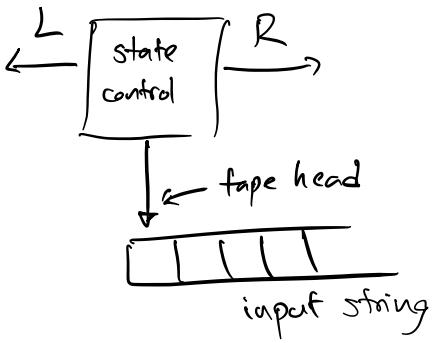
Announcements:

- HW 4 due Tues @ 11:59pm
- Tonight's virtual office hours:
  - ↳ 5:15–6pm: HW 2 & 3
  - ↳ 6:00–6:45: HW 4, TMs

Today: What TMs can do.

0. TM review
   - 0.1 TM state diagram & evaluation
1. Turing Machines: Beyond the Machine Level
   - 1.1. Tools & capabilities
   - 1.2. Simulating other automata.
2. The Church-Turing Thesis

0. Turing Machines

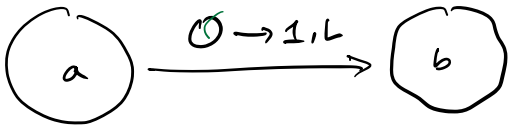


(1) read tape
(2) change internal state
(3) write to the tape
(4) move L or R

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$$

"give me a state and a symbol on the current tape square"

output a new state, a tape symbol to write, and a direction to move.



$\delta(a, 0) = (b, 1, L)$

Example:

$$A = \{0^{2^n} \mid n \geq 0\}$$

$0^{2^0} = 0$ ✓

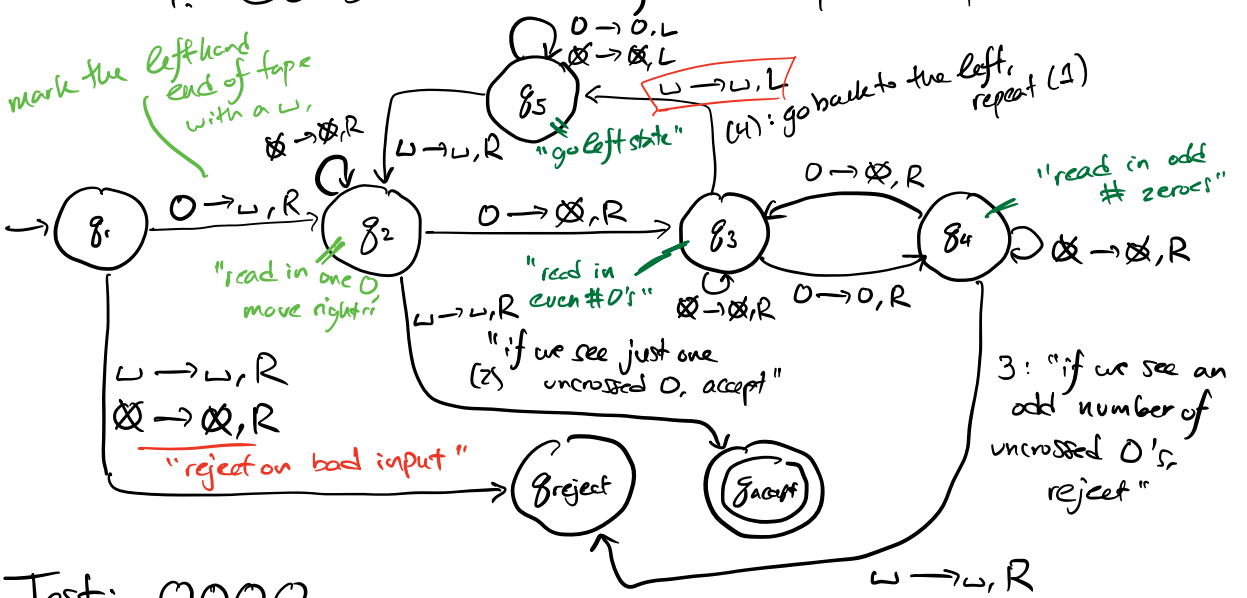
$0^{2^1} = 00$ ✓

$0^{2^3} = 0\,\cancel{0}\,\cancel{0}\,\cancel{0}\,\cancel{0}\,\cancel{0}\,\cancel{0}\,\cancel{0}$

00000 ✗

M_1 = "On input w:

1. Read input left to right, crossing off every other 0.
2. If we see one single uncrossed 0, accept.
3. If we see an odd number of uncrossed zeroes, reject.
4. Go back to the left and repeat step 1."



