Computational Histories

Mark Greenstreet, CpSc 421, Term 1, 2008/09

- Computational Histories
  - The Undecidability of E_{LBA}.
  - An Undecidability result for CFLs.
- The Post Correspondence Problem

Showing Universality

- So far, we've used two methods to demonstrate undecidable results:
  - If the computation is being performed by a TM, we show that solving some problem is at least has hard as solving the halting problem, or A_{TM}, or their complements.
  - We've shown that other models of computation have the equivalent undecidable problems by showing that they can simulate TMs. For example, we've shown that a 2-PDA can simulate a TM.
- Another very general approach is to represent the computation that a TM performs as a string.
  - Some computational models that are not powerful enough to (conveniently) simulate a TM are powerful enough to be able to recognize valid, accepting comptations.
  - We can then show that questions about the language recognized by these other models are undecidable.
  - Because the computation is given to us (we only have to check that it is valid), we can show that seemingly weaker models hit the same problems of undecidability as TMs.

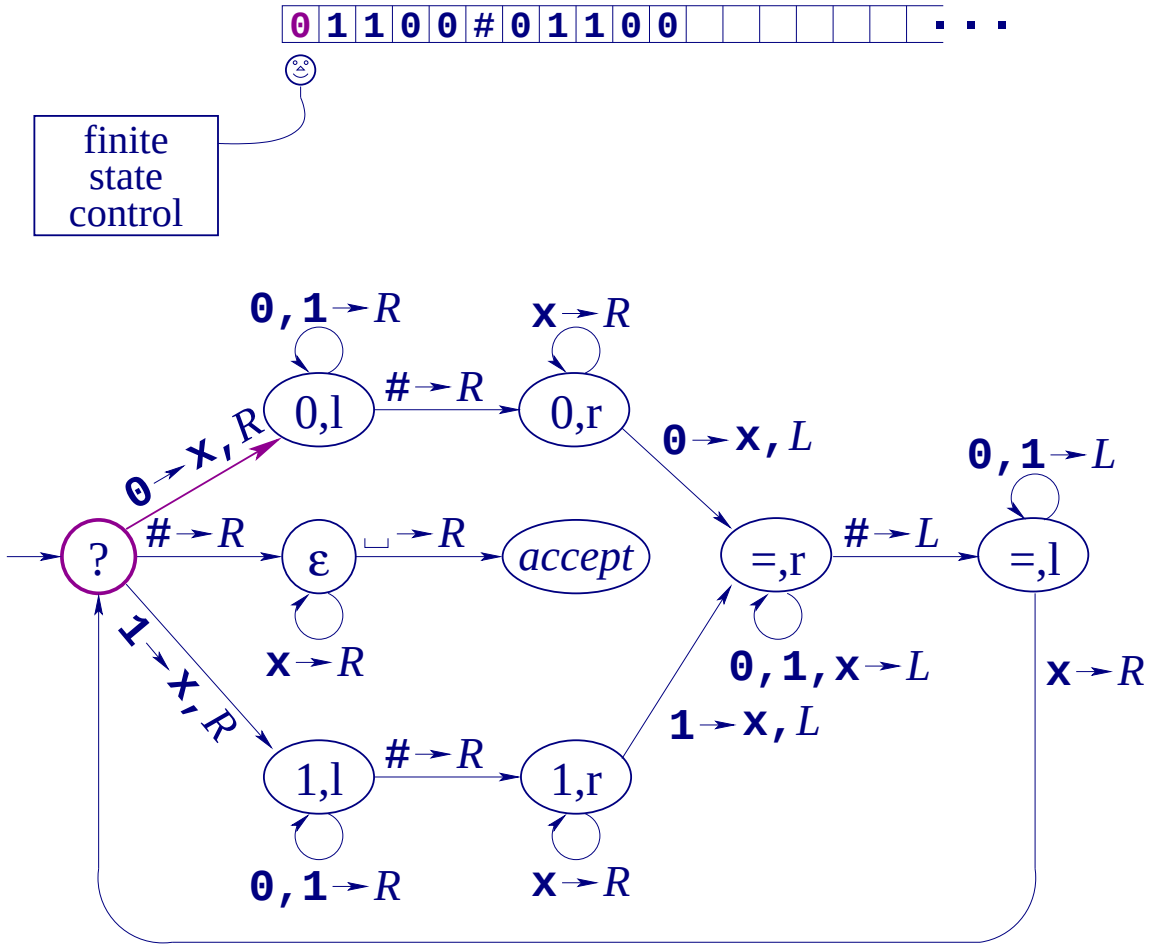
Turing Machine Configurations

- A Turing Machine is a 7-tuple: $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:
- A configuration is a 3-tuple (u, q, v) where
  - $u \in \Gamma^*$ is the tape content to the left of the head.
  - $v \in Q$ is the current state of the Turing machine.
  - $v \in \Gamma^*$ is the tape content starting at the head and to the right. The first symbol of v is the current symbol under the tape head. There are an infinite number of blanks to the right of v.
- M accepts w if there is a sequence of configurations, $C_0, C_1, \ldots C_k$ such that
  - $C_0 = q_0\, w$;
  - For all $0 \leq i < k$, $C_i \xrightarrow[M]{1} C_{i+1}$;
  - For all $0 \leq i < k$, the state for C_i is not the the reject state; and
  - The state for C_k is the accept state.

Computational Histories

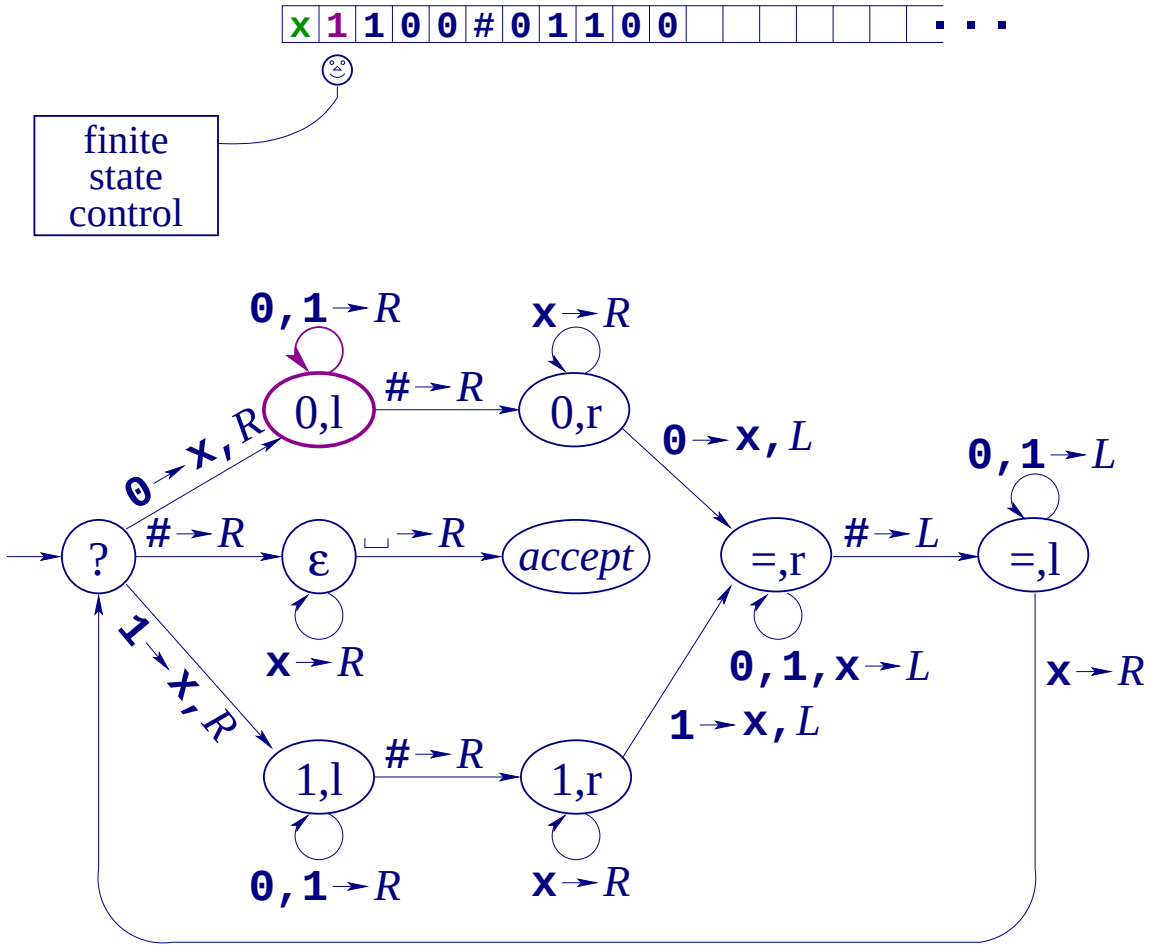
- TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ accepts w iff there is a sequence of configurations $C_0, C_1, \ldots C_k$ such that $C_0 = q_0\, w$, $\forall 0 \leq i < k.\ C_i \xrightarrow[M]{1} C_{i+1}$; $\forall 0 \leq i < k.\ C_i$ is not in the accept or reject state; and $C_k = u\, q_{accept}\, v$ for some $u, v \in \Gamma^*$.
- Let $\#$ be a symbol that is not in Q or Γ.
- The computational history for M accepting w is $C_0 \# C_1 \# \ldots \# C_k$.
- We can use computational histories to reduce A_{TM} and $\overline{A_{TM}}$ to other problems.

