Lecture 24 Last Class!

Announcements:

HW3 solns posted
HW4 solns posted later today
* Test 2 Review this Friday 4-6!

check announcements for zoom link if you can't attend in person.

Today:

Review for Test 2

Closing comments

* I will post solns to Review Qs in next day or so (but try to solve yourself first!)

This course is about how hard problems are

"languages" are problems. For now think of languages as problems.

We will characterize problems into some classes

Regular languages / DFAs

Context-free languages / PDAs

Computable/decidable languages / TMs

Complexity Theory: P, NP, NP-complete

Examples of Languages

$L1 = \{ w \in \{0,1\}^* \mid w \text{ has an even number of 1's} \}$

$L2 = \{ w \in \{0,1\}^* \mid w \text{ ends with } 011 \}$

$L3 = \{ w \in \{0,1\}^* \mid w = 0^n 1^n,\ n \geq 1 \}$

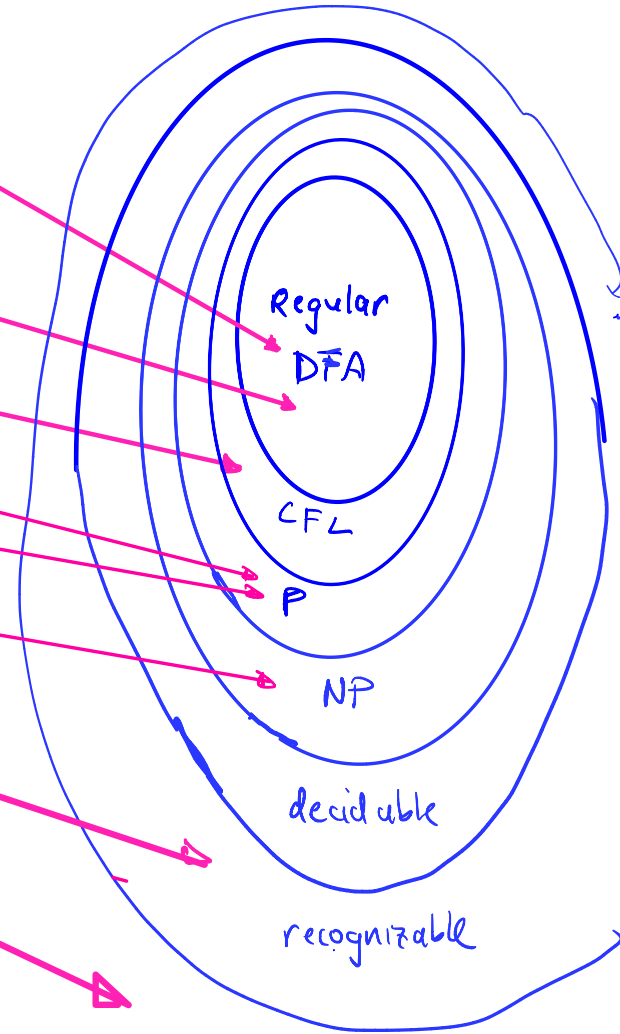
$L4 = \{ w \in \{0,1,2\}^* \mid w = 0^n 1^n 2^n,\ n \geq 1 \}$

$L5 = \{ w \in \{0,1\}^* \mid w \text{ encodes a connected graph} \}$

$L6 = \{ (g,k) \mid g \text{ contains a } k\text{-clique} \}$

$L7 = \{ \langle M, x \rangle \mid M \text{ halts on } x \}$

$L8 = \{ \langle M, x \rangle \mid M \text{ does not halt on } x \}$



Example of a problem that is decidable but probably not in NP:

MajSAT: input is a 3CNF formula ϕ over $x_1 \ldots x_n$

accept ϕ iff the # of satisfying assignments is $\geq 2^n/2$

Recursively Enumerable (RE) / Recognizable Languages

A Language $L \subseteq \Sigma^*$ is RE or recognizable if
there exists a TM M such that $\mathcal{L}(M) = L$.

