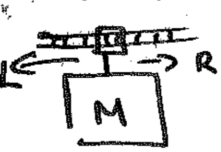
CSC463

→ Computability

TM $M = \{Q, \Sigma, \Gamma, \delta\}$ → transition function



- Q: states, $q_0, q_{acc}, q_{rej} \in Q$
- Σ → finite input alphabet
- Γ: finite tape alphabet

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

$(q, a) \mapsto (q', a', X)$

→ notice: we could also write $Q - \{q_{acc}, q_{rej}\}$

Def. M accepts x iff on input x, M reaches ~~at~~ q_{acc}

- $\mathcal{L}(M) = \{x \in \Sigma^* \mid M \text{ accepts } x\}$

We encode ~~tape~~ configurations of a TM with 3 pieces of info

1. current state
2. — symbol scanned
3. — tape

$\ldots \cancel{s_\eta}\, s_\gamma s_\beta s_\alpha\, q_X\, s_1 s_2 s_3 \ldots$

The computation of M on input ω is a series of configs $C_0, C_1 \ldots$ (where $C_0 = q_0\,\omega$)

M halts iff it reaches a halting config = {accepting conf. / rej — conf.}

Def. M is a decider iff it halts on all $\omega \in \Sigma^*$

M_c is an enumerator if it has a write-only tape (no input) and only prints out characters

→ $L \in SD$ iff $\exists M, L = \mathcal{L}(M)$

→ $L \in D$ iff $\exists M, L = \mathcal{L}(M) \wedge M$ is a decider

for decidable langs, we know which strings are NOT part as well

e.g. $PAL = \{\omega\omega^R \mid x \in \{0,1\}^*\} \in D$ ()

Church-Turing Thesis (1936)

Any 'algorithm' can be simulated ~~by~~ by a TM.

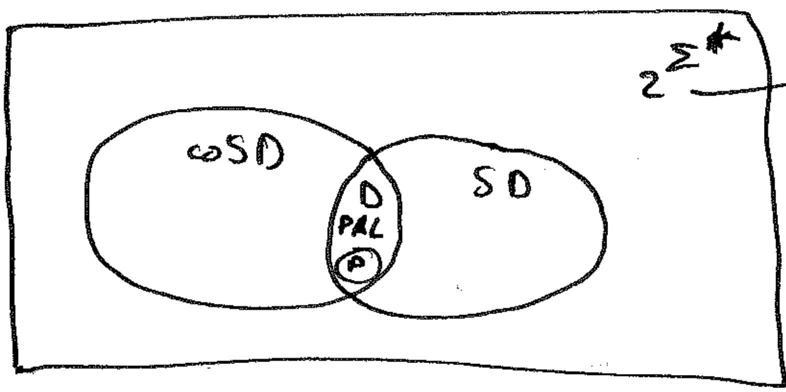
$\forall n \in \mathbb{N}$ — time complexity

Def. $P = \{L \mid \exists M\ (::TM), \exists c \in \mathbb{N}, T_M(n) \in O(n^c), L = \mathcal{L}(M)\}$

$SD = \{L \mid L \text{ is semi-decidable}\}$ — turing-recognizable, recursively enumerable

$D = \{L \mid L \text{ is decidable}\}$ — computable, recursive

$coSD = \{L \mid L \notin SD \wedge \overline{L} \in SD\}$



2^{Σ^*} → cardinality of powerset: all possible languages from fixed finite alphabet Σ.

$\langle TM, input \rangle$: any encoding of a TM in a language in 2^{Σ^*}.

e.g.

$A_{TM} = \{\langle M, \omega \rangle \mid M \text{ is } TM \wedge M \text{ accepts } \omega\}$

$All_{TM} = \{\langle M \rangle \mid M :: TM \wedge \mathcal{L}(M) = \Sigma^*\}$

$E_{TM} = \{ \text{\_\_\_\_\_\_\_\_\_\_\_\_} = \emptyset \}$

$HB = \{ \text{\_\_\_\_\_\_\_\_\_\_\_\_} \wedge M \text{ halts on } \emptyset \}$

$DiAG = \{ \text{\_\_\_\_\_\_\_\_\_\_\_\_} \wedge \langle M \rangle \notin \mathcal{L}(M) \}$

Closure

D is closed under $\cap$, $\cup$ and $\overline{}$.

SD is closed under $\cap$, $\cup$ but <u>not</u> $\overline{}$.

$DIAG = \{\langle M\rangle \mid M \text{ is a TM} \wedge \langle M\rangle \notin \mathcal{L}(M)\}$

<u>Thm</u> DIAG is not semi-decidable

<u>proof</u> Suppose $\mathcal{L}(M_D) = DIAG$

By contradiction: M_D accepts $\langle M_D\rangle$

$$\Rightarrow \begin{cases} \langle M_D\rangle \in DIAG \\ \langle M_D\rangle \notin DIAG \end{cases}$$

<u>Corollary</u> A_{TM} is not decidable

$A_{TM} = \{\langle M, \omega\rangle \mid M \text{ is a TM} \wedge M \text{ accepts } \omega\}$

A_{TM} decidable $\Rightarrow$ DIAG decidable

Church-Turing thesis

$\rightarrow$ no comp prog solves DIAG, A_{TM}

$HALT_{TM} = \{\langle M, \omega\rangle \mid M \text{ is a TM and } M \text{ halts on input } \omega\}$

<u>Thm</u> $HALT_{TM}$ is not decidable

<u>proof</u> assume $HALT_{TM}$ is decidable

given $\langle M, \omega\rangle$, we can determine whether M accepts ω.

run $HALT_{TM}$ on $\langle M, \omega\rangle$

1. if M doesnt halt $\Rightarrow \langle M, \omega\rangle \notin A_{TM}$
2. M halts on input $\omega \Rightarrow$ run M on input ω, accept iff M accepts ω —

Subset of decidable set is not necessarily decidable

Recall $\bar{A} = \{x \in \Sigma^* \mid x \notin A\}$

<u>Thm</u> A is decidable iff $A \wedge \bar{A}$ are semi-dec.

