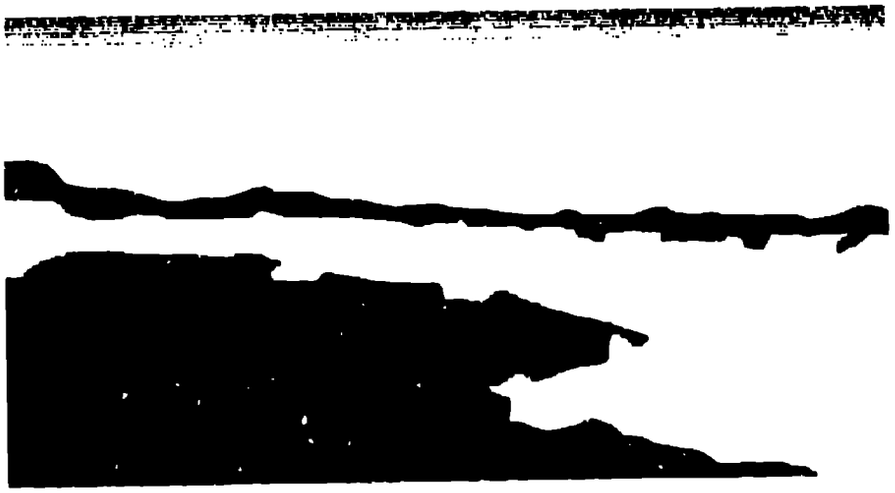
LA-UR- 93-1569

Title: THERMODYNAMICS OF COMPUTATION AND INFORMATION DISTANCE

Author(s): Charles H. Bennett
Peter Gacs
Ming Li
Paul M. B. Vitanyi
Wojciech H. Zurek, T-6

Submitted to: Symposium on Theory of Computation (STOC)
25th ACM Conf., San Diego, CA, June 7-11, 1993



Los Alamos
NATIONAL LABORATORY

4/12/93

Thermodynamics of Computation and Information Distance

Charles H. Bennett Péter Gács Ming Li
Paul M.B. Vitányi Wojciech H. Zurek

March 12, 1993

Abstract

Applying the tools of algorithmic information theory, we compare several candidates for an asymptotically machine-independent, absolute measure of the informational or "cognitive" distance between discrete objects x and y. The *maximum* of the conditional Kolmogorov complexities $\max\{K(y|x), K(x|y)\}$, is shown to be optimal, in the sense of being minimal within an additive constant among semicomputable, symmetric, positive semidefinite functions of x and y satisfying a reasonable normalization condition and obeying the triangle inequality. The optimal metric, in turn, differs by at most an additive logarithmic term from the size of the smallest program for a universal *reversible* computer to transform x into y. This program functions in a "catalytic" capacity, being retained in the computer before, during, and after the computation. Similarly, the *sum* of the conditional complexities, $K(y|x) + K(x|y)$, is shown to be equal within a logarithmic term to the minimal amount of information flowing out and in during a reversible computation in which the program is not retained. Finally, using the physical theory of reversible computation, it is shown that the simple difference $K(x) - K(y)$ is an appropriate (ie universal, antisymmetric, and transitive) measure of the amount of thermodynamic work required to transform string x into string y by the most efficient process.

1 Introduction

The algorithmic entropy, or Kolmogorov complexity, $K(x)$ of a string x may be defined as the size in bits of a minimal sized program to compute x on an optimal binary universal computer. Intuitively, $K(x)$ represents the amount of information required to generate x by an effective process. The conditional algorithmic entropy $K(y|x)$, of y relative to x, may be defined similarly as the size of a minimal sized program to compute y if x is furnished as an auxiliary input to the computation. Intuitively $K(y|x)$ represents the amount of

information required to generate y from x by an effective process. The functions $K(\cdot)$ and $K(\cdot|\cdot)$, though defined in terms of a particular machine model, are machine independent up to an additive constant, and acquire an asymptotically universal and absolute character through Church's thesis, from the ability of universal machines to simulate one another and execute any effective process.

It would be good to have a similarly universal measure of the informational "distance" between two strings, i.e. the minimal quantity of information sufficient to translate between x and y, generating either string effectively from the other. The conditional entropy $K(y|x)$ itself is of course unsuitable for this purpose because it is unsymmetric: $K(\Lambda|x)$, where Λ is the empty string, is small for all x, yet intuitively a long random string x is not close to the empty string.

This asymmetry can be remedied by defining the algorithmic informational distance between x and y to be the sum of the relative entropies[17]:

$$K(y|x) + K(x|y).$$

The resulting metric will overestimate the information required to translate between x and y if there is some redundancy between the information required to get from x to y and the information required to get from y to x.

We therefore inquire to what extent the information required to compute x from y can be made to overlap with that required to compute y from x. In some simple cases, *complete* overlap can be achieved, so that the same minimal program suffices to compute x from y as to compute y from x. For example if x and y are independent random binary strings of the same length n (up to additive contants $K(x|y) = K(y|x) = n$), then their bitwise exclusive or $x \oplus y$ serves as a minimal program for both computations. Similarly, if $x = uv$ and $y = vw$ where u, v, and w are independent random strings of the same length, then $u \oplus w$ is a minimal program to compute either string from the other.

Now suppose that more information is required for one of these computations than for the other, say,

$$K(y|x) > K(x|y).$$

Then the minimal programs cannot be made identical because they must be of different sizes. Nevertheless, in simple cases, the overlap can still be made complete, in the sense that the larger program (for y given x) can be made to contain all the information in the smaller program, as well as some additional information. This is so when x and y are independent random strings of unequal length, for example u and vw above. Then $u \oplus v$ serves as a minimal program for u from vw, and $(u \oplus v)w$ serves as one for vw from u.

A principal result of this paper is that, up to logarithmic error terms, the information required to translate between two strings can always be represented in this maximally overlapping way. [1] Namely, let

[1] On the other hand, the programs for going between independent random x and y can at

$$\begin{aligned} k_1 &= K(x|y), \\ k_2 &= K(y|x), \\ l &= k_2 - k_1 . \end{aligned}$$

Then there is a string d of length $k_1 + O(\log k_1)$ and a string q of length $l + \log l$ such that d serves as the minimal program both from xq to y and from y to xq. This means that the information required to pass from x to y is always maximally correlated with the information required to get from y to x. It is therefore never the case that a large amount of information is required to get from x to y and a large *but independent* amount of information is required to get from y to x. (It is very important here that the time of computation is completely ignored: this is why this result does not contradict the idea of one-way functions.)

