International Journal of Theoretical Physics, Vol. 36, No. 1, 1997

Algorithmic Information and Simplicity in Statistical Physics

Rüdiger Schack[1,2]

Received April 2, 1996

Applications of algorithmic information theory to statistical physics rely (a) on the fact that average conditional algorithmic information can be approximated by Shannon information and (b) on the existence of *simple states* described by short programs. More precisely, given a list of N states with probabilities $0 < p_1 \leq \cdots \leq p_N$, the average conditional algorithmic information $\bar{I}$ to specify one of these states obeys the inequality $H \leq \bar{I} < H + O(1)$, where $H = -\Sigma\, p_j \log_2 p_j$ and $O(1)$ is a computer-dependent constant. We show how any universal computer can be slightly modified in such a way that (a) the inequality becomes $H \leq \bar{I} < H + 1$ and (b) states that are simple with respect to the original computer remain simple with respect to the modified computer, thereby eliminating the computer-dependent constant from statistical physics.

1. INTRODUCTION

Algorithmic information theory (Solomonoff, 1964; Kolmogoroff, 1965; Zvonkin and Levin, 1970; Chaitin, 1987), in combination with Landauer's principle (Landauer, 1961, 1988), which specifies the unavoidable energy cost $k_B T \ln 2$ for the erasure of a bit of information in the presence of a heat reservoir at temperature T, has been applied successfully to a range of problems: the Maxwell demon paradox (Bennett, 1982), a consistent Bayesian approach to statistical mechanics (Zurek, 1989a,b; Caves, 1993a,b), a treatment of irreversibility in classical Hamiltonian chaotic systems (Caves, 1993b; Schack and Caves, 1992), and a characterization of quantum chaos relevant to statistical physics (Caves, 1993b; Schack and Caves, 1993, 1996a,b; Schack *et al.*, 1994). The algorithmic information for a physical state is defined as

[1] Center for Advanced Studies, Department of Physics and Astronomy, University of New Mexico, Albuquerque, New Mexico 87131-1156.

[2] Present address: Department of Mathematics, Royal Holloway, University of London, Egham, Surrey TW20 0EX, U.K.; e-mail: r.schack@rhbnc.ac.uk.

0020-7748/97/0100-0209$12.50/0 © 1997 Plenum Publishing Corporation

the length in bits of the shortest self-delimiting program for a universal computer that generates a description of that state (Zurek, 1989b; Caves, 1990). Algorithmic information with respect to two different universal computers differs at most by a computer-dependent constant (Chaitin, 1987). Although typically the latter can be neglected in the context of statistical physics, the presence of an arbitrary constant in a physical theory is unsatisfactory and has led to criticism (Denker and leCun, 1993). In the present paper, we show how the computer-dependent constant can be eliminated from statistical physics.

In the following paragraphs we give a simplified account of the role of algorithmic information in classical statistical physics. A more complete exposition including the quantum case can be found in Zurek (1989b) and Caves (1993b). We adopt here the information-theoretic approach to statistical physics pioneered by Jaynes (1983). In this approach, the *state* of a system represents the observer's knowledge of the way the system was prepared. States are described by probability densities in phase space; observers with different knowledge assign different states to the system. Entropy measures the information missing toward a complete specification of the system.

Consider a set of N states ($N \geq 2$) labeled by $j = 1, \ldots, N$, all having the same energy and entropy. The restriction to states of the same energy and entropy is not essential, but it simplifies the notation. Initially the system is assumed to be in a state in which state j is occupied with probability $p_j > 0$. We assume throughout that the states j are labeled such that $0 < p_1 \leq \cdots \leq p_N$. If an observation reveals that the system is in state j, the increased knowledge is reflected in an entropy decrease $\Delta S = -k_B \ln 2\, H$, where $H = -\Sigma\, p_j \log_2 p_j > 0$ is the original missing information measured in bits. To make the connection with thermodynamics, we assume that there is a heat reservoir at temperature T to which all energy in the form of heat must eventually be transferred, possibly using intermediate steps such as storage at some lower temperature. In the presence of this fiducial heat reservoir, the entropy decrease ΔS corresponds to a free energy increase $\Delta F = -T\Delta S = +k_B T \ln 2\, H$. Each bit of missing information decreases the free energy by the amount $k_B T \ln 2$; if information is acquired about the system, free energy increases.

The fact that entropy can decrease through observation—which underlies most proposals for a Maxwell demon—does not conflict with the second law of thermodynamics, because the observer's physical state changes as a consequence of the interaction with the system. Szilard (1929) discovered that no matter how complicated the change in the observer's physical state, the associated irreducible thermodynamic cost can be described solely in terms of information. He found that in the presence of a heat reservoir at temperature T each bit of information acquired by the observer has an energy cost at least as big as $k_BT \ln 2$. Total available work is reduced not only by missing information, but also by information the observer has acquired about the system. The physical nature of the cost of information was clarified by Bennett (1982), who applied Landauer's principle (Landauer, 1961, 1988) to the Maxwell demon problem and showed that the energy cost has to be paid when information is erased.

To keep the Landauer erasure cost of the observational record as low as possible, the information should be stored in maximally compressed form. The concept of a maximally compressed record is formalized in algorithmic information theory (Chaitin, 1987). Bennett (1982) and Zurek (1989a,b) gave Szilard's theory its present form by using algorithmic information to quantify the amount of information in an observational record. In particular, by exploiting Bennett's idea of a reversible computer (Bennett, 1982), Zurek (1989a) showed how an observational record can be replaced by a compressed form at no thermodynamic cost. This means that the energy cost of the observational record can be reduced to the Landauer erasure cost of the compressed form.

Let us denote by s_j a binary string describing the jth state ($j = 1, \ldots, N$). A detailed discussion of how a description of a physical state can be encoded in a binary string is given in Zurek (1989b). The exact form of the strings s_j is of no importance for the theory outlined here, however, because the information needed to generate a list of *all* the strings s_j can be treated as *background information* (Caves, 1990, 1993b). Background information is the information needed to generate a list $s = ((s_1, p_1), \ldots, (s_N, p_N))$ of all N states together with their probabilities; i.e., background information is the information the observer has before the observation. This formulation assumes that the probabilities p_j are completely specified by the background information—a natural assumption in the Bayesian approach to probabilities (Jaynes, 1983) adopted in this paper. A generalization to approximately specified probabilities is discussed in Bennett (1982).

