Logical Depth and Other Algorithmically Defined Properties of Finite Objects

C. H. Bennett
IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

The input/output relation of a universal computer, which by Church's thesis provides a microcosm of all deductive logic and all physical processes that might be described by such logic, can be used to define intrinsic properties of digitally represented physical objects. An object's intrinsic information content or arbitrariness may identified with its algorithmic entropy, *the negative logarithm of the object's probability of being produced as the output of a computation with random input. An object's intrinsic causal nontriviality or* logical depth *may be identified with number of steps in a typical one of the computation paths leading to the object. Although not effectively computable or efficiently approximable in practice, these properties are asymptotically roughly machine independent (and thus intrinsic), owing to the ability of any two efficiently universal computers to simulate one another at the cost of an additive increase in program size and a polynomial expansion of execution time.*

Because of the ability of universal computers to simulate one another and compute any compuable function, the input/output relation of a standard universal computer can be regarded as a microcosm of all of deductive logic, as well as all physical processes describable by such logic.

As suggested in the figure, the input/output relation of a universal computer can be viewed as a random computation, in which the proverbial monkey, instead of randomly striking the keys of a typewriter in hopes of producing meaningful literature, randomly strikes the keys of the computer in hopes of causing it to do a meanigful computation.

Experience suggests that under these circumstances the computer is most likely to embark on a trivial, quickly-terminating computation, perhaps issuing an error message, or else on a trivial nonterminating one, in which the computer loops endlessly and ignores all further input. However, every meaningful computation also has some chance of accidentally being programmed by the monkey and performed by the computer, eg computing the digits of pi, the infrared spectrum of toluene, or determining which side, if any, has a winning strategy in chess.

A computer U is universal if for any other computer V, there is some sequence of initial monkey bits that will cause U to simulate V on subsequent input. In other words, the computation tree of U contains an internal node, (marked "simulate V" in the figure), the subtree below which is isomorphic to the entire computation tree of V. If U is efficiently universal, as many simple Turing machines are, then this isomorphism in node structure will extend, within a polynomial, to the times of computation. Therefore, the probability that the monkey will perform a given computation on the U machine is at least a constant fraction of its probability of doing so on the V machine, and the time required to do so on the U machine is at most polynomially greater than the time required to perform the computation on the V machine. It is perhaps worth noting that because U can simulate any computer it can also simulate itself. Therefore the computation tree of U will be self-similar, containing infinitely many subtrees isomporphic to the entire tree.

The approximately invariant structure of the universal computation tree can be used to define correspondingly robust properties of finite objects such as long binary strings x, and by extension, of physical objects such as genomes that can be conveniently represented by such strings. These properties are approximately invariant with respect to changing from one universal machine to another, or applying computationally trivial mappings to the strings themselves (ie concisely programmable 1:1 mappings that are fast to compute and have fast inverses).

The simplest algorithmically defined string property [4][2][3] is the output probability of x, termed the

0-8186-3420-0/93 $03.00 © 1993 IEEE

universal semimeasure, a-priori probability, or algorithmic probability of x. It is machine-independent up to a multiplicative constant, and its base two logarithm, the algorithmic entropy $H(x)$, is equal within an additive constant to the size of the smallest monkey-program to compute x. Algorithmic entropy corresponds intuitively to a string's absolute information content or arbitrariness[4][2], and an "algorithmically random" string may be defined as one of near-maximal algorithmic entropy. Such strings are also termed "algorithmically incompressible", since they lack any algorithmic description significantly more concise than the string itself.

Another property, logical depth[1], measures the typical duration of computations leading to x. Typical duration cannot be taken simply to mean expected duration, since that is infinite for any x, but must be defined more carefully. Various slightly different definitions are possible[1]; one is to say that a string x is d-deep at confidence level 2^{-k} if computations running in time $< t$ contribute a less than a fraction 2^{-k} of the algorithmic probability of x. This formalizes the notion that it is more implausible for x to have arisen by a computation of fewer than t steps than for a tossed coin to yield k consecutive heads. A deep string thus contains internal evidence that a large amount of mathematical work was probably done to create it. If the string represents a physical object, and if one believes that physical processes in general can be efficiently simulated by digital computers, then this implies that the physical object also probably had a nontrivial causal history.

Frequently one wishes to consider how much additional information, or how much additional computational work, must be supplied to compute an object y if another object x is already present. These notions can be formalized as relative algorithmic entropy of y given x —the negative logarithm of the probability that a monkey will program U to transform x, supplied as an auxiliary input to the computer, into y— and relative depth—the time required for a significant fraction of these computations to finish.

Other properties referring to time of computation, but distinct from logical depth, may be defined in terms of the relation of a string x to the universal computation tree. Treating x as an input rather than an output, x might be called "ambitious" if it acts as a program for a long computation without giving evidence that the computation need actually to have been performed to generate x. Ambition is not a very robust property, since small changes in a program can greatly alter its run time. Another property distinct from depth is "crypticity"[1], the time required to infer a plausible origin for a thing given a knowledge of the thing itself. Formally, an object x can be called cryptic if every incompressible program for computing x is deep relative to x. Cryptograms are typically cryptic, at least in the common circumstance that they contain sufficient information to infer a unique plausible plaintext, while making it computationally infeasible to do so. Cryptic objects need not be deep, however, since good cryptograms need not take a long time to generate, only to analyze. Neither need deep objects be cryptic

All these properties are so closely connected with the halting problem as to be uncomputable in principle and not efficiently approximable in practice. Their main usefulness is therefore in proving theorems concerning circumstances under which objects having the properties will and will not be generated by computational or dynamical processes.