The situation is analogous to the inverse function theorem of multidimensional analysis. This theorem says that under certain conditions, if we have a vector function $f(x,d)$ then it has an inverse $g(y,d)$ such that in a certain domain, $f(d,x) = y$ holds if and only if $g(y,d) = x$. In the function going from y to x, the parameter d remains the same as in the function going from x to y.

The process of going from x to y way may be broken into two stages. First, add the string q; second, use the difference program d between qx and y. In the reverse direction, first use d to go from y to qx; second, erase q. Thus the computation from x to y needs both d and q, while the computation from y to x needs only d as program.

The foregoing is true of ordinary computations, but if one insists that the computation be performed *reversibly*, that is by a machine whose transition function is 1:1 [14, 4], then the full program $p = dq$ is needed to perform the computation in either direction. This is because reversible computers cannot get rid of unwanted information simply by erasing it as ordinary irreversible computers do. If they are to get rid of unwanted information at all, they must cancel it against equivalent information already present elsewhere in the computer. Reversible computations are discussed further in section 5.

Section 4 develops an axiomatic theory of information distance and argues that the function

$$E_1(x,y) \quad \max\{K(x|y), K(y|x)\}$$

is the most natural way of formalizing the notion of the minimal algorithmic informational distance between x and y. This function is symmetric, obeys the triangle inequality to within an additive constant, and is minimal among

one wishes, be made *completely independent*. For example use y to go from x to y and x to go from y to x. We suspect this may be true, in general, at least to within logarithmic terms.

a class of functions satisfying a normalization constraint appropriately limiting the number of distinct strings y within a given distance of any x.

Section 5 defines a reversible distance E_2 representing the amount of information required to program a reversible computation from x to y. The E_2 distance is equal within an additive constant to the length of the conversion program $p = dq$ considered above, and so is at most logarithmically greater than the optimal distance E_1. The reversible program functions in a catalytic capacity in the sense that it remains unchanged throughout the computation.

Section 6 instead considers reversible computations in which additional information r besides x is consumed, and additional information s besides y is generated in the course of the computation. The sum of these amounts of information is shown to be equal to within a logarithmic term to Zurek's sum metric $K(y|x) + K(x|y)$, which is typically larger than our proposed optimal metric because of the redundancy between r and s. Section 7 compares the topological properties of the optimal and sum metrics.

Finally Section 8 considers the problem of defining a thermodynamic *cost* of transforming x into y, and argues that it ought to be an antisymmetric, transitive function, in contrast to the informational metrics which are symmetric. Landauer's principle connecting logical and physical irreversibility is invoked to argue in favor of $K(x) - K(y)$ as the ideal thermodynamic cost of transforming x into y.

2 Known properties of algorithmic entropy

Let $l(p)$ denote the length of the binary string p. Let $\#S$ denote the number of elements of set S.

The following definitions and theorems spell out some basic properties of $K(x,y)$.

The notion of semicomputable functions was introduced into algorithmic information theory in [19]. That paper can also serve as a good introduction to the subject. We say that a real valued function $f(x,y)$ over strings is **upper semicomputable** if the set of triples

$$\{(x,y,d) : f(x,y) < d,\ d \text{ rational}\}$$

is recursively enumerable. A function f is **lower semicomputable** if $-f$ is upper semicomputable.

A **prefix set** is a set of strings such that no member is a prefix of any other member. A partial recursive function $F(p,x)$ is called a **prefix interpreter** if for each x the set $\{p : l(p)F(p,x) = y\}$ is a prefix set. The argument p is called a **self-delimiting program** for y from x, because, owing to the prefix property, no punctuation is required to tell the interpreter how much of p to use.

We define the **conditional algorithmic entropy** of y with condition x, with respect to the interpreter F as

$$K_F(y|x) = \min_{F(p,x)=y} l(p) .$$

(2.1) Theorem There is a prefix interpreter U with the property that for all other prefix interpreters F and for all p, x there is an additive constant c_F such that

$$K_U(p|x) \leq K_F(p, x) + c_F .$$

□

Such a prefix interpreter will be called **optimal**. We fix such an U and write

$$K(x|y) = K_U(x|y) .$$

We will call $K(x|y)$ the **algorithmic entropy** of x with respect to y.

From now on, we will denote by $\stackrel{+}{<}$ an inequality to within an additive constant, and by $\stackrel{+}{=}$ the situation when both $\stackrel{+}{<}$ and $\stackrel{+}{>}$ hold.

(2.2) Theorem

(a) $K(x|y)$ is an upper semicomputable function with the property $\sum_y 2^{-K(x|y)} \leq 1$.

(b) If $f(x, y)$ is an upper semicomputable function with $\sum_y 2^{-f(x,y)} \leq 1$ then $K(y|x) \stackrel{+}{<} f(x, y)$.

□

The proof of the following theorem can be found e.g. in [10] and [7].

(2.3) Theorem

$$K(x, y) \stackrel{+}{<} K(x) + K(y|x, K(x)) .$$

□

Let $B(i, x)$ be the set of strings y such that $K(y|x) \leq i$.

(2.4) Theorem We have

$$\log \#B(i, x) \stackrel{+}{<} i - K(i|x) .$$

□

3 Conversion programs

(3.1) Difference Theorem With the notation of the Introduction, suppose $k_1 \leq k_2$. Then there is a string p of length $k_2 + O(\log k_2)$ such that

$$U(p, 0x) = y,\ U(p, 1y) = x\ .$$

□

This is equivalent to asserting that there is a string d of length k_2 such that both $K(y|xq, d)$ and $K(xq|y, d)$ are bounded by $O(\log k_2)$. We call this theorem the Difference Theorem since it asserts the existence of a difference string p that converts both ways between x and y and at least one of these conversions is optimal. If $k_1 = k_2$ then the conversion is optimal in both directions.