<u>proof</u> . $\Rightarrow$ ✓

. $\Leftarrow$ suppose $A \wedge \bar{A}$ are semi-dec

$\Rightarrow$ let $A = \mathcal{L}(M_1)$, $\bar{A} = \mathcal{L}(M_2)$

$\rightarrow$ design a decider M_3 as such:

. on input x : run $M_1 \wedge M_2$ on input x

. accept if M_1 accepts x

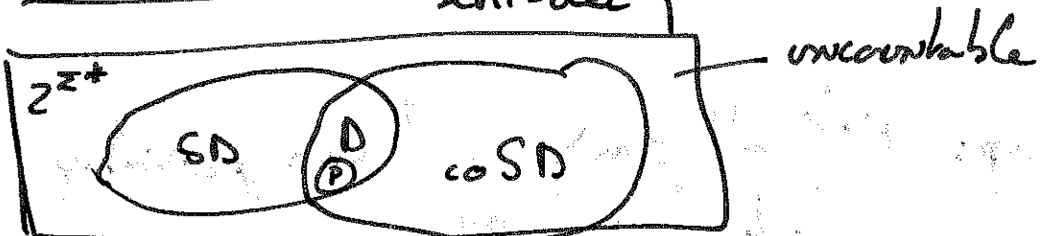
. reject if M_2 rejects x

<u>Corollary</u> $\overline{A_{TM}}$ is not semi-dec

proof: bc A_{TM} is $\neg$dec $\wedge$ is semi-dec

<u>Def</u> $D = \{A \subseteq \Sigma^* \mid A \text{ is decidable}\}$

$SD =$ ——————— semi-dec $\}$



$R(x,y)$ is a rela° on string of pairs

$R: \Sigma^* \times \Sigma^* \rightarrow \{0,1\}$

$L_R = \{\langle x,y\rangle \mid R(x,y) = 1\}$

<u>thm</u> $A \in SD$ iff $\exists$ $R(x,y)$ decidable s.t.

$$\forall x \in \Sigma^*,\ x \in A \Leftrightarrow \exists y, R(x,y)$$

y is a certificate showing $x \in A$

<u>e.g.</u> $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$

$R(x,y)$ – suppose $x = \langle M, w\rangle$

Let $y = C_0, C_1 \dots C_t$ ~ the computation of M on input ω. — must be an accepting config → finite

Thus $R(\langle M, \omega\rangle, y)$ holds iff y codes an accepting ~~state~~ config of M on input ω (R is decidable)

If x is not of the form $x = \langle M, \omega \rangle$ where M is a TM and $\omega \in \Sigma^* \Rightarrow R(x, y)$

• $x \in A_{TM}$ iff $\exists y\, R(x,y) = 0$
$x \in \langle M, \omega \rangle$

The certificate theorem

Let $A \subseteq \Sigma^*$, $A \in SD$ iff $\exists$ a decidable relation $R(x,y)$ s.t. ~~$x \in A \Leftrightarrow$~~ $x \in A \Leftrightarrow \exists y,\ R(x,y)$

When $R(x,y)$ holds, we say y is a certificate that proves $x \in A$

① $\Leftarrow$ proof Suppose that we have a decidable R satisfying
$x \in A$ iff $\exists y\ R(x,y)$

We will construct a machine M such that $A = \mathcal{L}(M)$

$\Sigma^* = \{\omega_0, \omega_1 \dots\}$
M on input x =
  for $i := 0$ to ∞
    if $R(x,y)$ holds, then accept
  end for

② $\Rightarrow$ proof Suppose $A = \mathcal{L}(M)$, for some TM M.
For any input x, let $C_0, C_1, C_2 \dots C_t$ be an accepting computation of M on x.

Let $y = \langle C_0, C_1 \ldots C_r \rangle$ be our certif.
Let $R(x,y)$ holds iff y is an accepting computa° of M on x.
Then R is decidable

<u>remark</u> the certif. th$^{\bar{e}}$ gives another way of proving that a language is in SD.

<u>e.g.</u> $E_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } \mathcal{L}(M) = \emptyset\}$

① is $\overline{E_{TM}} \in SD$?

~~pf~~ Let $y = z C_0 C_1 \ldots C_r$ be a certif where $C_0 C_1 \ldots C_r$ is an accept. comp. of M on x.

② is $E_{TM} \notin SD$? → use reduction

<u>Reducibility</u>

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

$DIAG = \{\langle M\rangle \mid M \text{ ——— } \wedge \langle M\rangle \notin \mathcal{L}(M)\}$

$\langle M\rangle \cancel{\notin} \in DIAG \iff \langle M\rangle \notin \overline{DIAG} \iff \langle M\rangle \notin \mathcal{L}(M)$
$\iff \langle M, \langle M\rangle\rangle \notin A_{TM}$

<u>Computable functions</u> A func° $f: \Sigma^* \to \Sigma^*$ is comp if $\exists$ ~~a~~ a TM M s.t. $\forall x \in \Sigma^*$, M halts on input x with just $f(x)$ on its tape.

<u>Many-one reducibility</u> (mapping reduc. in Sipser)

<u>Def</u>: Let $L_1, L_2 \subseteq \Sigma^*$. We say L_1 is many one reducible

to L_2, $L_1 \le_m L_2$, if $\exists$ a comp. func° $f : \Sigma^* \to \Sigma^*$ s.t. $\forall x \in \Sigma^*$, $x \in L_1 \Leftrightarrow f(x) \in L_2$.

$$\boxed{\text{historically } \forall \text{ and } \exists \text{ came from the fact it was easier to reverse A and E on typewriters}}$$

Claim: $\overline{DIAG} \le_m A_{TM}$ $f(\langle M \rangle)$

define $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$

theorem $L_1, L_2 \subseteq \Sigma^*$. Assume $L_1 \le_m L_2$

1. $\overline{L_1} \le_m \overline{L_2}$
2. $L_2 \in D \Rightarrow L_1 \in D$
3. $L_2 \in SD \Rightarrow L_1 \in SD$

use contrapositive to prove.

proof ① $L_1 \le_m L_2 \Rightarrow \exists f : \Sigma^* \to \Sigma^*$, $x \in L_1 \Leftrightarrow f(x) \in L_2$
~~call it~~ f.

$$x \notin L_1 \Leftrightarrow f(x) \notin L_2$$
$$x \in \overline{L_1} \Leftrightarrow f(x) \in \overline{L_2}$$
$$\overline{L_1} \le_m \overline{L_2}$$

② Assume $L_2 \in D$. Then call M_2, a TM s.t. $\mathcal{L}(M_2) = L_2$ and M_2 always halts.

Since f is comp. there is a TM that computes f. Call it M.
Define M_1 as follows:

- on input x, run M on x to compute $f(x)$
- run M_2 on $f(x)$, and accept or reject as M_2 does.

Clearly, M_1 always halts and $\mathcal{L}(M_1) = L_1$
$\Rightarrow L_1 \in D$.

③ Define M_1: . on x, RUN M to compute $f(x)$
. RUN M_2 on $f(x)$
. if M_2 accepts, accept
. — rej —
(similar to ② except M_2 might not halt)
Clearly: $\mathcal{L}(M_1) = L_1$

<u>Example</u>. $\overline{DIAG} \leq_m A_{TM}$
. $DIAG \notin D \Rightarrow \overline{DIAG} \notin D \Rightarrow A_{TM} \notin D$
. $DIAG \notin SD$ and (by ①) $DIAG \leq_m \overline{A_{TM}}$
Hence (by ③), $\overline{A_{TM}} \notin SD$

. $HB = \{\langle M\rangle \mid M$ is a TM and M halts on blank input tape$\}$
<u>Claim</u>: $HB \in SD \wedge HB \notin D$
<u>proof</u> we will show: $A_{TM} \leq_m HB$
let $x \in \Sigma^*$, we assume $x = \langle M, w\rangle$ where M is a TM.
<u>Define</u> $f(x) = \langle M'\rangle$ where M' works as follows:
. M' write w on the tape
. RUN M on w.
. if M halts $\wedge$ accept, M' accepts
— rej. M' loops forever
Clearly f is computable
Also: M accepts $w \Rightarrow M'$ halts $\Rightarrow \langle M'\rangle \in HB$
$\langle M, w\rangle \notin A_{TM} \Rightarrow M'$ doesn't halt
$\Rightarrow \langle M'\rangle \notin HB$

<u>Corrolary</u> $\overline{HB} \notin SD$
<u>proof</u> Suppose $\overline{HB} \in SD$, then since $HB \in SD$ we have $HB \in D$. contradic°.