Test: 0000

state	tape
q_1	$\check{0}000$
q_2	$\sqcup\check{0}00$
q_3	$\sqcup X\check{0}0$

state	tape
q_4	$\sqcup X0\check{0}$
q_3	$\sqcup X0X\check{\sqcup}$
$q_5, q_5, q_5 \cdots$	$\check{\sqcup}X0X$

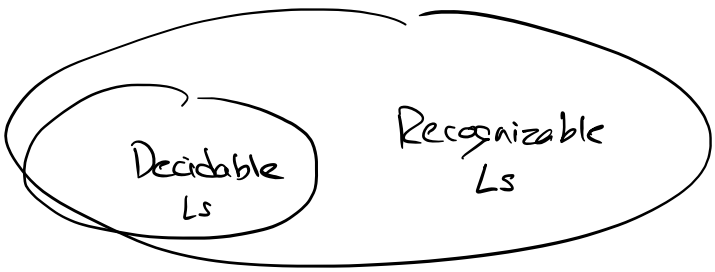
state	tape
q_2	$\sqcup X\check{0}X$
q_2, q_3	$\sqcup XX\check{X}$
q_3	$\sqcup XXX\check{\sqcup}$

state	tape
$q_5, q_5, q_5 \cdots$	$\check{\sqcup}XXX$
$q_2, q_2, q_2, \cdots$	$\sqcup XX\check{X}\sqcup$
q_{accept}	$\sqcup XXX\sqcup\check{\sqcup}$

Def. A language is (Turing)-decidable if some Turing Machine
- accepts on every string in the language
- rejects on every string not in the language.

Def. Given a TM M, the set of strings accepted by M is the language recognized by M.

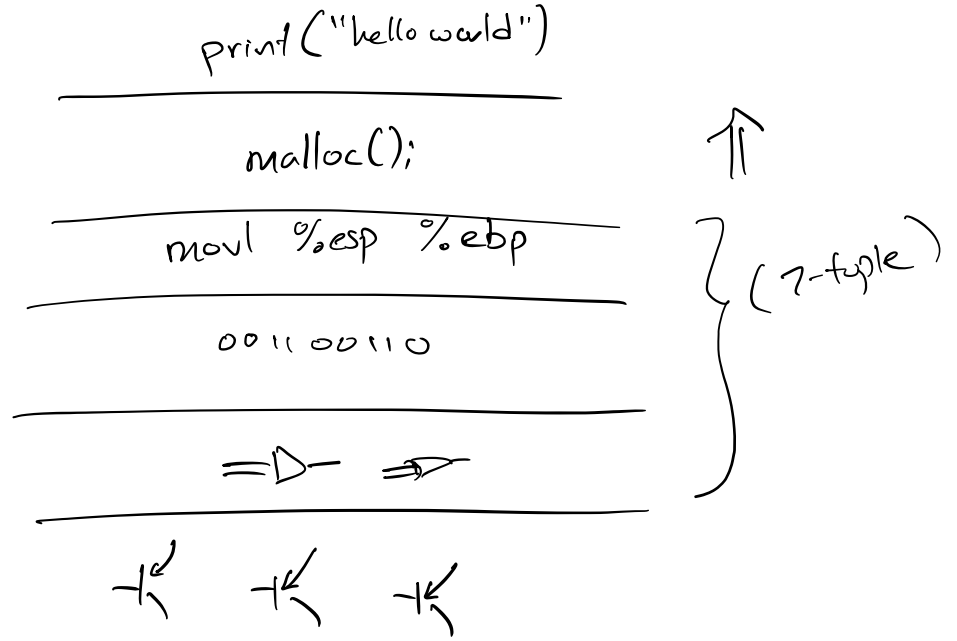
A language is (Turing)-recognizable if some TM recognizes it (= accepts all strings in the language, rejects or loops on strings not in the language.)



Break: back at 1:53

2. Beyond the Machine Level.

A "real" computer



implementation-level description:

- describe how the TM operates on a generic input string with words
- describe head movement
- describe memory management (what's written on tape)

Litmus test: could I take these instructions and create a formal description?

1) Matching strings.

$$B = \{ w \# w \mid w \in \{0,1\}^* \}$$

011 # 011

Our TM M_B that decides B will work as follows:

M_B = "On the input string S:

1. scan left to right, check if our input matches $(0 \cup 1)^* \# (0 \cup 1)^*$.

move across input, change nothing

can check if $w \in R$ because DFAs can do this!

Reject if we don't match.

2. scan back and forth between the substrings before and after the #. Each time, read the first uncrossed character from each substring, cross both off, and reject if they don't match.

Reject if we run out of chars in either substring.

3. Accept if all characters are crossed off.

0X1X#0XX

010#011

0XX1#0XX

2) Multiplying.

$$C = \{a^i b^j c^k \mid i \times j = k, \text{ and } i, j, k \geq 1\}$$

M_C = "On input string w:

1. Scan left to right, and check if we match $a^+b^+c^+$. Reject if not.

2. Scan back and forth, and for each a:
   - cross off this a.
   - scan back and forth between b's and c's. cross off one b and one c each time.
   - reject if we run out of c's. (*)
   - uncross all b's.