Proof Let S be the set of all binary strings. Let X, Y be two disjoint sets whose elements are in a one-to one correspondence with the elements of S: we could e.g. set $X = \{ (s, 0) : s \in S \}$ and $Y = \{ (s, 1) : s \in S \}$. Let $G = (X \cup Y, E)$ be the following infinite bipartite graph over $X \cup Y$ with set of edges of E where

$$E = \{ (x, y) : K(x|y) \leq k_1,\ K(y|x) \leq k_2 \}\ .$$

By definition, the maximum degree of the nodes in X is at most 2^{k_2+1} and in Y is at most 2^{k_1+1}.

Two edges are adjacent if they have common endpoints. A matching is a set of nonadjacent edges. We can partition E into at most 2^{k_2+2} matchings $M_1, M_2, \ldots$. If we can do this constructively we have a program p of length $k_2 + O(\log k_2)$ that takes $0x$ into y and $1y$ into x. Indeed, for a pair $(x, y) \in E$, the number i of the matching M_i containing (x, y) has length at most $k_2 + O(1)$. Knowing i and x gives y while knowing i and y gives x.

Let us do the partitioning constructively, in the most simple-minded way. By its definition, the set E can be enumerated into a sequence $e_1, e_2, \ldots$ of edges. In step t, a new edge e_t is given. We will put it into one of the nonempty matchings created so far, (it does not matter into which one) if this is possible; if it is not we create a new matching. For clarity, here is a formal definition. We define, recursively, a function $n(t)$ for each t such that $M_i = \{ e_t : n(t) = i \}$. Let

$$M_i^t = \{ e_u : n(u) = i,\ u < t \}\ .$$

Then $n(t)$ is the first i such that e_t is not adjacent to any edge of M_i^t.

Let us show that the number of nonempty matchings is indeed at most 2^{k_2+2}. Let M_i be a nonempty matching; then there is a t such that $i = n(t)$. The edge e_t is adjacent to some edge in each matching M_j for $j < i$. But the number of edges that an edge can be adjacent to is at most the sum of the degrees of the endpoints — actually, 2 less than that. Hence, $i - 1 < 2^{k_2+2} - 2$.

More explicitly, we describe the program p such that $U(p, bx) = y$ if $b = 0$ and $U(p, by) = x$ if $b = 1$. It contains the following parts.

The numbers k_2 and i.

Procedure to generate the sequence $e_1, e_2, \ldots$.

Procedure to simultaneously generate the matchings $M_1, M_2, \ldots$.

Procedure to generate M_i.

Procedure to find y using x, M_i if $b = 0$ and to find x using y, M_i if $b = 1$.

■

(3.2) Excess Theorem Let us use the above notation, with $l = k_2 - k_1$. There is a binary string q of length $l + O(\log l)$ such that

$$\begin{aligned} K(y|qx) &= k_1 + O(\log k_1)\,, \\ K(qx|y) &= k_1 + O(\log k_1)\,. \end{aligned}$$

□

Proof Similarly to the above proof, let the graph $G = (X \cup Y, E)$ be now such that

$$E = \{\, (x\ y) : K(x|y) \le k_1 \,\}\,.$$

Its edges are enumerated again as $e_1, e_2, \ldots$. Now the degree of the nodes in Y is still $\le 2^{k_1+1}$. Let us define a new graph $G' = (X' \cup Y, E')$ such that in it, the degree of the degree of nodes in X' is also bounded by 2^{k_1}. To do this, we will simply split the nodes $x \in X$ as soon as their degree would rise above 2^{k_1}. For each node $x \in X$, let $(x, y_1), (x, y_2), \ldots$ be the natural enumeration (in the order of the edges $e_1, e_2, \ldots$) of the set of edges $\{\, (x, y) : (x, y) \in E \,\}$. Let

$$m(n) = \lceil n/2^{k_1} \rceil\,.$$

We split each node x into a sequence of nodes $z_1(x), z_2(x), \ldots$. We have

$$\begin{aligned} X' &= \{\, z_m(x) : x \in X,\ m = 1, 2, \ldots \,\}\,, \\ E' &= \{\, (z_{m(i)}(x), y_i) : x \in X,\ i = 1, 2, \ldots \,\}\,. \end{aligned}$$

Thus, at node x, the first group of 2^{k_1} edges (x, y_i) will be attached to the new node $z_1(x)$, the second group to $z_2(x)$, etc.

Now the binary string q is a self-delimiting program for the number m such that $(z_m(x), y) \in E'$. By definition, $m < 2^{l+1}$, therefore $l(q) \le l + O(\log l)$. The pair x, m determines the node $z_m(x)$. Since the maximum degree of the graph G' is at most 2^{k_1} we have the desired upper bounds on the conditional algorithmic entropies. ■

4 Distance axioms

Let us identify digitized black-and-white pictures with binary strings. There are many distances defined for binary strings. For example, the Hamming distance and the Euclidean distance. Such distances are sometimes appropriate. For instance, if we take a binary picture, and change a few bits on that picture, then the changed and unchanged pictures have small Hamming or Euclidean distance, and they do look similar. However, this is not always the case. The positive and negative prints of a photo have the largest possible Hamming and Euclidean distance, yet they look similar in our eyes. Also, if we shift a picture one bit to the right, again the Hamming distance may increase by a lot, but the two pictures remain similar. Many approaches to pattern recognition try to define picture similarity. Let us show that the distance E_1 defined above is, in a sense, minimal among all reasonable similarity measures.

A distance measure must be nonnegative for all $x \neq y$, symmetric, and satisfy the triangle inequality. This is not sufficient since a distance measure like $D(x,y) = 1$ for all $x \neq y$ must be excluded. For each x and d, we want only finitely many elements y at a distance d from y. Exactly how fast we want the distances of the strings y from x to go to ∞ is not important: it is only a matter of scaling. For convenience, we will require the following **normalization property**:

$$\sum_y 2^{-D(x,y)} \leq 1 .$$

We consider only distances that are computable in some broad sense. This condition will not be seen as unduly restrictive. As a matter of fact, only upper semicomputability of $D(x,y)$ will be required. This is reasonable: as we have more and more time to process x and y we may discover new and new similarities among them, and thus may revise our upper bound on their distance. The upper semicomputability means exactly that $D(x,y)$ is the limit of a computable sequence of such upper bounds.

