The Limits of Computation

7C

Universal Computers (Turing Machines)

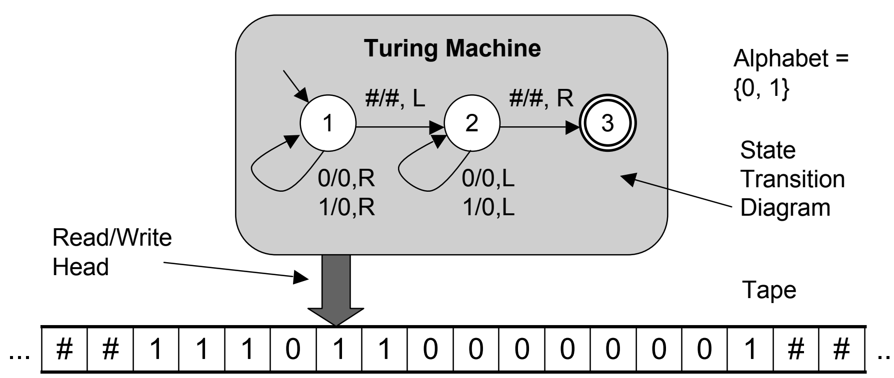
Alan Turing

- 1912-1954
- Consider by many to be one of the founding fathers of computer science.
- Developed the theoretical notion of a universal computer in the 1930s: Turing Machines
- Worked during WW II at Bletchley Park in England, helping to break encrypted messages by the Germans (the Enigma code)
- Formalized the fundamental principle of artificial intelligence: how to determine if a machine is "intelligent": The Turing Test (More on this soon.)

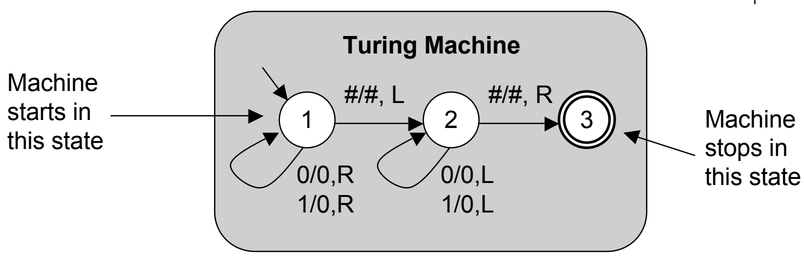
The Turing Machine

- A Turing Machine (TM) consists of:
  - A finite set of states.
  - A finite alphabet of symbols.
  - An infinite tape marked off into cells such that one symbol can be stored in each cell.
  - A sensing head that can read or write one symbol at a cell on the tape. The head can also move forward or back on the tape one cell at a time.
  - A state transition diagram that shows how the machine works.

The Turing Machine



The Turing Machine



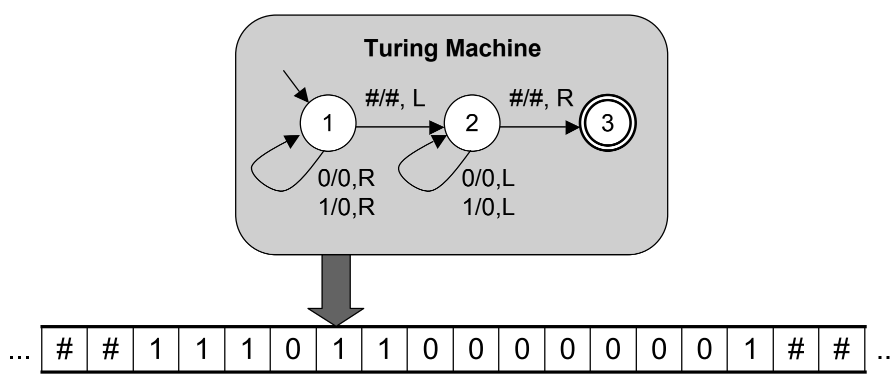
State Transitions for state 1:

0/0, R : If we're in state 1 and the head sees a 0 on the tape, write a 0 on the tape, move the head to the right one cell and remain in state 1.

