18.404/6.840 Lecture 10

Last time:

- The Reducibility Method for proving undecidability and T-unrecognizability
- General reducibility
- Mapping reducibility

Today: (Sipser §5.2)

- The Computation History Method for proving undecidability
- The Post Correspondence Problem is undecidable
- Linearly bounded automata
- Undecidable problems about LBAs and CFGs

Remember

To prove some language B is undecidable, show that A_{TM} (or any known undecidable language) is reducible to B.

Revisit Hilbert's 10th Problem

Recall $D = \{\langle p \rangle |$ polynomial $p(x_1, x_2, \dots, x_k) = 0$ has integer solution)

Hilbert's 10th problem (1900): Is D decidable?

Theorem (1971): No

Proof: Show A_{TM} is reducible to D. [would take entire semester]

Do toy problem instead which has a similar proof method.

Toy problem: The Post Correspondence Problem.

Method: The Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \left[\frac{t_1}{b_1} \right], \left[\frac{t_2}{b_2} \right], \ldots, \left[\frac{t_k}{b_k} \right] \right\}$$

a match is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\left[\frac{t_{i_1}}{b_{i_1}} \right] \left[\frac{t_{i_2}}{b_{i_2}} \right] \ldots \left[\frac{t_{i_l}}{b_{i_l}} \right]$ where $t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$

Example: $P = \left\{ \left[\frac{\text{ab}}{\text{aba}} \right], \left[\frac{\text{aa}}{\text{aba}} \right], \left[\frac{\text{ba}}{\text{aa}} \right], \left[\frac{\text{abab}}{\text{b}} \right] \right\}$

Match: ✓

Check-in 10.1

Let $P_1 = \left\{ \left[\frac{\text{aa}}{\text{aaba}} \right], \left[\frac{\text{ba}}{\text{ab}} \right], \left[\frac{\text{ab}}{\text{ba}} \right] \right\}$

Does P_1 have a match?

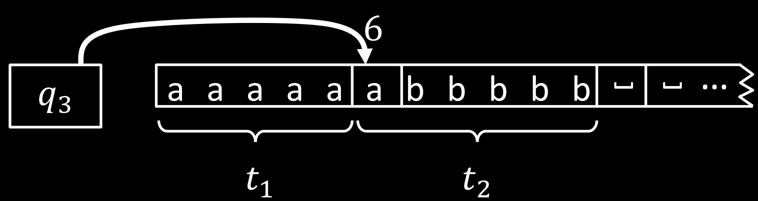
(a) Yes.

(b) No.

Check-in 10.1

TM Configurations

Defn: A <u>configuration</u> of a TM is a triple (q, p, t) where
q = the state,
p = the head position,
t = tape contents
representing a snapshot of the TM at a point in time.



Configuration: $(q_3, 6, \text{aaaaaabbbbb})$
Encoding as a string: aaaaa$q_3$abbbbb

Encode configuration (q, p, t) as the string $t_1 q t_2$ where
$t = t_1 t_2$ and the head position is on the first symbol of t_2.

TM Computation Histories

Defn: An <u>(accepting) computation history</u> for TM M on input w is a sequence of configurations $C_1, C_2, \ldots, C_{\text{accept}}$ that M enters until it accepts.

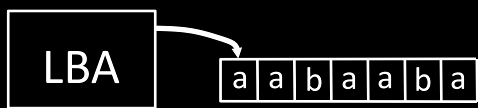
Encode a computation history $C_1, C_2, \ldots, C_{\text{accept}}$ as the string $C_1 \# C_2 \# \cdots \# C_{\text{accept}}$ where each configuration C_i is encoded as a string.

A computation history for M on $w = w_1 w_2 \cdots w_n$.
Here say $\delta(q_0, w_1) = (q_7, \text{a}, \text{R})$ and $\delta(q_7, w_2) = (q_8, \text{c}, R)$.

$$\underbrace{q_0 w_1 w_2 \cdots w_n}_{C_1} \; \# \; \underbrace{\text{a} q_7 w_2 \cdots w_n}_{C_2} \; \# \; \underbrace{\text{ac} q_8 w_3 \cdots w_n}_{C_3} \; \# \; \cdots \; \# \; \underbrace{\cdots\, q_{\text{accept}} \cdots}_{C_{\text{accept}}}$$

Linearly Bounded Automata

Defn: A linearly bounded automaton (LBA) is a 1-tape TM that cannot move its head off the input portion of the tape.



Tape size adjusts to length of input.

Let $A_{\text{LBA}} = \{\langle B, w\rangle \mid \text{LBA } B \text{ accepts } w\}$

Theorem: A_{LBA} is decidable

Proof: (idea) If B on w runs for long, it must be cycling.

