Incompleteness and Artificial Intelligence

Shane Legg

IDSIA — Switzerland

shane@idsia.ch

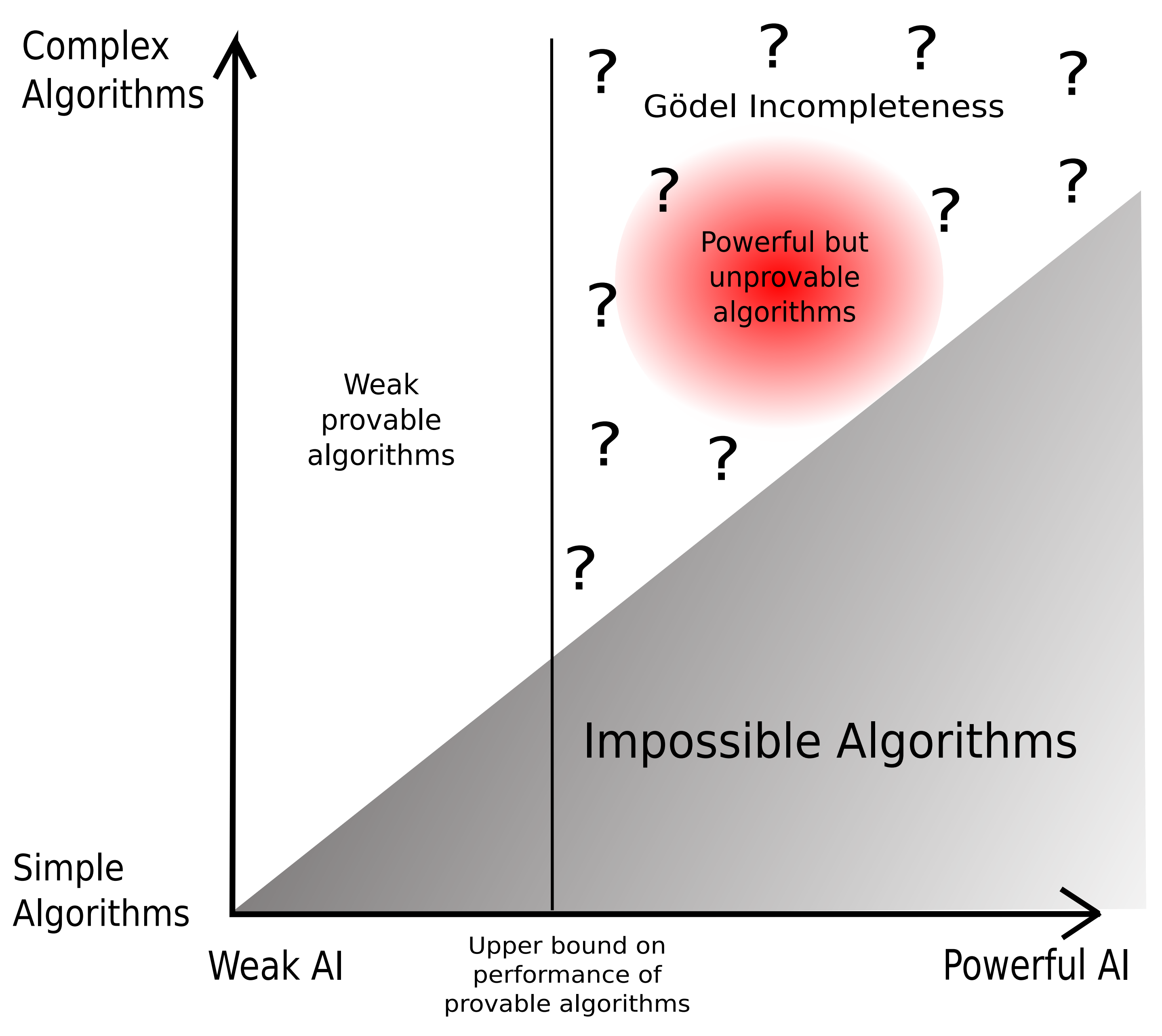
Predictor Complexity vs. Predictor Power

Fundamentally new results on the relationship between complexity, artificial intelligence and Gödel incompleteness are proven.

