Jack Copeland - Computable Numbers: A Guide

Tom Rochette <tom.rochette@coreteks.org>

December 21, 2025 — 77e1b28a

0.1 Context

0.2 Learned in this study

0.3 Things to explore

1 Overview

2 Notes

2.1 1. Turing Machines

- A scanner and a limitless memory-tape
- Atomic operations: erase, print, move and change state

2.2 2. Standard Descriptions and Description Numbers

2.2.1 Standard Description

- Single words of the form $\mathbf{q}_i S_j S_k M \mathbf{q}_l$
  - $\mathbf{q}_i$: Current state
  - S_j: Scanned symbol
  - S_k: Printed symbol
  - M: Direction of movement of the scanner
  - $\mathbf{q}_l$: Next state
- Read and printed symbols: Replace letters and numbers by D followed by n repetitions of C
  - -: D
  - 0: DC
  - 1: DCC
  - 2: DCCC
- State: Replace letters and numbers by D followed by n repetitions of A
  - a: DA
  - b: DAA
  - c: DAAA
- a-0Rb; b–Rc; c-1Rd; d–Ra;
- aDDCRb; bDDRc; cDDCCRd; dDDRa;

a-0Rb;	b–Rc;	c-1Rd;	d–Ra;
DADDCRDAA;	DAADDRDAAA;	DAAADDCCRDAAAA;	DAAAADDRDA;

- D is used to mark the beginning of a segment

2.2.2 Description Number

- A standard description can be converted into a description number

A	C	D	L	R	N	;
1	2	3	4	5	6	7

a-0Rb;	b–Rc;	c-1Rd;	d–Ra;
DADDCRDAA; 3133253117	DAADDRDAAA; 31133531117	DAAADDCCRDAAAA; 311133225311117	DAAAADDRDA; 3111335317

2.3 3. Subroutines

- Programmes used as components of other programmes

2.4 4. The Universal Computing Machine

- Control the function of a computing machine by storing a programme of symbolically encoded instructions in the machine's memory
- A single machine of fixed structure is able to carry out every computation that can be carried out by any Turing machine whatsoever, i.e. is universal
- Write a description that explains what the machine should do in every configuration in which it might find itself

2.5 5. Turing, von Neumann, and the Computer

- A *practical* programming code should not only be universal, but must in addition:
  - employ basic operations that can be realized simply, reliably, and efficiently by electronic means
  - enable the "actually important problems" to be solved on the machine as rapidly as the eletronic hardware permits
  - be as easy as possible for the human "problem planner" to work with

2.6 6. Turing and Babbage

- Differential Engine to produce mathematical tables
- Analytical Engine with memory and a central processing unit
  - Would have had conditional branching (select alternative actions based on the previous actions)
- The Analytical Engine was universal

2.7 7. Origins of the Term "Computer Programme"

- Programme: a planned sequence of events

2.8 8. Circular and Cirle-Free Machines

- 0/1 are symbols "of the first kind"
- 2, *, x, blank, etc. are symbols "of the second kind"
- A computing machine is said by Turing to be *circular* if it never prints more than a finite number of symbols of the first kind.
- A computing machine that will print an infinite number of symbols of the first kind is said to be *circle-free*

2.9 9. Computable and Uncomputable Sequences

- A sequence of binary digits is said to be a *computable* sequence if it is the sequence computed by some circle-free computing machine.
- 010 is NOT a computable sequence
- By definition, no finite sequence is a computable sequence
- Modern writers usually define "computable" in such a way that every finite sequence is a computable sequence (since each of them can be computed)
- Not every infinite sequence of binary digits is a computable sequence (through the diagonal argument)
- The diagonal argument basically states that the set of real numbers cannot be mapped to the set of integers (in other words, there are more real numbers than there are integers)

2.10 10. Computable and Uncomputable Numbers

- B: the sequence of binary digits printed by a given computing machine
- 0.B: the number computed by the machine
- Circular machines always compute rational numbers
- Circle-free machines may compute an irrational number (pi for example)
- A number computed by a circle-free machine is said to be a *computable number*
- π and e are computable
- Not all real numbers are computable
  - If S is an infinite binary sequence that is uncomputable, then 0.S is an uncomputable number

2.11 11. The Satisfactoriness Problem

- A standard description is said to be satisfactory if the machine it describes is circle-free.
- A number is satisfactory if it is a description number of a circle-free machine.
- A number is unsatisfactory if either it is a description number of a circular machine or it is not a description number at all.
- The satisfactoriness problem is to decide for any arbitrarily selected standard description or equivalent description number, whether or not it is satisfactory. This decision must be made in a finite number of steps.
- It is not possible to produce a computing machine **H** such that **H** can compute the successive digits of the diagonal sequence (the sequence of circle-free machines descriptions).
- What happens when **H** encounters a number that describes **H** itself (called the description number **K**)?
  - **H** must check if **K** is a description number
  - **H** must test if **K** is satisfactory
  - Since **H** is supposed to calculate the endless binary sequence β', **H** must be circle-free
  - **H** must say that **K** is satisfactory
  - At some point, **H** will have simulated itself through **K** which will have then to simulate **H** itself once again (thus recursively simulating itself forever)
- The result of this is that no computing machine can solve the satisfactoriness problem

2.12 12. The Printing and Halting Problems

2.12.1 The printing problem

Determine whether a standard description or equivalent description number will ever print a certain symbol ('0' for example).

Turing proves that if the printing problem were solvable by some computing machine, then the satisfactoriness problem would be too. Therefore neither is.

2.12.2 The halting problem

Determine whether a standard description or equivalent description number will eventually halt, i.e. stop moving (being done with computation).

Also known as the halting theorem.

2.13 13. The Church-Turing Thesis

- The universal Turing machine can perform any calculation that any human computer can carry out
- Any systematic method can be carried out by the universal Turing machine
- Anything that can be made completely precise can be programmed for a universal digital computer
  - This is false, as the printing, halting and satisfactoriness problems are completely precise but cannot be programmed for a universal computing machine
- Any number, or binary sequence, that can be computed by the universal Turing machine can be calculated by means of a systematic method

2.13.1 Systematic methods

- The method can, in practice or in principle, be carried out by a human computer working with paper and pencil
- The method can be given to the human computer in the form of a finite number of instructions
- The method demands neither insight nor ingenuity on the part of the human being carrying it out
- The method will definitely work if carried out without error
- The method produces the desired result in a finite number of steps; or, if the desired computable function fx.
- Church's thesis: Every function of positive result is some infinite sequence of symbols, then the method produces each individual symbol in the sequence in some finite number of steps

2.13.2 Church's contribution

- To each computable sequence S corresponds a computable function fx.
- Church's thesis: Every function of positive integers whose values can be calculated by a systematic method is lambda-definable.

2.14 14. The Entscheidungsproblem

- Find an effective method by which, given any expression Q in the notation of the system, it can be determined whether or not Q is provable in the system.

- A formal system can simply be defined to be any mechanical procedure for producing formulas, called provable formulas

- Turing and Church both showed that no consistent formal system of arithmetic is decidable

- No Turing machine can perform the task in question→There is no systematic method for performing the task

- There is a systematic method for performing the task→A Turing machine can perform a task in question (by transposition)

3 See also

4 References

- Turing, Alan, and B. Jack Copeland. The Essential Turing Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life, plus the Secrets of Enigma. Oxford: Clarendon

Press; 2004.