Algorithmic information is defined with respect to a specific universal computer U. We denote by $I_U(s_j \mid s)$ the conditional algorithmic information, with respect to the universal computer U, to specify the jth state, given the background information (Chaitin, 1987; Zurek, 1989b; Caves, 1990). More precisely, $I_U(s_j \mid s)$ is the length in bits of the shortest self-delimiting program for U that generates the string s_j, given a minimal self-delimiting program to generate s. For a formal definition of a universal computer U and of $I_U(s_j \mid s)$ see Section 2. It should be emphasized that a minimal program that generates the list s of descriptions of all states and their probabilities can be short even when a minimal program that generates the description s_j of a typical single state is very long (Zurek, 1989b).

Since total available work is reduced by $k_B T \ln 2$ by each bit of information the observer acquires about the system as well as by each bit of missing information, the change in *total free energy* or *available work* upon observing state j can now be written as

$$\begin{aligned} \Delta F_{j,\mathrm{tot}} &= -T[\Delta S + k_B \ln 2\, I_U(s_j \mid s)] \\ &= -k_B T \ln 2\, [-H + I_U(s_j \mid s)] \end{aligned} \tag{1}$$

This definition of total free energy is closely related to Zurek's definition of physical entropy (Zurek, 1989b). Average conditional algorithmic information $\overline{I_U(\cdot \mid s)} = \sum p_j I_U(s_j \mid s)$ obeys the double inequality (Zurek, 1989b; Caves, 1990)

$$H \leq \overline{I_U(\cdot \mid s)} < H + O(1) \tag{2}$$

where $O(1)$ denotes a positive computer-dependent constant (Chaitin, 1987). It follows immediately that the *average* change in total free energy, $\Delta F_{\mathrm{tot}} = \sum p_j \Delta F_{j,\mathrm{tot}}$, is zero or negative:

$$0 \geq \Delta F_{\mathrm{tot}} > -O(1) k_B T \ln 2 \tag{3}$$

The left side of this double inequality establishes that acquiring information cannot increase available work on the average. For standard choices for the universal computer U, e.g., a Turing machine or Chaitin's LISP-based universal computer (Chaitin, 1987), the computer-dependent $O(1)$ constant on the right is completely negligible in comparison with thermodynamic entropies. Condition (3) therefore expresses that on the average, with respect to a standard universal computer, total free energy remains essentially unchanged upon observation. Despite the success of this theory, the presence of an arbitrary constant is disturbing. To understand the issues involved in removing the arbitrary constant, we must introduce the notions of simple and complex states.

Although the average information $\overline{I_U(\cdot \mid s)}$ is greater than or equal to H, there is a class of low-entropy states that can be prepared without gathering a large amount of information. For example, in order to compress a gas into a fraction of its original volume, free energy has to be spent, but the length in bits of written instructions to prepare the compressed state is negligible on the scale of thermodynamic entropies. States that can be prepared reliably in a laboratory experiment usually are *simple states*, which means that there is a short verbal description of how to prepare such a state.

The concept of a simple state is formalized in algorithmic information theory. A simple state is defined as a state for which $I_U(s_j \mid s) << H$; i.e., descriptions for simple states can be generated by short programs. The total free energy increases, in the sense of equation (1), upon observing the system to be in a simple state. Simplicity is a computer-dependent concept. Standard universal computers like Turing machines reflect our intuitive notion of simplicity. It is easy, however, to define a universal computer for which there are no short programs at all; such a computer would not recognize simplicity.

Intuitively, simplicity ought to be an intrinsic property of a state. A computer formalizing the intuitive concept of simplicity should reflect this. In particular, for such a computer a simple state should have a short program independent of the probability distribution $p_1, \ldots, p_N$. This is not true for all universal computers. In Section 2 we introduce a universal computer U_ϵ for with $I_{U_\epsilon}(s_j | s)$ is determined solely by the probabilities $p_1, \ldots, p_N$. For this computer, a short program for the jth state reflects a large probability p_j, not an intrinsic property of the state. We will say that such a computer does not recognize intrinsically simple states.

Simple states are rare—there are fewer than 2^n states j for which $I_U(s_j | s) < n$ (Chaitin, 1987)—and thus arise rarely as the result of an observation, yet they are of great importance. Simple states are states for which the algorithmic contribution to total free energy is negligible. The concept of total free energy does not conflict with conventional thermodynamics because thermodynamic states are simple. If the theory does not have the notion of simple states, the connection with conventional thermodynamics is lost.

The opposite of a simple state, a *complex state*, is defined as a state for which $I_U(s_j | s)$ is of the same order as H. Complex states arise not just through Maxwell demon-like observations. We have shown (Caves, 1993b; Schack and Caves, 1992, 1993, 1996a,b; Schack *et al.*, 1994) that initially simple states of chaotic Hamiltonian systems in the presence of a perturbing environment rapidly evolve into extremely complex states (Caves, 1993a,b) for which the negative algorithmic contribution to total free energy is vastly bigger than H and thus totally dominates conventional free energy. In addition to giving insight into the second law of thermodynamics, this result leads to a new approach to quantum chaos (Caves, 1993b; Schack and Caves, 1993, 1996a; Schack *et al.*, 1994).

In this paper, we show how the computer-dependent $O(1)$ constant can be eliminated from the theory summarized above. In Section 2 we construct an optimal universal computer for which the $O(1)$ constant is minimal. It turns out, however, that optimal universal computers do not recognize intrinsically simple states and thus are unsatisfactory in formulating the theory. This difficulty is solved in Section 3, where we show that any universal computer U can be modified in a simple way such that (a) any state that is simple with respect to U is also simple with respect to the modified universal computer U_3 and (b) average conditional information with respect to U_3 exceeds average conditional information with respect to an optimal universal computer by at most 0.5 bit. Moreover, conditional algorithmic information with respect to the modified computer U_3 obeys the inequality $H \le \overline{I_{U_3}(\cdot | s)} < H + 1$. This double bound is the tightest possible in the sense that there is no tighter bound that is independent of the probabilities p_j.