A **permissible distance**, $D(x,y)$, is a total nonnegative function on the pairs x,y of binary strings that is 0 only if $x = y$, is symmetric, satisfies the triangle inequality, is semicomputable and normalized.

The following theorem shows that E_1 is, in some sense, the optimal permissible distance. We find it remarkable that this distance happens to also have a "physical" interpretation as the approximate length of the conversion program of theorem 3.1, and, as shown in the next section, of the smallest program that transforms x into y on a reversible machine.

(4.1) Theorem For an appropriate constant c, let $E(x,y) = E_1(x,y) + c$ if $x \neq y$ and 0 otherwise. Then $E(x,y)$ is a permissible distance function that is minimal in the sense that for every permissible distance function $D(x,y)$ we have

$$E(x,y) \stackrel{+}{<} D(x,y) .$$

□

Proof The nonnegativity and symmetry properties are immediate from definition. The addition theorem 2.3 implies that there is a nonnegative integer constant c such that

$$E_1(x,z) \leq E_1(x,y) + E_1(y,z) + c .$$

Let this c be the one used in the statement of the theorem, then $E(x,y)$ satisfies the triangle inequality without an additive constant.

The normalization property as well as the minimality follow from Theorem 2.2. ■

5 Reversible Computations

Reversible models of computation, in which the transition function is 1:1, have been explored especially in connection with the question of the thermodynamic limits of computation. Reversible Turing machines were introduced by Lecerf[14] and independently but much later by Bennett [4]; further results concering them can be found in [5][3][15].

Reversibility of a Turing machine's transition function can be guaranteed by requiring disjointness of the ranges of the quintuples, just as determinism is guaranteed by requiring disjointness of their domains. To assure that the machine's global input:output relation is also 1:1, it is necessary to impose a standard format on the initial and final instantaneous descriptions, in particular requiring that all working storage other than that used for the input and output strings be blank at the beginning and end of the computation. Let $\{\psi_i\}$ be the partial recursive function computed by the i'th such **reversible Turing machine**. As usual, we let $\{\phi_i\}$ denote the partial recursive function computed by the i'th ordinary (in general irreversible) Turing machine. Among the more important properties of reversible Turing machines are the following:

There is a universal reversible machine, i.e. an index u such that for all k and x, $\psi_u(k\dagger x) = k\dagger\psi_k(x)$. (Here $k\dagger$ denotes a self delimiting representation of the index k).

Two irreversible algorithms, one for computing y from x and the other for computing x from y, can be efficiently combined to obtain a reversible algorithm for computing y from x. More formally, for any two indices i and j one can effectively obtain an index k such that, for any strings x and y, if $\phi_i(x) = y$ and $\phi_j(y) = x$, then $\psi_k(x) = y$.

From any index i one may obtain an index k such that ψ_k has the same domain as ϕ_i and, for every x, $\psi_k(x) = \langle x, \phi_i(x) \rangle$. In other words, an arbitrary Turing machine can be simulated by a reversible one which saves a copy of the irreversible machine's input in order to assure a global 1:1 mapping.

The above simulation can be performed rather efficiently. In particular, for any $\epsilon > 0$ one can find a reversible simulating machine which runs in time $O(T^{1+\epsilon})$ and space $O(S \log T)$ compared to the time T and space S of the irreversible machine being simulated.

From any index i one may effectively obtain an index k such that if ϕ_i is 1:1, then $\psi_k = \phi_i$. The reversible Turing machines $\{\psi_k\}$, therefore, provide a Gödel-numbering of all 1:1 partial recursive functions.

The connection with thermodynamics comes from the fact that in principle the only thermodynamically costly computer operations are those that are **logically irreversible**, i.e. operations that map several distinct logical states of the computer onto a common successor, thereby throwing away information about the computer's previous state [12] [4],[9][5]. The thermodynamics of computation is discussed further in section 8. Here we show that the minimal program size for a reversible computer to transform input x into output y is equal within an additive constant to the size of the minimal conversion string p of theorem 3.1.

The theory of reversible minimal program size is conveniently developed using a reversible analog of the universal prefix interpreter U defined in Section 2.

A partial recursive function $F(p, x)$ is called a **reversible prefix interpreter** if

for each p, $F(p, x)$ is 1:1 as a function of x;

for each x the set $\{ p : \exists(y) F(p, x) = y \}$ is a prefix set; and

for each y the set $\{ p : \exists(x) F(p, x) = y \}$ is a prefix set.

Such an F may be thought of as the function computed by a reversible Turing machine which performs a 1:1 mapping on $x \leftrightarrow y$ under the control of a program p which remains on the program tape throughout the computation. Any other work tapes used during the computation are supplied in blank condition at the beginning of the computation and must be left blank at the end of the

computation. The program tape's head begins and ends scanning the leftmost square of the program, which is self-delimiting both for forward computations from each input x as well as for backward computations from each output y.

A **universal reversible prefix interpreter** UR, whose program size is minimal to within an additive constant, can readily be shown to exist, and the **reversible algorithmic entropy** $KR(y|x)$ defined as $\min\{\, l(p) : UR(p,x) = y \,\}$.

In the Section 3, it was shown that for any strings x and y there exists a conversion program p, of length at most logarithmically greater than $E_1(x,y) = \max\{K(y|x), K(x|y)\}$, such that $U(p,0x) = y$ and $U(p,1y) = x$. Here we show that the length of this minimal conversion program is equal within a constant to the length of the minimal *reversible* program for transforming x into y.

(5.1) Theorem

$$KR(y|x) \stackrel{+}{=} \min\{\, l(p) : U(p,0x) = y \text{ and } U(p,1y) = x \,\}.$$

□

Proof This proof is an example of the general technique for combining two irreversible programs, for y from x and for x from y, into a single reversible program for y from x. In this case the two irreversible programs are almost the same, since by theorem 3.1 the minimal conversion program p is both a program for y given $0x$ and a program for x given $1y$. The computation proceeds by several stages as shown in Table 1. To illustrate motions of the head on the self-delimiting program tape, the program p is represented by the string "prog" in the table, with the head position indicated by a caret.