The first result shows that the ability of an algorithm to learn to predict computable sequences is limited by the complexity of the prediction algorithm. In other words, only complex predictors can predict truly difficult sequences, simple but very powerful predictors are impossible. Thus, there is no elegant and extremely powerful computable theory of prediction. This is illustrated on the diagram by the greyed out region of impossible algorithms on the lower right.

The second result is that beyond some point powerful predictors become so complex that it is no longer possible to prove that any one of these predictors is indeed powerful. In other words, for any powerful predictor p (this set is not empty) the statement "p is a powerful predictor" is true but unprovable. This is represented by the upper right region on the diagram. It proves that the most powerful prediction algorithms cannot be mathematically discovered due to the problem of Gödel incompleteness.

The primary goal of artificial intelligence research is to discover intelligent algorithms, and fundamental to intelligent systems is their ability to make predictions about the future. For example, an intelligent system needs to be able to predict whether or not taking some action is likely to lead to success with respect to a goal. Thus, the problems of high complexity and Gödel incompleteness with predictors carries over into intelligent systems in general: Not only are very powerful artificial intelligences highly complex, they are also profoundly unknowable.



Technical Definitions

Let $\mathbb{B} := \{0,1\}$, let $\mathbb{B}^*$ be the set of (finite) binary *strings*, and let $\mathbb{B}^\infty$ be the set of (infinite) binary *sequences*. A *substring* of x is defined $x_{j:k} := x_j x_{j+1} \dots x_k$ where $1 \le j \le k \le n$. By $|x|$ we mean the length of the string x. We represent a monotone universal Turing machine by $\mathcal{U}$.

A sequence $\omega \in \mathbb{B}^\infty$ is a *computable sequence* if $\exists q \in \mathbb{B}^* : \mathcal{U}(q) = \omega$. We denote this set of sequences $\mathcal{C}$.

A *computable predictor* is an algorithm $p \in \mathbb{B}^*$ that on a universal Turing machine $\mathcal{U}$ computes a total function $\mathbb{B}^* \to \mathbb{B}$. We write $p(x)$ to mean the function computed by the program p when executed on $\mathcal{U}$ along with the input string x.

Having $x_{1:n}$ as input, the objective of a predictor is for its output, called its *prediction*, to match the next symbol in the sequence, that is, $p(x_{1:n}) = x_{n+1}$.

We say that a predictor p can *learn to predict* a sequence $\omega := x_1 x_2 \ldots \in \mathbb{B}^\infty$ if $\exists m \in \mathbb{N}, \forall n \ge m : p(x_{1:n}) = x_{n+1}$.

Let $\mathcal{P}(\omega)$ be the set of all predictors able to learn to predict ω. Similarly for sets of all sequences $\mathcal{S} \subset \mathbb{B}^\infty$, define $\mathcal{P}(\mathcal{S}) := \bigcap_{\omega \in \mathcal{S}} \mathcal{P}(\omega)$.

The standard measure of complexity for a sequence ω is its *Kolmogorov complexity* defined by

$$K(\omega) := \min_{q \in \mathbb{B}^*} \{|q| : \mathcal{U}(q) = \omega\}.$$

If no such q exists, we define $K(\omega) := \infty$. For the Kolmogorov complexity of a string $x \in \mathbb{B}^n$ we require that $\mathcal{U}(q)$ halts after generating x.

For $n \in \mathbb{N}$, let $\mathcal{C}_n := \{\omega \in \mathcal{C} : K(\omega) \le n\}$. Further, let $\mathcal{P}_n := \mathcal{P}(\mathcal{C}_n)$ be the set of predictors able to learn to predict all sequences in $\mathcal{C}_n$. The larger n is, the more powerful the predictors in the set $\mathcal{P}_n$ are.

As most results hold within an independent constant we define $f(x) \overset{+}{<} g(x)$ to mean that $\exists c \in \mathbb{R}, \forall x : f(x) < g(x) + c$.