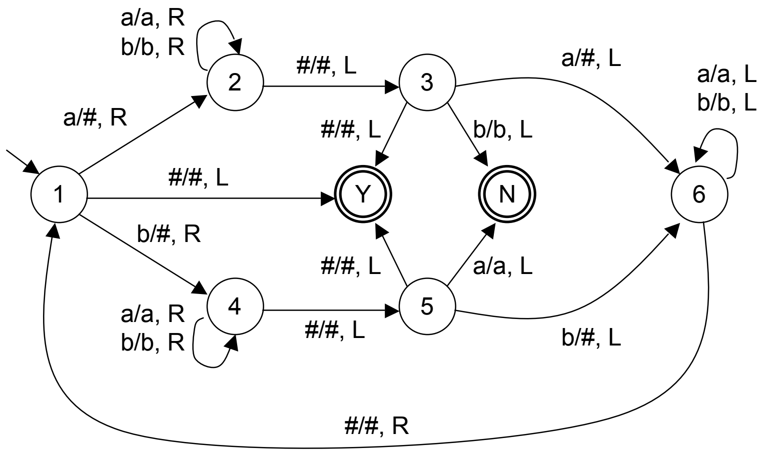
1/0, R : If we're in state 1 and the head sees a 1 on the tape, write a 0 on the tape, move the head to the right one cell and remain in state 1.

#,#, L: If we're in state 1 and the head sees a space on the tape,write a space on the tape, move the head to the left one cell and go to state 2.

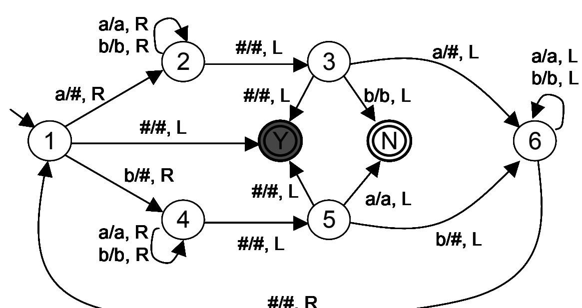
The Turing Machine



Palindrome Checker



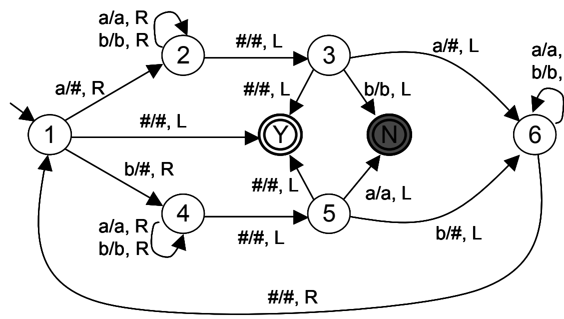
Palindrome Checker



ORIGINAL TAPE

...	#	#	#	#	a	b	b	a	a	b	b	a	#	#	#	#	#	#	...
...	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	...

Palindrome Checker



ORIGINAL TAPE

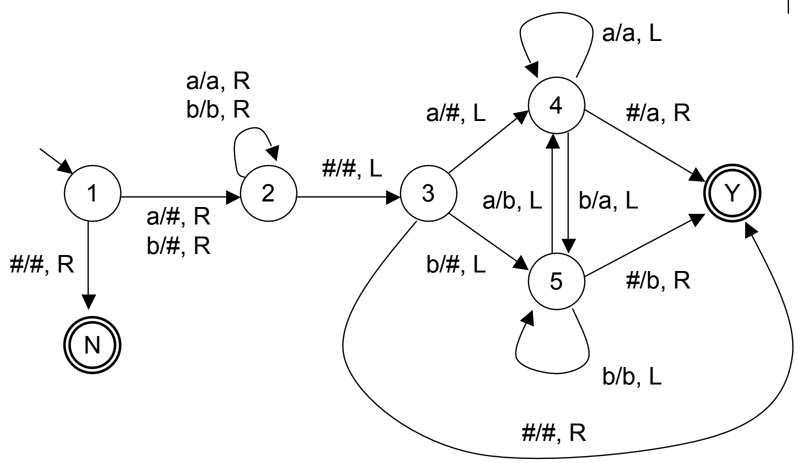
...	#	#	#	#	a	b	b	a	a	b	b	b	#	#	#	#	#	#	...

...	#	#	#	#	#	b	b	a	a	b	b	b	#	#	#	#	#	#	...

