Super Turing-Machines

B. Jack Copeland

1. Universal Turing Machines

A universal Turing machine is an idealised computing device consisting of a read/write head and a (potentially infinite) paper tape which passes through the head (figure 1).

FIGURE 1 ABOUT HERE

The tape is divided into squares, each square bearing a single symbol—'0' or '1', for example. This tape is the machine's general-purpose storage medium: the machine is set in motion with its input inscribed on the tape, output is written onto the tape by the head, and the tape serves as a short-term working memory for the results of intermediate steps of the computation. The program governing the particular computation that the machine is to perform is also stored on the tape. A small, fixed program that is 'hard-wired' into the head enables the head to read and execute the instructions of whatever program is on the tape. The machine's atomic operations are very simple—for example, 'move left one square', 'move right one square', 'identify the symbol currently beneath the head', 'write 1 on the square that is beneath the head', and 'write 0 on the square that is beneath the head'. Complexity of operation is achieved by the chaining together of large numbers of these simple atoms. Any universal Turing machine can be programmed to carry out any calculation that can be performed by a human mathematician working with paper and pencil in accordance with some algorithmic method. This is what is meant by calling these machines 'universal'.

Turing thought up his abstract computer in 1935 while pursuing research in mathematical logic which nobody could have guessed would lead to a practical application. A dozen years later the first stored-program electronic digital computers began to spring into existence. All were modelled on the universal Turing machine. Today's digital computers also are in essence universal Turing machines.

2. Is There a Known Upper Bound to Computability?

Many textbooks on the fundamentals of computer science offer examples of information-processing tasks that are, it is claimed, absolutely uncomputable, in the sense that no machine can be specified to carry out these tasks. For example, it is said that no machine can repond to any given (finite) string of binary digits in accordance with the following rules:

(1) Answer '1' if the string is a program that will cause a universal Turing machine on whose tape it is inscribed to execute only a *finite* number of operations (such programs are called 'terminating').
(2) Answer '0' if the string is not a terminating program; i.e. if the string is either not a well-formed Turing-machine program or is a well-formed program that does not terminate.
I will refer to this task as the terminating program test or TP test.

It is false that *no* conceivable machine can carry out the TP test. An AUTM, or Accelerating Universal Turing Machine, can carry out this task [1, 2, 3]. An AUTM executes the program on its tape at an accelerating rate, performing each atomic operation that the program calls for in half the time that was taken for the immediately preceding atomic operation. So if the machine takes one unit of time to perform the first atomic operation that the program calls for, the second is performed in half a unit, the third in one quarter of a unit, and so on. Since

$$1 + 1/2 + 1/4 + 1/8 + ... + 1/2^{n-1} + ...$$

is less than 2, the AUTM requires less than two units of running time to do everything that the program on its tape instructs it to do. This is true even in the case of a program that does not terminate, for example a program that runs on forever calculating each successive digit of π. Each of the infinite number of operations that the nonterminating program instructs the machine to perform will be completed before the end of the second unit of running time.

Figure 2 shows the tape of an AUTM that is set up to perform the TP test.

FIGURE 2 ABOUT HERE

Once the Start button has been pressed, the machine (under the control of the task program) determines whether or not the string being tested is a well-formed Turing-machine program. If the string is not a well-formed program then the task program simply shuts the machine down. If the string is a well-formed program then the task program hands over control to it and the machine carries out the instructions encoded in the string. If the string is a terminating program then control eventually returns to the task program, which directs the head to move to the initial square of the tape and to replace the 0 that was written there during the setting-up procedure by 1. The machine then shuts down. If, however, the string is a nonterminating program then the head never returns to the start of the tape.

Either way, at the end of the second unit of operating time the initial square contains the digit required by the above rules.

There is, of course, more than a whiff of unreality about a machine that performs infinitely many operations in a finite span of time. I will call a machine *finitely-operating* if it delivers each of its answers after only a finite number of atomic operations. The claim that *no* machine can carry out the TP test is evidently false but is this weaker form true: no finitely-operating machine can carry out the TP test? It is certainly the case that no finitely-operating universal Turing machine can perform the TP test—Turing proved this in 1936 [4]. Perhaps the finitely-operating universal Turing machines form an upper bound to the class of *realistic* machines, in the sense that every information-processing task that can be carried out by any sort of realistic machine can also be carried out by a finitely-operating universal Turing machine? The latter suggestion is sometimes called 'the Church-Turing thesis'. Unfortunately so, since neither Turing nor Church appears to have embraced it [5]. In fact, there is no reason to think that this thesis is true; and—to return to the previous question—it is possible to specify a finitely-operating machine that can carry out the TP test.