Claim $E_{TM} = \{\langle M\rangle \mid M \text{ is a TM} \wedge \mathcal{L}(M) = \emptyset\}$

① $\overline{E_{TM}} \in SD$

② $E_{TM} \notin SD$

proof ① show that: $\overline{HB} \leq_m E_{TM}$ (or $\Leftrightarrow$ $HB \leq_m \overline{E_{TM}}$)

Define $f(\langle M\rangle) = \langle M'\rangle$ where:

on input x:

- if x is not the blank tape, rej.
- if x is the blank tape, RUN M on x and accept if it halts.

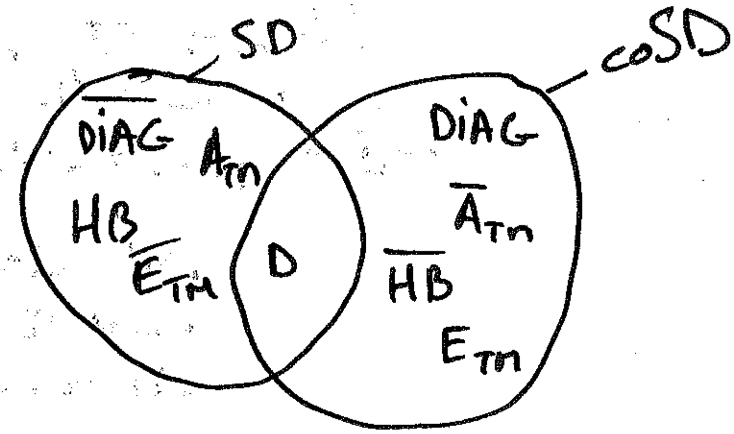
Hence: $\langle M\rangle \in HB \Leftrightarrow M'$ accepts the blank tape
$\Leftrightarrow M' \in \overline{E_{TM}}$
$\Leftrightarrow f(\langle M\rangle) \in \overline{E_{TM}}$

Example $\longleftarrow$ AOB = set of all C programs $\langle P\rangle$ s.t. P causes an array out of bound ref.

$HB \leq_m AOB$

Recall

$coSD = \{A \subseteq \Sigma^* \mid \overline{A} \in SD\}$



Theorem

① Let $A, B, C \subseteq \Sigma^*$. $A \leq_m B \leq_m C \Rightarrow A \leq_m C$

② Let $A \subseteq \Sigma^*$. $A \in SD \wedge A \leq_m \overline{A} \Rightarrow A \in D$

example of ①:

$\longrightarrow A_{TM} \leq_m HB \leq_m \overline{E_{TM}}$

proof (1)

- Assume $A \leq_m B$ via computable f — $f: \Sigma^* \to \Sigma^*$
  and $B \leq_m C$ — $g: \Sigma^* \to \Sigma^*$

- $g \circ f$ is computable:
  run TMs that comp f then g

- $\forall x \in A \iff f(x) \in B \iff g(f(x)) \in C$
  $\iff g \circ f(x) \in C$

proof (2)

- $A \leq_m \bar{A} \Rightarrow \bar{A} \leq_m A$
- $A \in SD \wedge \bar{A} \leq_m A \Rightarrow \bar{A} \in SD$
- $A \in SD \wedge \bar{A} \in SD \Rightarrow A \in D$

"total TM's"

Define $T_{TM} = \{\langle M \rangle \mid M$ is a TM and M is a decider$\}$

Claim $T \notin SD$ (1) $\wedge$ $\bar{T} \notin SD$ (2)

proof (1) show that $HB \leq_m T_{TM}$ ($\Rightarrow \overline{HB} \leq_m \bar{T}_{TM}$)

- define: $f(\langle M \rangle) = \langle M' \rangle$ where
  - if input $\neq$ blank M' rejects
  - input $=$ blank run M on it

$\to \langle M \rangle \in HB \iff M$ halts on blank tape
$\iff M'$ halts on all inputs
$\iff \langle M' \rangle \in T_{TM}$

(2) show that $HB \leq_m \bar{T}_{TM}$

- define: $f(\langle M \rangle) = \langle M' \rangle$ where on input x:
  - Run M on blank tape on $|x|$ steps
  - if M halts within x steps, M' loops
  - if not, M' accepts

Thus: . M doesnt halt on blank tape $\Rightarrow$ M' halts on all input

$\Rightarrow \langle M' \rangle \in T_{TM}$

$\Rightarrow \langle M' \rangle \notin \overline{T_{TM}}$

. M halts on blank tape in t steps

$\Rightarrow$ M' doesnt halt $\forall x, |x| \geq t$

$\Rightarrow$ M' is not total

$\Rightarrow \langle M' \rangle \in \overline{T_{TM}}$

Define $All_{TM} = \{\langle M \rangle \mid M \text{ is a TM} \wedge \mathcal{L}(M) = \Sigma^*\}$

Claim: $All_{TM} \notin SD \wedge \overline{All_{TM}} \notin SD$

(1) (2)

proof (1) show $T_{TM} \leq_m All_{TM}$

define $f(\langle M \rangle) = \langle M' \rangle$ where M' on x:

. RUN M on x

. if M halts, then M' accepts

accepts or rejects

then: $\langle M \rangle \in T_{TM} \Leftrightarrow \forall x \in$ input, M halts

$\Leftrightarrow$ ——, M' accepts

$\Leftrightarrow M' \in All_{TM}$

(2) show $\overline{All_{TM}} \notin SD$

Since $T_{TM} \leq_m All_{TM}$, we have $\overline{T_{TM}} \leq_m \overline{All_{TM}}$

since

Define $EQ_{TM} = \{\langle M \rangle \mid$

Chomsky Hierarchy

rec. enum.	TM	
CSL		
CFL	~~pushdown~~ PDA=NPDA	
REG	FSA=NFSA	

Finite automaton:

$M = \{\Sigma, Q, \delta, F\}$

$w_1 w_2 \ldots w_n$

q

at each step, head moves right one step $\to \delta(q, a_i) = q'$

M accepts w iff it reaches f ∈ F after scanning last symbol.

$$REG = \{\mathcal{L}(M) \mid M \text{ is a FSA}\}$$

$$REG \subsetneq CFL$$

e.g. PAL ∉ REG (pumping lemma)

NFSA can be converted to FSA w/ expo num of states.