Palindrome Checker

- If the TM halts and the tape is blank, then the original string must be a palindrome.
- If the TM halts and the tape is not completely blank, then the original string must not be a palindrome.
- To think about:
  - Does this TM work with strings of odd length?
  - How would you adapt this TM for alphabets with n symbols (rather than 2)?

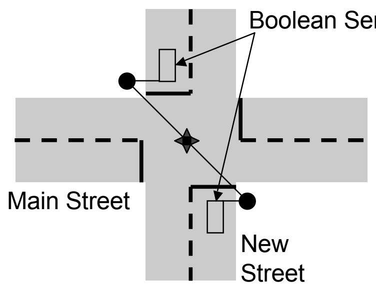
What does this TM do?

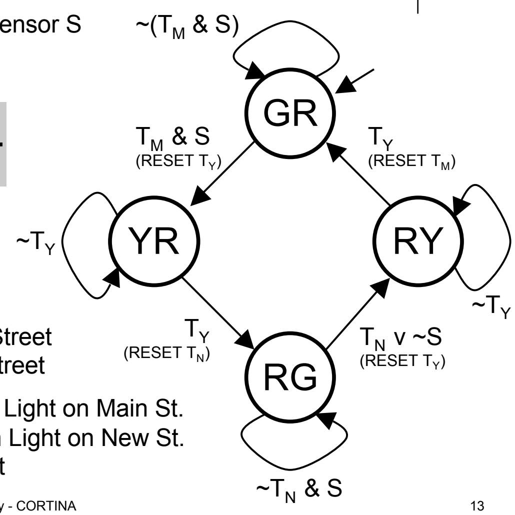


Finite State Machine (FSM)

- A model used to define the behavior of a system consisting of states and transitions.
- Some uses in computing:
  - Designing sequential circuits. (e.g. traffic signal controllers)
  - Building object-oriented systems. (e.g. cell phone state diagram in UML)
  - Communication protocols. (e.g. TCP/IP)
- A Turing machine is a FSM.

Traffic Light State Machine





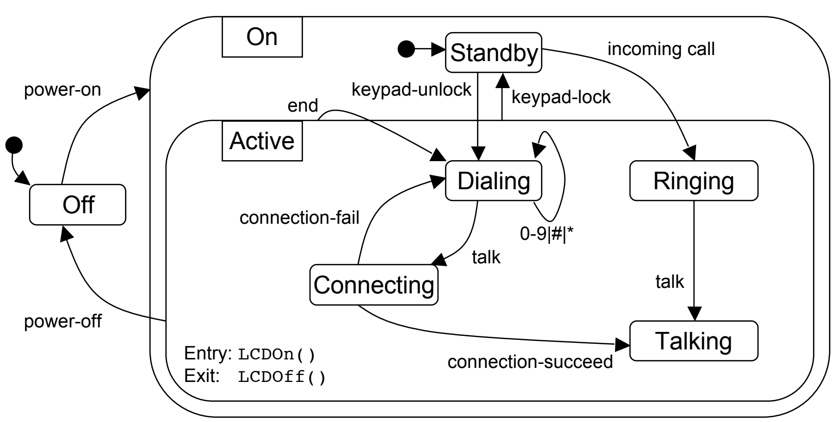
C_MC_N C_M = Light Color for Main Street
C_N = Light Color for New Street

T_M = Boolean Timer for Min. Green Light on Main St.
T_N = Boolean Timer for Max. Green Light on New St.
T_Y = Boolean Timer for Yellow Light

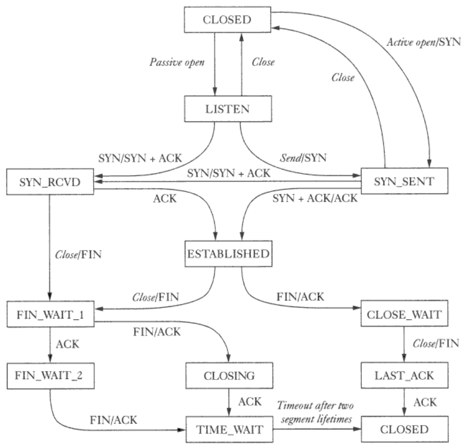
Cell Phone State Machine

in Unified Modeling Language (UML)

from Object-Oriented Software Development Using Java by Xiaoping Jia



TCP/IP State Machine



from http://www.ssfnet.org/Exchange/tcp/tcpTutorialNotes.html#ST

TCP/IP is the suite of Transmission Control Protocol and Internet Protocol.