2. AN OPTIMAL UNIVERSAL COMPUTER

The idea of an optimal universal computer is motivated by Zurek's discussion (Zurek, 1989b) of *Huffman coding* (Huffman, 1952) as an alternative way to quantify the information in an observational record. We consider only binary codes, for which the code words are binary strings. Before reviewing Huffman coding, we need to formalize the concept of a list consisting of descriptions of N states together with their probabilities.

Definition 1. A *list of states* s is a string of the form $s = ((s_1, p_1), \ldots, (s_N, p_N))$, where $N \ge 2$, $0 < p_1 \le \cdots \le p_N$, $\Sigma\, p_j = 1$, and s_j is a binary string ($j = 1, \ldots, N$). More precisely, the list of states s is the binary string obtained from the list $((s_1, p_1), \ldots, (s_N, p_N))$ by some definite translation scheme. One possible translation scheme is to represent parentheses, commas, and numbers (i.e., the probabilities p_j) in ascii code, and to precede each binary string s_j by a number giving its length $|s_j|$ in bits. The entropy of a list of states is $H(s) = -\Sigma\, p_j \log_2 p_j$. Throughout this paper, $|t|$ denotes the length of the binary string t.

The Huffman code for a list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$ is a prefix-free or instantaneous code (Welsh, 1988)—i.e., no code word is a prefix of any other code word—and can, like all prefix-free codes, be represented by a binary tree as shown in Fig. 1. The number of links leading from the root

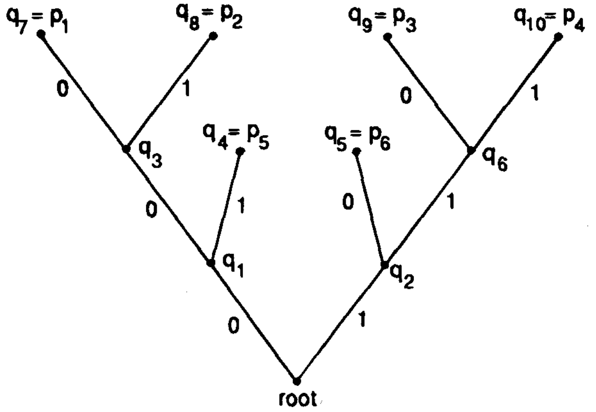

Fig. 1. Binary tree representing the Huffman code for six states with probabilities $p_1, \ldots, p_6$. The node probabilities q_k are defined recursively, i.e., $q_7 = p_1$, $q_8 = p_2$, $q_3 = q_7 + q_8$, etc. Code words correspond to branch labels; e.g., the code word for the third state (probability p_3) is 110.

of the tree to a node is called the *level* of that node. If the level-n node a is connected to the level-$(n + 1)$ nodes b and c, then a is called the *parent* of b and c; a's *children* b and c are called *siblings*. There are exactly N terminal nodes or *leaves*, each leaf corresponding to a state j. Each link connecting two nodes is labeled 0 or 1. The sequence of labels encountered on the path from the root to a leaf is the code word assigned to the corresponding state. The code-word length of a state is thus equal to the level of the corresponding leaf. Each node is assigned a probability q_k such that the probability of a leaf is equal to the probability p_j of the corresponding state and the probability of each nonterminal node is equal to the sum of the probabilities of its children.

A binary tree represents a Huffman code if and only if it has the *sibling property* (Gallager, 1978), i.e., if and only if each node except the root has a sibling, and the nodes can be listed in order of nonincreasing probability with each node being adjacent to its sibling in the list. The tree corresponding to a Huffman code and thus the Huffman code itself can be built recursively. Create a list of N nodes corresponding to the N states. These N nodes will be the leaves of the tree that will now be constructed. Repeat the following procedure until the tree is complete: Take two nodes with smallest probabilities and make them siblings by generating a node that is their common parent; replace in the list the two nodes by their parent; label the two links branching from the new parent node by 0 and 1.

The procedure outlined above does not define a unique Huffman code for the list of states s, nor does it give generally a unique set of code-word lengths. In the following, we will assume that we are given some definite algorithm to assign a Huffman code where the freedom in the coding procedure is used to assign to the first state (the one with smallest probability) a code word of maximum length consisting only of zeros.

Definition 2. Given a list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$, the binary string $c_j(s)$ with length $l_j(s) \equiv |c_j(s)|$ denotes the Huffman code word assigned to the jth state using a definite algorithm with the property that $c_1(s) = 0 \cdots 0$ and $l_j(s) \le l_1(s)$ for $j = 2, \ldots, N$. We denote the average Huffman code-word length by $\bar{l}(s) = \Sigma\, p_j l_j(s)$. The *redundancy* $r(s)$ of the Huffman code is defined by $r(s) = \bar{l}(s) - H(s)$.

The redundancy $r(s)$ obeys the bounds $0 \le r(s) < 1$, corresponding to bounds

$$H(s) \le \bar{l}(s) < H(s) + 1 \qquad (4)$$

for the average code-word length. Huffman coding is optimal in the sense that there is no prefix-free binary code with an average code-word length less than $\bar{l}(s)$. There can be, however, optimal prefix-free codes that are not Huffman codes.

The length $l_j(s)$ of the Huffman code word $c_j(s)$ cannot be determined from the probability p_j alone, but depends on the entire set of probabilities $p_1, \ldots, p_N$. The tightest general bounds for $l_j(s)$ are (Katona and Nemetz, 1976)

$$1 \leq l_j(s) < -\log_g p_j + 1 \tag{5}$$

where $g = (\sqrt{5} + 1)/2$ is the golden mean. The code-word length for some states j thus can differ widely from the value $-\log_2 p_j$. For most states j, however, the Huffman code-word length is $l_j(s) \simeq -\log_2 p_j$. The following theorem (Schack, 1994) is a precise version of this statement.