$\boxed{\text{REG is closed under } \cup, \cap, -}$

Pushdown automaton PDA

$w_1 \ldots w_n$

q — top

$M = \{\Sigma, \Gamma, Q, \delta, F\}$ — stack $\alpha\beta$

$\delta(q, a, b) = (q', \text{push or pop})$

b: stack symbol — allow ε moves for this to do nothing

$$CFL = \{\mathcal{L}(M) \mid M \text{ is a PDA}\}$$

$$DCFL \subsetneq CFL \quad \wedge \quad DCFL = \{\mathcal{L}(M) \mid M \text{ is a DPDA}\}$$

$\boxed{\text{DCFL is closed under } - \text{ but not } \cup, \cap}$

proof $-$: take opposite DPDA

$\cap$: $L_{ab\#}, L_{\#ab} \in DCFL$, $L_{ab\#} \cap L_{\#ab} = L_{abc} \notin CFL$

$\cup$: De Morgan w/ $-$ and $\cap$.

$$\text{All}_{CFG} = \{\langle G\rangle \mid G \text{ is a CFG} \wedge \mathcal{L}(G) = \Sigma^*\}$$

Claim: $\text{All}_{CFG} \notin SD$

proof: show $\overline{HB} \leq_m \text{All}_{CFG}$

- given a TM M, construct a CFG G_M s.t. M halts on a blank tape iff $\mathcal{L}(G_M) \neq \Sigma^*$
- idea: design G s.t. $\mathcal{L}(G)$ is the set of all strings which do not code a halting of M on a blank tape.

Let $C_1 \ldots C_T$ be a halting computation, where $C_i = x q y$

$$\# q_0 \sqcup \# \underbrace{\longleftarrow}_{\text{odd}} \# \underbrace{\longrightarrow}_{\text{even}} \# \underbrace{\longleftarrow}_{\text{odd}} \#$$

L_1 = 1st and last config are ok (REG)

L_2 = "even" transitions are ok (DCFL)

L_3 = "odd" transitions —— (DCFL)

$$L_1 \cap L_2 \cap L_3 = \begin{cases} \emptyset & \text{if } M \text{ loops} \\ \{\text{halting comp}\} & \text{if } M \text{ halts} \end{cases}$$

$$\overline{L_1 \cap L_2 \cap L_3} = \begin{cases} \Sigma^* \\ \Sigma^* - \{h.c.\} \end{cases}$$

$$\parallel$$

$$\underbrace{\overline{L_1}}_{REG} \cup \underbrace{\overline{L_2} \cup \overline{L_3}}_{DCFL}$$

most of the time, when a model of computation's accepting lang is closed under ¯ it's usually bc we can easily change accept ∧ reject states.

set of sets S, $|S| > 2$

Russel: ~~condition set with~~

$All_{CFG} \notin SD$

Given a TM M, find a ~~cfg~~ CFG G s.t. M doesn't halt on a blank tape iff $\mathcal{L}(G) = \Sigma^*$

Find G' over $\Sigma' = \Sigma \cup \Gamma \cup Q \cup \{\#\}$

- Use ASCII codes for symbols in Σ'

$L_{comp} = \overline{L_1 \cap L_2 \cap L_3}$ s.t.

$L_1 \cap L_2 \cap L_3 = \begin{cases} \emptyset \text{ if } M \text{ doesn't halt} \\ \{w_M\} \text{ o/w} \end{cases}$

with ASCII codes, $L_{comp} \subseteq \Sigma^*$

Computability redux

- Decidability relies on our model of (universal) comp: TM
- Church-Turing thesis -

Undecidable pbs

- Diophantine equations
- Axiomatizing arithmetic - gödel's incompleteness

use fact that $\overline{HB} \notin SD$

give a TM ...

$\varphi_M(x)$ asserts x codes a halt.comp

$\forall x \neg \varphi_M(x)$

if every true such statement has a finite proof then $\overline{HB} \in SD$

- Entsheidungs pb

s. look: google it for spelling

Church & Turing

→ Complexity Theory

WCTime complexity: Let M be a TM decider

call $t_M(x)$: # steps req. by M to halt on input x

$\cdot\ T_M(n) : \max_{|x|=n} (t_M(x))$

e.g. M_{PAL} decides PAL $\quad$ 0110 0110 $\quad n+(n-1)+\dots 1$

$T_{M_{PAL}} \in O(n^2)$

Def $TIME(t(n)) = \{A \subseteq \Sigma^* \mid \exists M$ is a TM $\land \mathcal{L}(M)=A$ and $T_M(n) \in O(t(n))\}$

interpr. as a func! i.e. $n \mapsto n^2$ $\quad$ e.g. $PAL \in TIME(n^2)$

Def $P = \bigcup_k TIME(n^k)$

e.g. $SQ = \{\langle a^2\rangle \mid a \in \mathbb{N}, a^2$ in binary$\}$

$SQ \in P$ → Newton's method: $x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$

Def $A \in NP$ iff $\exists$ polytime relation $R(x,y)$ and a poly $p(n)$ s.t. $x \in A \iff \exists y\ (|y| \leq p(x)$ and $R(x,y))$

e.g. uHAMPATH $= \{\langle G,s,t\rangle \mid G$ is an undirected graph w/ a Hamiltonian Path $s \leadsto t\}$

CLIQUE $= \{\langle G,k\rangle \mid$

Def $G = \langle V,E\rangle$ is k-colourable if there is a func. $c: V \to \{1..k\}$ s.t. if $(u,v) \in E$, then $c(u) \neq c(v)$

Cook: Omission in Sipser

→ Every NP-complete pb A has an associated search pb A-search.

Let $A \in NP$, so $x \in A \iff \exists y \,(|y| \leq p(|x|) \text{ and } R(x,y))$

<u>A-search</u>
- instance: $x \in \Sigma^*$
- output: $y \in \Sigma^*$ s.t. $|y| \leq p(|x|) \wedge R(x,y)$
  or 'NO' if no such y exists.

<u>Def</u> Let P_1, P_2 be pb (search or decision)

$P_1 \xrightarrow{P} P_2$ iff $\exists$ polytime alg which solves P_1 which can ask questions to an 'oracle' which solves P_2 (do not count time required by P_2).