This is a networked protocol that allows two machines to communicate with one another by sending data in labeled packets with acknowledgement packets to confirm reception.

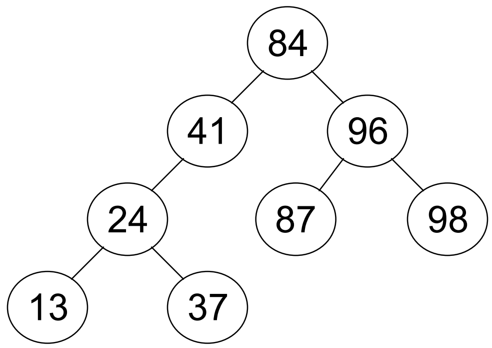
Encoding Data as Strings

- All data can be represented as a string (i.e. a sequence of symbols) that can be stored on a Turing machine tape.
- Examples:
  - An integer: **15105**
    11101100000001
  - A vector:

0	1	1	2	3	5	8	13	21	34	55	89

0*1*1*2*3*5*8*13*21*34*55*89

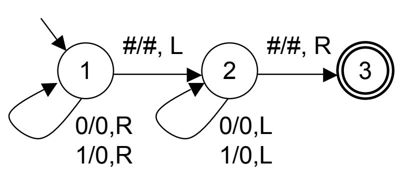
Representing Data as Strings



- A binary tree:
  `84*41*96*24**87*98*13*37`

Encoding Turing Machines

- We can even encode a Turing machine and store it on a tape of a Turing machine!



`*1*0*0*R*1**1*1*0*R*1**1*#*#*L*2**2*0*0*L*2**2*1*0*L*2**2*#*#*R*3*`

Noncomputability again

- Define Q as a TM that runs on a tape that includes an encoding of a TM and an input tape configuration for the encoded TM.

...	#	encoded TM	#	data for encoded TM	#	...

- Q halts with only a 1 on its tape if its encoded TM halts using the input tape configuration. Otherwise, it halts with a 0 on its tape.
  - Q is a TM that determines if the encoded TM on the tape halts if it were to run with the data stored after the encoded Turing Machine on the tape.

Noncomputability again

- Define S as a new TM that requires an encoded TM on its tape. It copies the encoded TM on its tape again, and then simulates Q. If the simulated Q writes 1 on the tape, S goes into a state that moves left forever. Otherwise, S halts.

...	#	encoded TM	#	...

...	#	encoded TM	#	encoded TM	#	...

- What happens if S runs with its own encoding on the tape?

Church-Turing Thesis

- Turing machines are capable of solving <u>any</u> solvable algorithm problem.
- Another way to look at it:
  - Any computer, no matter how powerful or how advanced can be mapped into a Turing machine.
  - A supercomputer and a personal computer can solve the same problems given an infinite amount of time and memory space.
  - Problems that cannot be solved by a computer also cannot be solved on a Turing machine.
- NOTE: This is a <u>thesis</u>.

Alternate Models

- Are there more powerful Turing machines?

  What if we had a tape that ran infinitely in only one direction?

  What if we had multiple tapes?
- All alternative models can be shown to be equivalent to the original TM we discussed already.
- All computers are equivalent in computational power, given unlimited time and memory space!

  Although Turing machines perform their operations in a very tedious way for anything but trivial tasks, Turing machines are only <u>polynomially less efficient</u> than the most efficient algorithms programmed on fast computers with modern programming languages.

P vs. NP again

- If someone were to show that a Turing machine cannot solve an NP-complete problem in polynomial time, then we can state from the C-T thesis that any sequential computing device (modern computer) cannot solve such a problem.
- If one particular NP-complete were to be shown to be unsolvable in polynomial time on a TM, then all NP-complete problems would not be solvable in polynomial time.
  - We would say here, then, that P is not equal to NP.

Summary

- A computer can't solve all computational problems!
  - Some are intractable.
    There are algorithmic solutions, but they require a huge amount of time, much longer than our lifetimes for anything but trivial cases.
  - Some are uncomputable.
    There are no algorithmic solutions that can be built to some these problems, no matter how hard we try to find them.
  - More powerful computers won't solve this dilemma since these will merely be faster Turing machines which have the same computational power.