Each of the stages can be accomplished without using any many-to-one operations. For example, appending a zero to the beginning of x in stage 1 is can be undone by changing the zero to a blank. In stage 2, the computation of y from x, which might otherwise involve irreversible steps, is rendered reversible by saving a history, on previously blank tape, of all the information that would have been thrown away. In stage 3, making an extra copy of the output onto blank tape is an intrinsically reversible process, and therefore can be done without writing anything further in the history. Stage 4 exactly undoes the work of stage 2, which is possible because of the history generated in stage 2. Perhaps the most critical stage is stage 7, in which x is computed from y for the sole purpose of generating a history of that computation. Then, after the extra copy of x is reversibly disposed of in step 8 by cancellation (the inverse of copying onto blank tape), stage 9 undoes stage 7, thereby disposing of the history and the remaining copy of x, while producing only the desired output y.

Not only are all its operations reversible, but the computations from x to y in stage 2 and from y to x in stage 7 take place in such a manner as to satisfy the requirements for a reversible prefix interpreter. Hence the minimal irreversible

Stage and Action	Program Tape		Work Tape	
0. Initial configuration	$\hat{\text{p}}\text{rog}$	x		
1. Append 0 to beginning of x	$\dot{\text{p}}\text{rog}$	$0x$		
2. Compute y, saving history	$\text{prog}\dot{}$	y	$(y\|x)$-history	
3. Copy y to blank region	$\text{prog}\dot{}$	y	$(y\|x)$-history	y
4. Undo comp. of y from x	$\dot{\text{p}}\text{rog}$	$0x$		y
5. Remove 0, swap x and y	$\hat{\text{p}}\text{rog}$	y		x
6. Append 1 to y	$\dot{\text{p}}\text{rog}$	$1y$		x
7. Compute x, saving history	$\text{prog}\dot{}$	x	$(x\|y)$-history	x
8. Cancel extra x	$\text{prog}\dot{}$	x	$(x\|y)$-history	
9. Undo comp. of x from y	$\dot{\text{p}}\text{rog}$	$1y$		
10. Remove 1 from y	$\dot{\text{p}}\text{rog}$	y		

Table 1: Combining irreversible computations of y from x and x from y to achieve a reversible computation of y from x.

conversion program p, with constant modification, can be used as a reversible program for UR to compute y from x.

Conversely, the minimal reversible program for y from x, with constant modification, serves as a program for y from x for the ordinary irreversible prefix interpreter U, because reversible prefix interpreters are a subset of ordinary prefix interpreters. This establishes the theorem.

■

We define the **reversible distance** between x and y as

$$E_2(x,y) = KR(y|x) = \min\{\, l(p) : UR(p,x) = y \,\}.$$

As just proved, this is within an additive constant of the size of the minimal conversion program of theorem 3.1. Although it may be logarithmically greater than the optimal distance E_1, it has the intuitive advantage of being the actual length of a concrete program for passing in either direction between x and y. The optimal distance E_1 on the other hand is defined only as the greater of two one-way program sizes, and may not correspond to the length of any two-way translation program.

E_2 may indeed be legitimately called a distance because it is symmetric and obeys the triangle inequality to within an additive constant (which can be removed by the additive renormalization technique described at the end of Section 4).

(5.2) Theorem

$$E_2(x,z) \stackrel{+}{<} E_2(x,y) + E_2(y,z)$$

□

Stage and Action	Program tape	Work Tape	
0. Initial configuration	$\hat{p}$progq$p$rog	x	
1. Compute $(y\|x)$, transcribing pprog.	$\hat{p}$progq$p$rog	y	pprog
2. Space forward to start of qprog.	pprog$\hat{q}$prog	y	pprog
3. Compute $(z\|y)$.	pprog$\hat{q}$prog	z	pprog
4. Cancel extra pprog as head returns.	$\hat{p}$progq$p$rog	z	

Table 2: Reversible execution of concatenated programs for $(y|x)$ and $(z|y)$ to transform x into z.

Proof We will show that, given reversible UR programs p and q, for computing $(y|x)$ and $(z|y)$ respectively, a program of the form spq, where s is a constant supervisory routine, serves to compute z from x reversibly. Because the programs are self-delimiting, no punctuation is needed between them. If this were an ordinary irreversible U computation, the concatenated program spq could be executed in an entirely straightforward manner, first using p to go from x to y, then using q to go from y to z. However, with reversible UR programs, after executing p, the head will be located at the beginning of the program tape, and so will not be ready to begin reading q. It is therefore necessary to remember the length of the first program segment p temporarily, to enable the program head to space forward to the beginning of q, but then cancel this information reversibly when it is no longer needed. A scheme for doing this is shown in Table 2, where the program tape's head position is indicated by a caret. To emphasize that the programs p and q are strings concatenated without any punctuation between them, they are represented respectively in the table by the expressions "pprog" and "qprog", and their concatenation pq by "pprogqprog". ∎

6 The information flux distance

The reversible distance E_2 defined in the previous section, is equal to the length of a "catalytic" program, which allows the interconversion of x and y while remaining unchanged itself. Here we consider noncatalytic reversible computations which consume some information p besides x, and produce some information q besides y. Even though consuming and producing information may seem to be operations of opposite sign, we can define a distance based on the notion of information flow, as the minimal *sum* of amounts of extra information flowing into and out of the computer in the course of the computation transforming x into y

For a function ψ computed on a reversible Turing machine, let

$$E_\psi(x,y) \quad \min\{\, l(p) + l(q) : \psi(\langle x,p\rangle) \quad \langle y,q\rangle \,\}\,.$$

(6.1) Lemma There is a universal (non-self-delimiting) reversible Turing machine ψ_u such that for all functions ψ computed on a reversible Turing machine, we have

$$E_{\psi_u}(x,y) \leq E_\psi(x,y) + c_\psi$$

for all x and y, where c_ψ is a constant which depends on ψ but not on x or y. □

Proof This is a consequence of the existence of universal reversible Turing machines(cf. section 5)[14][4]. ■