Claim: For inputs of length n, an LBA can have only $|Q|\times n\times|\Gamma|^n$ different configurations.

Therefore, if an LBA runs for longer, it must repeat some configuration and thus will never halt.

Decider for A_{LBA}:

$D_{\text{A-LBA}} =$ "On input $\langle B, w\rangle$

1. Let $n = |w|$.
2. Run B on w for $|Q|\times n\times|\Gamma|^n$ steps.
3. If has accepted, *accept*.
4. If it has rejected or is still running, *reject*."

must be looping

E_{LBA} is undecidable

Let $E_{\text{LBA}} = \{\langle B\rangle \mid B \text{ is an LBA and } L(B) = \emptyset\}$

Theorem: E_{LBA} is undecidable

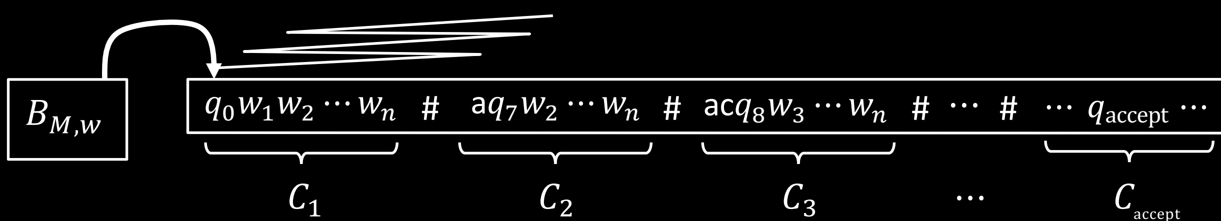
Proof: Show A_{TM} is reducible to E_{LBA}. Uses the computation history method.

Assume that TM R decides E_{LBA}

Construct TM S deciding A_{TM}

$S =$ "on input $\langle M, w\rangle$

1. Construct LBA $B_{M,w}$ which tests whether its input x is an accepting computation history for M on w, and only accepts x if it is.
2. Use R to determine whether $L(B_{M,w}) = \emptyset$.
3. *Accept* if no. *Reject* if yes."



Check-in 10.2

What do you think of the Computation History Method? Check all that apply.

(a) Cool !

(b) Just another theorem.

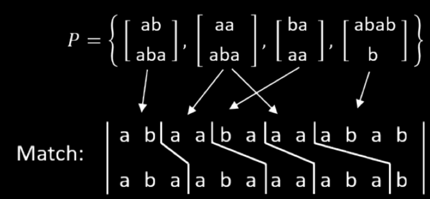
(c) I'm baffled.

(d) I wish I was in 6.046.

Check-in 10.2

PCP is undecidable

Recall $PCP = \{\langle P \rangle \mid P \text{ has a match }\}$



Theorem: PCP is undecidable

Proof: Show A_{TM} is reducible to PCP. Uses the computation history method.

Technical assumption: Match must start with $\begin{bmatrix} t_1 \\ b_1 \end{bmatrix}$. Can fix this assumption.

Assume that TM R decides PCP

Construct TM S deciding A_{TM}

$S =$ "on input $\langle M, w \rangle$

1. Construct PCP instance $P_{M,w}$ where a match corresponds to a computation history for M on w.
2. Use R to determine whether $P_{M,w}$ has a match.
3. *Accept* if yes. *Reject* if no."

Constructing $P_{M,w}$

Make $P_{M,w}$ where a match is a computation history for M on w.

$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} \# \\ \# q_0 w_1 \cdots w_n \# \end{bmatrix}$ (starting domino)

For each $a, b \in \Gamma$ and $q, r \in Q$ where $\delta(q, a) = (r, b, \mathrm{R})$

put $\begin{bmatrix} q & a \\ b & r \end{bmatrix}$ in $P_{M,w}$

(Handles right moves. Similar for left moves.)

Ending dominos to allow a match if M accepts:

$\begin{bmatrix} a \; q_{\text{accept}} \\ q_{\text{accept}} \end{bmatrix}$ $\begin{bmatrix} q_{\text{accept}} \; a \\ q_{\text{accept}} \end{bmatrix}$

Check-in 10.3

What else can we now conclude?

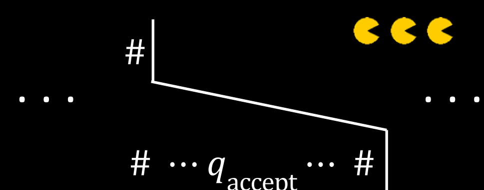
Choose all that apply.

