COMS W3261 – Lecture 8, Part 2:

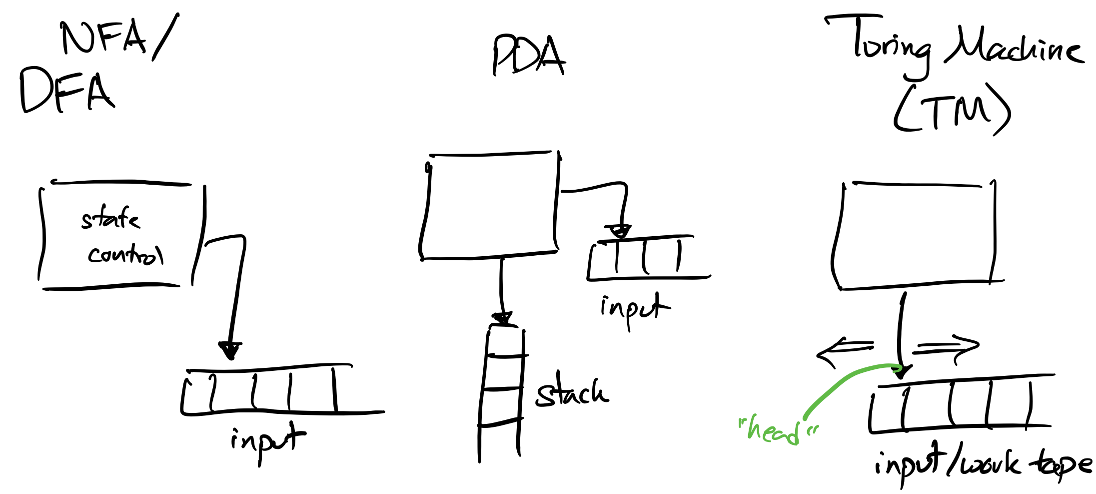
Turing Machines.

Alan Turing: (1912–1954)

- Invented the Turing Machine.
- Invented the Turing Test.
- Built a (sort of) computer during WWII. Used it to crack the Enigma cipher.

// see: Wiki; Alan Turing: The Enigma (Hodges), *The Imitation Game*

Turing Machine: an automaton that can read and write on an infinite memory tape.



At every step of computation:

(1) read a symbol off the current tape square

(2) enter a new internal state, write on the tape square, and move R or L.

<u>Def</u>. (Turing Machine). A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{reject}, q_{accept})$,

where Q is a finite set of states

Σ is the input alphabet

Γ is the tape alphabet. $(\sqcup \in \Gamma, \Sigma \subseteq \Gamma)$.

"blank symbol"

// note: deterministic transitions (for now)

$$\delta : \underbrace{Q \times \Gamma}_{\text{in some state read some symbol off current tape square}} \longrightarrow \underbrace{Q}_{\text{state}} \times \underbrace{\Gamma}_{\text{write some symbol on current square}} \times \underbrace{\{L, R\}}_{\text{move left or right.}}$$ is the transition function.

q_0, q_{accept}, and q_{reject} are start, accept, and reject states.

How a TM computes:

(1) Begin with an input string $\omega = \omega_1 \omega_2 \cdots \omega_n \in \Sigma^*$ written on the leftmost n squares of the tape.

(2) The "head" starts pointing at the leftmost square. Start state is q_0.

(3) Computation proceeds according to the transition function.
(If we attempt to move left from the leftmost tape square, we do not move.)

(4) Computation continues until we reach q_{accept} or q_{reject}, at which point we <u>immediately</u> accept/reject.

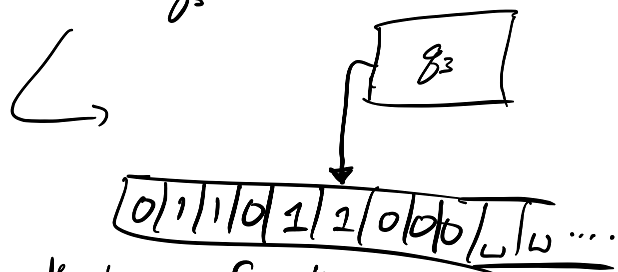
// Note: this means we don't necessarily stop. We might go into an infinite loop, and preventing this is (very) hard.

// Note: TMs are often too cumbersome to write out formally.

<u>Def</u>. (Configuration.) A configuration consists of a current state, the tape contents, and the head location.

Often written as uqv for a Turing Machine in state q, with the string u to the left of the head and the string v to the right (head is on first symbol of v.)

Example: $01101\, q_3\, 1000$



We'll also say that a configuration

$$u a q_i b v \text{ yields } u q_j a c v \quad \text{if } \delta(q_i, b) = (q_j, c, L).$$

in q_i, read b — go to q_j, write c, move L

$$u a q_i b v \text{ yields } u a c q_j v \quad \text{if } \delta(q_i, b) = (q_j, c, R).$$

$(a, b \in \Gamma,\ u, v \in \Gamma^*,\ q_i, q_j \in Q)$.

