Announcements:

- Quiz on Thur in class — note on typo in lecture & notes
- HW2 grades + feedback will be released today
  (HW2 solutions + Handout 2 already released)

$L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$

- After quiz we will
  - open HW2 for regrades
  - reopen checkup 4 for further attempts

Last time:

- TM model can simulate k-tape TM
- in fact, it can simulate many other models, (many ... any...?)
  including python, quantum computers, AI models, whatever you use on your laptop, ...

Church-Turing thesis

Every reasonable model of computation (or notion of "algorithm") can be simulated by a TM

Thus, to show a lang is TM-recognizable / TM-decidable

you can use a high level algorithm description (ie. "pseudo-code")

(as long as every step is implementable)

think about whether you could write this step in your favorite programming lang. with all the libraries available.

Example: $L = \{a^n b^n \mid n \geq 0\}$ is decidable:

* Formal level algorithm
* Implementation level algorithm
} see lecture 11

* high level algorithm:

M = "On input $x \in \{a,b\}^*$:
check whether $x = a^n b^n$ for some $n \geq 0$
if so - accept
else - reject"

This is a sufficient description, because each step can be implemented (checking if x is of the form $x = a^n b^n$ for some $n \geq 0$ can be done, eg by counting...)

M is a decider, and $L(M) = \{a^n b^n \mid n \geq 0\}$

Recall: To prove L is TM-recognizable:

- construct algorithm M
- prove $x \in L \Rightarrow M$ accepts x
  $x \notin L \Rightarrow M$ rejects x or M runs forever on x

equivalently: $x \in L \Leftrightarrow M$ accepts x

To prove L is TM-decidable

- construct algorithm M
- prove $x \in L \Rightarrow M$ accepts x
  $x \notin L \Rightarrow M$ rejects x
  ~~or M runs forever on x~~

equivalently: (1) $x \in L \Leftrightarrow M$ accepts x, AND (2) M halts on all inputs.

Thm: The class of TM-decidable languages is closed under complement

proof: Let L be a decidable language, and let M be a TM decider s.t. $L(M) = L$.
Construct M' that is the same as M, except with accept/reject states flipped

Construction

$x \in \overline{L} \Rightarrow x \notin L \Rightarrow M$ rejects $x \Rightarrow M'$ accepts x
(by def; M decider for L; by construction of M')

$x \notin \overline{L} \Rightarrow x \in L \Rightarrow M$ accepts $x \Rightarrow M'$ rejects x

analysis/proof

Thus, we proved M' is a decider for $\overline{L}$. ☒

(end of review from last time)

Q: Does the above proof work to show the class of TM-recognizable languages closed under complement?

A: No! If M is not a decider, flipping accept/reject doesn't give you a TM for $\overline{L(M)}$ because if M runs forever on x, $x \in \overline{L}$ but M' also runs forever on x. (instead of accepting)

Counter example: See checkup 4.

The proof fails here: $x \notin L \Rightarrow$ M rejects x no longer true, for general M, $x \notin L \Rightarrow$ M rejects x or runs forever on x.

Q: Is the class of TM-recognizable lang. closed under complement?

A: We will show later: NO!

Thm: The class of Tm-decidable languages is closed under union

proof: Let L_1, L_2 be decidable. Let M_1 be a decider for L_1, M_2 a decider for L_2. Construct a decider M for $L_1 \cup L_2$ as follows:

M: "on input x 1

Run M_1 on x 2

if it accepts - accept 3

Run M_2 on x 4

if it accepts - accept 5

note: we only get here if M_1, M_2 both rejected ↖ Reject " 6

Analysis:

$x \in L_1 \cup L_2 \Rightarrow x \in L_1$ or $x \in L_2 \Rightarrow M_1$ accepts x or M_2 accepts x

$\Rightarrow$ M accepts x [on line 3 or 5]

$x \notin L_1 \cup L_2 \Rightarrow x \notin L_1$ and $x \notin L_2 \Rightarrow M_1$ rejects x and M_2 rejects x

$\Rightarrow$ M rejects x [line 6]

So, M is a decider for $L_1 \cup L_2$, and we proved thm ☒

Q: Does the above proof work to show the class of TM-recognizable languages closed under union?

A: No! If x is such that M_1 runs forever on x, but M_2 accepts x, then $x \in L_1 \cup L_2$ but M will run forever in line 2! (thus, M does not recognize the right language $L_1 \cup L_2$)

Q: Is the class of TM-recognizable lang. closed under union?

A: yes! using another construction:

just run M_1 and M_2 in parallel

eg on a 2 tape TM, run M_1 on first tape M_2 on second.
Or repeat: run one step of M_1 on x then one step of M_2 on x

Thm: The class of TM-recognizable languages is closed under union.