<u>Theorem</u> $A \in$ NP-complete then A-search $\xrightarrow{P} A$

<u>e.g.</u> HAMPATH-search $\xrightarrow{P}$ HAMPATH

Here's the reducing algorithm →

input: $\langle G, s, t \rangle$

oracle: boolean procedure $HP(H)$ which solves HAMPATH

<u>alg:</u> if $HP(G) = 0$ then output 'NO'

~~else~~ $H \leftarrow G$

for $i = 1..m$

if $HP(H - e_i)$ then $H \leftarrow (H - e_i)$

Correctness : invariant = G has a HAMPATH from $s \leadsto t$ using every edge in $E_H \cap \{e_1, \dots e_{i-1}\}$

<u>Def</u> $A \leq_p B$ (Karp reducible)

If $\exists$ polytime func $f : \Sigma^* \to \Sigma^*$ s.t $x \in A \iff f(x) \in B$

NOTE : $A \leq_p B \Rightarrow A \leq_m B$

$A \leq_p B \Rightarrow A \xrightarrow{P} B$

<u>Def</u> A is NP-hard iff ~~$B \leq$~~ $\forall B \in NP$, $B \leq_p A$

<u>Def</u> $A \in NPC$ iff $A \in NP \wedge A$ is NP-hard

<u>Lemma</u> $A \leq_p B \wedge B \in P \Rightarrow A \in P$

$A \leq_m B \wedge B \in D \Rightarrow A \in D$

FP = class of polytime computable functions

<u>Corollary</u> $A \in NPC \wedge A \in P \Rightarrow P = NP$

- missed lecture -

. A is NP-hard iff $B \leq_p A \quad \forall B \in NP$

. $A \in NPC$ iff A is NP-hard $\wedge$ $A \in NP$

<u>Cook-Levin's Thm</u> $SAT \in NP$

$SAT = \{\langle \varphi \rangle \mid \varphi \text{ is bool formula} \wedge SAT\}$

<u>Lemma</u> $A \in NPH \wedge A \leq_p B \Rightarrow B \in NPH$

"A is hard and A is reducible to B, so B must be hard"

<u>proof</u> $\leq_p$ is transitive

<u>Def</u> $CNF = \{\langle \varphi \rangle \mid \varphi \text{ is in } CNF\}$

$kCNF = \{\langle \varphi \rangle \mid \varphi$ is in CNF, s.t. each clause has at most k $\vee$ literals$\}$

$\rightarrow kSAT = \{\langle \varphi \rangle \mid \varphi \text{ is in } kCNF \wedge \varphi \text{ is SAT}\}$

<u>Thm</u> $3SAT \in NPC$

<u>show</u> $SAT \leq_p 3SAT$

<u>proof</u> idea: introduce a new literal (<u>neg or nonneg var</u>) for every binary subformula of φ such that for every such α: $\alpha = (\eta \vee \theta)$, construct a

$D_\alpha \equiv x_\alpha \leftrightarrow (x_\eta \vee x_\theta)$

... then define $\varphi' = D_\alpha \wedge D_\beta \ldots$

$\underbrace{\underbrace{(A \vee B)}_{\alpha} \vee \underbrace{(A \wedge C)}_{\beta}}_{\gamma}$

"at most 1 per connective"

proof show $\varphi \leftrightarrow \varphi'$

$\leftarrow$ (1) sup τ SAT φ', for every bin subform τ SAT

$\alpha \equiv x_\alpha \wedge l_\varphi \equiv \varphi$ under τ _

$\rightarrow$ (2) sup τ SAT φ

then there is an extension τ' s.t. τ' SAT φ'

$\text{IND-SET} = \{\langle G,k\rangle \mid G \text{ is undirected with IS of size } k\}$

Def: $V' \subseteq V$ is IS of $G=(V,E)$ iff $u,v \in V' \Rightarrow (u,v) \notin E$

Thm $IS \in NPC$

$\to$ show $3SAT \leq_p \text{IND-SET}$ (cook 1971)

proof. assume φ has exactly 3 literals per clause wlog

$\varphi = C_1 \wedge \ldots \wedge C_m$, $C_i = (l_{i1} \vee l_{i2} \vee l_{i3})$

$V_\varphi = \{\underbrace{\langle i,j\rangle}_{l_{ij}} \mid 1 \leq i \leq m, 1 \leq j \leq 3\}$ $\}$ — occurances of literals in φ.

$|V_\varphi| = 3m$

~~Let~~ $k_\varphi = m$

$E_1 = \{(\langle i,j\rangle, \langle i,l\rangle) \mid i \in 1..m, 1 \leq j < l \leq 3\}$

$E_2 = \{(\langle i,j\rangle, \langle i',j'\rangle) \mid l_{ij} = \overline{l_{i'j'}}\}$

$E_\varphi = E_1 \cup E_2$

$G_\varphi = (V_\varphi, E_\varphi)$ has an IS of size m iff φ is SAT

① $\Rightarrow$ if τ SAT

② $\Leftarrow$ sup. G_φ has an IS V' of size m.
then V' must have exactly 1 literal per clause
$\to$ Choose τ to make all these literals $= 1$.

$\text{CLIQUE} = \{\langle G,k\rangle \mid \underset{(V,E)}{G} \text{ has clique of size } k\}$

Thm $\text{CLIQUE} \in NPC$

proof ① $\text{CLIQUE} \in NP$ since given a $\boxed{\text{clique } V'}$ $\subseteq V$ check that $|V'| = k$ in polytime

certificate to show polytime decidability $\to$ algorithmic possibility

② show $\text{IND-SET} \leq_p \text{CLIQUE}$

Suppose $V' \subseteq V$. Then V' is an ~~ind~~ IS of G iff V' is a CLIQUE in the complement of G, i.e. $\bar{G}$.

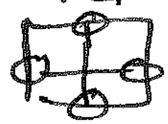
$\bar{G} = \{(V, \bar{E})$, $(u,v) \in \bar{E} \Leftrightarrow u \neq v, (u,v) \notin E$

VERTEX-COVER $= \{\langle G, k\rangle \mid G$ is an undirected graph with a vertex cover of size $k\}$

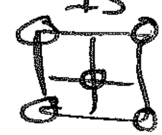
$V' \subseteq V$ is a VC for G iff $\forall (u,v) \in E \Rightarrow u \in V' \vee v \in V'$

Thm VERTEX-COVER $\in$ NPC

VC



IS



show IND-SET $\leq_p$ VERTEX-COVER

Lemma $V' \subseteq V$, V' is a VC for G iff $V - V'$ is an IS.

HAMPATH $= \{\langle G, s, t\rangle \mid G$ is a directed graph with a hampath from $s \rightsquigarrow t\}$

Thm HAMPATH $\in$ NPC proof $\to$ Sipser

Claim HAMPATH $\leq_p$ uHAMPATH

IDEA for each $u \in V$ $u \to v$
make $u^{in} \to u^{mid} \to u^{out}$

Back to Cook-Levin

SAT $\in$ NPC [1971 Cook]

SAT $\in$ NP ✓ $\forall L \in NP$, $L \leq_p$ SAT ?

proof Assume $A \in NP$
Given $x \in \Sigma^*$, find φ_x in polytime s.t. $x \in A \Leftrightarrow \varphi_x$ is SAT

By def, $A \in NP \Rightarrow \exists$ polytime relation $R(x,y) \wedge$ poly $p(n)$ s.t.
$x \in A \Leftrightarrow \exists y\, (|y| \leq p(|x|) \wedge R(x,y))$

Let M a TM which accepts $R(x,y)$ in time $T(n)$, $n = |x \not{b} y|$ (13)

→ input to M is $x\not{b}y$, assume wlog that $|y| = p(|x|)$

| $x \in A$ iff $\exists y_1 \cdots y_m$ s.t. comp $C_0 \ldots C_T$ of M (with $C_0^{\vec{x}\vec{y}}$)
| and C_T is an accept state, with $T(n) = cn^k$ (for some c, k)

→ Variables in φ_x ~~are such that~~ describe: →
- state set $Q = \{q_0 .. q_\ell\}$ for each C_t
- head pos for each t
- tape contents for each t

| $0 \leq t \leq T$

→ • q_{it}, $\begin{cases} 0 \leq i \leq \ell \\ 0 \leq t \leq T \end{cases}$: $q_{it} = 1$ iff M is in q_i at t.