(a) PCP is T-unrecognizable.

(b) $\overline{PCP}$ is T-unrecognizable.

(c) Neither of the above.

Illustration:
$w = 223$
$\delta(q_0, 2) = (q_7, 4, \mathrm{R})$



Match completed!
… one detail needed.

Check-in 10.3

ALL_{CFG} is undecidable

Let $ALL_{\mathrm{CFG}} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \Sigma^*\}$

Theorem: ALL_{CFG} is undecidable

Proof: Show A_{TM} is reducible to ALL_{PDA} via the computation history method.

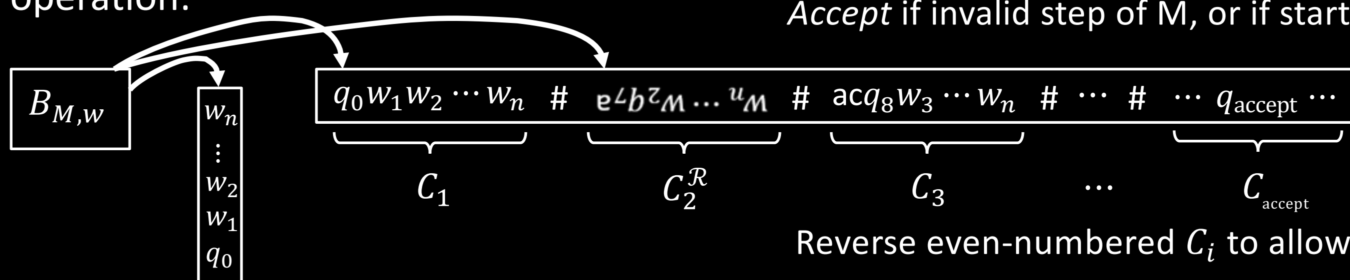
Assume TM R decides ALL_{PDA} and construct TM S deciding A_{TM}.

$S =$ "On input $\langle M, w\rangle$

1. Construct PDA $B_{M,w}$ which tests whether its input x is an accepting computation history for M on w, and only accepts x if it is NOT.
2. Use R to determine whether $L(B_{M,w}) = \Sigma^*$.
3. *Accept* if no. *Reject* if yes."

$B_{M,w}$ operation:

Nondeterministically push some C_i and pop to compare with C_{i+1}.
Accept if invalid step of M, or if start wrong, or if end isn't accepting.



Reverse even-numbered C_i to allow comparing with C_{i+1} via stack.

Computation History Method - recap

Computation History Method is useful for showing the undecidability of problems involving testing for the existence of some object.

D Is there an integral solution (to the polynomial equation)?

E_{LBA} Is there some accepted string (for the LBA)?

PCP Is there a match (for the given dominos)?

ALL_{CFG} Is there some rejected string (for the CFG)?

In each case, the object is the computation history in some form.

Quick review of today

1. Defined configurations and computation histories.
2. Gave The Computation History Method to prove undecidability.
3. A_{LBA} is decidable.
4. E_{LBA} is undecidable.
5. PCP is undecidable.
6. ALL_{CFG} is undecidable.

Eliminating the technical assumption

Technical assumption: Match must start with $\begin{bmatrix} t_1 \\ b_1 \end{bmatrix}$.

Fix this assumption as follows.

Let $P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \dots, \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$ where we require match to start with $\begin{bmatrix} t_1 \\ b_1 \end{bmatrix}$.

Create new $P' = \left\{ \begin{bmatrix} t_1 \\ \bar{\bar{b}}_1 \end{bmatrix}, \begin{bmatrix} \hat{t}_1 \\ \hat{b}_1 \end{bmatrix}, \begin{bmatrix} \hat{t}_2 \\ \hat{b}_2 \end{bmatrix}, \dots, \begin{bmatrix} \hat{t}_k \\ \hat{b}_k \end{bmatrix} \right\}$

For any string $u = u_1, \dots, u_k$, let

$\star u \;\; = * u_1 * u_2 * \cdots * u_k$

$u \star \;\; = \;\; u_1 * u_2 * \cdots * u_k *$

$\star u \star = * u_1 * u_2 * \cdots * u_k *$

Then let $P' = \left\{ \begin{bmatrix} \star t_1 \\ \star b_1 \star \end{bmatrix}, \begin{bmatrix} \star t_1 \\ b_1 \star \end{bmatrix}, \begin{bmatrix} \star t_2 \\ b_2 \star \end{bmatrix}, \dots, \begin{bmatrix} \star t_k \\ b_k \star \end{bmatrix}, \begin{bmatrix} *\$ \\ \$ \end{bmatrix} \right\}$