A Machine for $w\#w$



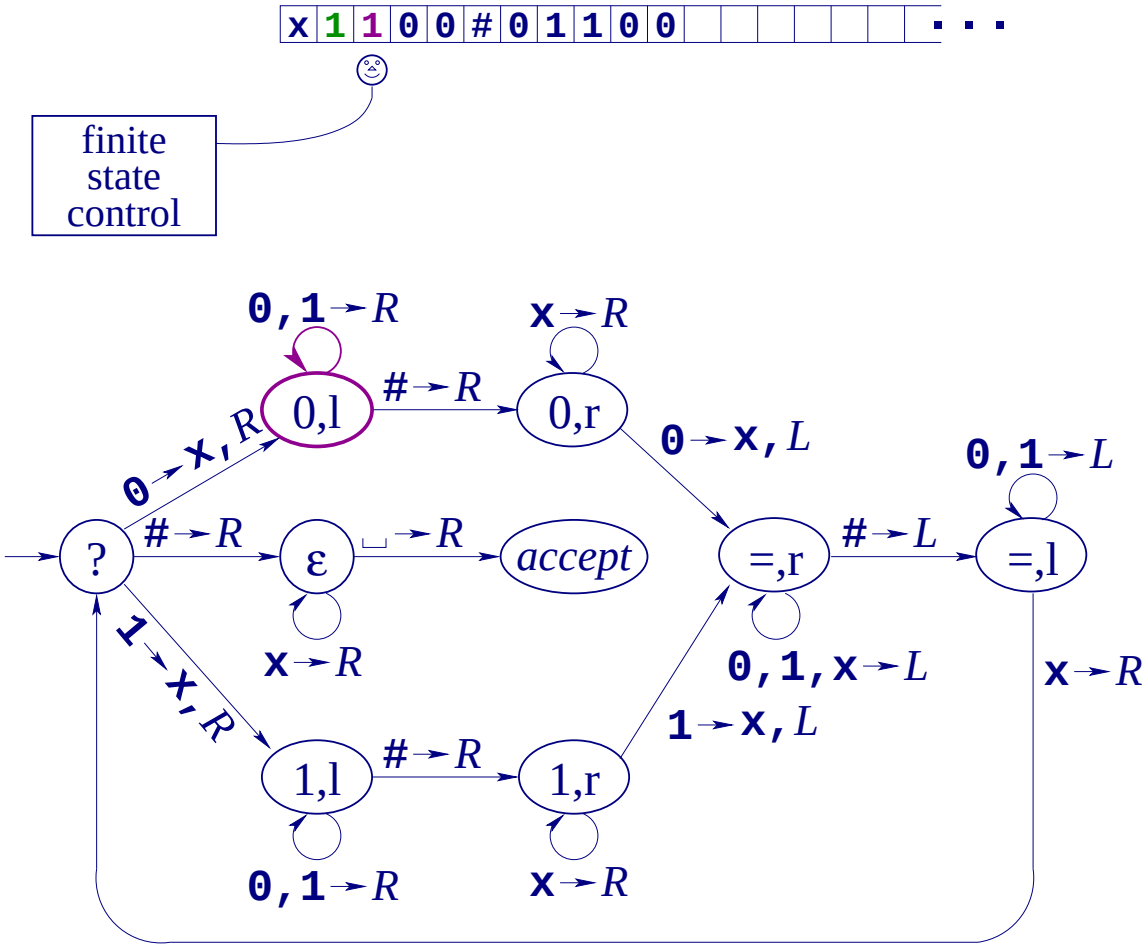
- History: $q_?01100\$01100$

A Machine for $w\#w$



