Lecture 11 CSE 431
Intro to Theory of Computation

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: A : x ⟼ B : $f(x)$

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 $x \in A \Leftrightarrow f(x) \in B$?

Sometimes I. 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: configuration $C \in \Gamma^* Q \Gamma \Gamma^*$ (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 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 A Linear Bounded Automaton (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.

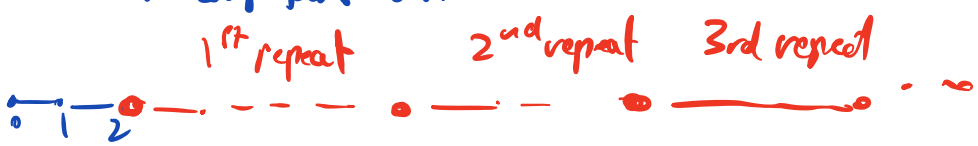
Let $n = |w|$. Can specify configuration of M on input w
- state $\in Q$
- head position $\in \{1, 2, \ldots, n\}$
- tape contents $\in \Gamma^n$

total $T = |Q| \cdot n \cdot |\Gamma|^n$

∴ 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.
(knowing 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

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

R if t is odd

Defn $\overline{\text{INTER}_{CFG}} = \{\langle G_1, G_2\rangle : G_1, G_2 \text{ are CFGs and } L(G_1)\cap L(G_2)\neq\emptyset\}$

i.e. G_1 and G_2 can generate a string in common

Thm $\overline{\text{INTER}_{CFG}}$ is undecidable.

Proof $A_{TM} \leq_m \overline{\text{INTER}_{CFG}}$
$\langle M, w\rangle \mapsto \langle G_1, G_2\rangle$

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 idea 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}

Maybe it is easy to compute

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

On exists iff M accepts w.

□

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

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 history.

↑ M accepts $w \Leftrightarrow$ 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 $\# q_0 w \#$ (easy to generate)
- it doesn't end with $\# u q_{accept} v \#$ 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 $C_i \vdash_M C_{i+1}$

Use ideas from

The Grammar $\langle G \rangle$ created for inputs $\langle M, w \rangle$ just needs rules to generate each of these cases.
which can be done as sketched above ☺

Next time: Gödel's Incompleteness Theorem (6.2) and some useful material about CFGs.