References

[1] C.H. Bennett, "Logical Depth and Physical Complexity" in *The Universal Turing Machine– a Half-Century Survey*, Rolf Herken, ed., Oxford University Press, 227-257, (1988).

[2] G. Chaitin, "A Theory of Program Size Formally Identical to Information Theory", *J. Assoc. Comput. Mach.* **22**, 329-340 (1975).

[3] L. Levin, "Randomness Conservation Inequalities; Information and Independence in Mathematical Theories," *Inform. and Control* **61**, 15-37 (1984).

[4] A.K. Zvonkin and L.A. Levin, "The Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by Means of the Theory of Algorithms", *Russ. Math. Surv.* **256** 83-124 (1970).

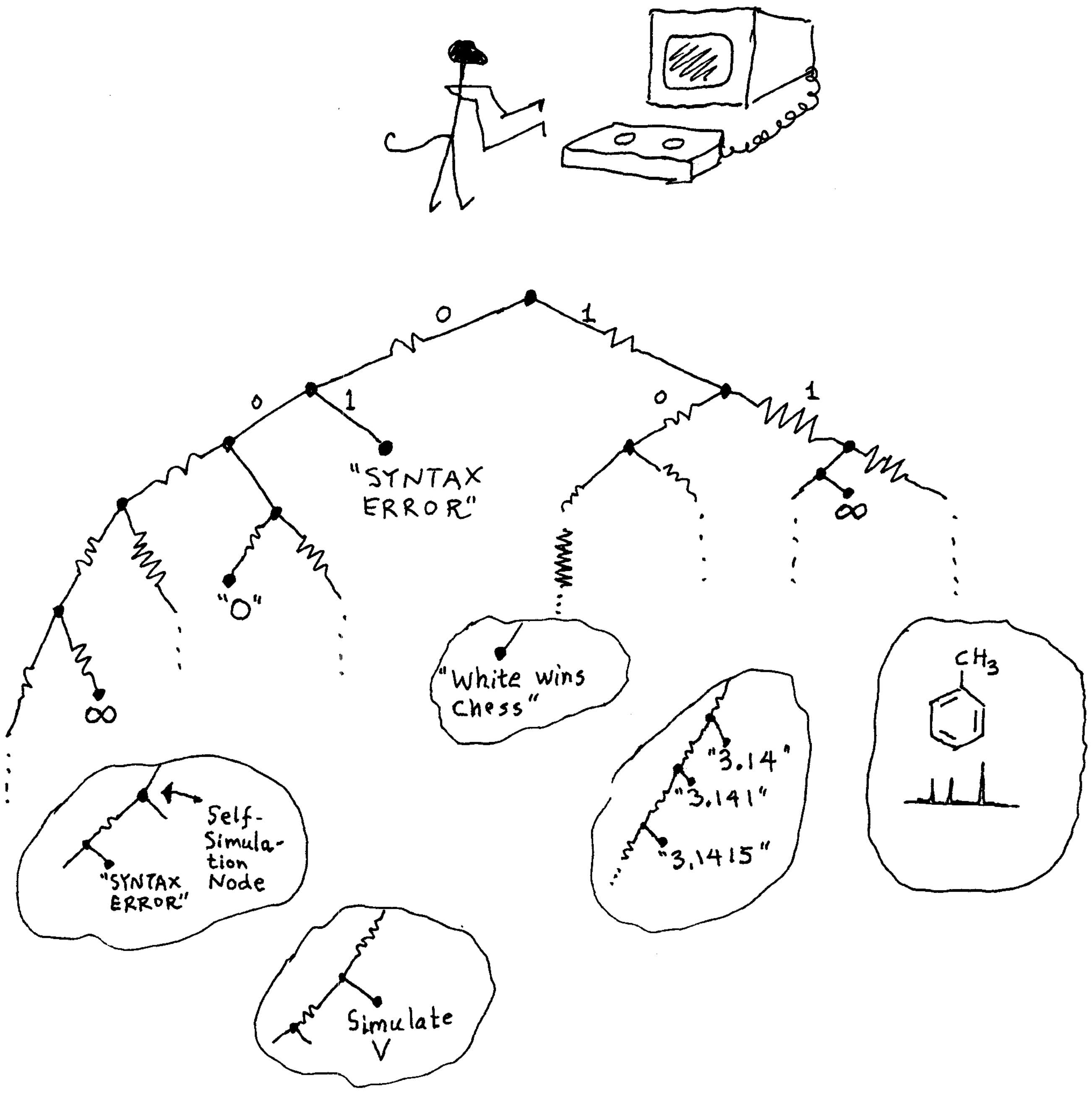

Figure Caption: Schematic computation tree of an efficiently universal computer U with random input. The computer starts in a standard initial state represented by the root of the tree (at top of figure). Whenever the computation requests an input bit, a random value is supplied, with the left branch representing a 0 and the right a 1. This might be done by having the computer unlock its keyboard until a monkey, typing at random, hits a 0 or 1, then lock it again until the next input bit is needed. Along each branch the computation proceeds until it halts (leaf of tree), requests another input bit (internal node), or embarks on a nonhalting computation that requests no further input (leaf marked ∞). The number of zig-zags in the path between a node and its parent schematically represents the duration of that leg of the computation.