Turing Completeness
Basic questions

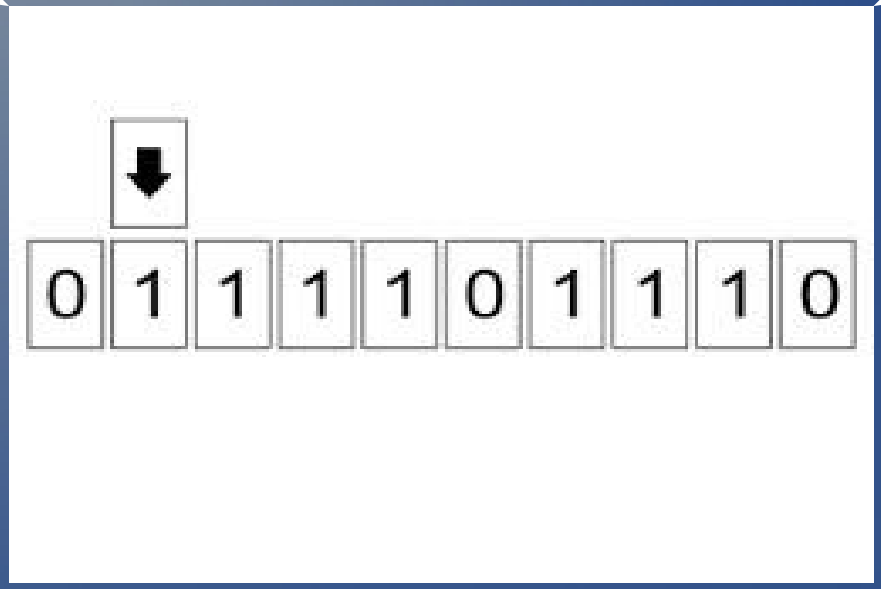
- What is Turing Completeness (TC)?
- Can computers be TC?
- What is and what isn´t TC?
- What makes a program TC?
- Why do we have both TC and non-TC programs?
- An example of a TC program

What is Turing Completeness?

Computer's instruction set

Programming language

Cellular automaton



Turing Complete

=

Computationally Universal

Can Computers be TC?

- Life expectancy
- Memory capacity

What is and what isn´t TC?

TC:

- More or less all general purpose programming languages

Not TC:

- Final automata (e.g. regular expressions, seq. logic circuits)
- A category of pushdown automata and context-free grammars, which are commonly used to generate parse trees in program compiling
- Markup Languages (e.g. XML, HTLM, JSON)

What makes a program TC?

- Recursive functions

OR

- Loops that may be infinite

Why both TC and not TC?

TC:

- Easier
- Flexibility outweighs its complexity

Not TC:

- For special domains TC may not be a smart feature

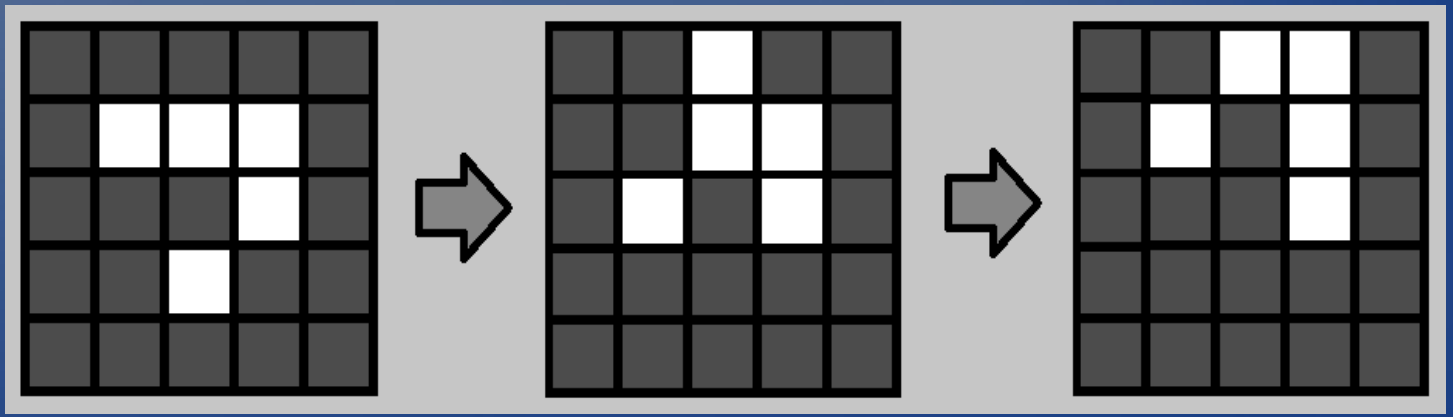
Cellular Automatons

- Rule 110

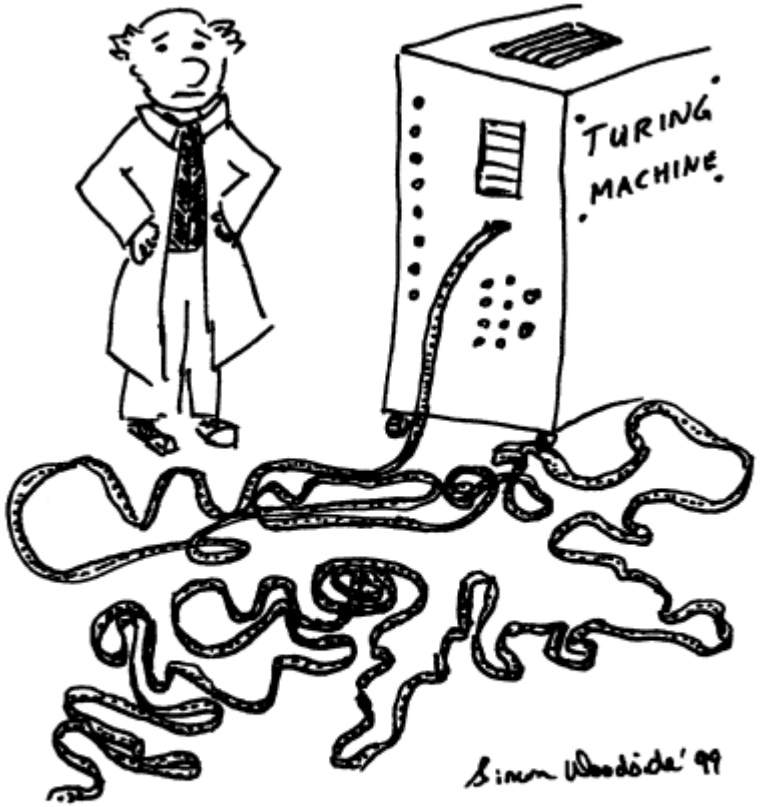
Rule 110 cellular automaton

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

- Game of Life (John Conway ~1970)



Game of Life



Turing solves the halting problem, only to discover that the REAL problem with his machine is what to do with all the tape.

Sources of information

- http://en.wikipedia.org/wiki/Turing_completeness
- http://lambda-the-ultimate.org/node/2846

Theory of Computation