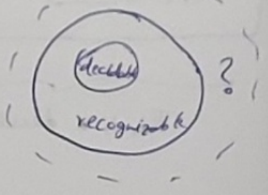
• Classes of Problems

recursive / decidable

re / T-recognizable

Turing machines — countable

Languages — uncountable } Hence ∃ languages that are not T-recognizable



We have ordering $w_1, w_2, \ldots$ of Σ^*

" " of all re. languages

$L = \{ \langle M_i \rangle \mid M_i \text{ does not accept } \langle M_i \rangle \}$

P.t. $\nexists$ an M_i s.t. $L = L(M_i)$

Proof

Suppose $\exists \langle M \rangle$ s.t. $L = L(M)$

Does $\langle M \rangle \in L(M) = L$ ↘ contradiction

• let $L_d = \{ \langle M \rangle \mid \langle M \rangle \notin L(M) \}$ ← Notation in Hopcroft-Ullman

↑

Thm L_d is not r.e.

Is $\bar{L}_d$ re.? Yes $\langle M_{\bar{L}_d} \rangle$

Proof:

~~$\bar{L}_d = \{ \langle M \rangle \mid \langle M \rangle \in L(M) \}$~~ ↖ iterate over all strings (in lexicographic order) and accept any that satisfy. Hence r.e.

M_1 M_2 M_3 . . .

$\langle M_1 \rangle$ ..

$\langle M_2 \rangle$

$\langle M_3 \rangle$

Wrong

if L is recursive $\Rightarrow \bar{L}$ is recursive

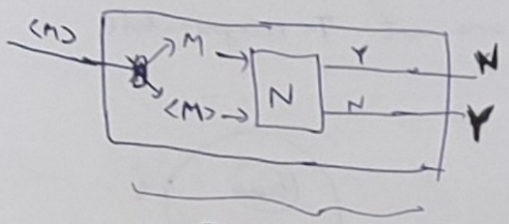
Proof??

- Show $L_u = \{ \langle M, w \rangle \mid w \in L(M) \}$ is undecidable, but r.e.

Idea: If $\langle M \rangle \in \bar{L}_d$,

$\langle M, \langle M \rangle \rangle \in L_u$

Suppose TM N decides L_u,



This is a decider for L_d. But then contradiction. Hence ∄ such N.

$L_d = \{ \langle M \rangle \mid \langle M \rangle \notin L(M) \}$

$\bar{L}_d$ not r.e.

L_d is not r.e.

$\bar{L}_d$ is r.e, not recursive

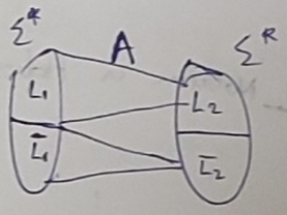
← reduction of L_d to L_u

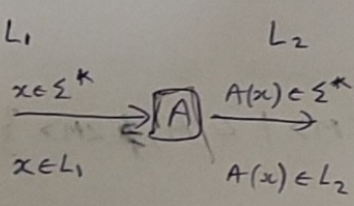
To show, L_u is r.e, . . . (simulate both M & ⟨M⟩)

Then, for $\bar{L}_u$ → can't be recursive (∵ then L_u would be recursive)
→ can't be r.e. also (∵ if $L, \bar{L}$ are r.e., then both are recursive as well).

- Reductions

$L_1 \leq L_2$





Many to one reduction

Def^n: If A is algo s.t. $x \in L_1$ iff $A(x) \in L_2$
then A is a "reduction" of L_1 into L_2.

So, $L_1 \leq L_2$ ← if L_1 can be reduced to L_2

Turing reduction

Algo to check if $x \in L_1$

uses subroutines like $x_1 \in L_2$?
$x_2 \in L_2$?

