Thermodynamics of Computation and Information Distance

Charles H. Bennett* Péter Gács† Ming Li‡
Paul M.B. Vitányi§ Wojciech H. Zurek¶

Abstract

Intuitively, the minimal information distance between x and y is the length of the shortest program for a universal computer to transform x into y and y into x. This measure will be shown to be, up to a logarithmic additive term, equal to the *maximum* of the conditional Kolmogorov complexities $E_1(x,y) = \max\{K(y|x), K(x|y)\}$.

Any reasonable distance to measure similarity of pictures should be an effectively approximable, symmetric, positive function of x and y satisfying a reasonable normalization condition and obeying the triangle inequality. It turns out that E_1 is minimal up to an additive constant among all such distances. Hence it is a universal 'picture distance', which accounts for any effective similarity between pictures.

A third information distance, based on the idea that one should aim for dissipationless computations, and hence for reversible ones, is given by the length $E_2(x,y) = KR(y|x) = KR(x|y)$ of the shortest reversible program that transforms x into y and y into x on a universal reversible computer. It is shown that also $E_2 = E_1$, up to a logarithmic additive term. It is remarkable that three so differently motivated definitions turn out to define one and the same notion.

Another information distance, E_3, is obtained by minimizing the total amount of information flowing in and out during a reversible computation in which the program is not retained, in other words the number of extra bits (apart from x) that must be irreversibly supplied at the beginning, plus the number of garbage bits (apart from y) that must be irreversibly erased at the end of the computation to obtain a 'clean' y. This distance is within a logarithmic additive term of the sum of the conditional complexities, $E_3(x,y) = K(y|x) + K(x|y)$.

Finally, using the physical theory of reversible computation, the simple difference $K(x) - K(y)$ is shown to be an appropriate (universal, antisymmetric, and transitive) measure of the amount of thermodynamic work required to transform string x into string y by the most efficient process.

*Address: T.J. Watson IBM Research Center, Yorktown Heights, NY 10598, USA. Email: bennetc@watson.ibm.com.

†Comp. Sci. Dept., Boston University, Boston, MA 02215. Email: gacscs.bu.edu Part of this research was done during the author's stay at IBM Watson Research Center. Partially supported by NSF grant CCR-9002614, and by NWO through NFI Project ALADDIN under Contract number NF 62-376 and Scientific Visitor Award B 62-394.

‡Partially supported by NSERC Operating grant OGP-046506. Address: Computer Science Dept, University of Waterloo, Waterloo, Ontario, N2L 3G1 Canada. Email: mli@math.uwaterloo.ca.

§Partially supported by NSERC International Scientific Exchange Award ISE0046203, and by NWO through NFI Project ALADDIN under Contract number NF 62-376. CWI and University of Amsterdam. Address: CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: paulv@cwi.nl.

¶LANL and Santa Fé Institute. Address: Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA. Email: whz@lanl.gov.

25th ACM STOC '93-5/93/CA,USA

1 Introduction

[1] The Kolmogorov complexity, (or algorithmic entropy), $K(x)$ of a string x is the length of the shortest binary program to compute x on a universal computer (such as a universal Turing machine). Intuitively, $K(x)$ represents the minimal amount of information required to generate x by any effective process. The conditional Kolmogorov complexity $K(y|x)$, 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. The functions $K()$ and $K(|)$, 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.

Our goal is to find the most appropriate informational "distance" between two strings, *i.e.* the min-

[1] Part of the results were announced in M. Li and P.M.B. Vitányi, pp. 42-46 In: *IEEE Proc. Physics and Computation Workshop*, 1992.

imal quantity of information sufficient to translate between x and y, generating either string effectively from the other. We first look at the length of the shortest binary program which computes x from y as well as computing y from x. Being shortest, such a program should take advantage of any redundancy between the information required to obtain from x to y and the information required to obtain from y to x. Therefore, we would like to know 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 shorter 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.

Section 3 exhibits a principal result of this paper that, up to logarithmic error terms, the information required to translate between two strings can always be represented in this maximally overlapping way. Namely, let

$$k_1 = K(x|y)\,,\ k_2 = K(y|x)\,,$$
$$l = k_2 - k_1\,.$$

Then there is a string d of length $k_1 + O(\log k_1)$ and a string q of length $l + O(\log l)$ 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. This demonstrates that $E_1 = \max\{K(y|x), K(x|y)\}$ equals the length of the shortest program $\langle d, q\rangle$ to compute x from y and y from x, up to a logarithmic additive term. (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 process of going from x to y 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 [16, 3], 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.

Let us note that the programs for going between independent random x and y can, if one wishes, also 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, at least with respect to some oracle), in analogy with the Slepian-Wolf Theorem of classical information theory [8].

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

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

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

Hence, three very different definitions arising from different concerns turn out to define the same fundamental notion of optimal effective information distance.

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, $E_3(x,y)$, of these amounts of information represents the minimal number of irreversible bit operations in a computation from x to y in which the program is not retained. E_3 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. However, using the program involved in E_1 we both consume it and are left with it at the end of the computation, accounting for $2E_1(x,y)$ irreversible bit operations, which is typically larger than $E_3(x,y)$.

Section 7 compares the dimensional properties of the optimal and sum metrics.

