and random patterns, respectively. Binary sequences, on the other hand, are classified as either random or what we call 'climber strings, low-complexity strings as defined in the following section. Thus, a pragmatic compression-as-comprehension test is designed and applied to various LLM models and versions, encompassing test elements of diverse complexity classes that can be understood and compared individually and collectively.

Algorithm 1. Pseudo-code for SuperARC framework

Require

- $D_{\text{low}}, D_{\text{medium}}, D_{\text{high}}$ (datasets of any type with low, medium, and high complexities with sizes given as $|\cdot|$. These are needed to ensure complexity diversity, but the choice of three groups is arbitrary and can be changed by the user.);
- enc (encoding chosen to put the datasets in a common format);
- $\mathcal{M}$ (complexity metric used to qualify the datasets and quantify the complexities of the models created by LLMs);
- $\mathcal{T}$ (test formula to evaluate a candidate model).

$c_{\mathcal{M}} \Leftarrow$ an array containing binary values.
$Aux_{\mathcal{M}} \Leftarrow$ an array containing auxiliary values.
$All_{\mathcal{M}} \Leftarrow$ an array containing complexity values.
for $k \in$ {low, medium, high} **do**
  $D_{k,\text{encoded}} \Leftarrow$ encoding of D_k using enc (the UTF-8 or ASCII binary representation of strings or a binary representation of integers, for example).
  for $j \in \{1, 2, \ldots, |D_{k,\text{encoded}}|\}$ **do**
    $R_{k,j} \Leftarrow$ the response obtained from prompting a LLM model to write a programme to reproduce the jth element of $D_{k,\text{encoded}}$.
  $c_{k,j} \Leftarrow$ a binary variable indicating if the output obtained after running $R_{k,j}$ is correct (equal to the input dataset) or not.
  $\mathcal{M}(R_{k,j}) \Leftarrow$ the complexity of $R_{k,j}$ according to $\mathcal{M}$.
  $a_{k,j} \Leftarrow$ a vector with real-valued variables representing the result of applying auxiliary functions to $R_{k,j}$.
    Append $c_{k,j}$ to $c_{\mathcal{M}}$.
    Append $\mathcal{M}(R_{k,j})$ to $All_{\mathcal{M}}$.
    Append $a_{k,j}$ to $Aux_{\mathcal{M}}$.
  end for
end for
$\mathcal{T}(c_{\mathcal{M}}, All_{\mathcal{M}}, Aux_{\mathcal{M}}) \Leftarrow$ the test score for the candidate model.

In other words, the SuperARC framework assesses how the LLM model is able to generate an algorithm $\mathcal{A}$ such that, when applied to the input data set τ, it is able to compress this input by learning its features and producing a compressed representation ∂. Then, by inverting such an algorithm and obtaining the algorithm $\mathcal{A}^{-1}$, the inputs τ are obtained losslessly with minimal complexity of the combined algorithms according to a complexity metric $\mathcal{M}$. From AIT, we have that universal induction indicates the best way to predict future elements of a sequence as favouring the simplest (i.e., the least complex) hypothesis or explanation—which aligns with the concept of Occam's razor, as discussed in the Supplementary Information. By minimising the complexity of the description of the data $(\mathcal{M}\left(\mathcal{A}^{-1}\circ\mathcal{A}\right))$, the theory effectively formalises prediction $(\mathcal{A}^{-1}\circ\mathcal{A}:\{\tau\to\partial\to\tau\})$. In this manner, the framework can be described as the pseudo-code in Algorithm 1 for which the LLM is presented with the following task:

$$\begin{aligned}&\text{minimize}_{\mathcal{A},\mathcal{A}^{-1}} \quad \mathcal{M}\left(\mathcal{A}^{-1}\circ\mathcal{A}\right)\\ &\text{subject to } \mathcal{A}^{-1}\circ\mathcal{A}:\{\tau\to\partial\to\tau\}\end{aligned} \tag{10}$$

It is important to clarify that the encoding *enc* in Algorithm 1 does restrict the analysis. For example, different data types could be encoded as vectors obtained in the latent space of a given deep neural network. As long as the encoder algorithm is known and common to all the input data, the framework can be applied because of the theorems in AIT. In particular, the information non-increase theorem [10] indicates that, for any computable function f, the inequality $K(f(x))\leq K(x) + K(f) + \mathbf{O}(1)$ holds. Therefore, once f is fixed for all data sets considered, $K(f)$ becomes an additive constant that does not affect the analysis when $K(x)$ is used to investigate the value of $K(f(x))$. In other words, the encoding is not important as long as it is known and kept fixed during the analysis.