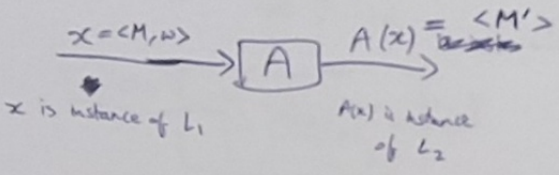
- $L_e = \{\langle M \rangle \mid L(M) = \phi\}$ ← need to show $\overline{L_e}$ is re, not recursive
  L_e is not recursive

$L_{ne} = \overline{L_e} = \{\langle M \rangle \mid L(M) \neq \phi\}$

Idea: $\overline{L_e}$ is r.e. - coz we can simulate M on all strings $w_1, w_2, \ldots$ ← (in a clocked way)
until any 1 string accepts ↓ using the pair generator (deterministic)

To show for L_{ne} is not recursive,

we show $L_u \leq L_{ne}$



where $\langle M, w \rangle \in L_u$ iff $\langle M' \rangle \in L_{ne}$

$\Rightarrow$ $w \in L(M)$ iff $L(M') \neq \phi$

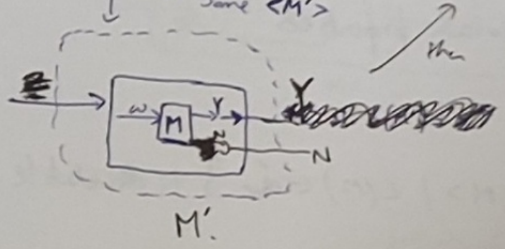
To construct this ↑,

given ⟨w, M⟩,

we need a machine that outputs some ⟨M'⟩

$w \in L(M) \Rightarrow L(M') \neq \phi$ (Σ^*)

$w \notin L(M) \Rightarrow L(M') = \phi$ (ϕ)

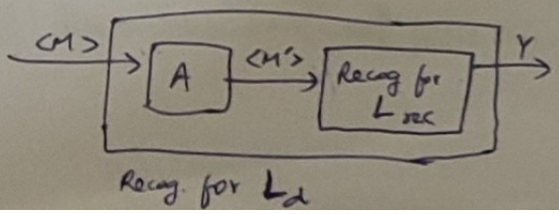


- $L_{rec} = \{\langle M \rangle \mid L(M) \text{ is recursive}\}$ ↖ this is not decidable, not re. even

$\overline{L_{rec}}$ is also not r.e.

Idea:

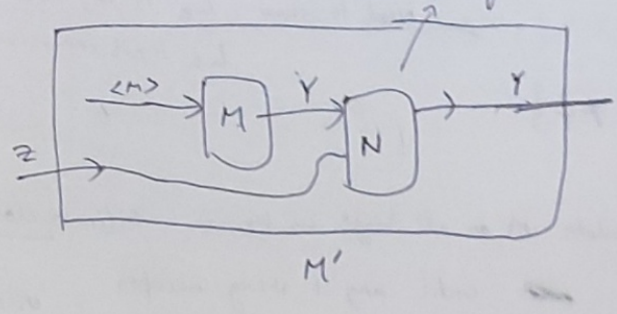
show $L_d \leq L_{rec}$



if $\langle M \rangle \notin L(M) \Rightarrow L(M')$ is recursive

if $\langle M \rangle \in L(M) \Rightarrow L(M')$ is not recursive

Then,

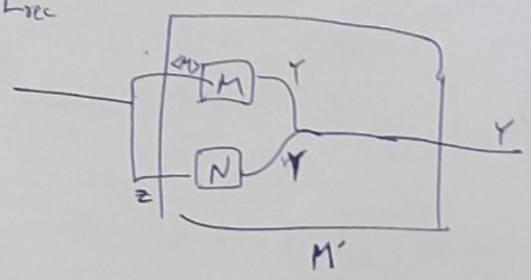


↳ Then if $\langle M \rangle \notin L(N) \Rightarrow L(M') = \phi$

if $\langle M \rangle \in L(M) \Rightarrow L(M') = L(N)$

for $\overline{L}_{rec}$,

to show $\overline{L}_{rec}$



- $L = \{ \langle M \rangle \mid L(M) \in \mathcal{L} \}$ ← can we use this as definition of r.e. langs

$\mathcal{L}$ ~~any~~ subset of r.e. langs (that may satisfy a "property")

$\mathcal{L} = \phi$
$\mathcal{L}$ = set of all r.e. langs } trivial properties

So, property $\mathcal{L}$ is decidable if $L = \{ \langle M \rangle \mid L(M) \in \mathcal{L} \}$ is decidable

$\mathcal{L} = \{ \langle M \rangle \mid M \text{ has 5 states} \}$ ← So not a property of r.e. langs

But $L(M) = L(M')$
↑ 5 states ↑ 6 states

But property of machines.

