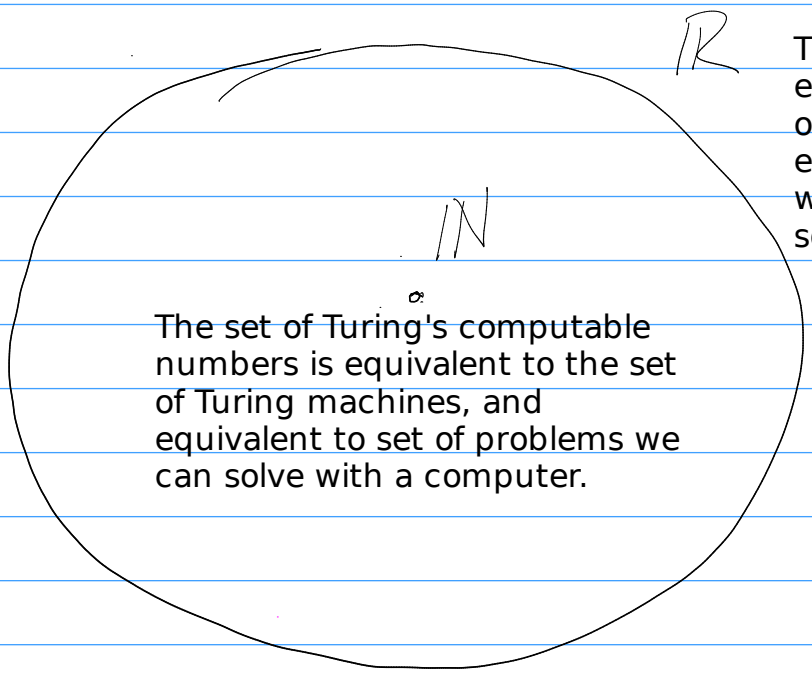
CS605
26 February 2025



The set of real numbers is equivalent to the set of languages over an arbitrary alphabet, and equivalent to the set of problems we might want a computer to solve.

Not to scale.

The set R is not a fixed number of times larger than the set N, it is infinitely larger -- mathematicians would say there is "probability zero" of finding a whole number if one randomly chooses any finite number (e.g. a trillion) real numbers at random.

Turing's diagonalisation 'algorithm'.

TM* \ Index	0	1	2	3	4	5	6	7	...
0	5	3	4	1	1	2	4	3	...
1	6	1	3	3	3	3	3	3	...
2	4	4	1	1	2	2	7	7	...
3	9	9	1	2	3	4	5	6	...
⋮									

* Only those TMs that output a computable (infinite decimal expansion).

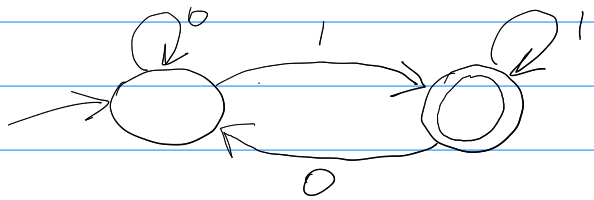
Diagonal 0.5112....

New number not in table: 0.6373.....
(an uncomputable number).

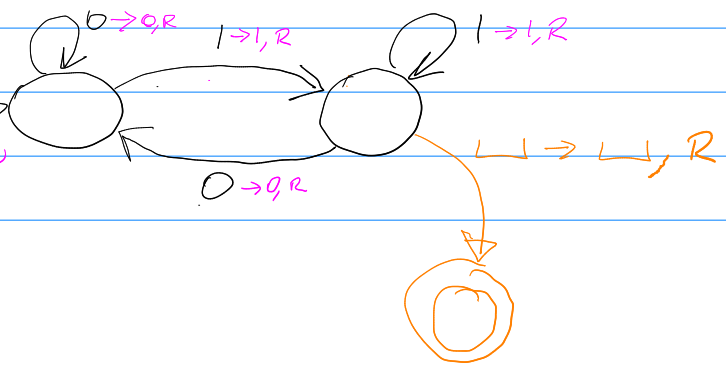
A contradiction!

TM programming is just an extension of FA programming

A FA to recognise $\{w \in \{0,1\}^* , w \text{ ends with } 1\}$



A TM to recognise the same language.



THE CHURCH–TURING THESIS

So far in our development of the theory of computation we have presented several models of computing devices. Finite automata are good models for devices that have a small amount of memory. Pushdown automata are good models for devices that have an unlimited memory that is usable only in the last in, first out manner of a stack. We have shown that some very simple tasks are beyond the capabilities of these models. Hence they are too restricted to serve as models of general purpose computers.