It should also be noticed that CTM/BDM is not purely a brute-force approach[35]. Although CTM alone would be a brute-force approach that seeks the shortest computer programmes explaining the data, BDM is not (see the section "CTM and BDM: A neurosymbolic approach to Superintelligence benchmarking"). CTM/BDM operates by exploiting the best of both worlds[35], operating at the fine balance between what traditional machine learning and deep learning approaches implement, while also combining it with optimal Bayesian causal inference[51] or algorithmic deconvolution[26]. As further discussed and explained in the Sup Inf, we have called this approach algorithmic information dynamics (AID)[27–29].

In order to present a quantitative implementation of a test following the SuperARC framework, an exploratory analysis is needed. This will be described in the next section, " Design of experiments".

Design of experiments

To evaluate how LLM models can be assessed within the SuperARC framework, we consider datasets composed of non-binary and binary sequences. It is worth highlighting that this choice is not mandatory, and all data should be encoded consistently. Different encodings may lead to different BDM values and thus other benchmarks may favour one type/structure of data or the other. Nevertheless, as BDM is an approximation to algorithmic complexity, AIT guarantees that algorithmic probability converges to the optimal solution in the asymptotic limit (if enough computational resources are provided).

Although prompting has been shown to considerably impact the performance of LLMs in a code generation task[93,94], we use the simplest possible prompt to avoid providing additional information to the LLM, which could bias its output (even if towards better codes). Also, for the same reasons, we performed zero-shot learning tasks.

The non-binary sequences of integers used in the questions were divided into three levels of complexity, as indicated in the previous subsection. Intuitively, the complexity levels could be explained as follows:

1. *Low complexity*: Sequences of digits or integers whose pattern is easily recognisable by a person and highly compressible. They have low CTM/BDM values.
2. *Medium complexity*: Sequences of digit integers generated recursively with longer formulas than those in the simpler set. They have intermediate CTM/BDM values.
3. *High complexity*: Random-looking sequences of digits or integers. They have high CTM/BDM values.

The following experiments were carried out:

- *Next-digit prediction task with binary and non-binary sequences*: We prompted LLMs specialising in time series forecasting to predict the digits of non-binary sequences of increasing complexity of two types. The first type are random binary sequences according to increasing CTM/BDM, and the second type are called '*climbers*'. – *Climbers* are strings that, when sorted by algorithmic probability in descending order (highest to lowest probability), or algorithmic complexity in ascending order (lowest to highest randomness), these binary sequences are longer than strings in their same complexity group, defined as strings with the same or very close complexity values as measured by BDM but of significantly longer length than them. This means that for these strings, their complexity is definitely not driven by string length only but by (simple) their internal structure, aligning with an intuitive understanding of simplicity vs. randomness in sequence structure[34]. In other words, these are strings that clearly correspond to lower randomness values because they show lower complexity estimations compared to shorter strings in the vicinity. For example, the sequence 0101010101... up to a certain finite size n is clearly less algorithmically random and therefore more algorithmically probable than any other more random-looking string, short or long of the same size n, and therefore such a patterned sequence must appear earlier in a complexity hierarchy if BDM works correctly. So, knowing these are highly structured strings with high algorithmic probability, we tested whether LLMs would identify them by producing short models and better predictions for them compared to others. – *Free-form generation task with binary and non-binary sequences*: We challenged advanced language models, including GPT-4o, GPT-o1, Claude 3.5 Sonnet, GPT-4o-mini, Grok, o1-mini, Qwen, and DeepSeek, to generate models, algorithms, formulas, or Python scripts capable of reproducing specific target sequences. – *Code generation task with non-binary sequences*: An answer was requested to generate source code that would produce sequences of numbers using prompts of the following type:

"With no additional explanations or comments or notes, write the code in {} programming language to produce the sequence [sequence].