Theorem 1. (a) $P_m^- = \Sigma_{j \in I_m^-} p_j < 2^{-m}$, where $I_m^- = \{i \mid l_i(s) < -\log_2 p_i - m\}$, i.e., the probability that a state with probability p has Huffman code-word length smaller than $-\log_2 p - m$ is less than 2^{-m}. (This is true for any prefix-free code.) (b) $P_m^+ = \Sigma_{j \in I_m^+} p_j < 2^{-c(m-2)+2}$, where $I_m^+ = \{i \mid l_i(s) > -\log_2 p_i + m\}$ and $c = (1 - \log_2 g)^{-1} - 1 \simeq 2.27$, i.e., the probability that a state with probability p has Huffman code-word length greater than $-\log_2 p + m$ is less than $2^{-c(m-2)+2}$.

Proof. See Schack (1994). ■

The probability of encountering a state j with a Huffman code word much longer than $-\log_2 p_j$ is therefore exponentially small. There are alternative coding schemes that avoid these long code words altogether at the cost of slightly increasing the average code-word length; one such scheme, Shannon–Fano coding, is discussed in Zurek (1989b). In the present paper, we use Huffman coding for specificity.

Suppose that one characterizes the information content of a state j by its Huffman code-word length $l_j(s)$. Then in condition (2) average algorithmic information $\overline{I_U(\cdot \mid s)}$ is replaced by average code-word length $\bar{l}(s)$, the $O(1)$ constant is replaced by 1, and condition (3) assumes the concise form $0 \geq \Delta F_{\text{tot}} > -k_B T \ln 2$. This way of eliminating the $O(1)$ constant, however, has a high price. Since Huffman code-word lengths depend solely on the probabilities $p_1, \ldots, p_N$—states with high probability are assigned shorter code words than states with low probability—Huffman coding does not recognize intrinsically simple states. This means that one of the most appealing features of the theory is lost, namely that the Landauer erasure cost associated with states that can be prepared in a laboratory is negligible.

In the present paper we show that it is possible to retain this feature of the theory, yet still eliminate the computer-dependent constant. We first attempt to do this by constructing an optimal universal computer, i.e., a universal computer for which the $O(1)$ constant in condition (2) is minimal. We find, however, that optimal universal computers do not recognize intrinsically simple states either. A solution to this problem will be given in Section 3, where we discuss a class of nearly optimal universal computers.

We will need precise definitions of a computer and a universal computer, which we quote from Chapter 6.2 in Chaitin (1987).

Definition 3. A *computer* C is a computable partial function that carries a program string p and a free data string q into an output string $C(p, q)$ with the property that for each q the domain of $C(., q)$ is a prefix-free set; i.e., if $C(p, q)$ is defined and p is a proper prefix of p', then $C(p', q)$ is not defined. In other words, programs must be self-delimiting. U is a *universal computer* if and only if for each computer C there is a constant sim(C) with the following property: if $C(p, q)$ is defined, then there is a p' such that $U(p', q) = C(p, q)$ and $|p'| \leq |p| + \mathrm{sim}(C)$.

In this definition, all strings are binary strings, and $|p|$ denotes the length of the string p as before. The self-delimiting or prefix-free property entails that for each free data string q, the set of all valid program strings can be represented by a binary tree.

For any binary string t we denote by $t^*(U)$ (or just t^* if no confusion is possible) the shortest string for which $U(t^*, \Lambda) = t$, where Λ is the empty string; i.e., t^* is the shortest program for the universal computer U to calculate t. If there are several such programs, we pick the one that is first in lexicographic order. This allows us to define conditional algorithmic information.

Definition 4. The *conditional algorithmic information* $I_U(t_1 | t_2)$ to specify the binary string t_1, given the binary string t_2, is

$$I_U(t_1 | t_2) = \min_{p \,|\, U(p, t_2^*) = t_1} |p| \tag{6}$$

In words, $I_U(t_1 | t_2)$ is the length of a shortest program for U that computes t_1 in the presence of the free data string t_2^*. In particular, the conditional algorithmic information $I_U(s_j | s)$ to specify the jth state, given a list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$, is

$$I_U(s_j | s) = \min_{p \,|\, U(p, s^*) = s_j} |p| \tag{7}$$

The average of $I_U(s_j | s)$ is denoted by $\overline{I_U(\cdot | s)} = \Sigma\, p_j I_U(s_j | s)$.

The next theorem puts a lower bound on the average information.

Theorem 2. For any universal computer U and any list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$, the average conditional algorithmic information obeys the bound

$$\overline{I_U(\cdot | s)} \geq H(s) + r(s) + p_1 \tag{8}$$

Proof. We denote by s_j' a shortest string for which $U(s_j', s^*) = s_j$. The N strings s_j' form a prefix-free code. If the N strings s_j' are represented by the leaves of a binary tree, then there is at least one node that has no sibling. Otherwise $U(p, s^*)$ would be defined only for a finite number N of programs p, and U would not be a universal computer. Let us denote by $\mathfrak{Q}$ a sibling-free node and by q its probability ($q \geq p_1$). Then a shorter prefix-free code $\{s''_j\}$ can be obtained by moving node $\mathfrak{Q}$ down one level. More precisely, for states j corresponding to leaves of the subtree branching from node $\mathfrak{Q}$, s''_j is obtained from s'_j by removing the digit corresponding to the link between node $\mathfrak{Q}$ and its parent; for all other states j, $s''_j = s'_j$. The code-word lengths of the new code are $|s''_j| = |s'_j| - 1$ if state j is a leaf of the subtree branching from node $\mathfrak{Q}$ and $|s''_j| = |s'_j|$ otherwise. Since the new code is prefix-free, its average code-word length is greater than or equal to the Huffman code-word length $\bar{l}(s)$. It follows that

$$\begin{aligned}\overline{I_U(\cdot\,|\,s)} = \sum_j p_j|s'_j| &= \sum_j p_j|s''_j| + q \geq \bar{l}(s) + p_1 \\ &= H(s) + r(s) + p_1 \end{aligned} \tag{9}$$

which proves the theorem. ■

We can now proceed to define an optimal universal computer.