So, these kinds of properties we don't consider.

- Rice's Theorem

Every non-trivial property of r.e. langs is undecidable.

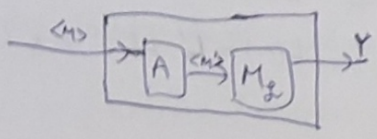
Proof:

Suppose $\mathcal{L}$ is a non-trivial property

Wlog, assume $\phi \notin \mathcal{L}$

Suppose $\exists$ a decider $M_{\mathcal{L}}$ which decides $L_{\mathcal{L}}$

Then to show $L_{re} \leq L_{\mathcal{L}}$ $\quad (L_{re} = \{\langle M \rangle \mid L(M) \neq \phi\})$

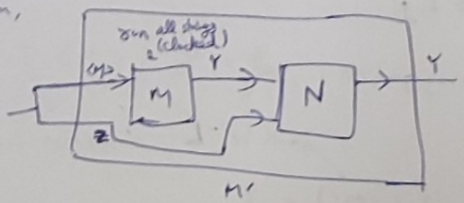


We need, $\quad L(M) \neq \phi \Rightarrow L(M') \in \mathcal{L}$ ($L(M)$ above $L(M')$)

$L(M) = \phi \Rightarrow L(M') \notin \mathcal{L}$ ($L(M') \downarrow \phi$)

N, $L(N) \in \mathcal{L}$

Then,



$L_{\mathcal{L}} = \{\langle M \rangle \mid L(M) \in \mathcal{L}\}$ property $\mathcal{L}$ (of r.e. langs) is about r.e. langs

When is $L_{\mathcal{L}}$ not r.e.?

Thm

- $L_{\mathcal{L}}$ is r.e. iff

① $L \in \mathcal{L}$ and L' is a r.e. lang s.t. $L' \supset L$, $\Rightarrow L' \in \mathcal{L}$

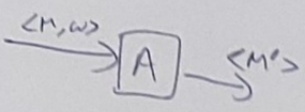
② $L \in \mathcal{L}$, $\exists$ a finite $L' \subset L$ s.t. $L' \in \mathcal{L}$

③ The set of finite language in $\mathcal{L}$ is enumerable. (i.e. $\exists$ enumerator that enumerates all finite langs in $\mathcal{L}$)

Proof of ①,

assume ① is false. Then, claim: $\overline{L_u} \leq L_{\mathcal{L}}$ (So, $L \in \mathcal{L}$, L' is r.e., $L \subset L'$, but $L' \notin \mathcal{L}$)

So,

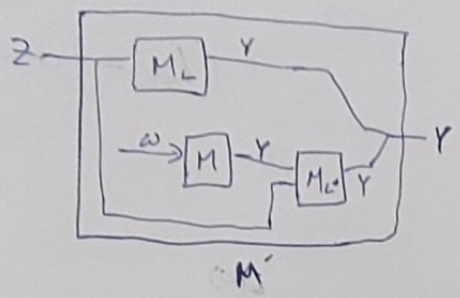


we need,

$w \in L(M) \Rightarrow L(M') \in \mathcal{L}$

$w \notin L(M) \Rightarrow L(M') \notin \mathcal{L}$

??