A full list of all sequences can be found in the Supplementary Information. Each prompt was submitted with varying values for the temperature parameter: [1, 0.7, 0.5, 0.2, 0.001], allowing for a comparison of its effect on the quality of the outputs.

Each prompt was formulated in such a way that it was expected that the LLM would return the code generating the defined sequences in the following programming languages: ArnoldC, C++, Python, Mathematica, Matlab, R, and JavaScript. After the codes were generated, they were executed, and their performance was compared.

Code and free-form generation tasks. Code generation in different programming languages was performed exclusively using non-binary

sequences of increasing complexity and was run only by ChatGPT. In contrast, free-form generation was conducted using both non-binary and binary sequences and prompted to a list of the most prominent LLMs. Depending on the case, the following processing steps were applied according to Algorithm 1:

For the jth element of $D_{k,\text{encoded}}$, $k \in$ {low, medium, high}, the output code (able to reproduce these elements) provided by the LLM model was $R_{k,j}$. Then, for these, after being logically evaluated to ensure that they produced the expected results, the following functions were applied.

- Auxiliary functions:
  - The script and model/formula lengths generated by LLMs were measured by the number of characters.
  - Since programme or model/formula length was taken as an indicator, and sequences were defined as either single- or multi-digit numbers, a process called normalisation was applied to the original code generated. This normalisation took out repetitions of the entire sequence from the code if this was included. For example, if a script that aims to reproduce the sequence '1, 2, 3, 4' were to be 'Print(1, 2, 3, 4)', after being normalised, it would be transformed into 'Print()'. In this way, we obtained lengths of normalised and non-normalised answers.
  - Compression: The zlib algorithm was applied to the normalised and non-normalised answers generated; also to the target sequences of digits alone in such a way that we obtained ASCII representation of the compressed and non-compressed variations of all scripts and their lengths.
  - For the code in different programming languates, a compression percentage measurement was designed: this is an indirect measurement of compression based on the number of elements of a sequence and their order of appearance in the answer to a question. For example, if the target sequence is "1, 2, 3, 4, 5" the code Print([1, 2, 3, 4, 5]) is considered to be 100% uncompressed, not only because it contains all elements of the original sequence, but it also keeps its original order. On the other hand, the code for $i = 1{-}5$ Print(i) is considered to have a higher degree of compression, since it only contains 2 of the original elements, but the logic to generate it "lives" in the code. Additionally, the code repeat print($n + 1$) is considered more compressed.
  - A set of filters was designed to study our results, and they were applied accordingly if non-binary or binary sequences were the target:
    - *Print code* (applicable to binary and non-binary sequences): this type of programme could be of two types: (a) the target sequence defined as a variable or a set of variables followed by a print(sequence), for example $a =$ '1,2,3', print(a), (b) a simple print(Sequence) without definition of variables, for example print('1,2,3').
    - *Correct code* (applicable to binary and non-binary sequences): if the given answer by any LLM models generated the target sequence.
    - *Print-correct* (applicable only to non-binary sequences): the combination of the two above.
    - *Incorrect-print* (applicable only to non-binary sequences): the negation of the previous one.
    - *Ordinal* (applicable only to binary sequences): The model or formula exclusively references the positional arrangement of digits to reproduce the target sequence.
  - The application of filters was done over all our measurements, allowing classification by averages of compressed, not compressed, normalised, and not normalised answers, filtered by prints, or correct and all its combinations.
- *Correctness variable*: Computer programmes and models/formulae were evaluated or executed in their respective compilers/interpreters to verify if they generated the target number sequences correctly.