Def. (TM acceptance, formally.) A TM M accepts a string w if there exists a sequence of configurations $C_1, C_2, \cdots C_k$ such that

C_1 is the start configuration $q_0 w$,

C_i yields C_{i+1} for $i < k$,

and C_k is an accept configuration.

Decidability & Recognizability.

Idea: Like other automata, TMs can accept or reject. However, they might loop forever. What does it mean to recognize a language?

Def. (Turing-recognizability.) The set of all strings a TM M accepts is the language $L(M)$ of M. A language is Turing-recognizable if some TM recognizes it.

Def. (Turing-decidability.) A language L is (Turing)-decidable, if some Turing machine decides it:
if accepts on all strings in L,
AND it rejects on all strings not in L.
(A TM that always accepts or rejects is called a decider.)

— Note: Decidability implies Turing-recognizability.

Example. A TM that recognizes $A = \{0^{2^n} \mid n \geq 0\}$.

1. An implementation-level description.

   ⌊ higher-level than formal 7-tuple or state diagram
   Describe how the head moves around and how it modifies the tape, but not individual states or transitions.

M_2 = "On input w:
1. Read input left to right, cross off every other 0.
2. (Base case) If we just saw a single 0, accept.
3. Otherwise, if the number of 0's was odd, reject.
4. Otherwise, return the head to the left end of the tape and repeat from state 1."

all English prose, but should be easily made precise.

Example. sixteen 0's → eight 0's → four 0's → two 0's → one 0.

fourteen 0's → seven 0's → ~~three~~ 0's.

2. A formal description of M_2. (In contrast to implementation-level description.)

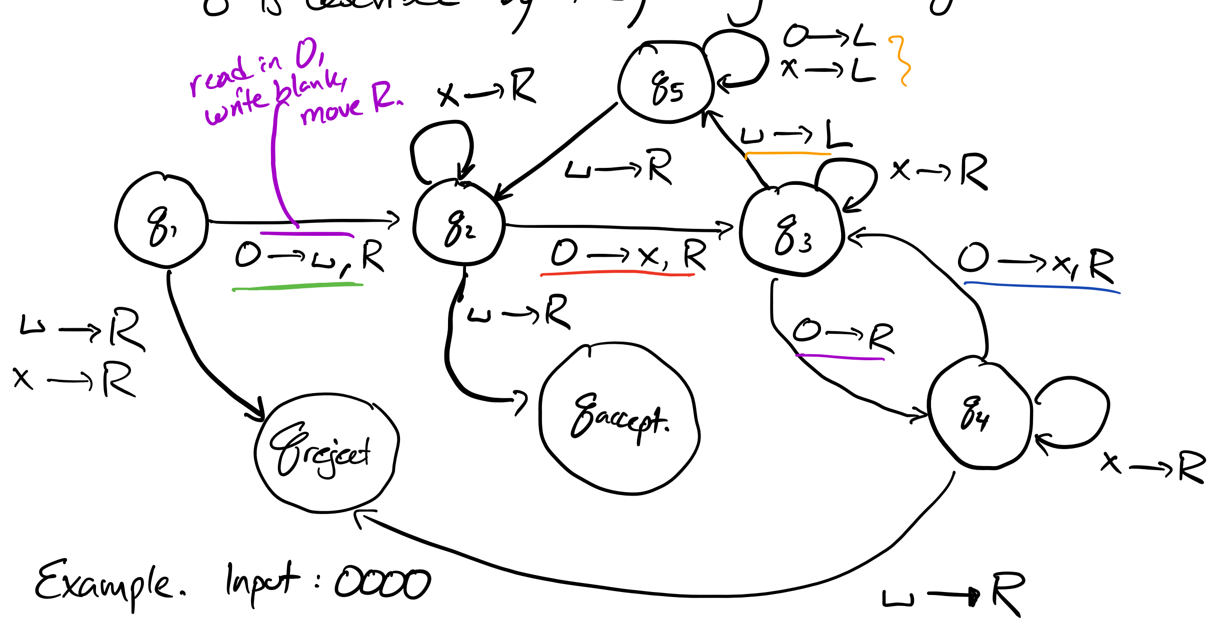
$$M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$$

$$Q = \{q_1, q_2, \dots, q_5, q_{accept}, q_{reject}\}$$

$$\Sigma = \{0\}$$

$$\Gamma = \{\sqcup, 0, x\}$$

δ is described by the following state diagram:



Example. Input: 0000

state	tape/head
q_1	0000
q_2	⊔000
q_3	⊔x00
q_4	⊔x00
q_3	⊔x0x⊔

state	tape/head
q_5	⊔x0x
q_5, q_5, q_5	⊔x0x
q_2	⊔x0x
q_2	⊔x0x
q_3	⊔xxx
q_3	⊔xxx⊔

state	tape/head
q_5	⊔xxx
q_5, q_5, q_5	⊔xxx
q_2	⊔xxx
q_2, q_2, q_2	⊔xxx⊔
q_{accept}	⊔xxx⊔⊔ ✓

// This is the only time we'll see a TM state diagram.

Next time: more TMs, decidability, enumerators,
Cantor — again!

Readings: 2.3 (CFPL), 3.1 (TMs)