• h_{it}, $\begin{cases} 0 \leq t \leq T \\ 0 \leq i \leq T \end{cases}$: $h_{it} = 1$ iff head is in pos i at time t

• $\begin{cases} z_{it} \\ u_{it} \\ b_{it} \end{cases}$, $\begin{cases} 0 \leq i \leq T \\ 0 \leq t \leq T \end{cases}$: $z_{it} = 0$ iff tape square at pos i has a 1 at time t.
$u_{it} \to 1$
$b_{it} \to b$

$$\varphi_x = \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4 \cdots \wedge \psi_?$$

<u>Def</u> $\underbrace{\text{Unique}(P_1 \ldots P_k)}$ ≡ exactly 1 of $P_1 \ldots P_k$ is true
↳ $(p_1 \vee p_2 \cdots p_k) \wedge \bigwedge_{1 \leq i < j \leq k} (\overline{p_i} \vee \overline{p_j})$

$$\psi_1 \equiv \bigwedge_{t=0}^{T} \text{Unique}(q_{0t} \cdots q_{\ell t})$$ – $\forall t$, M is in exactly 1 state

$$\psi_2 \equiv \bigwedge_{t=0}^{T} \text{Unique}(h_{0t} \cdots h_{Tt})$$ -

$$\psi_3 \equiv \bigwedge_{\substack{t=0 \\ j=0}}^{T} \text{Unique}(\overset{0}{z_{jt}}, \overset{1}{u_{jt}}, \overset{b}{b_{jt}})$$ -

e.g. initial config:
$0110 \not{b}\, y_1 \cdots y_m \not{b}$ → $\psi_4 \equiv q_{00} \wedge z_{00} \wedge u_{10} \wedge u_{20} \wedge z_{30} \wedge \overline{b_{40}} \wedge \overline{b_{50}} \cdots \wedge b_{(4+m)0}$ b_{T0}

~~initial configuration~~

$\psi_5 \equiv q_{\ell T}$ – end in an accept state

$$\psi_6 \equiv \bigwedge_{t=0}^{T-1} \bigwedge_{i=0}^{T} \left(\overline{h}_{it} \to (b_{it} \to b_{i,t+1}) \wedge (z_{it} \to z_{i,t+1}) \wedge (u_{it} \to u_{i,t+1}) \right)$$

– all tape symbols are unchanged except the scanned square –

ψ_7 – $\psi_?$ – transition function connections

$\to \forall \delta \in R$, ~~ψ~~ e.g. $[\delta(q_3,1)=(q_5,0,R)]$

$$\psi_7 \equiv \bigwedge_{t=0}^{T-1} \bigwedge_{i=0}^{T-1} \left[(q_{3t} \wedge h_{it} \wedge u_{it}) \to q_{5(t+1)} \wedge z_{i(t+1)} \wedge h_{(i+1)(t+1)} \right]$$

selfreducibility

$A \in NPC \Rightarrow A\text{-search} \leq_p A = \{x \mid \exists y, |y| \leq p(|x|), V_A(x,y) = yes\}$

poly alg

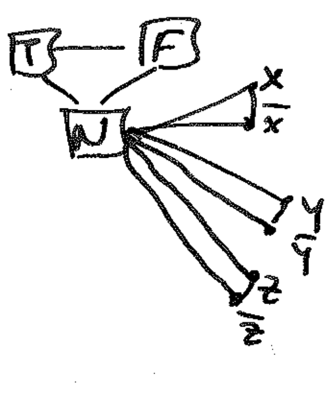
bc every language can be encoded in binary in poly time

```
C ← ε
while V_A(x,C) = "no"
    if C1 is a prefix of a certif.
        C ← C1
    else C ← C0
return C
```

Thm 3COLOR = $\{\langle G \rangle \mid G$ is 3colorable$\}$

show 3SAT $\leq_p$ 3COLOR

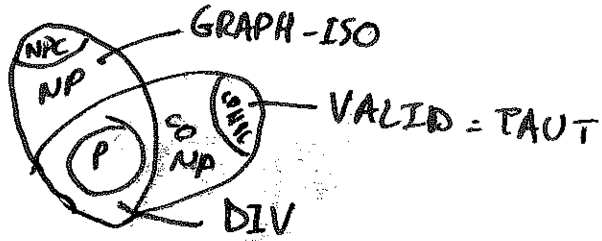
$x, y, z \mapsto$



- Complexity Redux

SAT $\in$ NPC (Cook-Levin)
| 3SAT
3SAT — IND-SET, HAMPATH
IND-SET — CLIQUE, VERTEX-COVER
HAMPATH — UHAMPATH

- DIV $\in$ NP $\cap$ coNP, $\overline{\text{DIV}} \in$ NP
- Assuming NP $\neq$ coNP ($\Rightarrow$ P $\neq$ NP) — bc P = coP



TAUT $\in$ NP iff $\exists$ proof sys s.t. $\exists c, k$, $\forall \phi \in$ TAUT, ϕ has a proof of length $\leq cn^k$, $n = |\phi|$

i.e. NP = coNP iff $\exists$ such a proof sys

Space Complexity

Let M be a TM, $x \in \Sigma^*$, $s_M(x)$: number of tape squares scanned in the compute of M on x.

$S_M(n) = \max\{s_M(x) \mid |x| = n\}$

$f: \mathbb{N} \to \mathbb{N}$

$\text{SPACE}(f(n)) = \text{DSPACE}(f(n)) = \{A \subseteq \Sigma^* \mid \exists \text{ TM } M, \mathcal{L}(M) = A \text{ and } S_M(n) \in O(f(n))\}$

Def $\text{PSPACE} = \bigcup_k \text{DSPACE}(n^k)$

Thm DTIME$(f) \subseteq$ DSPACE(f) proof obvious

$\Rightarrow$ P $\subseteq$ PSPACE ■ — Cook: P $\subsetneq$ PSPACE?

