CMS Notes
Volume 45 No. 2, March-April 2013

The Halting Problem

Cristian S. Calude
The University of Auckland, New Zealand

The *Entscheidungsproblem*, posed in 1928 by D. Hilbert, asks for an algorithm that takes as input a statement of a first-order logic and outputs 1 or 0 according to whether the statement is universally valid (i.e. in every structure satisfying the axioms of the logic) or not. By the completeness theorem of first-order logic, a statement is universally valid if and only if it can be deduced from the axioms of the logic, so solving the Entscheidungsproblem means finding an algorithm to decide whether a given statement is provable in the first-order logic.

Church [7] and Turing [9] published, independently, different negative solutions to the Entscheidungsproblem. In his proof, Turing introduced a computing machine, now called "Turing machine" (shortly, machine), as a mathematical model for the informal notion of algorithm, and showed how it can be programmed. He then constructed a *universal machine* which can simulate the execution of any machine; in a sense, the universal machine is the blueprint of the modern computer.

Page 247 of Turing's paper is a proof that a certain problem—now called (after M. Davis) *the halting problem* (for Turing machines)—cannot be solved by any machine. The halting problem asks for a machine $Halt$ that takes as input an arbitrary machine M and an input x, and outputs 1 or 0 according to whether M will come to a halt or not on x. We assume that a machine incorporates its input and a halting machine outputs a bit-string. As machines can be systematically enumerated (say, lexicographically) and $Halt$ itself always halts, Turing's proof follows by diagonalisation. The proof below gives also a "reason" for unsolvability. For convenience we assume from now on that all programs include their input data. Suppose, by absurdity, the existence of a machine $Halt$. Construct the following machine $T(N)$ (T from "trouble"):

> read a positive integer N in binary; list all machines up to N bits in size; use $Halt$ to remove from the list all machines which do not halt; simulate the running of all machines on the list; output the largest result computed by these machines plus 1.

For every N, the machine $T(N)$ halts. Its size is less than $\log_2 N + \mathrm{O}(1)$ bits, which is smaller than N, for large N. Accordingly, $T(N)$ generates itself at some stage of its computation, and a simple analysis of the output of $T(N)$ leads to a contradiction. Unsolvability is a consequence of coding scarcity.

In modern terms, machines are programs written in a programming language. In this framework, a universal machine is a universal programming language L, i.e. a language such that if any other language L' can program a machine in K bits, then L can do it in less than $K + c_{L,L'}$ bits, where $c_{L,L'}$ is a fixed constant. A universal programming language L can be constructed with the following five instructions: $=\mathtt{r1},\mathtt{r2},\mathtt{r3}$ (branching instruction), $\&\mathtt{r1},\mathtt{r2}$ (assigning instruction), $+\mathtt{r1},\mathtt{r2}$ (sum), $!\mathtt{r1}$ (read one bit), $\%$ (halt). Registers $\mathtt{r1},\mathtt{r2},\ldots$ can contain arbitrary large non-negative integers. A program consists of a finite list of labeled instructions from the above list, with the restriction that the halt instruction appears only once, as the last instruction. The input data is a bit-string which follows immediately after the halt instruction. The result of the computation, if any, is a non-negative integer stored in a fixed output register. A program not reading the whole data or attempting to read past the last input bit results in an error.

Systematically enumerate all programs (say, lexicographically) and define the real number H whose Nth bit h_N tells us whether or not the Nth program halts. Clearly, H is incomputable, i.e. the function $N \mapsto h_N$ is not computable by any program. This "coding" of the halting problem is rather wasteful, because N instances of the halting problem have only $\log_2 N$ bits of mathematical information: one only needs to know *how many* of these N programs halt to be able to determine which ones halt. One can obtain a more compact coding using the halting probability (or Chaitin's Ω) which is defined by the formula:

$$\Omega = \sum_{p \text{ halts}} 2^{-(\text{size of } p \text{ in bits})};$$

Ω is a probability because, due to the syntax of L, if p halts, then no prefix or extension of p halts. There are infinitely many halting probabilities, one for each universal programming language.

All Ω numbers share a few remarkable properties. Like H, Ω "codes" the halting problem, but in a more efficient way because if one "knows" the first N bits of Ω, then one can solve the halting problem for all programs of size smaller than $N + 1$: $2^{N+1} - 1$ instances of the halting problem are coded into N bits. While H is incomputable, there are programs computing infinitely many values h_N. No program can do a similar computation for Ω; in fact, Ω is strongly incomputable, that is, there is a positive integer Ω such that any program can compute at most ω values of digits of Ω (ω, which depends on the programming language, can be any non-negative integer). Any Ω has two apparently contradictory properties: computable enumerability, i.e. Ω is the limit of a computable sequence of rationals in the unit interval, and incompressibility (also called algorithmic randomness), i.e. the smallest program for computing the first N bits of Ω has at least $N - \mathrm{O}(1)$ bits. The converse result is also true: every real satisfying the above two conditions is an Ω number for some universal programming language. Finally, Ω is transcendental and normal. Two of the above properties can be expressed in terms of mathematical provability. Assume ZFC is sound. Strong incomputability implies unprovability: ZFC cannot prove more than Ω values of the bits of Ω (a quantum random generator can

© Canadian Mathematical Society 2013. All rights reserved.

behave in a similar manner, [1]). Incompressibility implies logical irreducibility: Any sound extension of ZFC that can prove the values of the first N bits of Ω must have at least $N - O(1)$ bits of axioms. See more in [2].

