

Physics, Information and Computation

Summary of a lecture given by Ziv Nevo on 17 Jun 2000,
for seminar No. 4 in CS by Dr. El-Yaniv.

Based on Chapter 8 in An Introduction to Kolmogorov Complexity
by Li and Vitanyi

1 Introduction

This lecture will cover two subjects. First we relate Shannon's entropy with the expected prefix complexity and with the expected plain Kolmogorov complexity. We then take a look at energy dissipation during computations. We shall introduce reversible computing as a theoretical solution for non-heating computers, and the reversible Turing machine will follow naturally. Finally we'll define the Kolmogorov complexity variant, associated with that machine.

2 Shannon's Entropy and Algorithmic Complexity

We recall that the entropy of a random variable X , with probability $P(X = x)$ for an outcome x is defined as:

$$H(P) = - \sum_x P(X = x) \log P(X = x).$$

We also recall the Noiseless Coding Theorem, stating that any prefix code E , with a minimal codeword length $L(P) = \sum_x P(x)l(E(x))$, satisfies

$$H(P) \leq L(P) \leq H(P) + 1.$$

We now look at a special prefix code E' , defined using our reference prefix machine. We define $E'(x) = x^*$, where x^* is the shortest, self-delimiting reference machine program for x . If the source words x are distributed with probability $P(x)$, then the expected code word length is $\sum_x P(x)l(E'(x)) = \sum_x P(x)K(x)$.

This code is guaranteed to be quite compact, as we take the shortest program for x . However it cannot always be **the** minimal prefix code, as $K(x)$ is fixed for a given x , and cannot "fit" itself to a certain distribution P . Surprisingly, under mild restrictions on P , the expectation of $K(X)$ achieves $H(P)$, up to a small additive constant, as stated by the following theorem.

Theorem 2.1 *Any recursive probability distribution P satisfies*

$$0 \leq \left(\sum_x P(X)K(X) - H(P) \right) \leq c_P,$$

where c_P is a constant depending only on P .

The left hand side of the inequality is immediate from the Noiseless Coding Theorem. The right hand side is proved using the definition of the universal enumerable distribution \mathbf{m} , and its properties. A detailed proof will be given in class.

We now examine how statistical entropy and expected **plain** Kolmogorov complexity are related. We use the popular inequality $C(x) \leq K(x) \leq C(x) + K(C(x))$, to transform Theorem 2.1 into

$$-c_P \leq H(P) - \sum_x P(x)C(x) \leq \sum_x P(x)K(C(x)). \quad (1)$$

We notice that the difference is bounded only if $\sum_x P(x)K(C(x))$ converges. Since it is impossible to bound the difference, we would like to show that the two quantities are asymptotically equal, as stated by the next theorem.

Theorem 2.2 *Let P run through a special sequence of distributions P_1, P_2, \dots , which satisfies $\lim_{i \rightarrow \infty} H(P) = \infty$ and $\lim_{i \rightarrow \infty} K(P)/H(P) = 0$. Then*

$$\lim_{i \rightarrow \infty} \frac{\sum_x P(x)C(x)}{H(P)} = 1.$$

The proof is based on Theorem 2.1, on equation 1, and on the special notation of the limit, as defined above. A full proof will be given in class.

The two theorems in this section, strengthen our intuition, that Kolmogorov complexity (and especially its prefix variation) is tightly related to the amount of information an individual object contains.

3 Reversible Computation

As we all know, computers produce a significant amount of heat, while doing their computations. Absorbing that heat, so the logic units will not melt, becomes more of a problem as circuits shrink, and as clock frequency increases. While some of the heat production can be eliminated using more advanced technologies, some basic rules of physics state that a specific amount of heat must be dissipated during the execution of certain logic operations. In the future we will have to limit our designs, so only non-dissipating operations are involved. Such computations are found to be of a very certain class.

Definition A computation is called *reversible* if its inputs can always be deduced from its outputs.

Around 1960, R. Landauer showed that there is no theoretical reason, why reversible computation should dissipate heat. It is only *irreversible computations*, where information is being lost, that must raise the temperature of their environment. In particular, an erasure of one bit (which is not reversible), can be shown to produce energy, equals to $kT \ln 2$ joules, where k is Boltzmann's constant, and T is the absolute temperature in Kelvin.

Landauer's explains this weird result as follows: suppose we erase n bits. Before the erasure, those bits could be in any of the 2^n possible states. After the erasure, we are left with only one state (no bits). This means we lost degrees of freedom in our physical system. The second law of thermodynamics states, that if the information wasn't transmitted anywhere, then this loss must be compensated for. The compensation must happen in the form of heat dissipation.

We would like therefore to check, whether reversible computation can replace our standard computation models: the gate-array (circuit) and the Turing machine.

3.1 Reversible Logic Circuits

Any Boolean function can be constructed using only AND and NOT gates, which form a universal set of operations. While the NOT gate is clearly reversible, the AND gate is clearly not, as it transforms two inputs into one output.