Thm NP $\subseteq$ PSPACE

proof $A \in$ NP iff $x \in A \Leftrightarrow \exists y, |y| \leq p(n) \wedge R(x, y)$

Since $|y| \leq p(n)$, we can use brute force to find y — $\Rightarrow$ exptime, but PSPACE

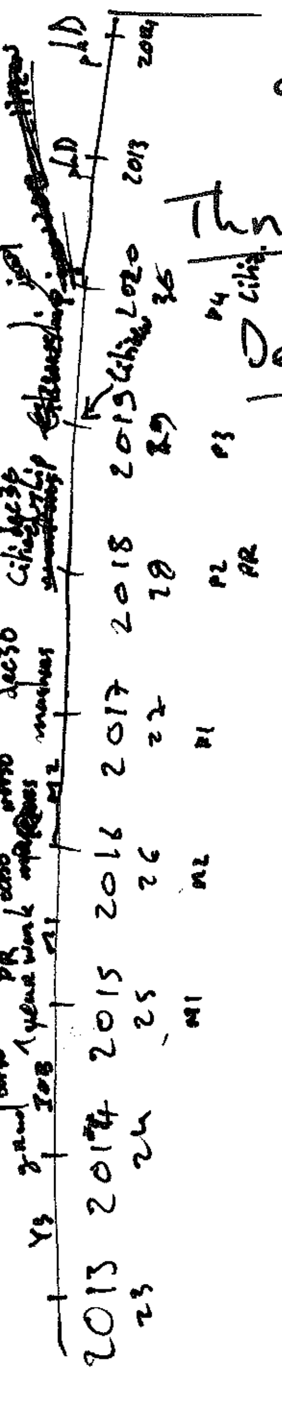
Space Complexity

TQBF

└ quantified Bool. form.

- QBF φ is a sentence iff it has no free vars.

TQBF = $\{\varphi \mid \varphi$ is a true QBF sentence$\}$



Claim SAT $\leq_p$ TQBF

given sat $\varphi(x_1 \ldots x_n)$, construct $\varphi' = \exists x_1, \ldots x_n, \varphi(x_1 \ldots x_n)$

Thm TQBF is PSPACE-complete.

Def Truth of a QBF (in prenex form) — quantif in prefix, every QBF can be put to that form.

$$\varphi(x_1 \ldots x_n) = Q_1 x_1 \ldots Q_n x_n \; \psi(x_1 \ldots x_n)$$

Induc.

Base: $n = 0$, $\varphi(\emptyset) = \psi(\mathrm{wwf}(0, 1, \wedge, \vee))$

I. step: case 1: $Q_1 = \exists$

$$\Rightarrow \varphi \equiv Q_2 \ldots Q_{n+1} x_{n+1} \psi(0, x_2 \ldots x_n) \vee \psi(1, x_2 \ldots x_n)$$

case 2: $Q_1 = \forall$

$$\Rightarrow \varphi \equiv \_\_\_\_ \wedge \_\_\_\_$$

Since we can build such an alg, that will eliminate quantifiers every time, we only need linear space with brute force (EXPTIME)

$\Rightarrow$ TQBF $\in$ L, $\in$ PSPACE show TQBF PSPACE hard:

let $A \in PSPACE$, $A = \mathcal{L}(M)$

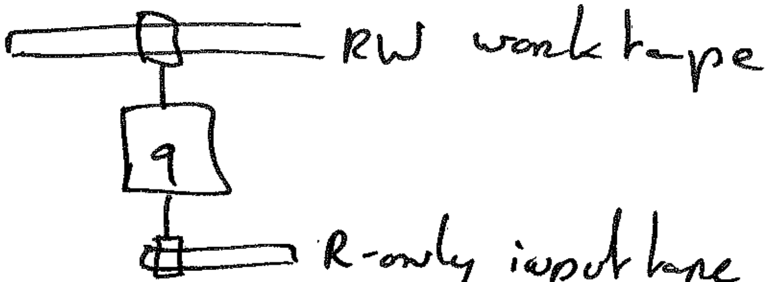
M on x has a comp using space $O(n^k)$

given $x \in \Sigma^*$, we must construct in poly time

a QBF sentence φ_x s.t. M accepts x iff φ_x is true.

<u>Def</u> $L = DSPACE(\log n)$; $NL = NSPACE(\log n)$

<u>log space TM</u>



$space_M(n)$ = # distinct work tape squares scanned

$MAJ = \{x \in \{0,1\}^* \mid x \text{ has more 1s than 0s}\}$

$MAJ \in L$ _ input head scans input $x_1 \ldots x_n$

work tape using binary notation

$PATH \in NL$

<u>proof</u> M on inp $\langle G, s, t\rangle$

M guesses $s = v_1, v_2 \ldots v_k = t$

checks $(v_i, v_{i+1}) \in E$, $1 \le i < k$

<u>Note</u> We may assume a log space TM has finitely many regs R_1, R_m

→ NO certificate theorem for NL

since you would get a certif for all

of NP! - Cook

Claim PATH $\in$ NL-complete under $\leq_L$

$L = NL$ iff PATH $\in L$

Claim $L \subseteq P$

Let M be a logspace TM. comp of M on $\langle x \rangle$, $|x| = n$

conf $C = \langle j, q w \rangle$ | work tape $\leq c \log_2 n$

head pos. $\leftarrow j$, $qw \rightarrow$ tape conf

$0 \leq j \leq n+1$ $\Rightarrow \exists\, O(n^k)$ possible confs

Def Logspace reduction

$FL = \{ f : \Sigma^* \to \Sigma^* \mid f \text{ comp in } O(\log) \text{ space} \}$

note model has extra W-only output.

$f \in FL \overset{\exists k}{\Rightarrow} |f(x)| = O(|x|^k)$ ~~for same~~

Because M must halt in time $O(n^k)$

e.g. $f_+(x,y) = x+y$; $f_\times(x,y) = x \cdot y$

Def $A \leq_L B$ iff $\exists f \in FL$, $x \in A \Leftrightarrow f(x) \in B$ $(\forall x \in \Sigma^*)$

Claim $A \leq_L B \Rightarrow A \leq_p B$ bc $FL \subseteq FP$

Claim $\leq_L$ is transitive bc FL is closed under composi"

proof $g \circ f$, $g \in FL$, $f \in FL$

Side note: probabilistic TM $\leadsto$ 2 defs: • NTM which 'chooses' a single branch based on probdist. • DTM w/ random tape

BPP = P for pTM

BP·NP = NP for pTM (hard to think about?)

(BP·NP = AM) $\leadsto$ cf Arthur-Merlin protocol

Savitch's theorem

→ Cook' 1st PhD student, 1970

$NL \subseteq DSPACE(\log^2 n)$

Suffices to show $PATH \in DSPACE(\log^2 n)$

(since PATH is NL-complete, i.e. $A \in NL \Rightarrow A \leq_L PATH$)

Lemma: $A \leq_L B \wedge B \in DSPACE(\log^2 n) \Rightarrow A \in DSPACE(\log^2 n)$

proof

Fact: if G is an undirected graph, G has a path of length at most $2l$ from u to v iff $\exists w \in N$, G has paths from u to w and w to v, each of length $\leq l$.

Idea: use D&C $|l| = \log n$

Alg: Bool procedure $Path(G, u, v, l)$

holds iff G has $|u \rightsquigarrow v| \leq 2^l$

if $l = 0$ and $u = v$, then ACCEPT

else for $w \in V$

if $Path(G, u, w, l-1)$ AND $Path(G, w, v, l-1)$ then ACCEPT

end for

REJECT

Space analysis: nesting depth: l

space per call: $O(\log |V|)$

→ total space: $O(l \cdot \log|V|)$

$\Rightarrow$ To solve PATH, call $Path(G, s, t, \lceil \log_2 n \rceil)$

Def $f(n)$ is space constructible iff $1^n \mapsto f(n)$ is computable in space $O(f(n))$.