That is: $\forall w \in L$ M on w halts and accepts, and
$\forall w \notin L$ M on w either halts & rejects or never halts

A Language $L \subseteq \Sigma^*$ is recursive or decidable if
there exists a TM M such that $\mathcal{L}(M) = L$ and M
halts on all inputs

That is: $\forall w \in L$ M on w halts and accepts
and $\forall w \notin L$ M on w halts and rejects

Church-Turing Thesis

Every reasonable model of computation can be simulated by a TM.

In other words, TMs can compute any function that can be computed by any current/future computational device

CLOSURE PROPERTIES

1. L recursive $\Rightarrow$ L r.e.

2. closure of recursive languages under $\cap, \cup, \neg$:

   L_1, L_2 recursive $\Rightarrow$ $L_1 \cup L_2$, $L_1 \cap L_2$, $\neg L_1$, $\neg L_2$ are recursive

3. Closure of r.e. languages under $\cap, \cup$

   L_1, L_2 r.e. $\Rightarrow$ $L_1 \cup L_2$, $L_1 \cap L_2$ are r.e.

What about closure of r.e. under $\neg$?

4. L is r.e. and $\bar{L}$ is r.e. $\Rightarrow$ L is recursive

   Let M_1 be TM that accepts L, M_2 same for $\bar{L}$

Say L is r.e. but not recursive.
what about $\bar{L}$? It is not r.e.

If L is recursive $\rightarrow \bar{L}$ is recursive

If L is r.e. $\Rightarrow \bar{L}$ could be r.e.
or not

Decidability of other Languages

① $D = \{\langle M \rangle \mid M(\langle M \rangle) \text{ does not accept}\}$ is not r.e. ← proof by diagonalization

Diagonal Language

② $\overline{D} = \{\langle M \rangle \mid M(\langle M \rangle) \text{ halts and accepts}\}$

is r.e. but not recursive

($\forall L \subseteq \Sigma^*$
If L recursive
then $\overline{L}$ also
recursive)

③ $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ is r.e. but not recursive

$\overline{A_{TM}}$ is not r.e.

④ $HALT = \{\langle M, w \rangle \mid M \text{ halts on } w\}$. $\overline{HALT}$ Not r.e.

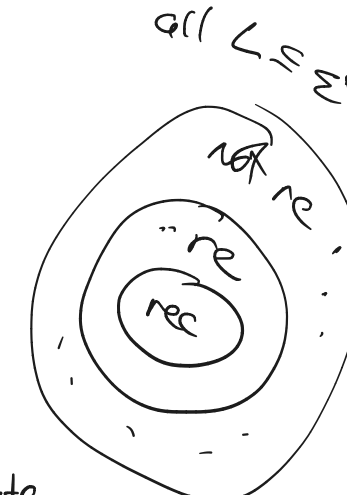
⑤ Nonempty : r.e. Not recursive

Empty : Not r.e.

Tips for characterizing a given Language

as (1) recursive; (2) recursive but not re; (3) Not r.e.

① Try obvious algorithms to see if you think L is recursive / r.e. (dovetailing technique useful to show r.e.)
Watch out for tricks -- if L defined based on some property of the machine (& not a property of L)



② To prove L is not r.e., sometimes helpful to look at $\bar{L}$
(If $\bar{L}$ is r.e. but not recursive then L not r.e.)

③ get reduction in correct direction!

④ Sometimes in reduction, need to construct an intermediate TM that ignores its own input.

The Class "P"

<u>Defn</u> Let $t : \mathbb{N} \to \mathbb{N}$ ($t \approx$ runtime)

$\text{TIME}(t(n)) = \{ L \mid L$ is a language decided by a $O(t(n))$-time TM $\}$

<u>Defn</u> $t : \mathbb{N} \to \mathbb{N}$ is <u>polynomial</u> if $t(n) = O(n^k)$ for some $k \geq 0$

<u>Ex</u> $t(n) = n^2$, $t(n) = n$, $t(n) = n \log n$ are polynomial

($t(n) = n^{\log n}$ or $t(n) = 2^n$ are <u>not</u> polynomial.)

