The Halting Problem Revisited

Cristian S. Calude
The University of Auckland, NZ
Isaac Newton Institute of Mathematical Science
Cambridge, UK

The Incomputable, Chicheley Hall, 14 June 2012

The halting problem for Turing machines cannot be solved by any Turing machine

Classical proofs use diagonalisation which seems artificial as the argument:

- looks like a linguistic trick,
- does not reveal “the cause” of the impossibility.

An information-theoretical argument

Assume that:

- we interest ourselves to (Turing) machines working with natural numbers as inputs and outputs;
- there exists a halting machine **HALT** which solves the halting problem for the above class of machines.

Construct the machine Trouble(N):

1. `read a natural` N;
2. `generate all machines and inputs` (T, n) `of up to` N `bits in size;`
3. `use` **HALT** `to remove all pairs` (T, n) `for which` T `does not stop on` n;
4. `run the remaining computations` $T(n)$ `till they stop;`
5. `compute the largest value` o `output by these machines and output` $2o + 1$.

Trouble(N) is in trouble

1. Trouble(N) **halts** for every N.
2. The size in bits of Trouble(N) is about $\log N$ plus a constant.
3. For large enough N, Trouble(N) has less than N bits in size.
4. For large enough N, Trouble(N) *generates itself* at some stage of the computation: by examining the output, we get a contradiction.

Coding the halting problem

Assume U is a universal prefix-free Turing machine.

The sequence

$$h(n) = \begin{cases} 1, & \text{if } U \text{ halts on the } n\text{th program}, \\ 0, & \text{otherwise}, \end{cases}$$

codes the halting problem for U and has the following properties:

- h is incomputable;
- the quantity of information in $h(\restriction n) = h(1)h(2)\cdots h(n)$ is about $\log n$, hence infinitely many bits of h can be computed.

Coding the halting problem

The sequence of bits of

$$\Omega_U = \sum_{U(p) \text{ stops}} 2^{-|p|} = 0.\omega_1\omega_2\cdots$$

codes the halting problem for U and has the following properties:

- the quantity of information in $\Omega_U(\upharpoonright n) = \omega_1\omega_2\cdots\omega_n$ is about n;
- Ω_U is bi-immune, i.e. no Turing machine can compute more than finitely many scattered bits of Ω_U.

For every universal prefix-free Turing machine U and natural $N > 0$, we can effectively construct another universal prefix-free Turing machine W such that:

1. $\Omega_W = \Omega_U$,
2. given W, ZFC can compute at most $N - 1$ bits of Ω_U, where the first bit of Ω_U equal to zero appears on the position N;if $\Omega_U < 1/2$, then ZFC cannot calculate any bit of Ω_U.

A probabilistic solution for the halting problem

Assume that U is a universal prefix-free Turing machine. We can effectively calculate a stopping time $s = s_U$ such that if $U(x)$ halts in time $t > s_U$, then t is not algorithmically random.

There exists a Turing machine which stops on every input (T, x), where T is a prefix-free Turing machine and x is an input, and outputs either:

- "T halts on x", and in this case the result is correct, or
- "T does not halt on x", and in this case the result may be wrong, but with probability less than an arbitrarily small number.

1. C. S. Calude. Chaitin Ω numbers, Solovay machines and incompleteness, *Theoret. Comput. Sci.* 284 (2002), 269–277.
2. C. S. Calude, G. J. Chaitin. What is . . . a halting probability? *Notices of the AMS* 57, 2 (2010), 236–237.
3. C. S. Calude, N. J. Hay, F. Stephan. Representation of left-computable ε–random reals, *Journal of Computer and System Sciences*, 77 (2011), 812–819.
4. C. S. Calude, M. A. Stay. Most programs stop quickly or never halt, *Advances in Applied Mathematics*, 40 (2008), 295–308.
5. R. M. Solovay. A version of Ω for which ZFC can not predict a single bit, in *Finite Versus Infinite. Contributions to an Eternal Dilemma*, C.S. Calude, G. Păun (eds.), pp. Springer-Verlag, London, 2000, 323–334.