→ was covered in class informally, this is the full proof added after class.

proof: Let L_1, L_2 be recognizable, Let M_1, M_2 be TMs s.t. M_1 recognizes L_1, M_2 recognizes L_2.

We want to construct a TM recognizing $L_1 \cup L_2$:

M: "on input x

1 Run M_1 and M_2 on x, in parallel

2 - if either of them accepts, accept

3 - if both reject, reject"

Analysis:

$x \in L_1 \cup L_2 \Rightarrow x \in L_1$ or $x \in L_2 \Rightarrow$ M_1 accepts x or M_2 accepts x

$\Rightarrow$ M accepts x [in line 2]

$x \notin L_1 \cup L_2 \Rightarrow x \notin L_1$ and $x \notin L_2 \Rightarrow$ Each of M_1, M_2 either rejects x or runs forever on x

$\Rightarrow$ either M rejects x [line 3, in case both M_1, M_2 reject)

or M runs forever on x [line 1, in case one of them runs forever on x]

Thus, M recognizes $L_1 \cup L_2$. We proved that class of TM-recognizable languages is closed under union.

Exercise: show closure under intersection, for both decidable and recognizable languages → is it easier/harder than what we did for union?

Theorem: L is decidable $\iff$ both L and $\overline{L}$ are recognizable

proof: ($\Rightarrow$): Let L be decidable, Let M be a decider for L. $\Rightarrow$ L is also recognizable (decidable is a special case of recognizable). Also, L decidable $\Rightarrow \overline{L}$ decidable by theorem we saw $\Rightarrow \overline{L}$ is recognizable.

($\Leftarrow$): Let L be s.t. $L, \overline{L}$ both recognizable let M_1, M_2 be TMs s.t. $L(M_1) = L$, $L(M_2) = \overline{L}$.

(i.e: M_1: $x \in L \Rightarrow M_1$ accepts x; $x \notin L \Rightarrow M_1$ rejects x or runs forever on x

M_2: same for $\overline{L}$, so. $x \notin L \Rightarrow M_2$ accepts x; $x \in L \Rightarrow M_2$ rejects x or runs forever on x.)

We want to construct a decider M for L.

Construction:

M: "on input x:

1 Run M_1 and M_2 on x in parallel

2 if M_1 accepts, accept

3 if M_2 accepts, reject "

$x \in L \Rightarrow M_1$ accepts $x \Rightarrow M$ accept [in line 2]

$x \notin L \Rightarrow x \in \overline{L} \Rightarrow M_2$ accepts $x \Rightarrow M$ rejects [in line 3]

So M is a decider for L. ☒

Note: For general M_1, M_2 the above M is not guaranteed to be a decider, since M_1, M_2 could both run forever on some input x. However, here it can't happen, since $\forall x$ exactly one of them will accept (since $L(M_1) = \overline{L(M_2)}$

Next: let's see some examples of decidable (or recognizable) languages

Important note on encodings:

- Every algorithm/TM can be encoded as a string (equivalently, as an integer)

eg. we define TM $M=(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ - this can be encoded as a string over some alphabet

a string $\rightarrow (\{q_0, q_1, q_2\}, \{a, b\}, \ldots)$

or a C++ program - also clearly a string, ...
And a string over any alphabet can be encoded using binary alphabet $\{0,1\}$ (which can also be viewed as some integer in binary).

We will not care about specifics of encoding.
The only important part is that there is some agreed upon encoding. We denote it by $\langle M \rangle$
the encoding of TM M

Since $\langle M \rangle$ is just a string, it can be input to another algorithm.

- given an input string x, we can check whether $x = \langle M \rangle$ is a valid encoding of a TM

(and if it is, we can do things with M, like count how many states it has, or run it on some input, or flip its accept and reject states, or run other algorithms...)

- Other objects can also be encoded as strings

e.g. DFAs, NFAs, regular expressions, integers, graphs, ...

- we'll use $\langle \dots \rangle$ to denote encoding
- we'll construct algorithms that take such encodings and check something about the encoded object.

Examples:

$A_{DFA} = \{ \langle D, w \rangle \mid D \text{ is a DFA, } w \text{ is a string, and } D \text{ accepts } w \}$

"Acceptance problem for DFAs"

$A_{NFA} = \{ \langle N, w \rangle \mid N \text{ is a NFA, } w \text{ is a string, and } N \text{ accepts } w \}$

"Acceptance problem for NFAs"

$E_{DFA} = \{ \langle D \rangle \mid D \text{ is a DFA, and } L(D) = \emptyset \}$

"Emptiness problem for DFAs"

(we'll show algorithms next time)

How would you prove these are decidable?