Next-digit prediction task. For the next-digit prediction task, we used binary and non-binary sequences. We compared results obtained with different LLMs specialising in time series forecasting to predict values in the sequences used in our experiments. The models used included Chronos, TimeGPT-1, and Lag-Llama. Our criteria for selecting these models can be summarised as follows:

1. researchers reported very high-quality predictions in zero-shot tasks, i.e., in time series never seen before;
2. they were compared to traditional machine learning models, showing superior results;
3. they are reported to capture dynamics in real-world datasets rather than relying on simple statistical patterns;
4. authors advocate for the superiority of LLM architectures in time-series forecasting;

We split our sequences into several segments, using the models described to predict the remaining portions, which correspond to 10%, 25%, 50%, and 75% of the sequence. This approach divided the sequence into a 'root' and a 'target'. For instance, given the sequence [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and a prediction of 25%, the 'root' (the context provided to the prediction model) would be [1, 2, 3, 4, 5, 6, 7, 8], with the 'target' [9, 10] expected to be predicted. An asymptotic distribution of test results $\varphi_1, \ldots, \varphi_n$ for growing n where $|s| = n$ should provide some insight into the generalisation of the capabilities of the LLMs to scale their reported abilities, if any.

We employed three methods to measure the accuracy of the predicted target:

1: *Sort similarity*: This measures how many elements in the target sequence were predicted correctly, with their order being considered.

2: *General similarity*: This measures the correctness of predicted elements, without considering their order.

3: *Levenshtein*: This measures the Levenshtein distance between the expected and predicted sequences after converting them to strings.

Data availability

The data generated for this work is available at https://github.com/AlgoDynLab/SuperintelligenceTestwhere a benchmark table will be updated regularly for frontier models after they are released and tested.

Code availability

The code generated for this work is available at https://github.com/AlgoDynLab/SuperintelligenceTest.

References

1. Spearman, C. general intelligence, objectively determined and measured. *Am. J. Psychol.* **15**, 201–292 (1904).
2. Gauvrit, N., Zenil, H., Soler-Toscano, F., Delahaye, J.-P. & Brugger, P. Human behavioral complexity peaks at age 25. *PLoS Comput. Biol.* **13**, e1005408 (2017).
3. Hernández-Orallo, J. & Minaya-Collado, N. A formal definition of intelligence based on an intensional variant of algorithmic complexity. In *International Symposium of Engineering of Intelligent Systems (EIS98) Kybernetics, Emerald Insight* (Mann C. ed.) 146–163 (1998).
4. Zenil, H., Marshall, J. A. R. & Tegnér, J. Approximations of algorithmic and structural complexity validate cognitive-behavioural experimental results. *Front. Comput. Neurosci.* **16**, (2023).
5. Zenil, H. On the complex behaviour of natural and artificial machines and systems. In *Metrics of Sensory Motor Integration in Robots and Animals*, *Cognitive Systems Monographs* (eds