We define the **sum distance** as

$$E_3(x,y) = E_{\psi_u}(x,y)\,.$$

(6.2) Theorem

$$E_3(x,y) = K(x|y) + K(y|x) + O(\log E_3(x,y))\,.$$

□

Proof Let us show first the lower bound

$$E_3(x,y) \geq K(y|x) + K(x|y)\,.$$

To compute y from x we must be given a program p to do so to start out with. By definition,

$$K(y|x) \leq l(p) + O(\log(p)).$$

The last term reflects the fact that p is externally delimited, while the minimal program used to define K is self-delimiting and may therefore need to be logarithmically longer. [2]

Assume the computation from x,p ends up with y,q. Since the computation is reversible we can compute x from y,q. Consequently, $K(x|y) \leq l(q) + O(\log(l(q)))$. Let us turn to the upper bound and assume $k_1 = K(x|y) \leq k_2 = K(y|x)$ with $l = k_2 - k_1$. According to Theorem 3.2, there is a string q of length $l + O(\log l)$ such that $K(qx|y) = k_1 + O(\log k_1)$ and $K(y|sx) = k_1 + O(\log k_1)$ We can even assume q to be self-delimiting: the price of this can be included into the $O(\log l)$ term. According to Theorem 3.1 and Theorem 5.1 there is a program p of length $k_1 + O(\log k_1)$ going reversibly between qx and y. Therefore with a constant extra program s, the universal reversible machine will go from (pq,x) to (p,y). And by the above estimates

$$l(pq) + l(p) = 2k_1 + l + O(\log k_2) = k_1 + k_2 + O(\log k_2)\,.$$

[2] I have retained the original definition of E_3, in terms of non self delimiting programs, because my suggested alternative definition in terms of UR machines used catalytic rather than information flow programs, and was thus not in the spirit of this section. CHB

■

Note that all bits supplied in the beginning to the computation, apart from input x, as well as all bits erased at the end of the computation, are *random* bits. This is because we supply and delete only shortest programs, and a shortest program p satisfies $K(p) \geq l(p)$, that is, it is maximally random.

The metrics we have considered can be arranged in increasing order. Here, the relation $\overset{\log}{<}$ means inequality to within an additive $O(\log)$, and $\overset{\log}{=}$ means $\overset{\log}{<}$ and $\overset{\log}{>}$.

$$\begin{aligned} E_1(x,y) &= \max\{K(y|x), K(x|y)\} \\ &\overset{\log}{=} E_2(x,y) = KR(y|x) \overset{+}{=} \min\{l(p) : U(p,0x) = y \text{ and } U(p,1y) = x\} \\ &\overset{\log}{<} K(x|y) + K(y|x) \overset{\log}{=} E_3(x,y) \\ &\overset{\log}{<} 2E_1(x,y)\,. \end{aligned}$$

The sum distance E_3, in other words, can be anywhere between the optimum distance E_1 and twice the optimal distance. The former occurs if one of the conditional entropies $K(y|x)$ and $K(x|y)$ is is zero, the latter if the two conditional entropies are equal.

7 Dimensional properties

In a discrete space with some distance function, the rate of growth of the number of elements in balls of size d can be considered as a kind of "dimension" of the space. The space with distance $E_1(x,y) = \max\{K(x|y), K(y|x)\}$ behaves rather simply from a dimensional point of view.

For a binary string x, let $B_1(d,x)$ be the set of strings y with $E_1(x,y) \leq d$.

(7.1) Theorem We have

$$d - K(d) \stackrel{+}{<} \log \#B_1(d,x) \stackrel{+}{<} d - K(d|x)\,.$$

The same bounds apply to $B_1(d,x) \cap \{y : l(y) = l(x)\}$. ()

Proof The upper bound is immediate from Theorem 2.4. For the lower bound, let $s = 2^{d-K(d)}$, let p_i be the i-th binary string of length $l(x)$. Let us consider all strings $y_i = x \oplus p_i$ where $\oplus$ means bitwise mod 2 addition. The number of such strings y_i is $2^{d-K(d)}$. We clearly have

$$E_1(x,y_i) \stackrel{+}{<} K(i) \stackrel{+}{<} d\,.$$

■

It is interesting that a similar dimension relation holds also for the larger distance $E_2(x,y) = K(y|x) + K(x|y)$.

(7.2) Theorem Let x be a binary string. There is a positive constant c such that for all sufficiently large d, the number of binary strings y with $E_3(x,y) \leq d$ is at most $2^d/d$ and at least $2^d/d^2$. □

Proof The upper bound follows from the previous theorem since $E_3 \geq E_1$. For the lower bound, consider strings y of the form px where p is a self-delimiting program. For all such programs, $K(x|y) \stackrel{+}{<} 0$, since x can be recovered from y. Therefore $E_3(x,y) \stackrel{+}{=} K(y|x) = K(p|x)$. Now just as in the argument of the previous proof, we obtain the lower bound $2^d/d^2$ for the number of such strings p with $K(p|x) \leq d$. ■

For the distance E_3, for the number of strings of length n near a random string x of length n, (i.e. a string with $K(x)$ near n) the picture is a little different from that of distance E_1. In this distance, "tough guys have few neighbors". In particular, a random string x of length n has only about $2^{d/2}$ strings of length n within distance d. The following theorem describes a more general situation.

(7.3) Theorem Let the binary strings x, y have length n. For each x the number of y's such that $E_3(x,y) \leq d$ is 2^α with

$$\alpha = \frac{n + d - K(x)}{2} \pm O(\log n),$$

while $n - K(x) \leq d$. For $n - K(x) \geq d$ we have $\alpha = d \pm O(\log n)$. □

Proof Let $K(x) = n - \delta(n)$. In the remainder of the proof all (in)equalities involving algorithmic entropies hold up to an $O(\log n)$ additional term.

($\geq$) We show that there are at least $2^{(d+\delta(n))/2}$ elements y such that $E(x,y) \leq d$ holds. Let $y = x^* z$ with $l(z) = \delta(n)$ and let x^* be the first program for x which we find by dovetailing all computations on programs of length less than n. We can retrieve z from y using at most $O(\log n)$ bits. There are $2^{\delta(n)}$ different such y's. For each such y we have $K(x|y) = O(1)$, since x can be retrieved from y using x^*. Now suppose we further divide $y = uw$ with $l(u) = l/2$ for an appropriate l and choose u arbitrary. Then, the total number of such y's increases to $2^{\delta(n)+l/2}$.

