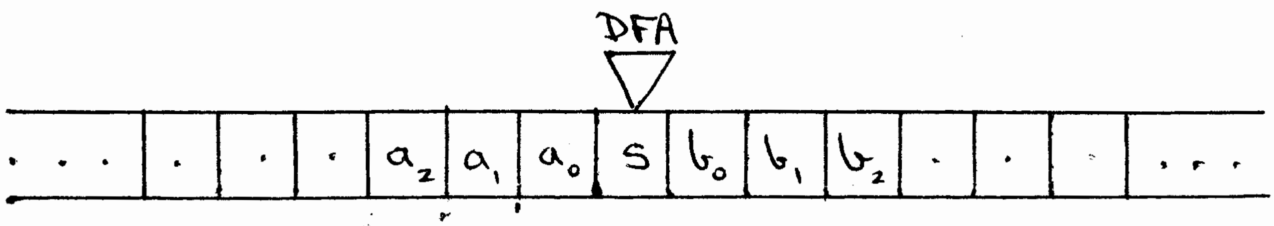
Computation Theory LECTURES 1, 6, 7, 8. p3q7 JKMM 2

Paper 3/10 (solution) 2004

Turing's Thesis is that the intuition of "what can be calculated" is precisely captured by the Turing machine model of computation. He argues this view by analogy with human calculation in his 1937 paper.



A Turing machine is a deterministic finite automaton that is at any instant positioned above one square – the "current" square – of an infinite tape; the tape is a linear medium, with potentially unbounded length both to left and to right.

Paper 3/10 (solution) ctd)

The DFA has a finite set Q of states, with a nominated initial state $q_0 \in Q$.

The Turing machine has a finite alphabet S, $|S| = k$ say, with a preferred blank symbol (maybe b, s_0 etc.). The action of the machine is deterministic, depending on the current state $q \in Q$ and the symbol $s \in S$ on the current square.

The computation is specified by giving initial data on the tape, and positioning the DFA (in its initial state) on a particular square; only a finite number of squares are non-blank.

Given (q, s), the machine makes 3 transitions:

Paper 3/10 (solution) ctd)

a) the DFA enters a new state $G(q,s) \in Q$;

b) the symbol on the current square is replaced by $F(q,s) \in S$;

c) there is movement $D(q,s)$ on the tape – the DFA is repositioned 1 square L or R.

One possible change of state is to HALT the machine – the computation terminates, and the result is the final state of the tape. Until the machine HALTs, operation is Black-Box.

A computation is determined by program logic + data; in the case of a Turing machine the initial data is the initial tape contents, and the program logic is defined by the 3 transition functions $\{(G(q,s), F(q,s), D(q,s)) \mid q \in Q, s \in S\}$.

Paper 3/10 (solution) ctd)

While a computation is in progress, its continuation is determined by the current state $q \in Q$, the head position, and the current tape contents. Evidently at any time t there will be only a finite number of non-blank symbols on the tape. We call this information the Turing machine configuration at time t.

Referring to the diagram on page 2, we give symbols $s \in S$ numerical values on scale k, making sure that the blank symbol has value 0. To specify a configuration we give current state $q \in Q$, and establish the tape contents relative to the current head position.

Paper 3/10 (solution) ctd,

The tape contents are specified by

s the current symbol

m the left half-tape, $\sum_{i=0}^{\infty} a_i k^i$

n the right half-tape, $\sum_{j=0}^{\infty} b_j k^j$.

The configuration is the quadruplet (q, s, m, n).

For head movement on the tape we use values $D(q,s) = 1$ (RIGHT), 0 (LEFT).

If the configuration at time t is (q, s, m, n), write $(\bar{q}, \bar{s}, \bar{m}, \bar{n})$ for the configuration at $(t+1)$.

$$\bar{q} = G(q,s)$$

$$\bar{s} = (n \text{ REM } k)\, D(q,s) + (m \text{ REM } k)(1 - D(q,s))$$

Paper 3/10 (Solution) ctd)

$$\bar{m} = (km + F(q,s))\, D(q,s) + (m \text{ DIV } k)(1 - D(q,s))$$

$$\bar{n} = (n \text{ DIV } k)\, D(q,s) + (kn + F(q,s))(1 - D(q,s))$$

We translate the quintuplet description of the Turing machine into the logic of the register machine by defining distinct sections of program for each state $q \in Q$.

It is convenient to allocate registers S, M, N to hold the three components of the current tape description. A single utility register Z is sufficient for the simple sums required.

First decode the current symbol $s \in S$, and switch to the program section for (q,s). We assume knowledge of $G(q,s)$, $F(q,s)$, $D(q,s)$.

Paper 3/10 (solution) ctd)

If $G(q,s) = \text{HALT}$, we just HALT the register machine, and can inspect the result of the Turing machine computation by considering s, m, n.

Otherwise $D(q,s) = L/R$ determines which sums must be done to compute the values $\bar{s}, \bar{m}, \bar{n}$. $F(q,s)$ is a known constant in the range $[0,k)$. We then continue the simulation with the program section corresponding to $G(q,s)$ and $\bar{s}$.

In this way we evidently simulate the Turing machine computation faithfully from one configuration to the next, HALTing when and only when the Turing machine would HALT.

Paper 3/10 (solution) ctd)

In 1930 Alonzo Church defined General Recursive Functions, and postulated – Church's Thesis – that they capture the intuition "what can be calculated" precisely. But that was a logician's approach, without Turing's motivation from human calculation.

Register machines were proposed much later, sharing the essential properties of Turing machines, but taking a model closer to digital computers. It is easy to calculate recursive functions on a register machine.

We've shown that register machines can also simulate Turing machines. It is easy to establish the converse as well. These results together have helped the acceptance of Turing's Thesis.