Lucky enough we can find a universal set of operations, which is made of a single reversible gate. Using that gate we can implement NOT and AND, and therefore can implement any Boolean function. An example for such a gate is the Fredkin gate. Fredkin gate is made of three inputs (Control, A and B) and three outputs (C , D and E). Its logic is described as:

$$(C, D, E) = \begin{cases} (\text{Control}, A, B) & \text{Control} = 0 \\ (\text{Control}, B, A) & \text{Control} = 1 \end{cases}$$

and we can easily see how its inputs can be reconstructed, given its outputs. Construction of AND and NOT, using Fredkin gate is simple, and will be shown in class.

We conclude, that there is no thermodynamic reason, why a computer, built from Fredkin gates only, should dissipate any heat. Any energy that should be put in order to start the computation, can be, in theory, be extracted at the end of it.

3.2 Reversible Turing Machines

In order to define a reversible Turing machine, we first define a variant of our standard model for TM. A Write/Move TM divides its transition relation into two types of quadruples: Write quadruples, where the scanned symbol is replaced, and Move quadruples, where the head is moved, without writing anything. It can be easily shown, that a Write/Move TM is equivalent to our standard model TM. The next step is to define when we cannot trace back our steps.

Definition Two quadruples *overlap in range* if they both cause the machine to enter the same state, and either write the same symbol b , or at least one of them is a move quadruple.

Definition A deterministic Write/Move TM is said to be *reversible* if its quadruples do not overlap in range pairwise.

The above definitions can be easily extended to define a k -tape reversible TM. This extension is needed in order to show that any partial recursive function can be computed by a reversible TM. The simulation proceeds as follows:

- We take the TM that computed the desired function and add one extra tape to the machine. This tape is called the “history tape”.
- We extend the TM’s quadruples to 6-tuples, so the extended machine will write on the “history tape”, the quadruples used by the original machine. Thus we can reversibly undo our computation.
- When computation is done, the simulating machine reversibly copies $f(x)$ to a safe place.
- The machine then undoes the computation from x to $f(x)$, while deleting the history tape.
- Finally when only x is left on the tape, the machine combines x and $f(x)$ into $\langle x, f(x) \rangle$. All other tape space is empty, as required.

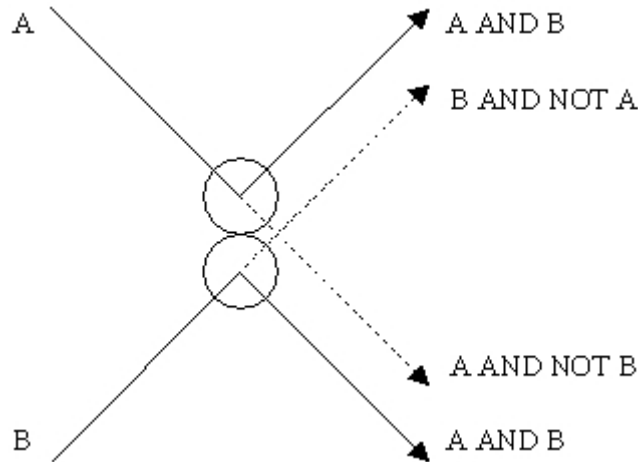


Figure 1: Implementing reversible AND and NOT gates

The manner in which the simulation proceeds, implies that the simulation is rather efficient. Bennet shows that for any $\epsilon > 0$, there is a reversible simulating machine that runs in time $O(t^{1+\epsilon})$ and space $O(s \log t)$, where the original machine runs in time t and space s .

Theorem 3.1 *Let ψ_i be the partial recursive function computed by the i th reversible prefix machine. Then there exists a reversible prefix machine UR, computing ψ_0 , such that for any k and x , we have that $\psi_0(\langle k, x \rangle) = \langle k, \psi_k(x) \rangle$.*

And we can define the Kolmogorov complexity variant for reversible computations.

Definition The *reversible prefix complexity* is $KR(y|x) \equiv \min\{l(p) : UR(p, x) = y\}$

3.3 Reversible Ballistic Computer

This model is rather a thought experiment, showing how reversible computing is possible. This experiment is using billiard balls, which are easier to realize.

We look at an idealized computer, where elastic frictionless billiard balls, represent our bits. The presence of a ball represents a 1 state for the bit, and its absence represents a 0 state. To put a 1 in our input bits, we fire a ball at a predefined speed. To put a 0 in our input, we do nothing. The balls are then moving in straight lines and in constant speed, until they hit another ball, or hit mirrors that were placed in advance. Those collision (elastic of course) represent the calculations we do. Finally we can measure the presence of the balls, at a certain moment, and determine our output bits. We can then use the kinetic energy of the balls, to launch them back, effectively undoing the computation. The balls will return to the place, from where they were initially launched, and another computation can be started. Figure 1 shows how AND and NOT gates can be implemented.

With additional mirrors, we can shift, delay and deflect a ball's path, and thus create an array of gates, which calculates any Boolean function. Since billiard balls tend to have frictional movement, and non-elastic collisions, we can replace those by molecules. However the system then, tends to be rather unstable. Any small error in position or in speed, is quickly multiplied to produce a huge error. Some researchers suggested different models to eliminate those problems. Today's attempts to build a quantum computer are one possible direction.