Finally Section 8 considers the problem of defining a thermodynamic entropy *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 Kolmogorov Complexity

Let $l(p)$ denote the length of the binary string p. Let $\#S$ denote the number of elements of set S. We give some definitions and basic properties of Kolmogorov complexity. (Some of us prefer the name 'algorithmic entropy'.) For details and attributions, we refer to [22, 11, 12, 18]. 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 machine** (interpreter) if for each x, $\{p : \exists(y)\ F(p,x) = y\}$ 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 machine how much of p to use. We define the **conditional Kolmogorov complexity**, (the 'self-delimiting' version) $K_F(y|x)$ of y with condition x, with respect to the machine F as the minimal $l(p)$ where the minimum is take over all strings p with $F(p,x) = y$. It is well-known that there is a prefix machine U with the property that for all other prefix machines 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 machine 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 **Kolmogorov complexity**, of x with respect to y. From now on, we will denote by $\stackrel{+}{<}$ an inequality to within an additive constant, and by $\stackrel{\pm}{=}$ the situation when both $\stackrel{+}{<}$ and $\stackrel{+}{>}$ hold.

Let us give a useful characterization of $K(x|y)$. It is easy to see that $K(x|y)$ is an upper semicomputable function with the property $\sum_y 2^{-K(x|y)} \leq 1$. But also, if $f(x,y)$ is an upper semicomputable function with $\sum_y 2^{-f(x,y)} \leq 1$ then $K(y|x) \stackrel{+}{<} f(x,y)$.

Kolmogorov complexity has the following addition property.

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

3 Conversion Programs

We show that $E_1(x,y) = \max\{K(x|y), K(y|x)\}$ equals the length the shortest program for the universal computer to compute x from y and y from x, up to a logarithmic additive term.

(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 p of length k_2 such that both $K(y|x,p)$ and $K(x|y,p)$ 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 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, $i - 1 \leq 2^{k_2+2} - 2$.

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 generating the sequence $e_1, e_2, \ldots$. Procedure generating simultaneously the matchings $M_1, M_2, \ldots$. Procedure generating 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*

$$K(y|qx) = K(qx|y) = k_1 + O(\log k_1).$$

The proof, based on the technique of the above proof, is omitted here.

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 x. 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**:

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

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 the definition. The addition property of complexity 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 the characterization of complexity mentioned in Section 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[16] and independently but much later by Bennett [3, 4]. Further results concering them can be found in [4, 5, 17].

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 [14, 3, 10, 4]. 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 machine U defined in Section 2. A partial recursive function $F(p,x)$ is called a **reversible prefix machine** if

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

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

for each y, $\{\, p : \exists(x) F(p,x) = y \,\}$ 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 machine** UR, whose program size is minimal to within an additive constant, can readily be shown to exist, and the **reversible Kolmogorov complexity** $KR(y|x)$ defined as $\min\{\, l(p) : UR(p,x) = y \,\}$.

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

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

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 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 machine. Hence the minimal irreversible 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 machine U, because reversible prefix machines are a subset of ordinary prefix machines. 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

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

The proof is omitted.

6 Information Flux Distance

The reversible distance E_2 defined in the previous section, is equal to the length of a "catalytic" pro-

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

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

gram, 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

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

It follows from the existence of universal reversible Turing machines mentioned in Section 5 that 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. We define the **sum distance** as

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

(6.1) 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. 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 $K(y|qx) = k_1 + O(\log k_1)$. 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

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

■

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 $\overset{\log}{>}$.

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

$$\stackrel{\log}{<} 2E_1(x,y)\,.$$

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 complexities $K(y|x)$ and $K(x|y)$ is is zero, the latter if the two conditional complexities 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*

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

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

The proof is omitted.

It is interesting that a similar dimension relation holds also for the larger distance $E_3(x,y) = K(y|x) + K(x|y)$. The proof is omitted.

(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 $2^d/d^2$.*

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. Its proof is omitted here.

(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 $\alpha = d \pm O(\log n)$.

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

(7.4) 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 Thermodynamic Cost

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** [14, 4, 20, 21, 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 [14, 13, 3, 4] [10, 15, 19, 1, 9] 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 Kolmogorov complexity $K(x)$. 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 entropy and Kolmogorov complexity.

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

$$W(y|x) = K(x) - K(y) \,,$$

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. This cost is obviously antisymmetric and transitive, but is not even semicomputable, being at best expressible as the *non-monotone* limit of a computable sequence of approximations.

$W(y|x)$ as well as a similar antisymmetric measure of the thermodynamic cost of data transformations,

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

were both considered by Zurek [20], who has also pointed out that they are nearly equal (that is, that they differ by at most a logarithmic additive term). Here we note that $W'(y|x)$ is slightly non-transitive. For example, it is known that there exist strings [11] x of each length such that $K(x^*|x) \approx \log l(x)$, where x^* is the minimal program for x. According to the W' measure, erasing such an x via the intermediate x^* would generate $K(x)$ less entropy than erasing it directly, while for the W 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 W and W' pointed out above (and resulting in a slight nontransitivity of W') depend on detailed assumptions which must be, ultimately, motivated by physics [21].

Bennett[4] and especially Zurek[21] 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 Kolmogorov complexity $K^t(x)$, *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. Logical reversibility of computation. *IBM J. Res. Develop.*, 17:525–532, 1973.

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

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

[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] I. Csiszár and J. Körner. *Information Theory*. Academic Press, New York, 1980.

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

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

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

[12] P. Gács. Lecture Notes on Descriptional Complexity and Randomness Technical Report 87-103, Computer Science Department, Boston University.

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

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

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

[16] 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.

[17] 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.

[18] M. Li and P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer Verlag, New York, 1993.

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

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

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

[22] 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.