3. Turing's Other Machines

In his PhD thesis (which was supervised by Church) Turing introduced the idea of machines able to perform tasks that cannot be performed by any finitely-operating universal Turing machine [6]. He described these as 'a new kind of machine' and called them 'O-machines'.

An O-machine is a universal Turing machine augmented with a 'black box'. The box carries out some information-processing task that cannot be done by a finitely-operating universal Turing machine, for example the TP test. Turing refers to the black box as an 'oracle'. As in the case of a universal Turing machine, the behaviour of an O-machine is determined by a program inscribed on the machine's tape. Sometimes an instruction in the program causes some string of symbols on the tape to be passed to the black box, which returns the answer '0' or '1' as appropriate. This process of 'querying the oracle' counts as one of the O-machine's atomic operations. Some O-machines are finitely-operating, so the class of finitely-operating information-processing machines is a superset of the class of finitely-operating universal Turing machines.

Turing gave no indication of how one of these logically specified black boxes might conceivably be implemented, saying only that an oracle works by 'unspecified means' and that 'we shall not go any further into the nature of [an] oracle'. (Equally, his earlier work provided no indication of how the atomic operations of a universal Turing machine—for example 'identify the symbol currently beneath the head'—might conceivably be implemented.) In fact, finitely-operating machinery that discharges the task of an oracle is not hard to concoct in the abstract. Consider this modified form of the TP task. Instead of thinking of the machine as responding in accordance with the above rules to a string of binary digits, think of it as responding to an integer. Since every finite binary string corresponds to an integer, and every integer corresponds to a finite binary string, the two formulations are equivalent. It is convenient to write 'a_n' to represent the correct answer to the TP task for any given integer n. a_n is always 0 or 1, of course. Now consider the following decimal specification of a number: $0{\cdot}a_1a_2a_3 \ldots$ [7]. Call this number 'τ'. Like π, τ is a definite—irrational—number. Perhaps the first few digits of τ are $0{\cdot}000000011 \ldots$ The numerical magnitude of some physical quantity might conceivably be exactly τ units [8]. Suppose that some device A does store exactly τ units of such a physical quantity, which for the sake of vividness one might call 'charge'. Suppose further that a mechanism B can measure the quantity of 'charge' stored in A to any specified number of significant figures. A and B jointly perform the function of the TP-task oracle. Given an integer n, B determines a_n in a finite number of steps by measuring A's charge to n significant figures and outputting the rightmost digit of the result.

A TP-task oracle designed in this way would not work very well in practice, because once n becomes very large, random noise would obstruct B's efforts to determine a_n accurately. No one yet knows whether it is possible to produce a realistic design for an oracle. But the search is on for a physically realizable architecture that is capable in the limit of computing more than a finitely-operating universal Turing machine—a 'super Turing-machine' [9, 10, 11].

REFERENCES

1. Stewart, I. 1991. 'Deciding the Undecidable'. *Nature*, 352, 664-5.

2. Copeland, B.J. 1998. 'Even Turing Machines Can Compute Uncomputable Functions'. In Calude, C.S., Casti, J., Dinneen, M.J. (eds) 1998, *Unconventional Models of Computation*, Singapore: Springer-Verlag.

3. Hamkins, J.D., Lewis, A. 1998. 'Infinite Time Turing Machines'. *Journal of Symbolic Logic* (forthcoming).

4. Turing, A.M. 1936. 'On Computable Numbers, with an Application to the Entscheidungsproblem'. *Proceedings of the London Mathematical Society*, Series 2, 42 (1936-37), 230-265.

5. Copeland, B.J. 1996. 'The Church-Turing Thesis'. In Perry, J., Zalta, E. (eds) *The Stanford Encyclopaedia of Philosophy* [http://plato.stanford.edu].

6. Turing, A.M. 1939. 'Systems of Logic Based on Ordinals'. *Proceedings of the London Mathematical Society*, 45, 161-228.

7. Chaitin, G.J. 1988. 'Randomness in Arithmetic'. *Scientific American*, 259, 80-85. (Chaitin defines a number Ω that is analogous to, but not the same as, τ.)

8. Copeland, B.J. 1997. 'The Broad Conception of Computation'. *American Behavioral Scientist*, 40, 690-716.

9. Copeland, B.J., Sylvan, R. 1998. 'Beyond the Universal Turing Machine'. *Australasian Journal of Philosophy* (forthcoming).

10. Siegelmann, H.T. 1995. 'Computation Beyond the Turing Limit'. *Science*, 268, 545-548.

11. Stannett, M. 1990. 'X-Machines and the Halting Problem: Building a Super-Turing Machine'. *Formal Aspects of Computing*, 2, 331-341.