<u>Defn</u> $P = \{ L \mid L$ is a language decided by a TM running in <u>polynomial</u> time $\}$

* We think of problems in P as those that have relatively efficient algorithms.

The (even more Famous) class NP

Defn 1 NP = { L | L is a language decided by a Nondeterministic TM running in polynomial time }

Equivalent Defn of NP

A verifier for language $L \subseteq \{0,1\}^*$ is an algorithm
$L = \{ w \mid V(w,c) \text{ accepts} \}$ where c is an additional string that we call a certificate or proof

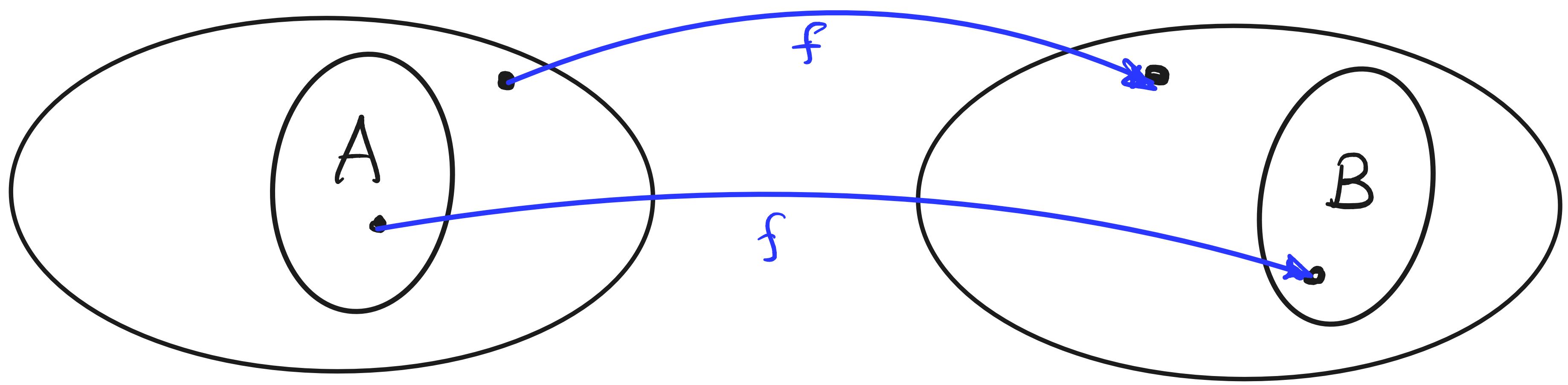
A verifier is polynomial-time if it runs in time polynomial in $|w|$.

* Note that if A is a polytime verifier then $|c|$ must also be polynomial in $|w|$.

Defn 2 NP = { L | L has a polytime verifier }

NP - completeness

Definition Language A is polynomial-time (mapping) reducible to B (written $A \leq_p B$) if there is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \iff f(w) \in B$



Definition

- A language $B \subseteq \{0,1\}^*$ is NP-hard if for every $A \in NP$ there is a polynomial time reduction from A to B ($A \leq_p B$)

- $B \subseteq \{0,1\}^*$ is NP-complete if: (i) B is in NP and
(ii) B is NP-hard

NP-completeness

Cook-Levin theorem

For every $k \geq 3$ 3-SAT is NP-complete

To show another language L is NP complete we just need to show:

(1) $L \in NP$

(2) Show $L' \leq_p L$

for some NP-complete language L'

Examples of other NP-complete Languages

(1) CLIQUE

(2) HAMPATH

Test 2 Format (Similar format as Test 1)

1. True False Question

2. Prove a language is recursive/re/not recursive

3. Prove a language is NP complete

4. short answer questions

5.