Definition 5. U is an *optimal universal computer* if there is a constant $\epsilon > 0$ such that for all lists of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$ with $p_1 \geq \epsilon$ the average conditional algorithmic information has its minimum value

$$\overline{I_U(\cdot\,|\,s)} = H(s) + r(s) + p_1 \tag{10}$$

Theorem 3. For any $\epsilon > 0$ there is an optimal universal computer U_ϵ.

Proof. Let U be an arbitrary universal computer and $\epsilon > 0$. For any list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$ with $p_1 \geq \epsilon$ we define $c'_1(s) = c_1(s) \circ 1 = 0\cdots01$ and $c'_j(s) = c_j(s)$ for $j = 2, \ldots, N$, where $\circ$ denotes concatenation of strings. The strings $c'_j(s)$ thus differ from the Huffman code $c_j(s)$ in that a 1 has been appended to the code word for the state $j = 1$. According to condition (5), $l_1(s) + 1 \leq N_0 \equiv \lfloor -\log_g \epsilon + 2 \rfloor$, where $g = (\sqrt{5} + 1)/2$ and $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. We denote by σ_0 a string composed of N_0 zeros; none of the strings $c'_j(s)$ is longer than σ_0.

For the definition of $U_\epsilon(p, q)$ we distinguish two cases. If the binary string q is of the form

$$q = \sigma_0 \circ q_s \qquad \text{with} \quad U(q_s, \Lambda) = s \tag{11}$$

for some list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$ with $p_1 \geq \epsilon$, then $U_\epsilon(p, q)$ is defined for

$$p \in D(q) \equiv \{\sigma_0 \circ p' \mid U(p', q) \text{ is defined}\} \\ \cup \{c_j'(s) \mid 1 \leq j \leq N\} \tag{12}$$

with

$$U_\epsilon(\sigma_0 \circ p', q) \\ = U(p', q) \qquad \text{whenever } U(p', q) \text{ is defined} \tag{13}$$

and

$$U_\epsilon(c_j'(s), q) = s_j \qquad \text{for} \quad j = 1, \ldots, N \tag{14}$$

If the binary string q is not of the form (11), then $U_\epsilon(p, q)$ is defined for

$$p \in D(q) \equiv \{\sigma_0 \circ p' \mid U(p', q) \text{ is defined}\} \tag{15}$$

with

$$U_\epsilon(\sigma_0 \circ p', q) = U(p', q) \qquad \text{whenever } U(p', q) \text{ is defined} \tag{16}$$

In both cases, the set $D(q)$, which is the domain of $U_\epsilon(\cdot, q)$, is clearly prefix-free. Moreover, since $U_\epsilon(\sigma_0 \circ p, q) = U(p, q)$ whenever $U(p, q)$ is defined and U is a universal computer, U_ϵ is also a universal computer, with the simulation constant sim(C) increased by N_0.

For any string t the minimal program on U_ϵ—i.e., the shortest program given an empty free data string—is $t^*(U_\epsilon) = \sigma_0 \circ t^*(U)$, where $t^*(U)$ is the minimal program for t on U. In particular, the shortest program for U_ϵ to compute s is $s^*(U_\epsilon) = \sigma_0 \circ s^*(U)$. Since $U_\epsilon(c_j'(s), s^*(U_\epsilon)) = s_j$ and $|c_j'(s)| \leq N_0$ for $j = 1, \ldots, N$, while $|p| \geq N_0$ for all other programs $p \in D(s^*(U_\epsilon))$, it follows immediately that

$$I_{U_\epsilon}(s_j|s) = |c_j'(s)| = |c_j(s)| + \delta_{1j} = l_j(s) + \delta_{1j} \tag{17}$$

and thus that

$$\overline{I_{U_\epsilon}(\cdot|s)} = \sum p_j I_{U_\epsilon}(s_j|s) = \sum p_j |c_j'(s)| = \sum p_j |c_j(s)| + p_1 \\ = \bar{l}(s) + p_1 = H(s) + r(s) + p_1 \quad \blacksquare \tag{18}$$

If $U(q_s, \Lambda) = U_\epsilon(\sigma_0 \circ q_s, \Lambda) = s$, i.e., if q_s is a program for U generating a list of states s, the programs p for which $U_\epsilon(p, \sigma_0 \circ q_s)$ is defined can be represented by a binary tree similar to Fig. 2. With respect to the binary tree representing the Huffman code (Fig. 1), the leaf for the $j = 1$ state has been moved up one level to make room for the new node labeled by U. This new node leads to a subtree representing all programs p' for which $U(p', \sigma_0 \circ q_s)$ is defined.

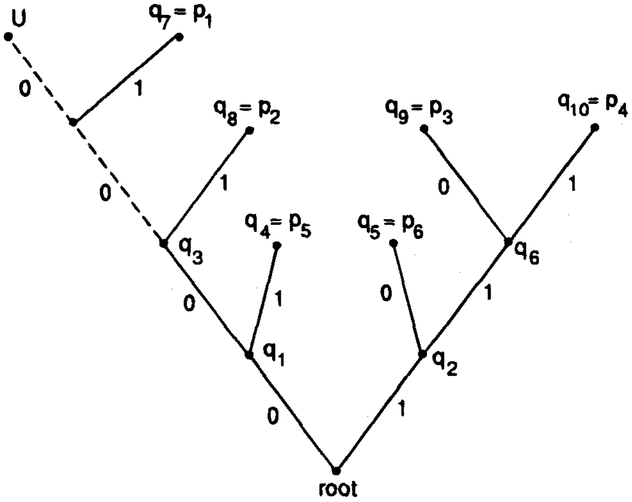

Fig. 2. Binary tree representing all valid programs for the optimal universal computer U_ϵ in the presence of a free data string generating a list of states $((s_1, p_1), \ldots, (s_6, p_6))$. With respect to the tree in Fig. 1, the node labeled $q_7 = p_1$ has been moved up one level to make room for the subtree representing programs for U.