An extensive simulation, whose correctness was mathematically proved, solved the halting problem for all programs in L of up to 80 bits in size and calculated the first 40 bits of $\Omega = \Omega_L$ [3]: 0001000000010000101001111011 1000011111010.

Because the Riemann hypothesis can be written in the form $\forall n P(n)$, where P is a computable predicated, programs in L can systematically search for a counter-example; such a program RH stops if and only if the conjecture is false. Hence, if we knew the first 2,745 bits—the length of such an RH—of Ω_L we would know whether the Riemann hypothesis is true [6]. Searching for a counter-example for the conjecture $P \neq NP$—which is of the form $\forall n \exists m R(n, m)$, where R is a computable predicate—is more difficult; the solution is to use the *same* programs in L, but with a different semantics. A classical computation produces a result only in case the computation stops; the result is then recorded in a special output register. An *inductive* program P produces the same results as the classical program P, but sometimes a result is obtained when P runs an infinite computation (of course, not all inductive computations produce results). In [4] an inductive program PNP of 6,495 bits was constructed to return 0 if and only if $P \neq NP$; PNP is more complex than RH not only because it is longer, but also because it's solution needs a programming language with higher computational power. Indeed, there is an inductive program in L which solves the halting problem for all classical programs in L. However, no inductive program can solve the halting problem for all inductive programs.

While deterministically unsolvable, the halting problem can be probabilistically solved [5, 8]. Fix a rational ε in (0,1). One can effectively construct a program which stops on every program p (as input) and outputs either: a) p *halts*, and in this case the result is correct, or b) p *does not halt*, and in this case the probability that the result is wrong is less than ε.

CMS SMC
Use Social Media? So do we!
Aimez la SMC sur Facebook

CMS SMC
Les bourses, ça vous intéresse? Nous aussi!
Cliquez **http://smc.math.ca/Bourses/Moscou/**

References

[1] A. A. Abbott, C.S. Calude, J. Conder, K. Svozil. Strong Kochen-Specker theorem and incomputability of quantum randomness, *Physical Review A* **86**, 6 (2012), DOI: 10.1103/PhysRevA.00.002100.

[2] C. S. Calude. *Information and Randomness: An Algorithmic Perspective*, Springer, Berlin, 2002, 2nd ed.

[3] C. S. Calude, M. J. Dinneen. Exact approximations of omega numbers, *Int. J. Bifurcat. Chaos* **17**, 6 (2007), 1937–1954.

[4] C. S. Calude, E. Calude, M. S. Queen. Inductive complexity of P versus NP problem. Extended abstract, *Proc. UCNC 2012*, LNCS 7445, Springer, (2012), 2–9.

[5] C. S. Calude, M. A. Stay. Most programs stop quickly or never halt, *Adv. Appl. Math.* **40** (2008), 295–308.

[6] E. Calude. The complexity of Riemann's Hypothesis, *Mult-Valued Log. S.* **18**, 3-4 (2012), 257–265.

[7] A. Church. An unsolvable problem of elementary number theory, *Am. J. Math.* **58** (1936), 345–363.

[8] Yu. Manin. Infinities in quantum field theory and in classical computing: renormalization program, *Proc. CiE 2010*, LNCS 6158, Springer, Heidelberg, 2010, 307–316.

[9] A. M. Turing. On computable numbers with an application to the Entscheidungsproblem, *Proc. Lond. Math. Soc.* ser. 2, **42** (1936), 230–265; correction: ser. 2, **43** (1937), 544–546.

2013 CMS MEMBERSHIP RENEWALS

RENOUVELLEMENTS 2013 À LA SMC

CMS SMC

REMINDER: Your membership reminder notices have been e-mailed. Please renew your membership as soon as possible. You may also renew on-line by visiting our website at **www.cms.math.ca/forms/member**

RAPPEL : Les avis de renouvellements ont été envoyés électroniquement. Veuillez s-il-vous-plaît renouveler votre adhésion le plus tôt possible. Vous pouvez aussi renouveler au site Web **www.cms.math.ca/forms/member?fr=1**

© Société mathématique du Canada 2013. Tous droits réservés.