Computer Science Theory, Test 2 Review Problems
Prof. Toniann Pitassi

1. Answer True or False for each statement. No justification is needed.

   T (a) $n = O(n^2)$ — $n = O(n)$, $n = O(\frac{n}{4})$

   F (b) $n \log n = O(n)$

   (c) $n^n = O(2^n)$

   Let $A \leq_m B$ → (d) Let A be mapping reducible to B. If B is decidable then A must be decidable.

   (e) Let A be mapping reducible to B. If A is decidable then B must be decidable.

   (f) If the complement of a language L is not recognizable then both L and $\neg L$ are not recognizable.

   (g) If A is NP-complete, $A \subseteq B$, and B is in NP then B is NP-complete.

   (h) If B is NP-complete and $A \subseteq B$ and A is in NP then A is NP-complete.

2. Let Double-CLIQUE denote the language consisting of all pairs (G, k) such that G is an undirected graph containing two disjoint cliques each of size k. Prove that Double-CLIQUE is NP-complete.

3. Prove that the following set is countable.

$$S = \{(i, j) \mid i \geq 0 \text{ and } j > i\}$$

4. Prove that the following set is [redacted]countable.

$$S = \{L \subseteq \{0,1\}^* | \text{the number of strings in } L \text{ is finite}\}$$

5. Prove that NP is closed under union. That is, for every $L_1, L_2 \in$ NP, $L_1 \cup L_2$ is also in NP.

6. Prove that NP is closed under concatenation.

7. Let L be the language consisting of all pairs $< M >$ such that M encodes a Turing machine and M accepts at least two inputs.

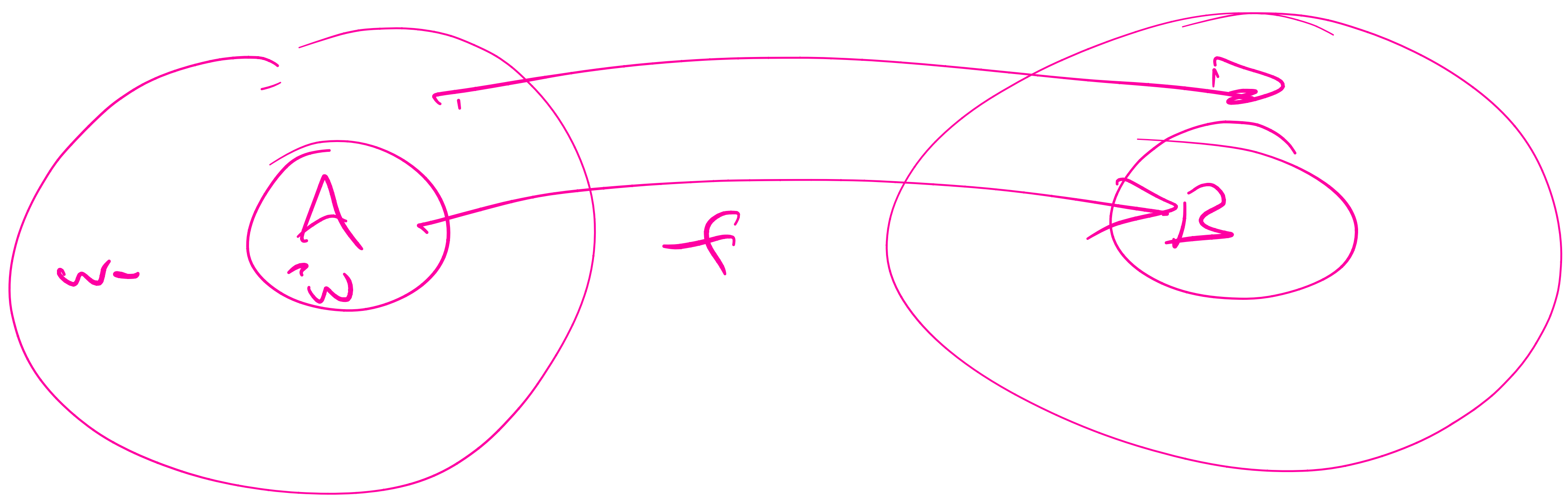
   (a) Prove that L is recognizable.