The operation of the optimal universal computer U_ϵ can be described in the following way. When U_ϵ reads a string that begins with N_0 zeros from its program tape, U_ϵ disregards the N_0 zeros and interprets the rest of the string as a program for the universal computer U, executing it accordingly. If U_ϵ encounters the digit 1 while reading the first N_0 digits from its program tape, U_ϵ interrupts reading from the program tape, reads in the free data string, and executes it. If the result of executing the free data string is a list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$, U_ϵ establishes the modified Huffman code $\{c_j'(s)\}$ for s, continues reading digits from the program tape until the string read matches one of the code words, say $c_{j0}'(s)$, and then prints the string s_{j_0}. The output of U_ϵ is undefined in all other cases.

Since $r(s) + p_1 < 1$ (Gallager, 1978), $H(s) \le \overline{I_U(\cdot\,|\,s)} < H(s) + 1$ for any optimal universal computer U. For the particular optimal universal computer U_ϵ defined in the proof of Theorem 3, however, the information $I_{U_\epsilon}(s_j\,|\,s)$ is completely determined by the Huffman code-word length for the jth state and therefore is completely determined by the probabilities $p_1, \ldots, p_N$. This optimal universal computer does not recognize intrinsically simple states. As an aside, note that U_ϵ cannot give a short description of the background information for any probability distribution, because a minimal program for computing the list of states s on U_ϵ must begin with N_0 zeros. It turns out that all optimal universal computers, not just U_ϵ, are unable to recognize intrinsically simple states. The following theorem formulates this inability for all optimal universal computers in a slightly weaker form than holds for U_ϵ. As a consequence, the use of algorithmic information with respect to an optimal universal computer to quantify the information in an observational record presents no advantage over the use of Huffman coding.

Theorem 4. For any optimal universal computer U and any list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$ for which $\overline{I_U(\cdot \mid s)} = H(s) + r(s) + p_1$, the following holds: If $p_i > p_j$, then $I_U(s_i \mid s) \leq I_U(s_j \mid s)$. Optimal universal computers therefore do not recognize intrinsically simple states.

Proof. To prove the theorem, we show that $\overline{I_U(\cdot \mid s)} > H(s) + r(s) + p_1$ for any universal computer U and any list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$ for which there are indices i and j such that $p_i > p_j$ but $I_U(s_i \mid s) > I_U(s_j \mid s)$. We denote by s_j' a shortest string for which $U(s_j', s^*) = s_j$. The strings s_j' form a prefix-free code. Following an argument similar to the proof of Theorem 2, we can shorten that code on the average by moving a sibling-free node one level down and in addition by interchanging the code words for states i and j. The resulting shorter code must obey the Huffman bound, from which the inequality $\overline{I_U(\cdot \mid s)} > \bar{l}(s) + p_1 = H(s) + r(s) + p_1$ follows. ■

3. PRESERVING SIMPLE STATES BY GIVING UP 1/2 BIT

Although the discussion in the last section shows that optimal universal computers present no advantages over Huffman coding, the main idea behind their construction can be further exploited. If the subtree representing the programs for the universal computer U is not attached next to the $j = 1$ leaf as in Fig. 2, but instead is attached close to the root as in Fig. 3, the resulting universal computer U_3 combines the desirable properties of Huffman coding and the computer U. This is the content of the following theorem.

Theorem 5. For any universal computer U there is a universal computer U_3 such that

$$I_{U_3}(t_1 \mid t_2) \leq I_U(t_1 \mid t_2) + 3 \tag{19}$$

for all binary strings t_1 and t_2, and that

$$H(s) \leq \overline{I_{U_3}(\cdot \mid s)} < H(s) + 1 \tag{20}$$

and

$$\overline{I_{U_3}(\cdot \mid s)} \leq H(s) + r(s) + 1/2 \tag{21}$$

for all lists of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$.

Proof. Let U be an arbitrary universal computer. For any list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$ we define the set of strings $c_j'(s)$ as follows. We

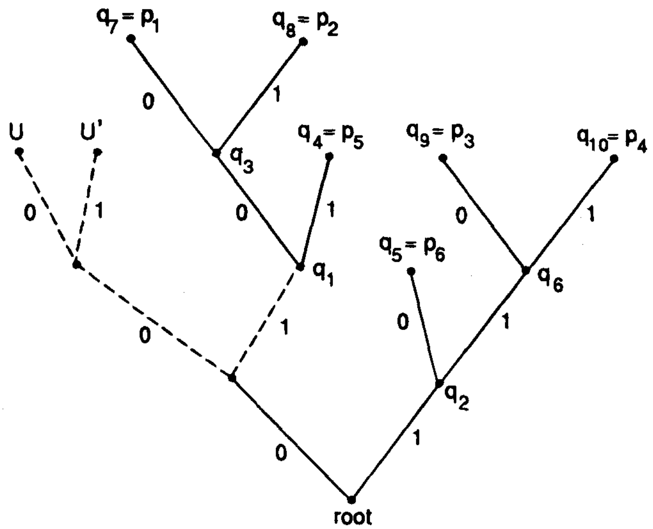

Fig. 3. Binary tree representing all valid programs for the universal computer U_3 in the presence of a free data string generating a list of states $s = ((s_1, p_1), \ldots, (s_6, p_6))$. With respect to the tree in Fig. 1, the level-1 node labeled q_1 has been moved up one level to make room for the subtrees representing programs for U. More precisely, the binary tree represents the programs p for which $U_3(p, 000 \circ q_s)$ is defined if $U_3(000 \circ q_s, \Lambda) = s$. The node labeled U is the root of a subtree corresponding to the programs p' for which $U(p', 000 \circ q_s)$ is defined, and the node labeled U' is the root of a subtree corresponding to the programs p' for which $U(p', q_s)$ is defined.

start from the binary tree formed by the Huffman code words $c_j(s)$, where we denote by q_1 the probability of the level-1 node connected to the root by the link labeled 0 (see Fig. 1). According to the value of q_1, we distinguish two cases. In the case $q_1 \leq 1/2$, $c_j'(s) = 01 \circ c_j^+(s)$ if $c_j(s)$ is of the form $c_j(s) = 0 \circ c_j^+(s)$, and $c_j'(s) = c_j(s)$ if $c_j(s)$ is of the form $c_j(s) = 1 \circ c_j^+(s)$. In the case $q_1 > 1/2$, $c_j'(s) = 01 \circ c_j^+(s)$ if $c_j(s)$ is of the form $c_j(s) = 1 \circ c_j^+(s)$, and $c_j'(s) = 1 \circ c_j^+(s)$ if $c_j(s)$ is of the form $c_j(s) = 0 \circ c_j^+(s)$.

