Last Class

Mapping Reducibility

Def Given $A, B \subseteq \Sigma^*$ we say that A is **mapping reducible** to B (mapping reduction)

If $\exists$ computable function $f: \Sigma^* \to \Sigma^*$ s.t. $\forall w \in \Sigma^*$

$w \in A \iff f(w) \in B$

Notation: $A \leq_m B$

Idea: Roughly: A is (roughly) as easy as B
B is (roughly) as hard as A
Let's make this precise:

Thm If $A \leq_m B$ and B is decidable
then A is decidable.

If $A \leq_m B$ and B is T-rec
then A is T-rec.

Cor If $A \leq_m B$ and A is not decidable
then B is not decidable

(we will use this a lot)

(Similarly for not T-rec)

Def $HALT_{TM} = \{\langle M, w\rangle : M$ is a TM that halts on input $w\}$

Thm $HALT_{TM}$ is undecidable

Claim: $A_{TM} \leq_m HALT_{TM}$

Want: $\langle M, w\rangle \xmapsto{f} \langle M', w\rangle$

s.t. $\langle M, w\rangle \in A_{TM} \iff \langle M', w\rangle \in HALT_{TM}$

i.e. M accepts $w \iff M'$ halts on input w

Design for TM M': Just like M except that instead of rejecting if goes into an infinite loop

map $\langle M, w\rangle \mapsto \langle M', w\rangle$ easily computable

Correctness:

$\langle M', w\rangle \in HALT_{TM} \iff M'$ halts on input w

$\iff M'$ accepts w

$\iff M$ accepts w ∎

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

Thm E_{TM} is undecidable

Proof: We show that $A_{TM} \leq_m \overline{E_{TM}}$.

Want: $\langle M, w\rangle \xrightarrow{f} \langle M_w\rangle$ (machine depends on M and w.)

s.t. M accepts $w \iff L(M_w) \neq \emptyset$

Design TM M_w as follows:

M_w: On input x

Ignore x (e.g. erase it)

Run M on input w and accept iff M accepts

(w is hard-coded into the instructions for M_w or is the code of M.)

The mapping f that takes input $\langle M, w\rangle$ to produce $\langle M_w\rangle$ is clearly computable.

By construction $L(M_w) = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$

$\langle M, w\rangle \in A_{TM} \iff M$ accepts w

$\iff L(M_w) \neq \emptyset$

$\iff \langle M_w\rangle \in \overline{E_{TM}}$.

$\therefore A_{TM} \leq_m \overline{E_{TM}}$ and so $\overline{E_{TM}}$ is not decidable.

(Also $\overline{E_{TM}}$ is not T-rec.) ☐

Another problem:

$EQ_{TM} = \{\langle M_1, M_2\rangle : M_1, M_2 \text{ are TMs}$
$\& L(M_1) = L(M_2)\}$

Thm Neither EQ_{TM} nor $\overline{EQ_{TM}}$ is T-rec

Proof (a) EQ_{TM} is not T-rec:

Claim: $\overline{A_{TM}} \leq_m EQ_{TM}$

Want f s.t. $\langle M, w\rangle \xmapsto{f} \langle M_1, M_2\rangle$
and $\langle M, w\rangle \in \overline{A_{TM}} \iff L(M_1) = L(M_2)$

i.e. M does not accept w $\iff L(M_1) = L(M_2)$

Idea: $\langle M, w\rangle \longmapsto \langle M_w, M_\emptyset\rangle$

where (from reduction for E_{TM}):
- M_w is the TM that ignores its input and runs M on input w
- $M_\emptyset$ is a simple TM that always rejects

Clearly f is computable

Now $L(M_w) = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$

$L(M_\emptyset) = \emptyset$

Correctness: $L(M_\emptyset) \neq L(M_w) \iff M$ does not accept w.