   (b) Prove that L is not decidable.

8. Recall that 3SAT is the set of all 3-CNF formulas ϕ such that ϕ is satisfiable. Let Search-3SAT be the following *search* problem: Given a 3CNF formula ϕ, output a satisfying assignment for ϕ if one exists, and otherwise output "ϕ is unsatisfiable". Prove that if 3SAT $\in$ P, then Search-3SAT can be solved in polynomial-time by a deterministic TM.

(1d) $A \leq_m B$. And B is decidable.

Is A also decidable?



To decide A:

on input w: compute $f(w)$

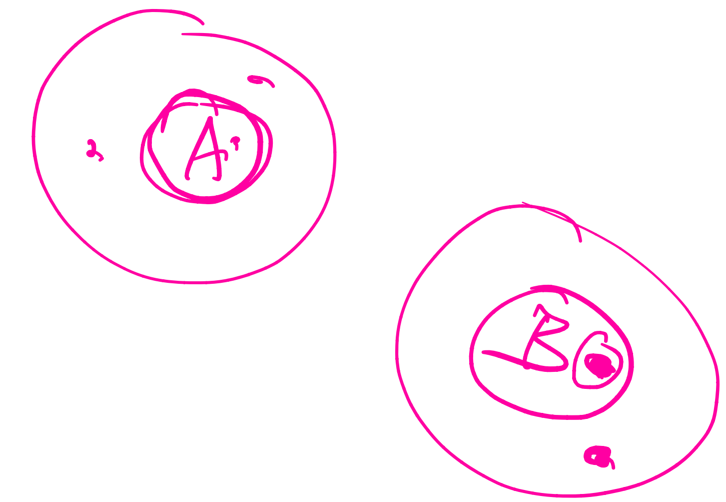
If $f(w) \in B$ (can decide this since B is decidable)

then $w \in A$

If $f(w) \notin B$ then $w \notin A$

(1e) $A \le_m B$. Now A is decidable.

Is B decidable? False.



want to show a counterexample.

Let $A = \{ w \in \{0,1\}^* \mid w \text{ ends in a } 1 \}$ ← decidable

Let B = HALT ← not decidable

Show: $A \le_m B$

$f(w)$: Check if w ends in a 1

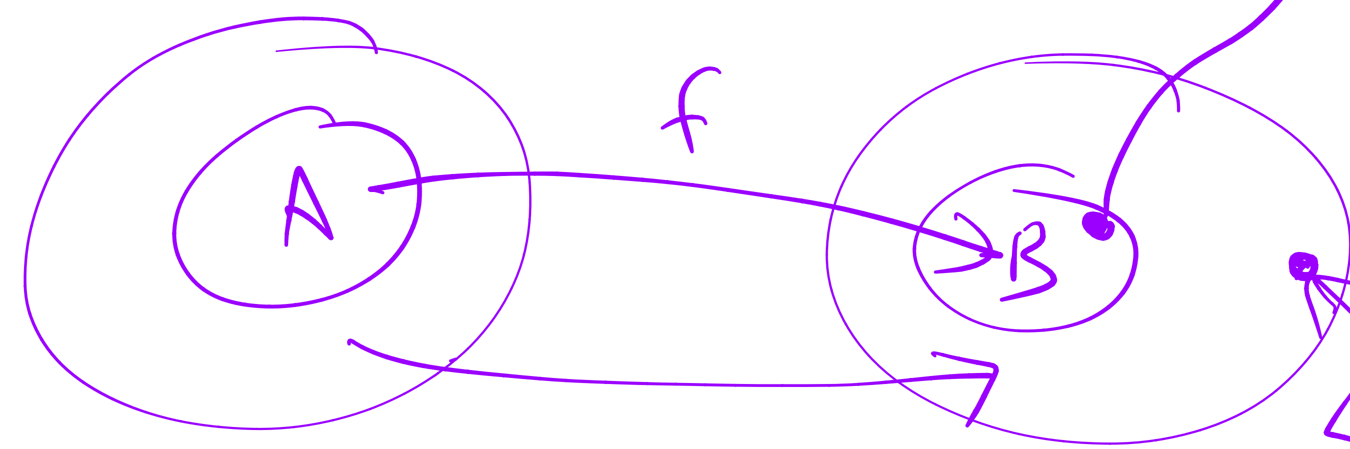
if yes then map $f(w) \longrightarrow \langle M_{\text{ALWAYS-HALT}}, 0 \rangle$

