(H2) Necessary Rudiments of Computability Theory

Selmer Bringsjord
Philosophy of AI

September 24, 2001

1 Suns Won't Do

At the very heart of the issues we'll be considering are the concepts of 'computer' and 'computation,' so it's probably not a bad idea if we at least get a workable account of both out on the table before we begin our debate about AI in earnest.

Obviously, we need an account of computerhood which isn't tied to any particular brand of computer: no one in AI claims that people are fundamentally, say, Sun workstations. In fact, we need an as-broad-as-possible conception of 'computer' and 'computation.' Fortunately, logic and computer science, since the 1930s, have had on hand a number of suitable conceptions (which, soon after their arrival, gave rise to the physical computers with which all of you are acquainted). One of these conceptions, undoubtedly the most popular, is a Turing machine, a mathematical account of computerhood created by Alan Turing [the same guy who gives his name to the famous Turing Test, about which you'll soon be reading: paper (T) on the syllabus]. (Some of the other conceptions were due to the logicians Church, Kleene, Post, Gödel, and von Neuman.)

2 Turing Machines

Put intuitively, TMs include a **two-way infinite tape** divided into squares, a **read/write head** for writing and erasing **symbols** (from some finite, fixed **alphabet**) on and off this tape, a **finite control unit** which at any step in a computation is in one particular state from among a finite number of possible states, and a set of **instructions** (= program) telling the machine what to do, depending upon what state it's in and what (if anything) is written on the square currently scanned by it's head.

There are many readily understandable ways to capture the full set-theoretic description of TMs. One such method is the state diagram approach, which is

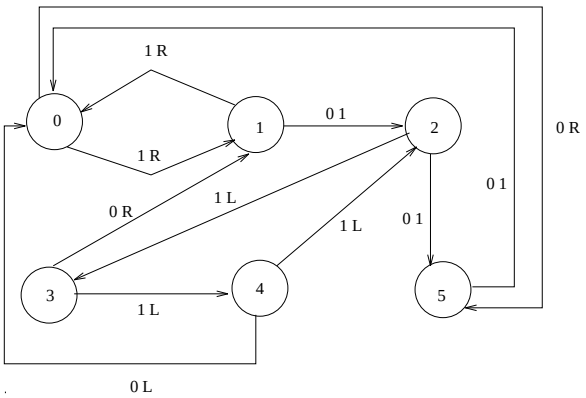

Figure 1: Gordon's 19 in 186

used in Figure 1. This TM, dubbed "Gordon's 19 in 186," is designed to start on a 0–filled infinite tape and produce, after 186 steps, 19 1's.

Let's "hand simulate" an initial segment of the computation of Gordon's TM—let's label the machine G— so that we completely fix the core mathematical concepts. The alphabet used is simply $\{0, 1\}$. The initial state of G is 0 (represented by the node labelled 0), and at the outset we'll assume that the tape is filled with 0's. The first thing G does is check to see what symbol it finds under its read/write head. In this case it initially finds a 0, so the arc labelled with 0 R[1] is taken, which means that the head moves one square to the right and the machine enters state 5. At this point, since there is another 0 found beneath the head, the 0 is changed to a 1, and the machine reenters state 0. It now finds a 1, and hence takes the arc labelled 1 R to state 1 (i.e., the machine moves its head one square to the right, and then enters state 1)—etc. The machine's activity can be perfectly captured by a tedious catalogue of its configurations from start to finish (Figure 2).

If this is your first exposure to TMs, you will doubtless be struck by how primitive and unassuming they are. But the surprising thing is that TMs apparently capture computation *in all its guises*. More precisely, whatever can be accomplished by way of an algorithm, by way of a programmed supercomputer, by way of a neural network, a cellular automaton, etc.—whatever can be accomplished by any of these can be accomplished by a TM.[2] Furthermore, we know that adding capabilities to our TMs doesn't give them any additional power.[3]

[1]Note that this is an ordered pair composed of zero and R for 'right,' not the word 'OR'.

[2]See my "Is the Connectionist-Logicist Clash One of AI's Wonderful Red Herrings?" for a discussion of the consequences of this fact for AI.

[3]The capabilities added must, however, be expressible in the language of set theory. If one considers a *physical* Turing Machine, then there are perhaps ways of "souping up" such machines so that they can process uncomputable functions.

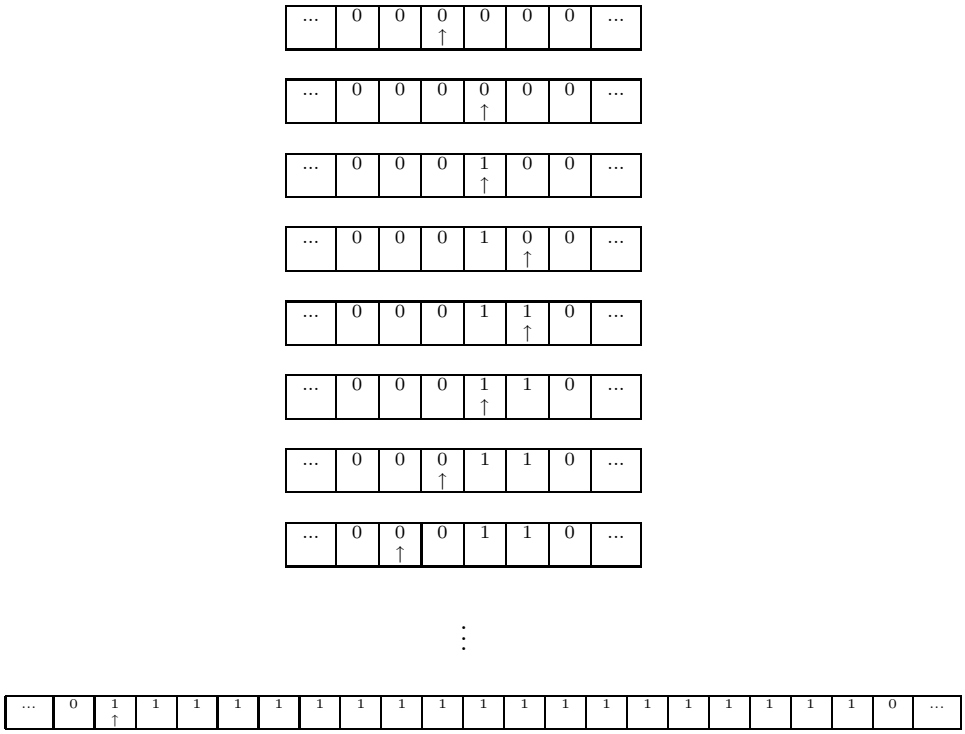

Figure 2: Tedious Catalogue of Gordon's TM

For example, if we give a TM *two* tapes rather than one, nothing that was impossible for the one-tape machine becomes doable for the two-tape creature.[4]

3 Pongid TMs

My favorite version of the TM is the pongid variety:

I imagine an old, rickety, pump-style boxcar riding on a railroad track whose squares are blackboards. The boxcar is powered and controlled by a muscular monkey armed with chalk and an eraser. The monkey follows instructions about moving the boxcar, and about writing and erasing on the squares; the instructions are commands like "If you're presently scanning a square marked by a, erase this symbol and write b, and then move left (right) one square."

4 Conclusion

So, what's computation, and what's a computer? The received answer, in very rough-and-ready form, is that computation is a computer at work, and a computer at work is that which can be modelled as a Turing machine at work.

[4]The interested reader can consult an octet of books I find useful: see me.