(b) $\overline{EQ_{TM}}$ is not T-rec

Claim: $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$

ie $A_{TM} \leq_m EQ_{TM}$

Want f with $\langle M, w\rangle \xmapsto{f} \langle M_1, M_2\rangle$

s.t. M accepts $w \iff L(M_1) = L(M_2)$

Similar idea: $\langle M, w\rangle \xmapsto{f} \langle M_w, M_{\Sigma^*}\rangle$

Clearly f is computable

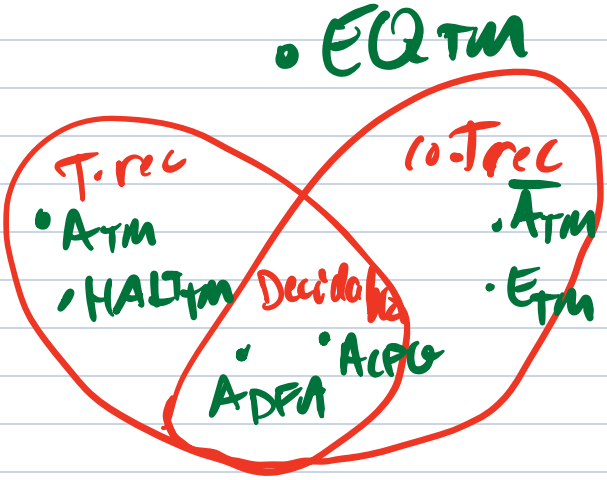
where M_{Σ^*} is a TM with input alphabet Σ that always accepts

$\therefore L(M_{\Sigma^*}) = \Sigma^*$

and $L(M_w) = \Sigma^* = L(M_{\Sigma^*})$
iff M accepts w.

$\therefore$ Reduction is correct ∎

The picture now:



Another property: Can we replace M by a DFA?

$REGULAR_{TM} = \{\langle M \rangle : M$ is a TM s.t. $L(M)$ is regular$\}$

$IRREGULAR_{TM} = \{\langle M \rangle : M$ is a TM s.t. $L(M)$ is not regular$\}$

Thm IRREGULAR_{TM} is not decidable

Proof We show $A_{TM} \leq_m \text{IRREGULAR}_{TM}$

We use the fact that $\emptyset$ is regular and $\{0^n1^n : n \geq 0\}$ is not

and design a TM M_w' s.t.

regular →

$$L(M_w') = \begin{cases} \emptyset & \text{if } M \text{ doesn't accept } w \\ \{0^n1^n : n \geq 0\} & \text{if } M \text{ does accept } w. \end{cases}$$

not regular ↗

M_w' = On input x:

Run M on input w

If M accepts w

Check to see if x is of the form 0^n1^n for some n and accept if it is.

else reject.

Like M_w except that we don't ignore x and only accept if it is the right form

Can easily compute $\langle M'_w \rangle$ from $\langle M, w \rangle$.

By construction.

M accepts $w \Leftrightarrow L(M'_w)$ is not regular

as required

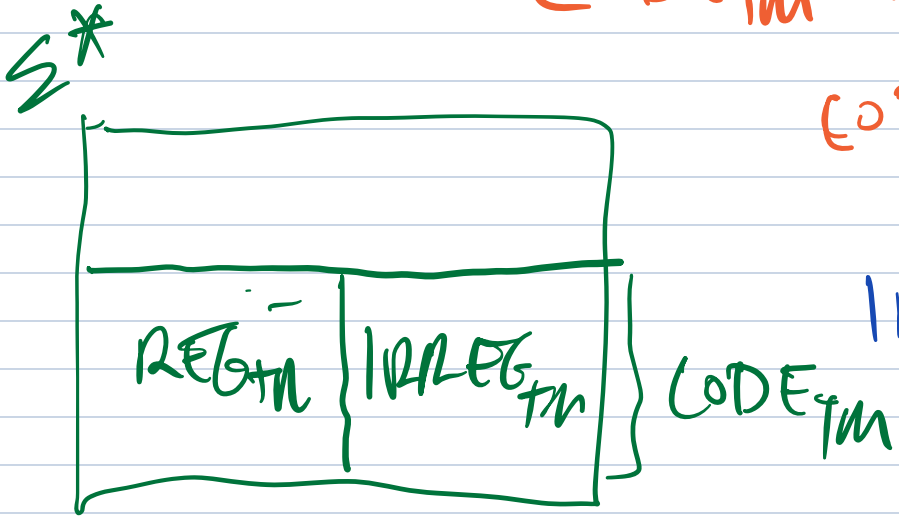
Cor $REGULAR_{TM}$ is not decidable

Proof: Note that $REGULAR_{TM}$ is "almost" the complement of $IRREGULAR_{TM}$:

only difference is that both only contain codes of TMs.