If no then map $f(w) \longrightarrow \langle M_{\text{ALWAYS-LOOP}}, 0 \rangle$

where $M_{\text{always halt}}$: halts immediately on all inputs

$M_{\text{ALWAYSLOOP}}$: loop forever on all inputs

$A \longrightarrow B$



$\langle M_{halt} \rangle$ = ignore input & halt

$\langle M_{Loop} \rangle$: ignore input and infinitely loop forever

If B decidable then A is decidable True

If A is decidable then B is decidable FALSE

Counterexample:

$B = \text{halt}$

$A = \{ w \in \{0,1\}^* \mid w \text{ has one } 1 \}$

$f(w)$: see if w has $\geq$ one 1 if so $f(w) \rightarrow \langle M_{HALT} \rangle$

ow $f(w) \rightarrow \langle M_{Loop} \rangle$

(7) $L = \{ \langle M \rangle \mid M \text{ accepts at least 2 inputs} \}$ $\quad \mathcal{L}(M) \subseteq \{0,1\}^*$

(a) prove L is r.e.

Warmup (Easier): $L' = \{ \langle M \rangle \mid M \text{ accepts at least one input} \}$ is r.e.

(Let $x_1, x_2, \ldots$ be an enumeration of all strings in $\{0,1\}^*$)

Alg:

For $t = 1, 2, \ldots$

Simulate M on strings $x_1 \ldots x_t$ for t steps each.

If M accepts any of them $\rightarrow$ halt & accept

OW

(7.) $L = \{ \langle M \rangle \mid M$ accepts at least 2 inputs$\}$ $\mathcal{L}(M) \subseteq \{0,1\}^*$

(a) prove L is r.e.

Alg:

For $t = 1, 2, \ldots$

Simulate M on the first t strings, $x_1 \ldots x_t$ each for t steps

Keep a count for how many are accepted.

If count reaches 2 halt and accept

7. $L = \{ \langle M \rangle \mid M \text{ accepts at least 2 inputs} \}$

(b) Prove L is not recursive

~~use~~

HALT not recursive

HALT is r.e.

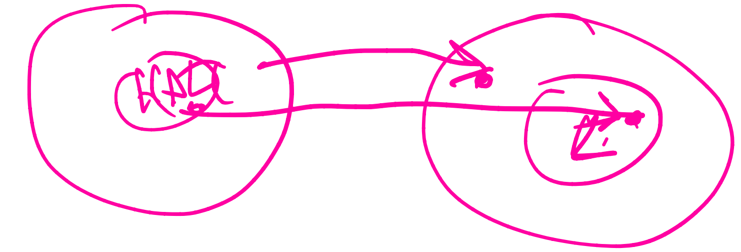
A_{TM} is not recursive

it is r.e.

To prove L is not recursive we will give a Turing / mapping reduction $\cancel{f} : \text{HALT} \longrightarrow L$

$\text{HALT} = \{ \langle M, x \rangle \mid M \text{ halts on } x \}$

7. $L = \{ \langle M \rangle \mid M$ accepts at least 2 inputs $\}$



(b) Prove L is not recursive

~~use~~

HALT not recursive

HALT is re.

A_{TM} is not recursive

it is r.e.

Assume L is decidable, and

Let N be a TM that decides L.

