L9:

Turing Machines

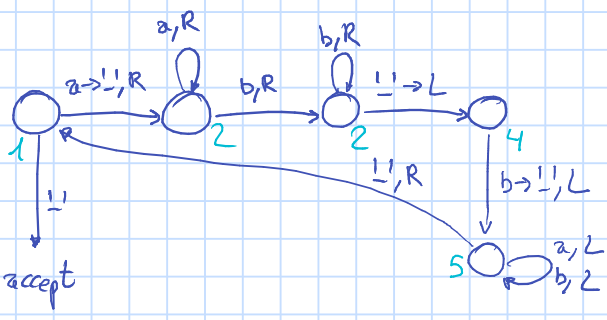
TM = DFA + infinite tape.
- → input written on tape; blanks elsewhere.
- → initially pointing to first input character.
- → each step reads from tape, writes, and moves L/R.
- → two special "accept" & "reject" states.

e.g. A TM deciding $L = \{a^n b^n\}$.

Informal: move between as & bs, crossing them out alternately. When no longer possible, check all characters were crossed out.

Formal:
1. cross out a (write '_').
2. move right until last b.
3. if no b found: reject.
4. cross out b (write '_').
5. move left until first a.
6. if no a found:
   move right
   if '_': accept
   else: reject
7. go to 1.

Diagram:



Obs: transitions consist of (read, write, L/R).
if write = read, we can omit it.
two special states accept & reject without transitions.
deterministic.
omitted transitions implicitly point to reject.

def TM. $(Q, \Sigma, \Gamma, \delta, q_0)$

Q: states.
Σ: input alphabet. $\_ \notin \Sigma$.
Γ: tape alphabet. $\_ \in \Gamma$. $\Sigma \subseteq \Gamma$.
δ: transition function.
$\delta: Q \times \Gamma \to Q \cup \{acc, rej\} \times \Gamma \times \{L, R\}$.
q_0: initial state.

def. computation.

sequence of configurations (s, q, i)
where s is tape contents, q a state, and i tape position.
initially s = input, $q = q_0$, $i = 1$.
sometimes write $(s_1 \dots s_{i-1})\, q\, (s_i \dots s_n)$ as shorthand.

a computation accepts if last state is (accept) and rejects if last state is (reject).

def M recognizes L if $\forall w \in L$, M accepts w.

def M decides L if $\forall w \in L$, M accepts w.
$\forall w \notin L$, M rejects w.

What is the difference?

∀ different ways to define TMs.

th the following models recognize the same languages:

(1) PDAs with $k \geq 2$ stacks
(2) TMs with one-sided infinite tape (indexed by $\mathbb{N}$).
(3) TMs with two-sided infinite tape (indexed by $\mathbb{Z}$).
(4) TMs with k tapes.

proof sketch:

(1)→(2). push input to first stack, then transfer to second

since this time, first stack contains tape left of pointer, and second contains tape right of pointer.
move left/right means pop 1 + push 2 / vice-versa.

(2)→(3). position i encodes tape positions i and -i.

alphabet is $\Gamma \times \Gamma \cup \{\#\}$.

when we reach position 0, we switch directions.
i.e. we have two copies of the TM, with directions reversed, and we switch between copies when we see an input of the form $(x, \#)$.

e.g. A TM deciding $L = \{a^n : n = 2^m\}$.

Informal: recursive construction. If only one a left, accept. Otherwise cross out every other a. If one a unmatched, reject. Repeat.

Formal:
1. replace first 0 with #.
2. move right, replacing 0s with $\dot{0}$ and 0 alternately.
3. if no replacement, accept
4. if last replacement was 0, reject
5. move left until #.
6. go to 2.

Diagram: