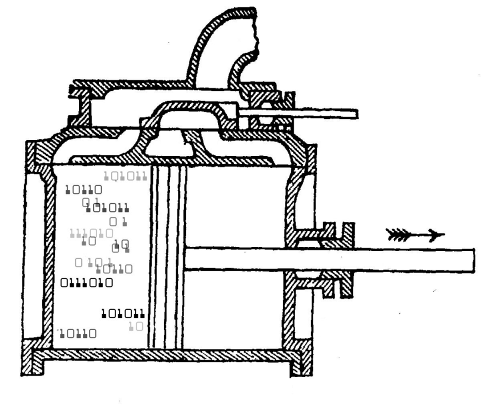
Computation and Thermodynamics



John Baez, U. C. Riverside

Santa Fe Institute, 16 November 2016

Any Turing machine *M* computes some *partially defined* function from $\mathbb{N}$ to $\mathbb{N}$. We write this function as *M*.

A partially defined function from $\mathbb{N}$ to $\mathbb{N}$ is **partial recursive** if it is computed by some Turing machine.

If $f\colon \mathbb{N} \to \mathbb{N}$ is everywhere defined and computed by some Turing machine we call it a **recursive function**.

Church–Turing Thesis. Any function $f\colon \mathbb{N} \to \mathbb{N}$ that is computable by any kind of systematic procedure is recursive.

There are lots of non-recursive functions $f\colon \mathbb{N} \to \mathbb{N}$:

- There must be uncountably many of them.
- Let $H(i) = 0$ if the ith Turing machine does not halt when given the input 0, and let $H(i) = 1$ if it does halt. H is not recursive.
- List statements in your favorite axiomatic system of mathematics. Let $P(i) = 0$ if the ith statement is not provable and $P(i) = 1$ if it is. If your system has
    - finitely many axioms,
    - is at least as powerful as Peano arithmetic, and
    - is consistent

  then P is not recursive.
- Many other explicit examples.

Universal prefix-free Turing machines

Instead of using lots of Turing machines, we can use one ‘universal’ Turing machine. ‘Prefix-free’ Turing machines work best. To define these, think of Turing machines as accepting bit strings rather than natural numbers as inputs.

Let a **string** be a bit string: a finite, possibly empty, list of 0’s and 1’s.

If x and y are strings, let xy be the concatenation of x and y. A **prefix** of a string z is a string x such that $z = xy$ for some y. A **prefix-free** set of strings is one in which no element is a prefix of any other.

If S is a prefix-free set of strings,

$$\sum_{x \in S} 2^{-|x|} < \infty$$

where $|x|$ is the **length** of the string x.

The **domain** of a Turing machine M is the set of strings x for which $M(x)$ is defined. That is, the machine eventually halts when given input x.

A **prefix-free Turing machine** is one whose domain is a prefix-free set.

A prefix-free machine U is **universal** if for any prefix-free machine M there exists a constant c such that for each string x, there exists a string y with

$$U(y) = M(x) \text{ and } |y| < |x| + c.$$

Theorem. There exists a universal prefix-free Turing machine U.

Indeed, there are many, but fix one!

Kolmogorov complexity

The **Kolmogorov complexity** of $n \in \mathbb{N}$ is the length of the shortest string x with $U(x) = n$.

Intuitively, it's the length of the shortest program that prints out n.

We can also talk about the Kolmogorov complexity of a string, since we can encode strings as natural numbers. Indeed we can define the Kolmogorov complexity of any sort of data.

Kolmogorov complexity versus Shannon entropy

Shannon entropy works for *probability distributions on strings*, while Kolmogorov complexity works for *individual strings*.

However, the Kolmogorov complexity of a long randomly produced string is typically close to the Shannon entropy of the probability distribution that gave rise to it!

Let’s make that precise.

Theorem. Suppose we have a probability distribution on k-bit strings:

$$p\colon \{0,1\}^k \to [0,1].$$

Let

$$S(p) = -\sum_{x \in \{0,1\}^k} p(x) \log(p(x))$$

be its Shannon entropy.

Suppose we choose n random strings $x_1, \ldots, x_n$ from this probability distribution. The concatenation $x_1 \cdots x_n$ is a string with Kolmogorov complexity $K(x_1 \cdots x_n)$.

Then with probability 1,

$$\lim_{n \to \infty} \frac{K(x_1 \cdots x_n)}{nS(p)} = 1.$$

The Complexity Barrier

But there's a problem:

Theorem. The Kolmogorov complexity

$$K\colon \mathbb{N} \to \mathbb{N}$$

is not a recursive function.

More surprisingly, there's an upper limit on how complex we can prove anything is!

Theorem. Choose your favorite set of axioms for math. If it's finite and consistent, there exists $C \geq 0$, the **complexity barrier**, such that for no $n \in \mathbb{N}$ can you prove $K(n) > C$.

Levin's time-bounded complexity

What to do? Instead of minimizing the length of a program that prints out n, let's minimize the *length of the program plus the logarithm of its runtime!*

More precisely: if our universal machine U halts when given the input x, let $t(x)$ be its runtime. Define the **Levin complexity** $L(n)$ to be

$$L(n) = \min_{x \text{ such that } U(x)=n} \Big(|x| + \ln t(x) \Big)$$

Theorem. $L\colon \mathbb{N} \to \mathbb{N}$ is recursive.

Algorithmic thermodynamics

Now let's unify these notions of complexity — and unify them with statistical mechanics!

Let X be the domain of our universal prefix-free Turing machine: the set of strings x for which $U(x)$ is defined.

Define the **partition function**

$$Z(\beta, \gamma) = \sum_{x \in X} e^{-\beta \ln(t(x)) - \gamma |x|}$$

Theorem. The sum for Z converges when $\beta \geq 0$ and $\gamma \geq \ln 2$. If also $\beta > 0$, Z is computable to arbitrary accuracy. For $\beta = 0$ it is not computable.

As usual in statistical mechanics, the probability distribution

$$p(x) = \frac{e^{-\beta \ln(t(x)) - \gamma |x|}}{Z}$$

maximizes entropy subject to a constraint on the expected values of the log runtime $\ln(t(x))$ and input length $|x|$.

$$\sum_{x \text{ such that } U(x)=n} p(x)$$

is the probability that a randomly chosen input will cause our universal machine U to print out n.

Intuitively,

$$\mathrm{Surprise}(n) = -\ln \left(\sum_{x \text{ such that } U(x)=n} p(x) \right)$$

is the **surprise** we should experience when a randomly chosen input causes U to print out n.

Theorem. There is a constant $C > 0$ such that for any $n \in \mathbb{N}$, the Kolmogorov complexity $K(n)$ obeys

$$|K(n) - \mathrm{Surprise}(n)| < C$$

when $\beta = 0$, $\gamma = \ln 2$, and the Levin complexity $L(n)$ obeys

$$|L(n) - \mathrm{Surprise}(n)| < C$$

when $\beta = 1$, $\gamma = \ln 2$.

We can go ahead and do “algorithmic thermodynamics”. If we write the expected log runtime and input length as

$$E = \langle \ln t(x) \rangle, \qquad V = \langle |x| \rangle$$

then

$$E = -\frac{\partial}{\partial \beta} \ln Z, \qquad V = -\frac{\partial}{\partial \gamma} \ln Z$$

as usual.

If we define the **algorithmic temperature** T and **algorithmic pressure** P by

$$\frac{1}{T} = \beta, \qquad \frac{P}{T} = \gamma$$

then we get Maxwell’s relations, etc. For details, see:

- John Baez and Mike Stay, Algorithmic thermodynamics.