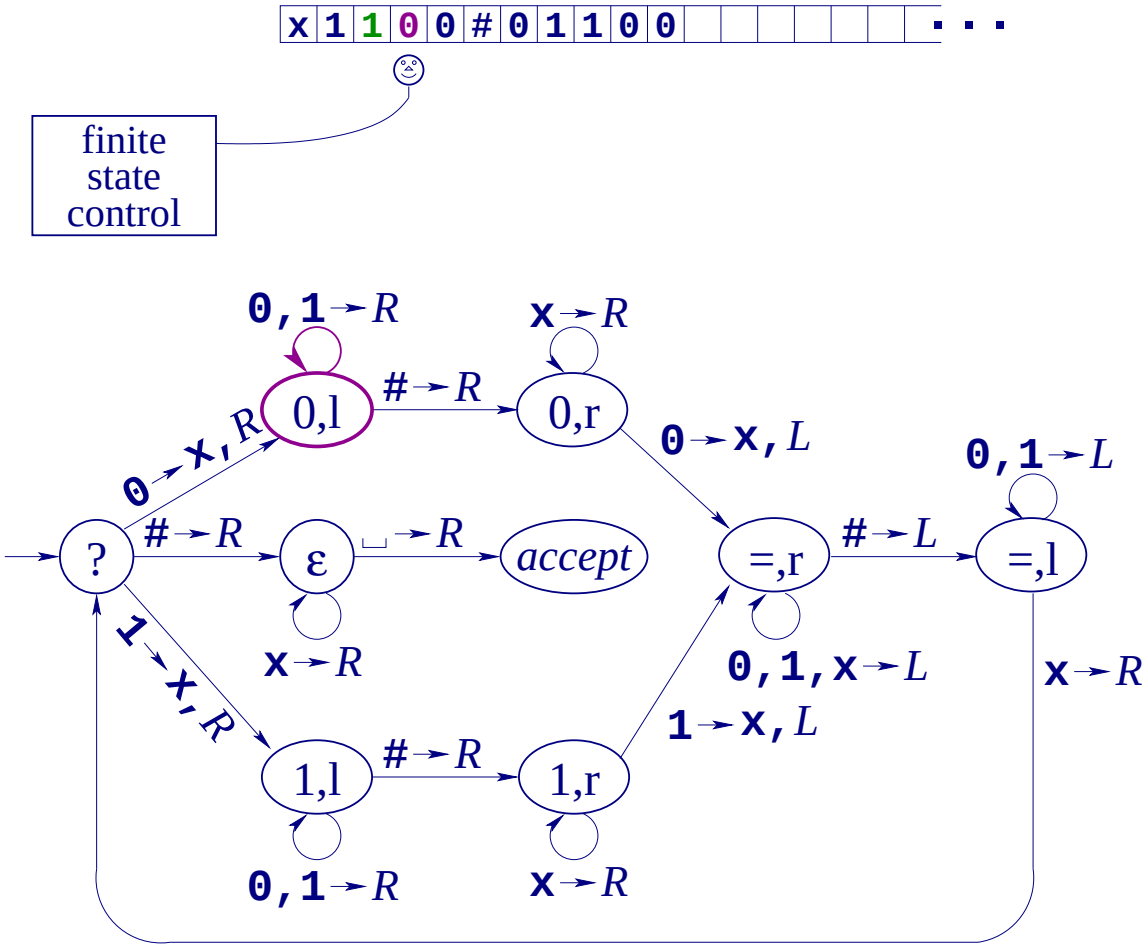
- History: $q_?01100\$01100 \quad \#xq_{(0,1)}1100\$01100$