Then, $L(M) = L$ when $w \in L(M)$

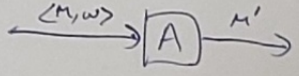
$L(M) = L'$ when $w \notin L(M)$

Proof of ②,

assume ② is false. Then claim: $\overline{L_u} \leq L_{\mathcal{L}}$

So, $L' \subset L$ and $L' \notin \mathcal{L}$ (finite)

So,



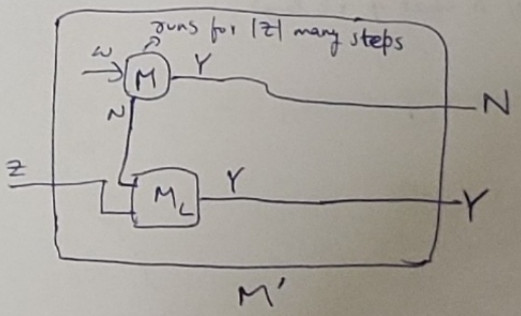
will be L

$w \notin L(M) \Rightarrow L(M') \in \mathcal{L}$

$w \in L(M) \Rightarrow L(M') \notin \mathcal{L}$

will be finite subset of L

$\{z \in L \mid |z| \leq$ steps for w to accept on $M\}$



Proof of ③,

go through in "clocked" method.

Idea: go over all "$w_1, \ldots, w_m$" and construct $\langle M \rangle$ s.t. $L(M) = \{w_1, \ldots, w_m\}$

and verify that $L(M) \in \mathcal{L}$ (i.e. $\langle M \rangle \in L_{\mathcal{L}}$)

will eventually output all finite langs in $\mathcal{L}$

Thm: ① and ② and ③ $\Rightarrow L_{\mathcal{L}}$ is r.e.

Proof: Given $\langle M \rangle \in L_{\mathcal{L}}$.

Run enumerator for finite langs (using ③)

For each = "$w_1, \cdots, w_m$"

Verify if $L(M) = \{w_1, \cdots, w_m\}$ (i.e. M accepts $w_1, \cdots w_m$) ← existence of finite subset using ②

(using ①)

* if yes, accept $\langle M \rangle$

Also can have properties/langs based on machine,

Like

$L = \{ \langle M, w \rangle \mid M \text{ writes 3 consecutive 1's when input is } w \}$

↳ we can show $L_u \leq L$. hence L is undecidable

↳ we can have $\langle M', w \rangle$ s.t. $L(M') = L(M)$ but $\langle M', w \rangle$ is not in L.

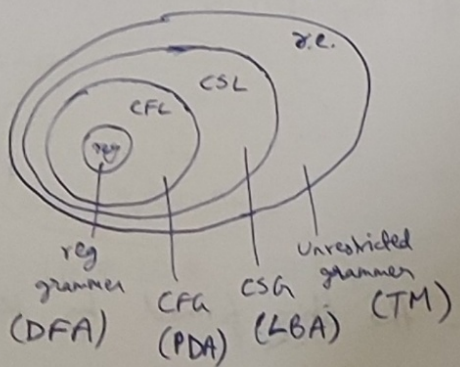
Another ex:

$L = \{ \langle M \rangle \mid \nexists M', L(M) = L(M') \text{ s.t. } |\langle M \rangle| < |\langle M' \rangle| \}$

?? not recursive

- Classes of computability

Chomsky Hierarchy



Context Sensitive Grammers

$\beta A \delta \rightarrow \beta \alpha \gamma$

$\beta' A \gamma' \rightarrow \beta' \alpha' \gamma'$

Unrestricted Grammers

$\alpha \rightarrow \beta$

$aAb\beta \rightarrow aCbbC$

- CSL - equivalent to class of languages recognized by TM with ~~linear~~ space (Proof ??)

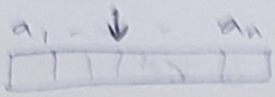
if input x, use atmost $|x|$ cells

↓

equivalent to $O(|x|)$ cells (by changing alphabet)

} called a Linear Bounded Automata (LBA)

If $|\Gamma| = k$,



then $\underbrace{k^n |Q| n}$

total # of 'configurations' of the LBA.

Hence a deterministic LBA will repeat (if long enough time)

So, $L = \{ \langle M, \omega \rangle \mid M$ is LBA which acc $\omega \}$ is decidable

---

- DTIME $(f(n))$ = class of all langs which are accepted by DTM with time complexity $f(n)$

~~Proof~~

using diagonalisation, we can show,

$$DTIME(n^k) \subsetneq DTIME(n^{k+1})$$ ← see hierarchy

→ Post's Correspondence Principle (PCP)

← Follow sipser

Σ,

$P = \{ (b, ca), (a, ab), (ca, a), (abc, c) \}$ ↑ finite set of elements from $(\Sigma^*)^2$

one instance of a PCP problem

Problem

Assume the pairs are tiles as such.