Figure 3 illustrates the binary tree formed by the code words $c_j'(s)$ for the case $q_1 \leq 1/2$. Of the two main subtrees emerging from the level-1 nodes in Fig. 1, the subtree having smaller probability is moved up one link and attached to the node labeled 01, and the subtree having larger probability is attached to the node labeled 1. In this way, the node labeled 00 is freed for the subtrees representing the valid programs for U.

For the definition of $U_3(p, q)$ we distinguish three cases. If the binary string q is of the form

$$q = 000 \circ q_s \qquad \text{with} \qquad U(q_s, \Lambda) = s \tag{22}$$

for some list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$, then $U_3(p, q)$ is defined for

$$p \in D(q) \equiv \{000 \circ p' \mid U(p', q) \text{ is defined}\}$$
$$\cup \{001 \circ p' \mid U(p', q_s) \text{ is defined}\}$$
$$\cup \{c_j'(s) \mid 1 \leq j \leq N\} \tag{23}$$

with

$$U_3(000 \circ p', q)$$
$$= U(p', q) \qquad \text{whenever } U(p', q) \text{ is defined} \tag{24}$$
$$U_3(001 \circ p', q)$$
$$= U(p', q_s) \qquad \text{whenever } U(p', q_s) \text{ is defined} \tag{25}$$

and

$$U_3(c_j'(s), q) = s_j \qquad \text{for} \quad j = 1, \ldots, N \tag{26}$$

If the binary string q is of the form

$$q = 000 \circ q' \tag{27}$$

but there is *no* list of states s such that $U(q', \Lambda) = s$, then $U_3(p, q)$ is defined for

$$p \in D(q) \equiv \{000 \circ p' \mid U(p', q) \text{ is defined}\}$$
$$\cup \{001 \circ p' \mid U(p', q') \text{ is defined}\} \tag{28}$$

with

$$U_3(000 \circ p', q)$$
$$= U(p', q) \qquad \text{whenever } U(p', q) \text{ is defined} \tag{29}$$

and

$$U_3(001 \circ p', q)$$
$$= U(p', q') \qquad \text{whenever } U(p', q') \text{ is defined} \tag{30}$$

Finally, if q is not of the form (27), then $U_3(p, q)$ is defined for

$$p \in D(q) \equiv \{000 \circ p' \mid U(p', q) \text{ is defined}\} \tag{31}$$

with

$$U_3(000 \circ p', q)$$
$$= U(p', q) \qquad \text{whenever } U(p', q) \text{ is defined} \tag{32}$$

In all three cases, the set $D(q)$, which is the domain of $U_3(\cdot, q)$, is clearly prefix-free. Moreover, since $U_3(000 \circ p, q) = U(p, q)$ whenever $U(p, q)$ is defined and U is a universal computer, U_3 is also a universal computer, with the simulation constant sim(C) increased by 3. Equation (19) holds because of the following. The minimal program for t_2 on U_3 in the presence of an empty free data string is $t_2^*(U_3) = 000 \circ t_2^*(U)$ since $U_3(p, \Lambda)$ is defined only if $p = 000 \circ p'$ and $U(p', \Lambda)$ is defined, in which case $U_3(p, \Lambda) = U(p', \Lambda)$. If p is a minimal program for t_1 on U in the presence of the minimal program for t_2, i.e., if

$$U(p, t_2^*(U)) = t_1, \qquad |p| = I_U(t_1 | t_2) \tag{33}$$

then

$$\begin{aligned} U_3(001 \circ p, t_2^*(U_3)) &= U_3(001 \circ p, 000 \circ t_2^*(U)) \\ &= U(p, t_2^*(U)) = t_1 \end{aligned} \tag{34}$$

and therefore

$$I_{U_3}(t_1 | t_2) \leq |001 \circ p| = |p| + 3 \tag{35}$$

The strings $c_j'(s)$ form a prefix-free code with an unused code word of length 2, for which $\sum p_j |c_j'(s)| < H(s) + 1$ according to Theorem 3 in Gallager (1978). [In Gallager (1978) the inequality appears with a $\leq$ sign, but equality can occur only if the smallest probability p_1 is equal to zero, a case we have excluded.] The shortest program for U_3 to compute s is $s^*(U_3) = 000 \circ s^*(U)$, where $s^*(U)$ is the shortest program for U to compute s. Since $U_3(c_j'(s), s^*(U_3)) = s_j$ for $j = 1, \ldots, N$, it follows immediately that $I_{U_3}(s_j | s) \leq |c_j'(s)|$ and thus that

$$\overline{I_{U_3}(\cdot | s)} = \sum p_j I_{U_3}(s_j | s) \leq \sum p_j |c_j'(s)| < H(s) + 1 \tag{36}$$

which establishes the upper bound in condition (20). The lower bound in (20) holds for all universal computers. Equation (21) follows from

$$\begin{aligned} \sum p_j |c_j'(s)| &= \sum p_j |c_j(s)| + \min(q_1, 1 - q_1) \\ &= \bar{l}(s) + \min(q_1, 1 - q_1) \\ &\leq H(s) + r(s) + 1/2 \quad \blacksquare \end{aligned} \tag{37}$$

If $U(q_s, \Lambda) = U_3(000 \circ q_s, \Lambda) = s$, i.e., if q_s is a program for U generating a list of states s, the programs p for which $U_3(p, 000 \circ q_s)$ is defined can be represented by a binary tree similar to Fig. 3. The level-3 node labeled U is the root of a subtree corresponding to the programs p' for which $U(p', 000 \circ q_s)$ is defined, and the level-3 node labeled U' is the root of a subtree corresponding to the programs p' for which $U(p', q_s)$ is defined.