Bonsignorio, F. P., del Pobil, A. P., Messina, E. & Hallam, J.) 111–125 (Springer, 2019).
6. Zenil, H. Compression is comprehension, and the unreasonable effectiveness of digital computation in the natural world. In *Unravelling Complexity: The Life and Work of Gregory Chaitin*(eds Wuppuluri, S. & Doria, F.) 173–208 (World Scientific Publishing, 2019).
7. Chaitin, G. J. Gödel's theorem and information. *Int. J. Theor. Phys.* **21**, 941–954 (1982).
8. Solomonoff, R. *The Application of Algorithmic Probability to Problems in Artificial Intelligence* 473–491 (Elsevier, 1986).
9. Hutter, M. *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability. Texts in Theoretical Computer Science. An EATCS Series* (Springer, 2005).
10. Calude, C. S. *Information and Randomness: An Algorithmic Perspective* 2nd edn (Springer-Verlag, 2002).
11. Downey, R. G. & Hirschfeldt, D. R. *Algorithmic Randomness and Complexity. Theory and Applications of Computability* (Springer New York, 2010).
12. Li, M. & Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications* 4th edn (Springer, 2019).
13. Chaitin, G. *Algorithmic Information Theory* 3rd edn (Cambridge University Press, 2004).
14. Sutskever, I. *Talk at the Simons Institute: Ilya Sutskever (OpenAI)*. Video (Simons Institute for the Theory of Computing, 2023).
15. AI News. *AI Compresses Reality to Small Vector Space: Elon Musk* https://analyticsindiamag.com/ai-news-updates/ai-compresses-reality-to-small-vector-space-elon-musk/ (2021).
16. Legg, S. & Hutter, M. *Tests of Machine Intelligence* 232–242 (Springer, 2007).
17. Zenil, H. A turing test-inspired approach to natural computation. In *Turing in Context II, Historical and Contemporary Research in Logic, Computing Machinery and Artificial Intelligence* (eds Primiero, G. & De Mol, L.) (Royal Flemish Academy of Belgium for Science and the Arts, 2013).
18. Schmidhuber, J. Gödel machines: fully self-referential optimal universal self-improvers. In *Artificial General Intelligence, Cognitive Technologies* (eds Goertzel, B. & Pennachin, C.) 199–226 (Springer, 2007).
19. Levin, L. Universal search problems and algorithmic probability. *Probl. Inf. Transm.* **9**, 265–266 (1973).
20. Solomonoff, R. A formal theory of inductive inference. *Inf. Control* **7**, 1–22 (1964).
21. Rissanen, J. Modeling by shortest data description. *Automatica* **14**, 465–471 (1978).
22. Hernández-Orallo, J. *C-Tests Revisited: Back and Forth with Complexity* 272–282 (Springer International Publishing, 2015).
23. Belcak, P., Schenker, F., Kastrati, A. & Wattenhofer, R. Fact: learning governing abstractions behind integer sequences. In Oh *Proc. of the 36th International Conference on Neural Information Processing Systems* (Koyejo S. et al. eds), *NIPS '22* (Curran Associates Inc., 2022).
24. Hernández-Orozco, S. et al. Algorithmic probability-guided machine learning on non-differentiable spaces. *Front. Artif. Intell.* **4**, 25 (2021).
25. Zenil, H. et al. An algorithmic information calculus for causal discovery and reprogramming systems. *iScience* S2589-0042(19) 30270-6 (2019).
26. Zenil, H., Kiani, N., Zea, A. & Tegnér, J. Causal deconvolution by algorithmic generative models. *Nat. Mach. Intell.* **1**, 58–66 (2019).
27. Zenil, H., Kiani, N., Abrahão, F. & Tegner, J. Algorithmic information dynamics. *Scholarpedia* (2020).
28. Zenil, H., Soler Toscano, F. & Gauvrit, N. *Methods and Applications of Algorithmic Complexity: Beyond Statistical Lossless Compression* (Springer, 2022).
29. Zenil, H., Kiani, N. A. & Tegnér, J.*Algorithmic Information Dynamics: A Computational Approach to Causality with Applications to Living Systems* (Cambridge University Press, 2023).
30. Chollet, F. On the measure of intelligence. *arXiv Preprints* arXiv:1911.01547 (2019).
31. LeCun, Y. *A Path Towards Autonomous Machine Intelligence*. OpenReview Archive https://openreview.net/forum?id=BZ5a1r-kVsf (2022).
32. Assran, M. et al., Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture, 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, pp. 15619-15629 (2023).
33. Delahaye, J.-P. & Zenil, H. Numerical evaluation of algorithmic complexity of short strings: a glance into the innermost structure of algorithmic randomness. *Appl. Math. Comput.* **219**, 63–77 (2012).
34. Soler-Toscano, F., Zenil, H., Delahaye, J.-P. & Gauvrit, N. Calculating Kolmogorov complexity from the output frequency distributions of small turing machines. *PLoS ONE* **9**, e96223 (2014).
35. Zenil, H., Hernández-Orozco, S., Kiani, N., Soler-Toscano, F. & Rueda-Toicen, A. A decomposition method for global evaluation of Shannon entropy and local estimations of algorithmic complexity. *Entropy* **20**, 605 (2018).
36. Manicka, S. & Levin, M. The cognitive lens: a primer on conceptual tools for analysing information processing in developmental and regenerative morphogenesis. *Philos. Trans. R. Soc. B: Biol. Sci.* **374**, 20180369 (2019).
37. Feldman, J. The simplicity principle in perception and cognition. *WIREs Cogn. Sci.* **7**, 330–340 (2016).
38. Planton, S. et al. A theory of memory for binary sequences: evidence for a mental compression algorithm in humans. *PLoS Comput. Biol.* **17**, e1008598 (2021).
39. Tan, M., Merrill, M. A., Gupta, V., Althoff, T. & Hartvigsen, T. Are language models actually useful for time series forecasting? In *Proc. of the 38th Annual Conference on Neural Information Processing Systems* (2024).
40. Zhao, Y., Zhang, R., Li, W. et al. Assessing and Understanding Creativity in Large Language Models. Mach. *Intell. Res.* **22**, 417–436 (2025).
41. Dziri, N. et al. Faith and fate: limits of transformers on compositionality. In *Proc. of the 37th International Conference on Neural Information Processing Systems*, (Oh, A. et al. eds.)*NIPS '23* (Curran Associates Inc., 2023).
42. Arkoudas, K. Gpt-4 can't reason. Preprint at https://arxiv.org/abs/2308.03762 (2023).
43. Frieder, S. et al. Mathematical capabilities of ChatGPT. In *Proc. of the 37th International Conference on Neural Information Processing Systems*, (Oh, A. et al. eds) *NIPS '23* (Curran Associates Inc., 2023).
44. Kambhampati, S. et al. Position: LLMS can't plan, but can help planning in LLM-modulo frameworks. In *Proc. of the 41st International Conference on Machine Learning, ICML'24* (JMLR.org, 2024).
45. Zenil, H. Compression is comprehension and the unreasonable effectiveness of digital computation in the natural world. In *Unravelling Complexity: The Life and Work of Gregory Chaitin* 201–238 (World Scientific, 2020).
46. Hernández-Orozco, S., Kiani, N. & Zenil, H. Algorithmically probable mutations reproduce aspects of evolution, such as convergence rate, genetic memory, and modularity. *R. Soc. Open Sci.* **5**, 180399 (2018).
47. Soler-Toscano, F. & Zenil, H. A computable measure of algorithmic probability by finite approximations with an application to integer sequences. *Complexity* **2017**, Article ID 7208428 (2017).
48. Soler-Toscano, F., Zenil, H., Delahaye, J.-P. & Gauvrit, N. Correspondence and independence of numerical evaluations of algorithmic information measures. *Computability* **2**, 125–140 (2013).