3. If all a's are crossed off, accept if we've crossed off all c's. Otherwise, reject.

~~a~~~~a~~ bbb ~~c~~~~c~~~~c~~~~c~~~~c~~ ✓

aa bbb ccccc ✗

~~a~~~~a~~ bbb ~~c~~~~c~~~~c~~~~c~~~~c~~ c ✗

3) Element Distinctness

$$E = \{\#x_1 \#x_2 \#x_3 \cdots \#x_\ell \mid \text{each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for } i \neq j\}$$

M_4: "On input w:

1. scan to make sure we match $(\#(0 \cup 1)^*)^+$.
2. (accept if there's only one input substring).

Otherwise, mark the first and second # with a dot. (#̇)

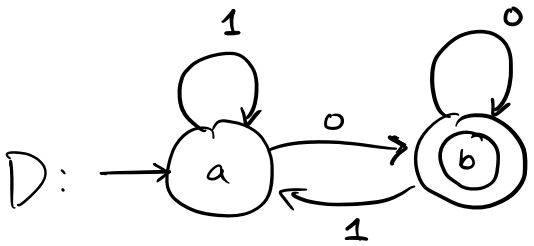
#̇ 0011 #̇ 101 #00 #1010

3. Use our program for matching strings to test if the strings next to the #̇ symbols are equal. If so, reject.

4. – If possible, move the second marked #̇ to the right and repeat (3).
   – If the second mark is on the last #, increment the first mark by one #, and move the second mark back next to it. Repeat (3).
   – If both marks are at the last two # already, accept."

Simulating automata

$(Q, \Sigma, q_0, F, \delta)$



$\Sigma = \{1, 0\}$

"ends in 0" machine.

Encoding of D:

$\langle D \rangle = [\{a, b\}, \{0, 1\}, a, \{b\},$
$\{[a, 0, b], [a, 1, a], [b, 0, b], [b, 1, a]\}]$

This is a string over $\Sigma = \{[,], ,, \{, \}, a, b, 0, 1\}$

This can be input to a TM!

$A_{DFA} = \{\langle D, w \rangle \mid D$ is an encoded DFA on $\Sigma = \{0, 1\}$, $w \in \{0,1\}^*$, and D accepts $w\}$.

$\langle$"encoded somehow"$\rangle$

How can we recognize/decide this language?

$M_{A_{DFA}}$ = "On input s:

1. Make sure s encodes a DFA D according to our encoding, and a binary string w. Reject if not.
2. Read <D> and write down the start state at the rightmand end of the tape.
3. Mark the first character of w with a dot (·).
4. Until I reach the last input character, do the following:
   - read the current DFA state, input char, and find the matching entry on the encoded trans. function δ.
   - follow δ: update my DFA state at the right end of the tape, increment my (·) on the input string.
5. Accept if the current DFA state written at the end of the string is in F after we finish reading w. Otherwise, reject."

---

5) $A_{TM} = \{\langle M, w\rangle \mid M \text{ is an encoded Turing Machine, and } M \text{ accepts } w\}$

"some encoding"

U = "on input s:

1. check if s encodes a TM M and string w.
2. simulate M on w by following M's transition function (just as for A_{DFA}).
3. accept if M(w) accepts. reject if M(w) rejects."

"M run on input w"

what if $M(w)$ loops?

we can't write an if/else statement to catch this.

U is the "universal TM." U recognizes A_{TM}, but it is not a decider.

Break: Back at 3:19.

TM programming puzzles.

(1) $S = \begin{cases} 0 & \text{if there is life on Mars} \\ 1 & \text{if not.} \end{cases}$

$A = \{S\}$. Is A decidable? (Decidable := TM always accepts $w \in A$, rejects $w \notin A$.)

We know that either $A = \{0\}$, or $A = \{1\}$. Both of these are decidable.

(2) If B is Turing-decidable, is $\overline{B}$ Turing-decidable?

Yes — flip accept/reject or simulate.

If B is Turing-recognizable, is $\overline{B}$ Turing-recognizable?

M_B says yes on $w \in B$, loops on strings $\notin B$.

(*) If B, $\overline{B}$ recognizable: by simulating both machines we can decide B.

(3) $E_{DFA} = \{\langle D \rangle \mid D \text{ is an encoded DFA that rejects all strings.}\}$

Can we decide E_{DFA}?

Using BFS? Mark all states reachable from $\rightarrow$O, accept if there are no reachable ◎.

(+) $A_{REX} = \{\langle R, w \rangle \mid R \text{ is an encoded reg. ex that generates } w\}$

Can we decide A_{REX}?

We have a deterministic procedure for turning

Reg. ex's $\Rightarrow$ equivalent NFAs

NFAs $\Rightarrow$ equivalent DFAs

Follow this procedure and then use our machine for A_{DFA}.

$(0 \cup 1)^*$

Church–Turing Thesis:

"tasks that can be done by any well-specified procedure" $\approx$ "tasks TMs can do"

Turing-Complete systems:

:= can do everything a TM can do.

- (almost) all programming languages.
- LaTeX
- Conway's Game of Life.
- Excel and Powerpoint
- Minecraft, Portal, Magic the Gathering