These choices of y must satisfy $E_3(x,y) \leq d$. Clearly, $K(y|x) \leq \delta(n) + l/2$. Moreover, $K(x|y) \leq l/2$ since we can retrieve x by providing $l/2$ bits. Therefore,

$$K(x|y) + K(y|x) \leq l/2 + \delta(n) + l/2.$$

Since the left hand side has value at most value d, the largest l we can choose is, up to the suppressed additional term $O(\log n)$, given by $l = d - \delta(n)$.

This puts the number of y's such that $E(x,y) \leq d$ at least at $2^{(\delta(n)+d)/2 + O(\log n)}$.

($\leq$) Assume, to the contrary, that there are at least $2^{(d+\delta(n)-2+c}$ elements y such that $E_3(x,y) \leq d$ holds, with c some large constant. Then, for some y,

$$K(y|x) \geq \frac{d+\delta(n)}{2} + c\,. \tag{7.4}$$

By assumption

$$K(x) = n - \delta(n),\ K(y) \leq n\,.$$

By the addition theorem 2.3 we find

$$n + \frac{d-\delta(n)}{2} + c \leq n + K(x|y)\,.$$

But this means that

$$K(x|y) \geq \frac{d-\delta(n)}{2} + c\,, \tag{7.5}$$

which, by Equations (7.4) and (7.5), contradicts $K(x|y) + K(y|x) \leq d$. ■

It follows from our estimates out that in every set of low algorithmic entropy almost all elements are far away from each other in terms of the distance E_3. Here, the algorithmic entropy $K(S)$ of a set is the length of the shortest binary program that enumerates S and then halts.

(7.6) Theorem For a constant c, let S be a set with $\#S = 2^d$ and $K(S) = c \log d$. Almost all pairs of elements $x, y \in S$ have distance $E_1(x,y) \geq d$, up to an additive logarithmic term. □

The proof of this theorem is easy. A similar statement can be proved for the distance of a string x (possibly outside S) to the majority of elements y in S. If $K(x) \geq n$, then for almost all $y \in S$ we have $E_1(x,y) \geq n + d - O(\log dn)$.

8 A thermodynamic potential

Thermodynamics, among other things, deals with the amounts of heat and work ideally required, by the most efficient process, to convert one form of matter to another. For example, at 0 C and atmospheric pressure, it takes 80 calories of heat and no work to convert a gram of ice into water at the same temperature and pressure. From an atomic point of view, the conversion of ice to water at 0 C is a reversible process, in which each melting water molecule gains about 3.8 bits of entropy (representing the approximately $2^{3.8}$ fold increased freedom of motion it has in the liquid sate), while the environment loses 3.8 bits. During this ideal melting process, the entropy of the universe remains constant, because the entropy gain by the ice is compensated by an equal entropy loss by the environment. Perfect compensation takes place only in the limit of slow melting, with an infinitesimal temperature difference between the ice and the water

Rapid melting, *e.g.* when ice is dropped into hot water, is thermodynamically irreversible and inefficient, with the environment (the hot water) losing less entropy than the ice gains, resulting in a net and irredeemable entropy increase for the universe as a whole.

Turning again to ideal reversible processes, the entropy change in going from state X to state Y is an antisymmetric function of X and Y; thus, when water freezes at 0 C by the most efficient process, it gives up 3.8 bits of entropy per molecule to the environment. When more than two states are involved, the entropy changes are transitive: thus the entropy change per molecule of going from ice to water vapor at 0 C (+32.6 bits) plus that for going from vapor to liquid water (−28.8 bits) sum to the entropy change for going from ice to water directly. Because of this antisymmetry and transitivity, entropy can be regarded as a thermodynamic potential or state function, each state has an entropy, and the entropy change in going from state X to state Y by the most efficient process is simply the entropy difference between states X and Y.

Thermodynamic ideas were first successfully applied to computation by Landauer. According to **Landauer's principle** [12, 5, 17, 18, 6] an operation which maps n states onto a common successor state must be accompanied by an entropy increase of $\log_2 n$ bits in other, non-information-bearing degrees of freedom in the computer or its environment. At room temperature, this is equivalent to the production of $kT \ln 2$ (about $7 \cdot 10^{-22}$) calories of waste heat per bit of information discarded.

Landauer's priniciple follows from the fact that such a logically irreversible operation would otherwise be able to decrease the thermodynamic entropy of the computer's data without a compensating entropy increase elsewhere in the universe, thereby violating the second law of thermodynamics.

Converse to Landauer's principle is the fact that when a computer takes a physical *randomizing* step, such as tossing a coin, in which a single logical state passes stochastically into one of n equiprobable successors, that step can, if properly harnessed, be used to remove $\log_2 n$ bits of entropy from the computer's environment. Models have been constructed, obeying the usual conventions of classical, quantum, and thermodynamic thought experiments [12][11] [4] [5] [9] [13] [16] [1] [8] [2] showing both the ability in principle to perform logically reversible computations in a thermodynamically reversible fashion (i.e. with arbitrarily little entropy production), and the ability to harness entropy increases due to data randomization within a computer to reduce correspondingly the entropy of its environment.

In view of the above considerations, it seems reasonable to assign each string x an effective thermodynamic entropy equal to its algorithmic entropy $K(x)$. A computation that erases an n bit random random string would then reduce its entropy by n bits, requiring an entropy increase in the environment of at least n bits, in agreement with Landauer's principle.

Conversely, a randomizing computation that starts with a string of n zeros and produces n random bits has, as its typical result, an algorithmically random

n-bit string x, *i.e.* one for which $K(x) \approx n$. By the converse of Landauer's principle, this randomizing computation is capable of removing up to n bits of entropy from the environment, again in agreement with the identification of the thermodynamic and algorithmic entropy.