Define $CODE_{TM} = \{\langle M \rangle : M \text{ is a TM}\}$

$CODE_{TM}$ is decidable.



$IRREG_{TM} = CODE_{TM} \cap \overline{IRREG_{TM}}$

If $REGULAR_{TM}$ were decidable we could decide $IRREGULAR_{TM}$ by the following:

Check if input is a TM code
if not, reject
else
Run decider for $REGULAR_{TM}$
and do the opposite:

∎

E_{TM}, $REGULAR_{TM}$, $IRREGULAR_{TM}$ are each defined by properties of languages.

We show Rice's Theorem which says that figuring (almost) any interesting property of $L(M)$ is undecidable

Def^n Any property of languages P defines a language

$$P_{TM} = \{\langle M \rangle : M \text{ is a TM s.t. } L(M) \text{ has property } P\}$$

P_{TM} Rice

Examples of properties P: "$= \emptyset$": empty; "regular"

$P_{TM} = CODE_{TM}$ ← "Turing-recognizable"

decidable — $P_{TM} = \emptyset$ ← "not T-rec"

"trivial properties"

Def^n We say that a property P is non-trivial iff

- There is a TM M_1 s.t. $L(M_1)$ has property P
- There is a TM M_0 s.t. $L(M_0)$ does not have property P

Rice's Theorem: For any non-trivial property P

L_{TM} is undecidable.

ie. Can't tell I/O behavior just from looking at the code.

Proof: We split this into two cases:

Case 1: ϕ does not have property P,

this was the case for P = "is regular" and the proof will be almost the same except we use the TM M_1.

Claim $A_{TM} \leq_m L_{TM}$:

$\langle M, w \rangle \xmapsto{f} \langle M''_w \rangle$

M''_w: On input x

Run M on input w.

If M accepts w

Run M_1 on input x

and accept iff M_1 accepts

else reject

Map is clearly computable

By definition:

$$L(M''_w) = \begin{cases} L(M_1) & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$$

$L(M_1)$ has property P

$\emptyset$ does not have property P.

$\therefore \langle M, w \rangle \in A_{TM} \iff L(M''_w) \in P_{TM}$ and $A_{TM} \leq P_{TM}$ so P_{TM} is undecidable. required.

Case 2: $\emptyset$ has property P

In this case we consider the complement of property P, $\overline{P}$

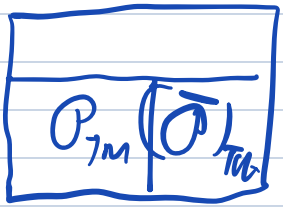
$\overline{P}$ is also not trivial (swap M_0, M_1)

By Case 1, $(\overline{P})_{TM}$ is not decidable

We now apply the same idea as in the proof for REGULAR$_{TM}$. If we could decide P_{TM} then we could decide

$$(\overline{P})_{TM} = CODE_{TM} \cap \overline{P_{TM}}$$

which we cannot.



Reductions summary:

Reductions

Thm If $A \leq_m B$ then
- if B is decidable then A is too
- if A is undecidable then B is too
- if B is T-rec then A is too
- if A is not T-rec then B also isn't T-rec

Steps to showing $A \leq_m B$

Step 0: Figuring things out:

Look for: $x \longmapsto f(x)$

x: in input format for A — $f(x)$: in input format for B

s.t. $x \in A \Rightarrow f(x) \in B$

$x \notin A \Rightarrow f(x) \notin B$

Write-up: Step 1: What is f?

Step 2: Why is f computable?

Step 3: Why does f satisfy $x \in A \Leftrightarrow f(x) \in B$?

Correctness: can do both steps together

(a) $x \in A \Rightarrow f(x) \in B$

(b) $x \notin A \Rightarrow f(x) \notin B$

Today a new method that can let us show problems for weaker models than TMs are also undecidable

{Accepting / Rejecting} Computation Histories of TM M on input w.

Recall: configurations $C \in \Gamma^* Q \Gamma \Gamma^*$ (upav.)

where M accepts w $\Leftrightarrow$ $q_0 w \vdash_M^* D$, $D \in \Gamma^* q_{accept} \Gamma^*$

$\exists t, C_0 = q_0 w, C_1, \ldots, C_t = D$ s.t.