TURING MACHINES

We turn now to a much more powerful model, first proposed by Alan Turing in 1936, called the ***Turing machine***. Similar to a finite automaton but with an unlimited and unrestricted memory, a Turing machine is a much more accurate model of a general purpose computer. A Turing machine can do everything that a real computer can do. Nonetheless, even a Turing machine cannot solve certain problems. In a very real sense, these problems are beyond the theoretical limits of computation.

The Turing machine model uses an infinite tape as its unlimited memory. It has a tape head that can read and write symbols and move around on the tape. Initially the tape contains only the input string and is blank everywhere else. If the machine needs to store information, it may write this information on the tape. To read the information that it has written, the machine can move its head back over it. The machine continues computing until it decides to produce an output. The outputs *accept* and *reject* are obtained by entering designated accepting and rejecting states. If it doesn't enter an accepting or a rejecting state, it will go on forever, never halting.

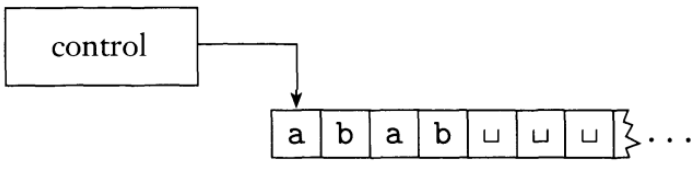


FIGURE 3.1
Schematic of a Turing machine

The following list summarizes the differences between finite automata and Turing machines.

1. A Turing machine can both write on the tape and read from it.
2. The read–write head can move both to the left and to the right.
3. The tape is infinite.
4. The special states for rejecting and accepting take effect immediately.

FORMAL DEFINITION OF A TURING MACHINE

The heart of the definition of a Turing machine is the transition function δ because it tells us how the machine gets from one step to the next. For a Turing machine, δ takes the form: $Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$. That is, when the machine is in a certain state q and the head is over a tape square containing a symbol a, and if $\delta(q, a) = (r, b, \mathrm{L})$, the machine writes the symbol b replacing the a, and goes to state r. The third component is either L or R and indicates whether the head moves to the left or right after writing. In this case the L indicates a move to the left.

DEFINITION 3.3

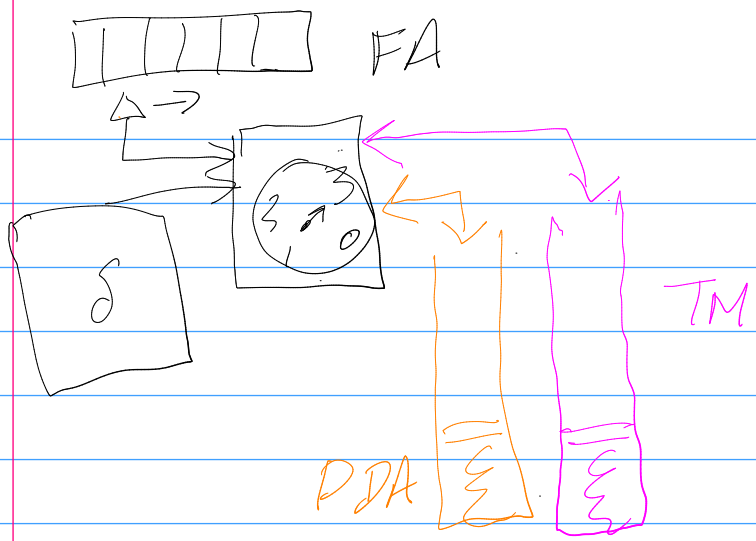
A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

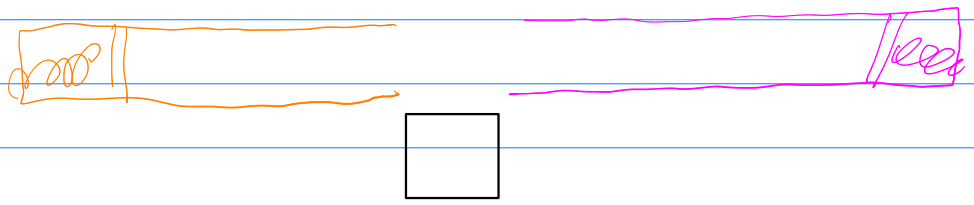
1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta\colon Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Formal definition of a TM computation

A Turing machine M ***accepts*** input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where

1. C_1 is the start configuration of M on input w,
2. each C_i yields C_{i+1}, and
3. C_k is an accepting configuration.





Tape head

Adding stack to a FA gives us a PDA.
Adding a second stack to the a gives us a TM.
How?
Popping a symbol from the left stack and pushing it on the right stack is equivalent to the tape head moving one position to the left on the tape.
Popping a symbol from the right stack and pushing it on the left stack is equivalent to the tape head moving one position to the right on the tape.