The operation of the universal computer U_3 can be described in the following way. When U_3 reads a string that begins with the prefix 000 from its program tape, U_3 disregards the prefix and interprets the rest of the string as a program for the universal computer U, executing it accordingly. When U_3 reads a string that begins with the prefix 001 from its program tape, the output is only defined if the free data string begins with 000, in which case U_3 disregards the first three digits of the program and free data strings and interprets the rest of the strings as program and free data strings for the universal computer U, executing it accordingly. If U_3 encounters the digit 1 while reading the first two digits from its program tape, U_3 interrupts reading from the program tape, reads in the free data string, and executes it. If the result of executing the free data string is a list of states $s = ((s_1, p_1), \ldots, (s_N, p_N))$, U_3 establishes the modified Huffman code $\{c'_j(s)\}$ for s, continues reading digits from the program tape until the string read matches one of the code words, say $c'_{j0}(s)$, and then prints the string s_{j_0}. The output of U_3 is undefined in all other cases.

The computer U_3 compromises between the desirable properties of algorithmic information and Huffman coding. Since algorithmic information defined with respect to U_3 exceeds algorithmic information relative to U by at most 3 bits, states that are simple with respect to U are simple with respect to U_3. Those 3 bits are the price to pay for a small upper bound on average information. The average conditional algorithmic information $\overline{I_{U_3}(\cdot \mid s)}$ obeys the close double bound (20) and exceeds the Huffman bound $\bar{l}(s)$ by at most 0.5 bit. This half bit is the price to pay for the recognition of intrinsically simple states.

4. CONCLUSION

We have shown that any universal computer U can be modified in such a way that (i) the modified universal computer U_3 recognizes the same intrinsically simple states as U and (ii) average algorithmic information with respect to U_3 obeys the same close double bound as Huffman coding, $H(s) \leq \overline{I_{U_3}(\cdot \mid s)} < H(s) + 1$. If for any choice of a universal computer U, total free energy is defined with respect to the corresponding modified universal computer U_3, i.e., if the change of total free energy due to finding the system in the jth state is $\Delta F_{j,\mathrm{tot}} = -k_\mathrm{B}T \ln 2\, [-H(s) + I_{U_3}(s_j \mid s)]$, then the bounds for the average change in total free energy are given by

$$0 \geq \Delta F_{\mathrm{tot}} > -k_\mathrm{B}T \ln 2 \tag{38}$$

instead of by (3).

This result effectively eliminates the undetermined computer-dependent constant from applications of algorithmic information theory to statistical physics. Except for an unavoidable loss due to the coding bounded by $k_\mathrm{B}T$ ln 2, on the average available work is independent of the information the observer has acquired about the system, any decrease of the statistical entropy being balanced by an equal increase in algorithmic information.

ACKNOWLEDGMENT

The author wishes to thank Carlton M. Caves for suggesting the problem and for many enlightening discussions.

REFERENCES

Bennett, C. H. (1982). *International Journal of Theoretical Physics*, **21**, 905.

Caves, C. M. (1990). In *Complexity, Entropy, and the Physics of Information*, W. H. Zurek, ed., Addison-Wesley, Redwood City, California, p. 91.

Caves, C. M. (1993a). *Physical Review E*, **47**, 4010.

Caves, C. M. (1993b). In *Physical Origins of Time Asymmetry*, J. J. Halliwell, J. Pérez-Mercader, and W. H. Zurek, eds., Cambridge University Press, Cambridge, p. 47.

Chaitin, G. J. (1987). *Algorithmic Information Theory*, Cambridge University Press, Cambridge.

Denker, J. S., and leCun, Y. (1993). In *Workshop on Physics and Computation: PhysComp '92*, IEEE Computer Society Press, Los Alamitos, California, p. 122.

Gallager, R. G. (1978). *IEEE Transactions on Information Theory*, **IT-24**, 668.

Huffman, D. A. (1952). *Proceedings IRE*, **40**, 1098.

Jaynes, E. T. (1983). In *Papers on Probability, Statistics, and Statistical Physics*, R. D. Rosenkrantz, ed., Kluwer, Dordrecht, Holland.

Katona, G. O. H., and Nemetz, T. O. H. (1976). *IEEE Transactions on Information Theory*, **IT-22**, 337.

Kolmogoroff, A. N. (1965). *Problemy Peredachi Informatsii*, **1**, 3 [*Problems of Information Transmission*, **1**, 1 (1965)].

Landauer, R. (1961). *IBM Journal of Research and Development*, **5**, 183.

Landauer, R. (1988). *Nature*, **355**, 779.

Schack, R. (1994). *IEEE Transactions on Information Theory*, **IT-40**, 1246.

Schack, R., and Caves, C. M. (1992). *Physical Review Letters*, **69**, 3413.

Schack, R., and Caves, C. M. (1993). *Physical Review Letters*, **71**, 525.

Schack, R., and Caves, C. M. (1996a). *Physical Review E*, **53**, 3257.

Schack, R., and Caves, C. M. (1996b). *Physical Review E*, **53**, 3387.

Schack, R., D'Ariano, G. M., and Caves, C. M. (1994). *Physical Review E*, **50**, 972.

Solomonoff, R. J. (1964). *Information and Control*, **7**, 1.

Szilard, L. (1929). *Zeitschrift für Physik*, **53**, 840.

Welsh, D. (1988). *Codes and Cyptography*, Clarendon Press, Oxford.

Zurek, W. H. (1989a). *Nature*, **341**, 119.

Zurek, W. H. (1989b). *Physical Review A*, **40**, 4731.

Zvonkin, A. V., and Levin, L. A. (1970). *Uspekhi Matematicheskikh Nauk* **25**, 85 [*Russian Mathematical Surveys*, **25**, 83 (1970)].