$C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \cdots \vdash_M C_t$

that is
- $C_0 = q_0 w$
- $C_i \vdash C_{i+1}$ $i < t$
- $C_t \in \Gamma^* q_{accept} \Gamma^*$

Def'n Let $\# \notin \Gamma \cup Q$. An accepting computation history of M on input w is a string $x \in (\Gamma \cup Q \cup \{\#\})^*$ s.t.

$x = \# C_0 \# C_1 \# \cdots \# C_t \#$ for some t

where
- $C_0 = q w$
- $C_i \vdash_M C_{i+1}$ for $i < t$
- $C_t \in \Gamma^* q_{accept} \Gamma^*$

if M accepts w there is precisely one such string
M doesn't accept w There is no such string.

The first model we use this for is that linear bounded automata

Def^n A Linear Bounded Automata (LBA) is exactly like a (1-tape) TM except that
- it can never move outside the original cells containing the input
  - right move from right end stays where it is just like at left end.

(formal parts are the same as TM except that execution is different)

Note: This is deterministic. The non-deterministic version of LBA's are equivalent to "context-sensitive languages"
- these have rules of form $aA \to aw$ instead of $A \to w$
- substitution can only happen in certain contexts

Now we see that problems about LBA's can be much easier than for TM's

$$A_{LBA} = \{\langle M, w\rangle : M \text{ is an LBA s.t. } w \in L(M)\}$$

Thm A_{LBA} is decidable

Proof Obvious alg to try:

On input $\langle M, w\rangle$:
Simulate M on input w step by step
if M accepts then accept
M rejects then reject

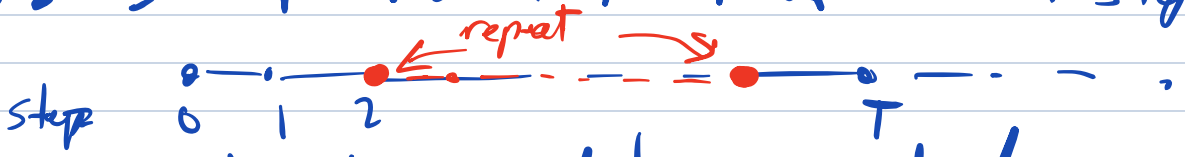
Problem: What if M runs forever?

Key idea to fix this: use bound on size of tape to bound # of possible configurations of M.

Let $n = |w|$. Can specify configuration of M on input w
- state $\in Q$
- head position $\in \{1, 2, \ldots, n\}$
- tape contents $\in \Gamma^n$

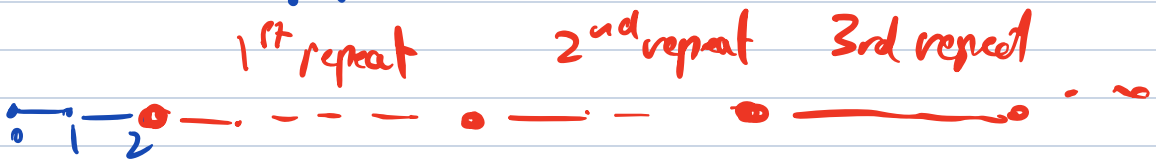
total $T = |Q| \cdot n \cdot |\Gamma|^n = T_M(n)$

∴ If computation of M runs for $\geq T$ steps



then there must be a repeated configuration (pigeonhole principle).

But then the computation will run forever



∴ Decider for A_{LBA}:

On input $\langle M, w \rangle$ where M is an LBA:
Simulate M on input w step by step for up to T steps
- if M accepts then accept
- if M rejects then reject
- if M reaches T steps then reject

$E_{LBA} = \{\langle M \rangle : M \text{ is an LBA with } L(M) = \emptyset\}$

Thm E_{LBA} is undecidable

Proof: We prove that $A_{TM} \leq \overline{E_{LBA}}$

$\langle M, w \rangle \longmapsto \langle K_{M,w} \rangle$

$K_{M,w}$ is an LBA

We want

M accepts $w \Rightarrow L(K_{M,w}) \neq \emptyset$

M doesn't accept $w \Rightarrow L(K_{M,w}) = \emptyset$

idea: Design $K_{M,w}$ to check whether its input is an accepting computation history of M on input w

