TURING MACHINES

Alan Turing

Alan Turing was one of the founding fathers of CS.

- His computer model –the Turing Machine– was inspiration/premonition of the electronic computer that came two decades later
- Was instrumental in cracking the Nazi Enigma cryptosystem in WWII
- Invented the "Turing Test" used in AI
- Legacy: The Turing Award.

A Thinking Machine

- First Goal of Turing's Machine: A model that can compute anything that a human can compute. Before invention of electronic computers the term "computer" referred to a *person* who's line of work is to calculate numerical quantities.
- As this is a philosophical endeavor, it can't really be proved.
- Turing's Thesis: Any "algorithm" can be carried out by one of his machines

A Thinking Machine

Second Goal of Turing's Machine: A model that's so simple, that can actually prove interesting epistemological results. Eyed Hilbert's 10th problem, as well as a computational analog of Gödel's Incompleteness Theorem in Logic.

Philosophy notwithstanding, Turing's programs for cracking the Enigma cryptosystem prove that he really was a true hacker! Turing's machine is actually easily programmable, if you really get into it. Not practically useful, though…

A Thinking Machine

Imagine a super-organized, obsessive-compulsive human computer. The computer wants to avoid mistakes so everything written down is completely specified one letter/number at a time. The computer follows a finite set of rules which are referred to every time another symbol is written down. Rules are such that at any given time, only one rule is active so no ambiguity can arise. Each rule activates another rule depending on what letter/number is currently read, EG:

A Thinking Machine

- It was hard for the ancients to believe that *any* algorithm could be carried out on such a device. For us, it's much easier to believe, especially if you've programmed in assembly!
- However, ancients did finally believe Turing when Church's lambda-calculus paradigm (on which lisp programming is based) proved equivalent!

Turing Machines

A Turing Machine (**TM**) is a device with a finite amount of *read-only* "*hard*" memory (states), and an unbounded[1] amount of read/write tape-memory. There is no separate input. Rather, the input is assumed to reside on the tape at the time when the TM starts running.

Just as with Automata, TM's can either be input/output machines (compare with Finite State Transducers), or yes/no decision machines. Start with yes/no machines.

A Thinking Machine
EG **`Successor`** Program

Sample Rules:

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read ☐, write 1, HALT!

Let's see how they are carried out on a piece of paper that contains the *reverse* binary representation of 47:

A Thinking Machine
EG **`Successor`** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

1	1	1	1	0	1				

A Thinking Machine
EG `Successor` Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	1	1	1	0	1				

A Thinking Machine
EG **Successor** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	0	1	1	0	1				

A Thinking Machine
EG **`Successor`** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read $\square$, write 1, HALT!

0	0	0	1	0	1				

A Thinking Machine
EG **`Successor`** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	0	0	0	0	1				

A Thinking Machine
EG **Successor** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read ☐, write 1, HALT!

0	0	0	0	1	1				

A Thinking Machine
EG `Successor` Program

So the successor's output on 111101 was 000011 which is the reverse binary representation of 48.

Similarly, the successor of 127 should be 128:

A Thinking Machine
EG `Successor` Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

1	1	1	1	1	1	1			

A Thinking Machine
EG **`Successor`** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read ☐, write 1, HALT!

0	1	1	1	1	1	1			

A Thinking Machine
EG **`Successor`** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	0	1	1	1	1	1			

A Thinking Machine
EG **`Successor`** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	0	0	1	1	1	1			

A Thinking Machine
EG **`Successor`** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	0	0	0	1	1	1			

A Thinking Machine
EG **Successor** Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	0	0	0	0	1	1			

A Thinking Machine
EG `Successor` Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	0	0	0	0	0	1			

A Thinking Machine
EG `Successor` Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read ☐, write 1, HALT!

0	0	0	0	0	0	0			

A Thinking Machine
EG `Successor` Program

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read □, write 1, HALT!

0	0	0	0	0	0	0	1		