49. Zenil, H., Kiani, N. A. & Tegnér, J. Low algorithmic complexity entropy-deceiving graphs. *Phys. Rev. E* **96**, 012308 (2017).
50. Abrahão, F. S., Hernández-Orozco, S., Kiani, N. A., Tegnér, J. & Zenil, H. Assembly theory is an approximation to algorithmic complexity based on lz compression that does not explain selection or evolution. *PLoS Complex Syst.* **1**, e0000014 (2024).
51. Zenil, H. A review of methods for estimating algorithmic complexity: options, challenges, and new directions. *Entropy* **22**, (2020).
52. Hubert, K. F., Awa, K. N. & Zabelina, D. L. The current state of artificial intelligence generative language models is more creative than humans on divergent thinking tasks. *Sci. Rep.* **14**, 3440 (2024).
53. Mirzadeh, I. et al. Gsm-symbolic: understanding the limitations of mathematical reasoning in large language models. Preprint at https://arxiv.org/abs/2410.05229 (2024).
54. Schuurmans, D., Dai, H. & Zanini, F. Autoregressive large language models are computationally universal. *arXiv preprint* arXiv:2410.03170 (2024).
55. Aljanabi, M., Ghazi, M., Ali, A. H., Abed, S. A. et al. ChatGPT: open possibilities. *Iraqi J. Comput. Sci. Math.* **4**, 62–64 (2023).
56. Yizhen, L. et al. Exploring the comprehension of ChatGPT in traditional Chinese medicine knowledge. *arXiv preprint*arXiv:2403.09164 (2024).
57. Bayani, D. Testing the depth of ChatGPT's comprehension via cross-modal tasks based on ASCII-Art: GPT 3. 5's abilities in regard to recognizing and generating ASCII—art are not totally lacking. *In Findings of the Association for Computational Linguistics: EACL 2024, pages 2063–2077, St. Julian's, Malta. Association for Computational Linguistics.*
58. Wei, F., Chen, X. & Luo, L. Rethinking generative large language model evaluation for semantic comprehension. In *ICML'24: Proceedings of the 41st International Conference on Machine Learning*, (Salakhutdinov R. & Kolter Z. eds) (2024).
59. Zhong, T. et al. Evaluation of OpenAI o1: opportunities and challenges of AGI. *arXiv preprint* arXiv:2409.18486 (2024).
60. Si, C., Yang, D. & Hashimoto, T. Can LLMs generate novel research ideas? A large-scale human study with 100+ NLP researchers. The Thirteenth International Conference on Learning Representations (2024).
61. Marcus, G. Deep learning is hitting a wall. *Nautilus* https://nautil.us/deep-learning-is-hitting-a-wall-238440/ (2022).
62. Feng, L., Zhang, L. & Lai, C. H. Optimal machine intelligence at the edge of chaos. *arXiv preprint* arXiv:1909.05176 (2019).
63. Marcus, G.*The Algebraic Mind: Integrating Connectionism and Cognitive Science* (MIT Press, 2001).
64. Bishop, J. M. Artificial intelligence is stupid and causal reasoning will not fix it. *Front. Psychol.* **11**, 513474 (2021).
65. Levin, L. A. Various measures of complexity for finite objects (axiomatic description). *Sov. Math. Dokl.* **17**, 522–526 (1976).
66. Levin, L. A. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Probl. Inf. Transm.* **10**, 206–210 (1974).
67. Levin, L. A. On the notion of a random sequence. *Sov. Math. Dokl.* **14**, 1413–1416 (1973).
68. von Mises, R. *Wahrscheinlichkeit, Statistik und Wahrheit* (Springer-Verlag, 1928).
69. Schnorr, C.-P. *Zufälligkeit und Wahrscheinlichkeit. Eine Algorithmische Begründung der Wahrscheinlichkeitstheorie* (Springer, 1971).
70. Schnorr, C.-P. A unified approach to the definition of random sequences. *Math. Syst. Theory* **5**, 246–258 (1971).
71. Zenil, H. et al. Minimal algorithmic information loss methods for dimension reduction, feature selection and network sparsification. *Inf. Sci.* **720**, 122520 (2025).