$f(n)$ can be written in unary / binary:

If $f(n)$ is space constr., then can mark off $f(n)$ tape squares on work tape in space $O(f(n))$

$(n)_{bin} \in O(\log n)$

General Savitch

$f(n)$ is space constr. $\left(f(n) \geq \log_2 n\right) \Rightarrow NSPACE(f(n)) \subseteq DSPACE(f^2(n))$

1st method: proof Assume $A \in NSPACE(f(n))$

- Let $A = \mathcal{L}(M)$, M is a nondet $O(f(n))$ TM
- use the configuration graph for M on x.

2nd method: proof (Sipser ex: 9.13) Padding argument:

def: $pad(w, l) = \{ w\#^j \mid j = \begin{cases} 0 & \text{if } l \leq |w| \\ l-|w| & \text{o/w} \end{cases} \}$, $w \in \Sigma^*$

def: $pad(A, f) = \{ pad(w, f(|w|)) \mid w \in A \}$, $A \subseteq \Sigma^*$

Assume $f(n)$ is sp. cons. $A \in NSPACE(\log n)$

$B = pad(A, 2^{f(n)}) \in NL$

By special Savitch, $B \in DSPACE(\log^2 n)$

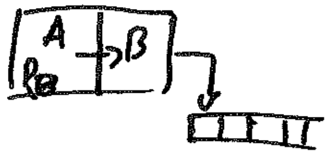
Thus: $A \in DSPACE(f(n)^2)$

Recursion Theorem

→ $\exists q : \Sigma^* \to \Sigma^*,\ w \mapsto P_w \wedge q$ is comp ← TM that outputs w

Q : on w, print $\langle P_w \rangle$ where P_w : on x, $\begin{cases} \text{erase } x \\ \text{write } w \\ \text{halt} \end{cases}$, halt

SELF:

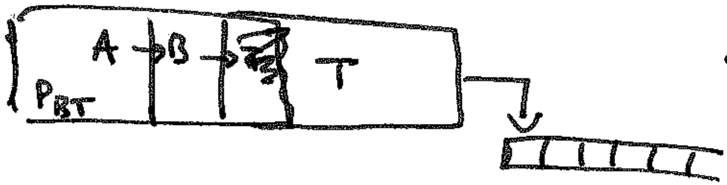


SELF = $\begin{cases} A \text{ prints } \langle B \rangle \\ B \text{ computes } q(\langle B \rangle) = \langle A \rangle \\ \quad \text{prints } \langle AB \rangle \end{cases}$ | $A = P_{\langle B \rangle}$, $B = Q(\langle B \rangle)$

Build a ~~TM that~~ can compute on its description (avoiding selfref paradox)

→ $\forall t : \Sigma \times \Sigma \to \Sigma,\ \exists R : \Sigma \to \Sigma,\ w \mapsto t(\langle R \rangle, w)$

R:



~~R = { A : print $P'_{\langle BT \rangle}$~~ (erase x → P')

B : ~~compute $q'(\langle BT \rangle) = \langle A \rangle$~~ print

$A = P'_{\langle BT \rangle}$, $A \to \langle BT \rangle$ | tape $w\langle BT \rangle$

$B = Q'(\langle BT \rangle)$, $B \to \langle ABT \rangle$ | $\langle ABT \rangle w$

Space Hierarchy

For any space cons. $f : \mathbb{N} \to \mathbb{N}$, $\exists A$ decidable in $O(f(n))$ space but not decidable in $o(f(n))$ space

<u>proof idea</u> build D using diagonalization

D on input $\langle M \rangle$:
- D runs M on $\langle M \rangle$ within $f(n)$ bound
- If M halts within $f(n)$, do the opposite of M.
- If uses more space, doesn't halt.

~~$h: \{0,1\}^* \to \{0,1\}^*$~~

Technical issues

① M runs in $o(f(n))$ space may use more than $f(n)$ space for small n when asymptotic beh kicks in –
 $\to$ accept $\langle M \rangle 1^k$ as input

② M might loop forever
 $\to$ since if it runs in $o(f(n))$ space, it will use at most $2^{o(f(n))}$ time $\Rightarrow$ keep a counter

D, on ω : • if $\omega \neq \langle M \rangle 1^k$ for some M, k ($M \to$ TM, $k \to \mathbb{N}$)

• let $n = |\omega|$, since $f(n)$ is space cons, mark off $f(n)$ space. If later steps ever use more than $f(n)$ space, reject.
• Simulate M on ω, reject iff M accepts or counter $> 2^{f(n)}$
• else accept.

Corrolary ① f is space cons. $\wedge$ $g \in o(f) \Rightarrow DSPACE(g) \subsetneq DSP(f)$

② $\forall \varepsilon_1, \varepsilon_2$, $0 \leq \varepsilon_1 < \varepsilon_2 \Rightarrow DSP(n^{\varepsilon_1}) \subsetneq DSP(n^{\varepsilon_2})$

③ $NL \subsetneq PSPACE$

<u>proof</u> By Savitch $NL \subseteq DSP(\log^2 n)$
By SHT, $DSP(\log^2 n) \subsetneq DSP(n)$

④ $PSPACE \subsetneq EXPSPACE$

Question 3

We show that SET-PARTITION is NP-complete.

(1) ~~Given~~ A certificate for $\langle S_1 \dots S_m, k\rangle \in$ SET-PARTITION is a subset C of $\{1 \dots m\}$ of size k such that the sets S_i corresponding to elements $i \in C$ are pairwise disjoint. Clearly, we can verify in polytime each $|C| \times (|C|-1)$ pair for empty intersection, and that $|C| = k$. Hence SET-PARTITION $\in$ NP.

(2) show IND-SET $\leq_m^P$ SET-PARTITION

Given a graph $G = (V,E)$ and natural number k, we build $m = |V|$ sets S_i $(1 \leq i \leq m)$, such that each corresponds to a node in G, and contains every edge incident to that node.

Thus, G will have an indep. set of size k iff we have k nodes (each corresponding to an S_i) such that no two nodes share an edge (every S_i pair is disjoint).

By (1) and (2), SET-PARTITION is NP-complete.

10/10 Correct

Question 4

We show μHAMPATH $\leq^P_m$ μHAMCYCLE.

Given a graph $G=(V,E)$ and $s,t \in V$, we construct a graph $G'=(V',E')$ where $V'=V\cup\{v\}$ ($v\notin V$ is an extra node) and $E'=E\cup\{v,s\}\cup\{v,t\}$.
Hence, if G has a hamiltonian path from $s \rightsquigarrow t$, then the new graph G' has a hamiltonian cycle from $s \rightsquigarrow t \to v \to s$.
The converse holds since v is only connected to s and t, hence every hamiltonian cycle has to include (s,v) and (v,t), thus there must be a hamiltonian path from $s \rightsquigarrow t$.

10/10 Correct.