If M does accept w, $L(K_{M,w})$ will contain such a history

If M doesn't accept w, $K_{M,w}$ won't accept any strings

$K_{M,w}$: Input alphabet = $(Q \cup \Gamma \cup \{\#\})$ for M

On input x

- Check that x begins and ends with # and has strings in $\Gamma^* Q \Gamma^*$ between every pair of #.
- Check that x begins with $\#q_0 w \#$
- Check that each $\#C_i \#C_{i+1}\#$ satisfies $C_i \vdash_M C_{i+1}$
- Check that x ends with $\#C\#$ where $C \in \Gamma^* Q_{accept} \Gamma^*$

Two consecutive strings that differ in only a few positions

Like checking $\{x\#x : x \in \{0,1\}^*\}$ as in TM from week 1

All the computation can be done without moving off X.
so an LBA can do them.
(Know how w and rules of M hardwired.)

We can do something similar to show problems for CFG's are hard:

Problem: CFG's can't produce $\{x\#x : x \in \{0,1\}^*\}$
so we can't use same form of accepting computation history
but they can generate
$\{x\#x^R : x \in \{0,1\}^*\}$ which will be good enough.

Defn A reversing accepting computation history of M on input w is a string X of the form

R if C_i is odd

$$\#C_0\#C_1^R\#C_2\#C_3^R\#C_4\#\cdots\#C_t$$

Defn $INTER_{CFG} = \{\langle G_1, G_2\rangle : G_1, G_2 \text{ are CFGs and } L(G_1) \cap L(G_2) \neq \emptyset\}$
i.e. G_1 and G_2 can generate a string in common

Thm $INTER_{CFG}$ is undecidable.

Proof $A_{TM} \leq_m INTER_{CFG}$
$\langle M, w\rangle \mapsto \langle G_1, G_2\rangle$

Idea: Create two CFGs G_1, G_2.

whose only possible strings in common would be a reversing accepting computation history.

$q_0 w$

$C_0 \# C_1^R \# C_2 \# C_3^R \# C_4 \# \cdots \# C_t$

Using same ideas as for $\{x\#x^R : x \in \{0,1\}^*\}$ (same as $\{x^R\#x : x \in \{0,1\}^*\}$ like palindromes)

to generate strings of form

$C \# D^R$ where $C \vdash_M D$

or $C^R \# D$ where $C \vdash_M D$

$G_1 = G_{even}$ will generate all strings of above s.t.

$C_{2i} \vdash_M C_{2i+1}$ for all i (C_{2i-1} and C_{2i} may be unrelated)

and C_t contains q_{accept}

$G_2 = G_{odd}$ will generate all strings as above s.t.

$C_0 = q_0 w$

$C_{2i+1} \vdash_M C_{2i+2}$ for all i (C_{2i} and C_{2i+1} may be unrelated)

and C_t contains q_{accept}

May it easy to compute

Only possible string generated by both G_1, G_2 are reversing accepting computation histories.

On exists iff M accepts w.

$\square$

$ALL_{CFG} = \{\langle G \rangle : G \text{ is a CFG s.t. } L(G) = \Sigma^*\}$

Thm ALL_{CFG} is undecidable

Proof Claim $A_{TM} \leq_m \overline{ALL_{CFG}}$

Idea: On input $\langle M, w \rangle$ design CFG
— that generates all strings in
$(\Gamma \cup Q \cup \{\#\})^*$
that are **not** rejecting
accepting computation histories.

M accepts $w \iff$ this is not Σ^*

Note: Can easily get CFG G for
$\{x \# y^R : x \neq y\}$
(which is the same as $\{x^R \# y : x \neq y\}$)

If a string $\# C_0 \# C_1^R \# \cdots \# C_t^{(R?)} \#$
is not an accepting computation history
then it satisfies one of the following
- it doesn't begin with $\# C_0 w \#$ (easy to generate)
- it doesn't end with $\# u \text{ accept } v \#$ for $u, v \in \Gamma^*$ (easy to generate)
- there is some i s.t.
$\# C_i \# C_{i+1}^R \#$ doesn't satisfy $C_i \vdash_M C_{i+1}$
or $\# C_i^R \# C_{i+1} \#$ doesn't satisfy $C_i \vdash_M C_{i+1}$

the ideas from

The Grammar $\langle G \rangle$ created for inputs $\langle n, w \rangle$ just needs rules to generate each of these cases.

which can be done as sketched above.