We will use N to create M that decides Halt:

M: on input $\langle M, x \rangle$

want

$f: \langle M, x \rangle \longrightarrow \langle M' \rangle$

s.t. M halts on x iff

M' accepts ≥ 2 inputs

M': on input w

~~if $w =$ 'o' halt & accept~~

~~ow (for all $w \neq$ 'o')~~: simulate M on x

if M halts on $x \Rightarrow$ halt & accept w

Mapping reduction f on input $\langle M, x \rangle$ out

$f(\langle M, x \rangle) \rightarrow \langle M' \rangle$

where M' is

M': on input w

~~if w = '0' halt & accept~~

~~ow (for all $w \neq$ '0')~~: simulate M on x

if M halts on x $\Rightarrow$ halt & accept w

want

$f: \langle M, x \rangle \longrightarrow \langle M' \rangle$

s.t. M halts on x iff

M' accepts ≥ 2 inputs

show

① $\langle M, x \rangle \in HALT \Rightarrow f(\langle M, x \rangle) \in L$

By defn of M'

if $\langle M, x \rangle \in$ Halt

then $\mathcal{L}(M')$ = all strings

so $\langle M' \rangle \in L$

② $\langle M, x \rangle \notin HALT \rightarrow f(\langle M, x \rangle) \notin L$

By defn of M'

if $\langle M, x \rangle \notin$ Halt then $\mathcal{L}(M') = \phi$ so $\langle M' \rangle \notin L$

8. Search-3SAT : Input is a 3SAT Formula ϕ

Output : $\begin{cases} \text{UNSAT} & \text{if } \phi \text{ is unsatisfiable} \\ \text{a satisfying assignment} & \text{if } \phi \text{ is satisfiable} \end{cases}$

Prove: If $3SAT \in P$ then Search-3SAT $\in P$

Example: Let $\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3)(\bar{x}_1 \vee \bar{x}_2 \vee x_4)(x_3)(\bar{x}_4)$

(For 3CNF ϕ over $x_1 \ldots x_n$

$$\underbrace{8\binom{n}{3}}_{\text{all clauses of size 3}} + \underbrace{4\binom{n}{2}}_{2} + \underbrace{2\binom{n}{1}}_{1} = O(n^3)$$

Idea On input ϕ $\qquad \phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3)(\bar{x}_1 \vee \bar{x}_2 \vee x_4)(x_3)(\bar{x}_4)$

1. Call 3SAT (ϕ)

   If ~~accepts~~ rejects $\rightarrow$ halt & output UNSAT

2. OW (ϕ is satisfiable):

   See if $\exists$ a sat. ass with $x_1 = 1$.

   If so recursively see if $\exists$ sat. ass with $x_1=1, x_2=1$

   $\vdots$

   If not $\exists$ a sat ass w/ $x_1 = 0$

   so rec. see if $\exists$ sat. ass with $x_1 = 0, \ldots$

Let $x_1 = 1$. Let $\phi|_{x_1=1} = \phi$ where we substitute $x_1 = 1$ and simplify

in our example $\phi|_{x_1=1}$: $(\bar{x}_2 \vee x_4)(x_3)(\bar{x}_4)$

Call 3SAT($\phi_{x_1=1}$).

Idea on input ϕ $\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3)(\bar{x}_1 \vee \bar{x}_2 \vee x_4)(x_3)(\bar{x}_4)$

alg $\alpha = \{\}$, ϕ

Loop: $i = 1, 2, \ldots n$:

Let $x_i = 1$. Let $\phi = \phi|_{x_i = 1}$

(in our example $\phi|_{x_1=1}$: $(\bar{x}_2 \vee x_4)(x_3)(\bar{x}_4)$)

Call 3SAT($\phi_{x_i=1}$).

If accepts, let α = old $\alpha \cup \{x_i = 1\}$, $\phi = \phi|_\alpha$

If rejects let α = old $\alpha \cup \{x_i = 0\}$, $\phi = \phi|_\alpha$

Output α