Lecture 13:

* Quiz on Thursday !!! (exciting, fun and more)
* Hw 2 grades + feedback will be released today
  (solutions + Handout 2 already released)
* After quiz we will
  - open HW2 for regrades
  - re-open checkup 4 for further attempts

The usual question:

On input w, a TM M can:

* accept
* reject
* run forever

Last time:

- A standard TM can simulate a k-tape TM
  => in fact, it can simulate many other models, including python, quantum computers, whatever you use on your laptop etc...

Church - Turing thesis

Every reasonable model of computation (our notion of algorithms) can be simulated by a TM

To show a language is TM recognizable / TM decidable

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

as long as: every step is implementable

↳ think about whether you could write that step in your favorite programming language (with all the libraries you like)

(ie: If William thinks about the color purple tomorrow do X else do Y)

↳ kind of impossible to implement

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

* Formal level algorithm (5-tuple)
* Implementation level algorithm

) lecture 11

↳ explain how the TM works (move head left/right etc...)

* high level algorithm:

M

"On input $x \in \{a,b\}^*$:

check whether $x = a^n b^n$ for some $n > 0$

If yes: accept
else: reject

This is a sufficient description, because each step can be implemented (checking if x is of the form $x = a^r b^r$ can be done by counting #of a's and b's, and checking all a's are before b's)

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

($x \in L \iff$ M accepts)

or M runs forever

M accepts $\Rightarrow x \in L$ equiv $x \notin L \Rightarrow$ M doesn't accept

to prove L is TM-decidable

- construct algorith M
- prove $x \in L \Rightarrow$ M accepts x
  $x \notin L \Rightarrow$ M rejects x

($x \in L \iff$ M accepts + and M halts on all inputs)

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

If L is decidable $\Rightarrow \bar{L}$ is decidable

proof: Let L be a decidable language and let

construction: M be a TM decider s.t $L(M) = L$
Construct M' that is the same as M, except with accept/reject states flipped
(swap q_{acc} & q_{rej})

analysis: $x \in \bar{L} \Rightarrow x \notin L \Rightarrow$ M rejects $x \Rightarrow$ M' accept x

proof ‖ $x \notin \bar{L} \Rightarrow x \in L \Rightarrow M$ accepts $x \Rightarrow M'$ reject x

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

Does the above work for the class of TM-recogni-zable languages to show closure under complement?

Answer No! flipping accept/reject states doesn't work.

Why? Say M recognizes L, and we build M' by switching the accept/reject states of M.

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

If $x \in \bar{L} \Rightarrow x \notin L \Rightarrow$ { M rejects $x \Rightarrow M'$ accepts x ; M runs forever on x }

M' runs forever on x

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

A: We will later show the answer is No!

(there exists a language L that is recognizable but $\bar{L}$ isn't)

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

Proof: Let L_1, L_2 be TM-decidable languages.
We want to show $L_1 \cup L_2$ is decidable.
Let M_1, M_2 be TM deciding L_1 and L_2.
We build a decider for $L_1 \cup L_2$ as follows:

M: "on input x

- Run M_1 on x — 2
  if it accepts: accept — 3
- Run M_2 on x: — 4
  if it accepts: accept — 5

Reject — 6

Analysis: If $x \in L_1 \cup L_2$: then $x \in L_1$ or $x \in L_2$

- If $x \in L_1$ => M_1 accepts on line 2
  M accepts on line 3
- If $x \notin L_1$ but $x \in L_2$ => M_1 rejects on line 2
  so we go to line 4. M_2 accepts x. So M accepts on line 5.

If $x \notin L_1 \cup L_2$ => M_1 rejects on line 2

M_2 rejects on line 4

M rejects on line 6

So M accepts iff $x \in L_1 \cup L_2$ & M never runs forever

so M decides $L_1 \cup L_2$

Is the class of TM-recognizable languages closed under union?

W.T.S if L_1, L_2 are TM-recognizable, $L_1 \cup L_2$ is TM-recognizable.

Yes!

Proof: Let L_1, L_2 be TM-recognizable languages. We have TM M_1, M_2 recognizing L_1, L_2.

We build a reccgnizer for $L_1 \cup L_2$ as follows:

M: "On input x:

Run M_1 and M_2 on x in parallel 1

If either M_1 or M_2 accepts: 2

accept x 3

If both M_1 and M_2 reject ↵ 4
reject x ↵ 5

in parallel? We could run one step of M_1 on x then one step of M_2 on x and repeat

Or we use two tapes, run M_1 on x on the first tape, M_2 on x on the second tape

Analysis: If $x \in L_1 \cup L_2 \Rightarrow x \in L_1$ or $x \in L_2$

$\Rightarrow M_1$ or M_2 accepts x on line 2

$\Rightarrow M$ accepts x on line 3,

If $x \notin L_1 \cup L_2 \Rightarrow$ then neither M_1 or M_2 will accept x

so either M will run forever or M will reject on line 5. ↳ when M_1 or M_2 runs forever on x

So M recognizes $L_1 \cup L_2$. So TM-recognizable languages are closed under union.

Exercise: show closure under $\cap$ for decidable and recognizable.

An important note on encoding:

- We want to give as input to our algorithms:
  - graph, DFA, TM

- Every TM / algorithm can be encoded as a string: we won't care about the details of that encoding. The only important part is that there is an agreed upon encoding: * we can check if a string is a valid encoding of a TM / DFA / graph
  * we can decode it.

Example: a TM => $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$

a DFA => $(Q, \Sigma, \delta, q_0, F)$ ← these are strings

We'll use $\langle \dots \rangle$ to denote an encoding. Ex $\langle M \rangle$ represents an encoding of a Turing Machine.

- We'll build algorithms that take such strings as input and check something about the object encoded.

Ex: $A_{DFA} = \{ \langle D, w \rangle \mid$ D is a DFA, w is a string (over any alphabet) and D accepts w $\}$ (accept)

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

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

$EQ_{DFA} = \{ \langle D_1, D_2 \rangle \mid D_1, D_2$ are DFAs, $L(D_1) = L(D_2) \}$

$A_{TM} = \{ \langle M, w \rangle \mid M$ is a Turing and M accepts w$\}$

Ex: A_{DFA} is decidable. We build a decider M for A_{DFA}:

$A_{DFA} = \{ \langle D, w \rangle \mid$ D is a DFA, w is a string (over any alphabet) and D accepts w $\}$ (accept)

M: On input x

(1) Check $x = \langle D, w \rangle$ i.e x is a valid encoding of a DFA D and a string w
If not reject.

(2) Run D on w:

If D accepts : accept (3)
If D rejects : reject (4)

Analysis: If $x \in A_{DFA}$, then $x = \langle D, w \rangle$ where D is a DFA that accepts w. So on line 2 D accepts and M accepts x on line (3)

If $x \notin A_{DFA}$, then (1) $x \neq \langle D, w \rangle$ => we reject on line (1)

else (2) $x = \langle D, w \rangle$ but D rejects w, so M rejects x on line (4)

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

is decidable.