$\left[\frac{b}{ca}\right]$ $\left[\frac{a}{ab}\right]$ $\left[\frac{ca}{a}\right]$ $\left[\frac{abc}{c}\right]$

Then does there exist a seq. of tiles s.t. the string formed by the concat of upper strings is same as that formed by concat of lower.

For ex:

$\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right]$ ---- abcaaabc
---- abcaaabc) same

Hence, $\langle P \rangle$ is a yes instance.

Thm: PCP problem is undecidable.

Define PCP = set of all ^{yes} instances

formally = $\{\langle P \rangle \mid P \text{ is yes instance}\}$

PCP can be used to prove statements on decidability

like if 2 CFLs have the same language

Proof:

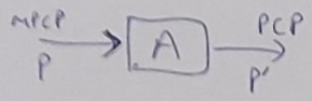
- **Modified PCP**

  ↳ extra restriction that $\left[\frac{u_1}{v_1}\right] = \left[\frac{t_1}{x_1}\right]$

  ↳ first tile

Idea: $L_u \leq_{MPCP} \leq PCP$, then PCP is undecidable

For $MPCP \leq PCP$,



Introduce new character $\star$ in Σ,

Then, if $P = \left\{\left[\frac{t_1}{x_1}\right], \cdots \left[\frac{t_n}{x_n}\right]\right\}$

$$P' = \left\{\left[\frac{\star \bar{t}_1}{\star \bar{x}_1 \star}\right], \cdots \underset{i=2\cdots n}{\left[\frac{\star \bar{t}_i}{\bar{x}_i \star}\right]} \cdots, \left[\frac{\star}{\epsilon}\right]\right\}$$

where $x = a_1 \cdots a_k$, $\bar{x} = a_1 \star a_2 \cdots \star a_k$

Clearly, P is MPCP yes iff P' is PCP yes.

for $L_u \leq MPCP$,

if $w = a_1 \cdots a_n$

let $ID_0 = q_0 a_1 \cdots a_n$

$ID_0 \# ID_1 \# \cdots \# ID_t$ ← computation history for a LBA

$\delta(q_0, a_1) = (q_1, b, R)$

$ID_1 = b q_1 a_2 \cdots a_n$

$\# q_0 a_1 a_2 \# \mid b q_1 a_2 \mid \cdots \#$

$\# q_0 a_1 \cdots a_n \# \mid b q_1 a_2 \mid \cdots \#$

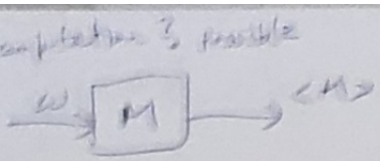
$\left[\frac{q_0 a_1}{b q_1}\right], \left[\frac{a_i}{a_i}\right], \left[\frac{\#}{\#}\right]$

$ID_0 \# \ldots \# ID_{k-1} \#$

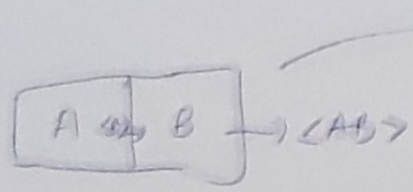
$ID_0 \# \ldots \# ID_{k-1} \# \sqsubset$

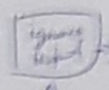
- Proof that "CFG ambiguous" is not decidable (MISSED)

- Self-replication is possible



SELF



Here A is 

English equivalent ⟹ "print 2 of these, second in quotes"
"print 2"

- Recursion Thm If $t: \Sigma^* \times \Sigma^* \to \Sigma^*$ is a computable function, then $\exists$ TM R s.t.