Complexity of Predictors

Lemma 1. $\forall n \in \mathbb{N}, \exists p \in \mathcal{P}_n : K(p) \overset{+}{<} n + O(\log n)$.

In words: Prediction algorithms exist that can learn to predict all computable sequences up to a given complexity, and these predictors need not be significantly more complex than the sequences they can predict.

Proof idea: Define a prediction algorithm p that knows that h algorithms up to a length of n generate infinite sequences. When asked to predict the next bit of input string $x \in \mathbb{B}^*$ of length l, the predictor simulates all programs up to length n waiting for h of these to output $l+1$ symbols. When this computation has finished, the predictor simply predicts by using the generated sequence that is most similar to x.

As $l \to \infty$ this prediction algorithm must converge to always making correct predictions whenever the program generating the string x has length less than or equal to n. This is because in this case the true sequence to be predicted is in the set of sequences that the predictor is simulating and thus it will be eventually identified and used for prediction. Furthermore, we see that algorithm p contains some fixed length code plus an encoding of $h \le 2^n$ which requires $n + O(\log n)$ bits to encode.

Theorem 1. $\forall n \in \mathbb{N} : p \in \mathcal{P}_n \Rightarrow K(p) \overset{+}{>} n$.

In words: Powerful and very general prediction algorithms are necessarily complex. Simple but truly powerful prediction algorithms are impossible.

Proof idea: Take a simple prediction algorithm p and convert it into a sequence generation algorithm q that always outputs the opposite to what p would predict at each step.

By construction, p can never learn to predict the sequence generated by q. Furthermore, as q is only a slightly modified version of p, these two algorithms have about the same Kolmogorov complexity. Thus, for every simple predictor there exists a simple computable sequence which it can never learn to predict. As this holds for all computable predictors, the result immediately follows.

Incompleteness of Predictors

Theorem 2. *In any consistent formal axiomatic system $\mathcal{F}$ that is sufficiently rich to express statements of the form "$p \in \mathcal{P}_n$", there exists $m \in \mathbb{N}$ such that for all $n > m$ and for all predictors $p \in \mathcal{P}_n$ the true statement "$p \in \mathcal{P}_n$" cannot be proven in $\mathcal{F}$.*

In words: Although there exist prediction algorithms that can learn to predict all sequences up to any given complexity, it is impossible to discover these powerful algorithms using mathematical proof due to Gödel incompleteness.

Proof idea: The proof has a similar structure to Chaitin's information theoretic proof of Gödel incompleteness. We show that if a simple proof search algorithm could find a proof that some algorithm was powerful, then this prediction algorithm must have low Kolmogorov complexity, contradicting Theorem 1. Therefore, such a proof cannot exist.

Proof: For each $n \in \mathbb{N}$ let T_n be the set of statements expressed in the formal system $\mathcal{F}$ of the form "$p \in \mathcal{P}_n$", where p is filled in with the complete description of some algorithm in each case. As the set of programs is denumerable, T_n is also denumerable and each element of T_n has finite length. From Lemma 1 and Theorem 1 it follows that each T_n contains infinitely many statements of the form "$p \in \mathcal{P}_n$" which are true.

Fix n and create a search algorithm s that enumerates all proofs in the formal system $\mathcal{F}$ searching for a proof of a statement in the set T_n. As the set T_n is recursive, s can always recognise a proof of a statement in T_n. If s finds any such proof, it outputs the corresponding program p and then halts.

By way of contradiction, assume that s halts, that is, a proof of a theorem in T_n is found and p such that $p \in \mathcal{P}_n$ is generated as output. The size of the algorithm s is a constant (a description of the formal system $\mathcal{F}$ and some proof enumeration code) as well as an $O(\log n)$ term needed to describe n. It follows then that $K(p) \overset{+}{<} O(\log n)$.

However from Theorem 1 we know that $K(p) \overset{+}{>} n$. Thus, for sufficiently large n, we have a contradiction and so our assumption of the existence of a proof must be false. That is, for sufficiently large n and for all $p \in \mathcal{P}_n$, the true statement "$p \in \mathcal{P}_n$" cannot be proven within the formal system $\mathcal{F}$.