72. Calude, C. S. & Stay, M. A. Most programs stop quickly or never halt. *Adv. Appl. Math.* **40**, 295–308 (2008).
73. Wolfram, S. *A New Kind of Science* (Wolfram Media, 2002).
74. Schmidhuber, J. Driven by Compression Progress: A Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes. In *Anticipatory Behavior in Adaptive Learning Systems. ABiALS 2008. Lecture Notes in Computer Science* (Pezzulo, G., Butz, M.V., Sigaud, O. & Baldassarre, G. eds), Vol 5499. (Springer, Berlin, Heidelberg, 2009).
75. Abrahão, F. S. & Zenil, H. Emergence and algorithmic information dynamics of systems and observers. *Philos. Trans. R. Soc. A* **380**, (2022).
76. Zenil, H. (ed.) *A Computable Universe: Understanding Computation and Exploring Nature as Computation* (World Scientific, Singapore, 2012).
77. Kirchherr, W., Li, M. & Vitányi, P. The miraculous universal distribution. *Math. Intell.* **19**, 7–15 (1997).
78. Marvin Minsky & World Science Festival. The limits of understanding (2014). https://www.youtube.com/watch?v=DfY-DRsE86s&t=5392s. Marvin Minsky discusses the importance of algorithmic probability and universal induction in this panel discussion.
79. Zenil, H., Soler-Toscano, F. & Joosten, J. J. Empirical encounters with computational irreducibility and unpredictability. *Minds Mach.* **22**, 149–165 (2012).
80. Agrawal, A., Gans, J. & Goldfarb, A. *Prediction Machines: The Simple Economics of Artificial Intelligence* (Harvard Business Review Press, 2018).
81. Agrawal, A., Gans, J. & Goldfarb, A. *Power and Prediction: The Disruptive Economics of Artificial Intelligence* (Harvard Business Review Press, 2022).
82. Goertzel, B. & Pennachin, C. (eds.) *Artificial General Intelligence* (Springer, 2007).
83. Lake, B. M., Ullman, T. D., Tenenbaum, J. B. & Gershman, S. J. Building machines that learn and think like people. *Behav. Brain Sci.* **40**, e253 (2017).
84. Bengio, Y. et al. Meta-learning of parameters for deep networks. *arXiv preprint*arXiv:1901.08981 (2019).
85. Marcus, G. & Davis, E. The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177* (2020).
86. Glazer, E. et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in AI. *arXiv preprint* arXiv:2411.04872 (2024).
87. Zenil, H., Soler-Toscano, F., Delahaye, J.-P. & Gauvrit, N. Two-dimensional Kolmogorov complexity and validation of the coding theorem method by compressibility. *PeerJ Comput. Sci.* **1**, e23 (2015).
88. Ozelim, L. et al. Assembly Theory Collapses to Dictionary Compression and Is Rendered Redundant by Common Statistical Algorithms. *npj Complexity* (2026).
89. Sheth, A., Roy, K. & Gaur, M. Neurosymbolic AI—why, what, and how. *IEEE Intelligent Systems Neurosymbolic Artificial Intelligence* **38**, 56–62 (2023).
90. Zenil, H. & Minary, P. Training-free measures based on algorithmic probability identify high nucleosome occupancy in DNA sequences. *Nucleic Acids Res.* **47**, gkz750 (2019).
91. Zenil, H., Delahaye, J.-P. & Gaucherel, C. Image characterization and classification by physical complexity. *Complexity* **17**, 26–42 (2012).
92. Uthamacumaran, A., Abrahão, F. S., Kiani, N. A. & Zenil, H. On the salient limitations of the methods of assembly theory and their classification of molecular biosignatures. *npj Syst. Biol. Appl.* **10**, 82 (2024).
93. Wang, C.-Y., DaghighFarsoodeh, A. & Pham, H. V. Selection of prompt engineering techniques for code generation through predicting code complexity https://arxiv.org/abs/2409.16416 (2024).