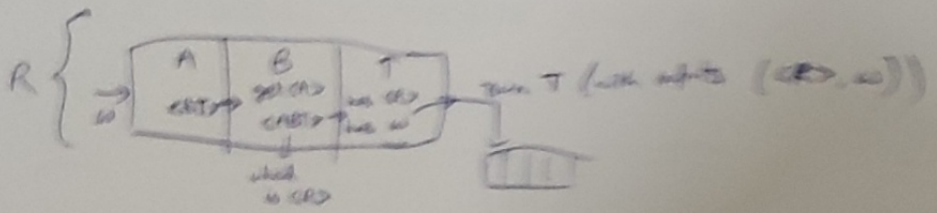
$$t(\langle R \rangle, w) = \gamma(w) \quad \forall w$$

where γ is the function computed by TM R

↑

Can be used to perform a task like, "get my self-description $\langle R \rangle$ for any arbitrary TM R."

Proof



$L = \{ \langle M \rangle \mid \nexists M' \text{ s.t. } L(M') = L(M) \text{ and } |\langle M' \rangle| < |\langle M \rangle| \}$

↖ lang of all minimal TM's

Thm: L is not r.e.

Proof:

Assume r.e. $\exists$ enumerator for L. (say E)

Then let M: (on input w)

Obtain $\langle M \rangle$

Run enumerator E to get a TM M' s.t. $|\langle M' \rangle| > |\langle M \rangle|$

Simulate M' on w and print $M'(w)$

↗

Notice $L(M) = L(M')$

But we chose M' which appears in E:

but $|\langle M' \rangle| > |\langle M \rangle|$.

Contradiction

- let $\sigma \in$ total rec. func

  $\exists$ TM M s.t. func computed by M and $\sigma(\langle M \rangle)$ are same.

Starting complexity theory

- **Complexity class P**

$DSPACE(f(n))$ → class of langs accepted by DTM with space complexity $f(n)$

Similarly, $DTIME(\cdot)$, $NSPACE(\cdot)$, $NTIME(\cdot)$ } for NTM's

For class $\mathcal{C}$, $co\text{-}\mathcal{C} = \{\bar{L} \mid L \in \mathcal{C}\}$

- For const $c < 1$,

if L is accepted by a DTM with time com. $f(n)$,

then $\exists$ a DTM with time com. $cf(n)$ which also accepts L.

Thm

$$DTIME(n^k) \subsetneq DTIME(n^{k+1})$$

Proof ??

Def^n

$$P = \bigcup_{k \in \mathbb{N}} DTIME(n^k) \quad , \quad PSPACE = \bigcup_{k>0} DSPACE(n^k)$$

And

$$P \subset PSPACE \subset EXPTIME \subset EXPSPACE \subset \dots$$

$O(f(n)) \subseteq O(2^{f(n)})$ (space ↓, time ↓)

$$L_{d,LBA} = \{\langle M \rangle \mid M \text{ is LBA s.t. } L(M) \notin \langle M \rangle\}$$

$$L_{d,n^k} = \{\langle M \rangle \mid M \text{ is TM with time com. } n^k \text{ s.t. } L(M) \notin \langle M \rangle\}$$

↳ can construct a machine with time com. n^{k+1} that accepts it.

Runs for atmost n^k steps with each step needing 'n' time to read the input

Wrt time & space com., DTM and NTM are diff^{n}. ?.

NTIME $(f(n)) \subseteq$ DTIME $(2^{O(f(n))})$

Def^{n}

$$NP = \bigcup_{k>0} NTIME(n^k) \quad , \quad NPSPACE = \bigcup_{k>0} NSPACE(n^k)$$

?.

Def: If $\exists$ a polytime algo s.t $x \in L'$ iff $A(x) \in L$.

then $L' \leq_m^p L$

Thm:
- Suppose $L' \leq_m^p L$ then

$$L \in P \Rightarrow L' \in P$$

- Suppose $L' \leq_m^p L$, then

$$L \in NP \rightarrow L' \in NP$$

- Q - whether $P = NP$?

- Def$^{\Delta}$: L is NP-Hard if $\forall L' \in NP$, $L' \leq_m^p L$

(class of 'hardest' probs in NP)

- Def$^{\Delta}$: L is NP-complete if L is NP-Hard & $L \in NP$.

→ Cook-Levin thm

↳ SAT is NP-complete.

↓

set of all satisfiable boolean formula

Proof: For some $L \in NP$, we must show $L \leq_A^p SAT$