What about computations that start with one random string x and end with another y? By the transitivity of entropy changes one is led to say that the thermodynamic cost, *i.e.* the minimal entropy increase in the environment, of a transformation of x into y, should be

$$C'(y|x) = K(x) - K(y),$$

because the transformation of x into y could be thought of as a two-step process in which one first erases x, then allows y to be produced by randomization. By the elementary properties of self delimiting programs, this cost measure is transitive within an additive constant. $C'(y|x)$ as well as a similar antisymmetric measure of the thermodynamic cost of data transformations,

$$C''(y|x) = K(x|y) - K(y|x)$$

were both considered by Zurek [17], who has also pointed out that they are nearly equal (that is, that they differ by at most a logarithmic additive terms). Here we note that $C''(y|x)$ is slightly non-transitive. For example, there exist strings[10] x such that $K(x^*|x) \approx K(K(x))$, where x^* is the minimal program for x. According to the C'' measure, erasing such an x via the intermediate x^* would generate $K(x)$ less entropy than erasing it directly, while for the C' measure the two costs would be equal within an additive constant. Indeed, erasing in two steps would cost only $K(x|x^*) - K(x^*|x) + K(x^*|0) - K(0|x^*) \stackrel{+}{=} K(x) - K(x^*|x)$ while erasing in one step would cost $K(x|0) - K(0|x) = K(x)$.

Subtle differences as the one between C' and C'' pointed out above (and resulting in a slight nontransitivity of C'') depend on detailed assumptions which must be, ultimately, motivated by physics. For instance, if one were to follow Chaitin [7] and define conditional information $K(y|x)$ in terms of the size of a self delimiting program to produce y from x^* (rather than x), joint information would be given directly by

$$K(x,y) = K(x) + K(y|x),$$

rather than by Theorem 2.3 above, and

$$C''(y|x) = C'(y|x)$$

would hold without logarithmic corrections. This alternative is worth considering especially because the resulting algorithmic joint and conditional entropies satisfy equalities which also obtain for the statistical entropy (*i.e.* Gibbs-Shannon entropy defined in terms of probabilities) without logarithmic corrections. This makes it a closer analog of the thermodynamic entropy. Moreover, as discussed by Zurek [18], in a cyclic process of a hypothetical Maxwell demon operated engine involving acquisition of information through measurement,

expansion, and subsequent erasures of the records compressed by reversible computation – the optimal efficiency of the cycle could be assured only by assuming that the relevant minimal programs are already available.

These remarks lead one to consider a more general issue of entropy changes in nonideal computations. Bennett[5] and especially Zurek[18] have considered the thermodynamics of an intelligent demon or engine which has some capacity to analyze and transform data x before erasing it. If the demon erases a random-looking string, such as the digits of π, without taking the trouble to understand it, it will commit a thermodynamically irreversible act, in which the entropy of the data is decreased very little, while the entropy of the environment increases by a full n bits. On the other hand, if the demon recognizes the redundancy in π, it can transform π to an empty string by a reversible computation, and thereby accomplish the erasure at very little thermodynamic cost. More generally, given unlimited time, a demon could approximate the semicomputable function $K(x)$ and so compress a string x to size $K(x)$ before erasing it. But in limited time, the demon will not be able to compress x so much, and will have to generate more entropy to get rid of it. This tradeoff between speed and thermodynamic efficiency is superficially similar to the tradeoff between speed and efficiency for physical processes such as melting, but the functional form of the tradeoff is very different. For typical physical state changes such as melting, the excess entropy produced per molecule goes to zero inversely in the time t allowed for melting to occur. But the time-bounded algorithmic entropy $K_t(x)$, i.e. the size of the smallest program to compute x in time $< t$, in general approaches $K(x)$ only with uncomputable slowness as a function of t and x.

References

[1] P.A. Benioff. Quantum mechanical Hamiltonian models of discrete processes that erase their histories: applications to Turing machines. *Int'l J. Theoret. Physics*, 21:177–202, 1982.

[2] P.A. Benioff. Quantum mechanical hamiltonian models of computers. *Ann. New York Acad. Sci.*, 480:475–486, 1986.

[3] C. H. Bennett. Time/space trade offs for reversible computation. *S I A M. Journal on Computing*, 18,766–776, 1989.

[4] C.H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, 17,525–532, 1973.

[5] C.H. Bennett. The thermodynamics of computation – a review. *Int'l J. Theoret. Physics*, 21:905–940, 1982.

[6] C. M. Caves, W. G. Unruh, and W. H. Zurek. Comment on quantitative limits on the ability of a Maxwell Demon to extract work from heat. *Phys. Rev. Lett.*, 65:1387, 1990.

[7] G. Chaitin. A theory of program size formally identical to information theory. *J. Assoc. Comput. Mach.*, 22:329–340, 1975.

[8] R.P. Feynman. Quantum mechanical computers. *Optics News*, 11:11, 1985.

[9] E. Fredkin and T. Toffoli. Conservative logic. *Int'l J. Theoret. Physics*, 21(3/4):219–253, 1982.

[10] P. Gács. On the symmetry of algorithmic information. *Soviet Math. Doklady*, 15:1477–1480, 1974. Correction, Ibid., 15(1974), 1480.

[11] R.W. Keyes and R. Landauer. Minimal energy dissipation in logic. *IBM J. Res. Develop.*, 14:152–157, 1970.

[12] R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Develop.*, pages 183–191, July 1961.

[13] R. Landauer. *Int. J. Theor. Phys.*, 21:283, 1982.

[14] Yves Lecerf. Machines de Turing reversibles. Recursive insolubilite en $n \in N$ de l'equation $u = \theta^n$ ou θ est un "isomorphism des codes". *Comptes Rendus*, 257:1597–2600, 1963.

[15] R. Y. Levine and A. T. Sherman. A note on Bennett's time-space trade-off for reversible computation. *S.I.A.M. Journal on Computing*, 19:673–677, 1990.

[16] K. Likharev. Classical and quantum limitations on energy consumption on computation. *Int'l J. Theoret. Physics*, 21:311–326, 1982.

[17] W. H. Zurek. Thermodynamic cost of computation, algorithmic complexity and the information metric. *Nature*, 341:119–124, 1989.

[18] W. H. Zurek. Algorithmic randomness and physical entropy. *Phys. Rev.*, A40:4731–4751, 1989

[19] 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. *Russian Math. Surveys*, 25(6):83–124, 1970.