94. Li, J., Li, G., Li, Y. & Jin, Z. Structured chain-of-thought prompting for code generation. *ACM Trans. Softw. Eng. Methodol.* **34**, https://doi.org/10.1145/3690635 (2025).
95. Hutter, M., Quarel, D. & Catt, E. *An Introduction to Universal Artificial Intelligence* (CRC Press, 2024).

Author contributions

H.Z. conceived the test, supervised, reviewed the results, and wrote the non-methodological sections of the manuscript. A.H.-E. and L.O. developed different methods and analysed the data, F.S.A. and H.Z. wrote and developed the theoretical content.

Funding

F.S.A. acknowledges support from the São Paulo Research Foundation (FAPESP), grants 2021/14501-8 and 2023/05593-1.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-026-73289-5.

Correspondence and requests for materials should be addressed to Hector Zenil.

Peer review information *Nature Communications* thanks Yuqing Kong and the other anonymous, reviewer(s) for their contribution to the peer review of this work. A peer review file is available.

[1]Oxford Immune Algorithmics, Oxford University Innovation & London Institute for Healthcare Engineering, Oxford and London, UK. [2]Algorithmic Dynamics Lab, Center of Molecular Medicine, Karolinska Institute & King's College London, London, UK. [3]Centre for Logic, Epistemology and the History of Science, University of Campinas (UNICAMP), Campinas, Brazil. [4]DEXL, National Laboratory for Scientific Computing (LNCC), Petrópolis, Brazil. [5]Algorithmic Dynamics Lab, Department of Biomedical Computing, School of Biomedical Engineering and Imaging Sciences, London, UK. [6]King's Institute for Artificial Intelligence, King's College London, London, UK. ✉e-mail: hector.zenil@kcl.ac.uk