A Machine for $w\#w$



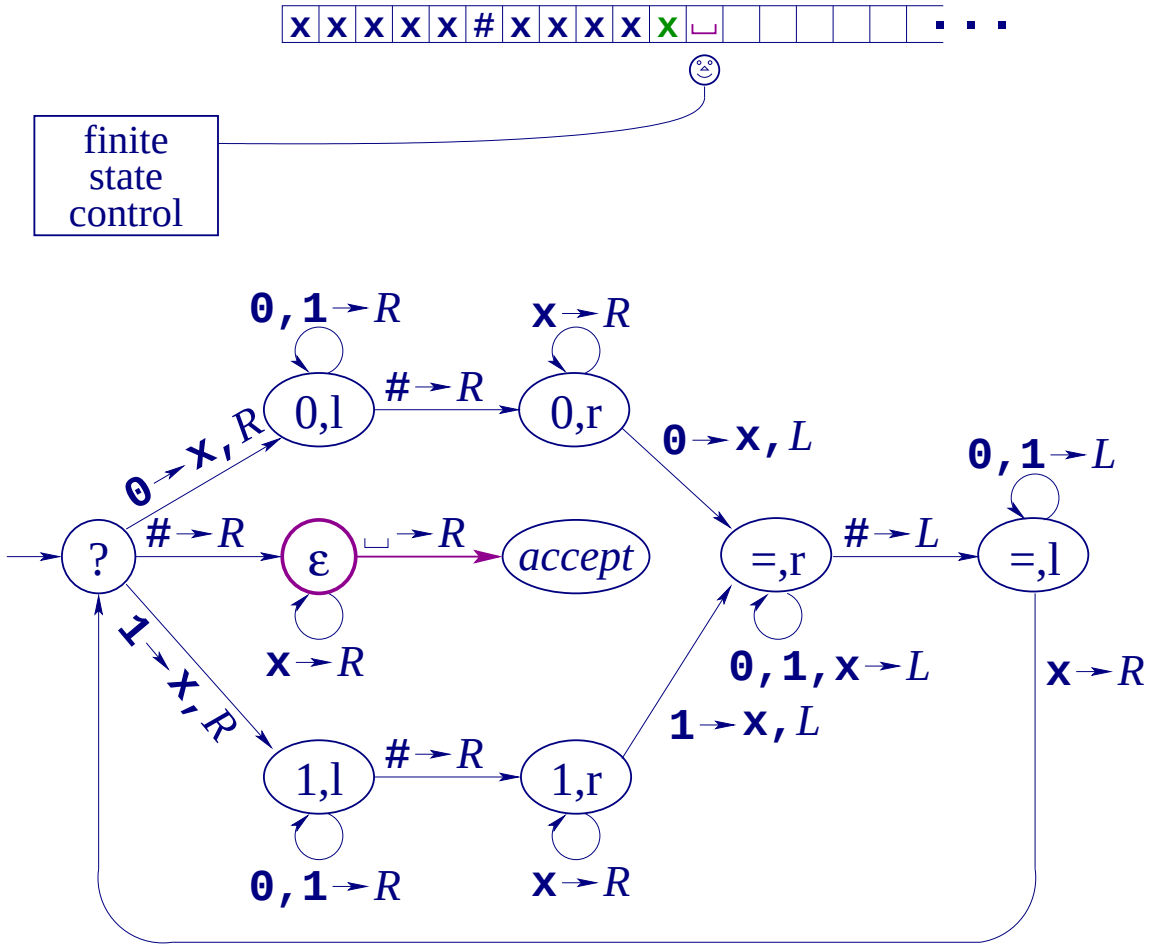
- History: $q_?01100\$01100\ \#xq_{(0,1)}1100\$01100\quad \#x1q_{(0,1)}100\$01100$

A Machine for $w\#w$



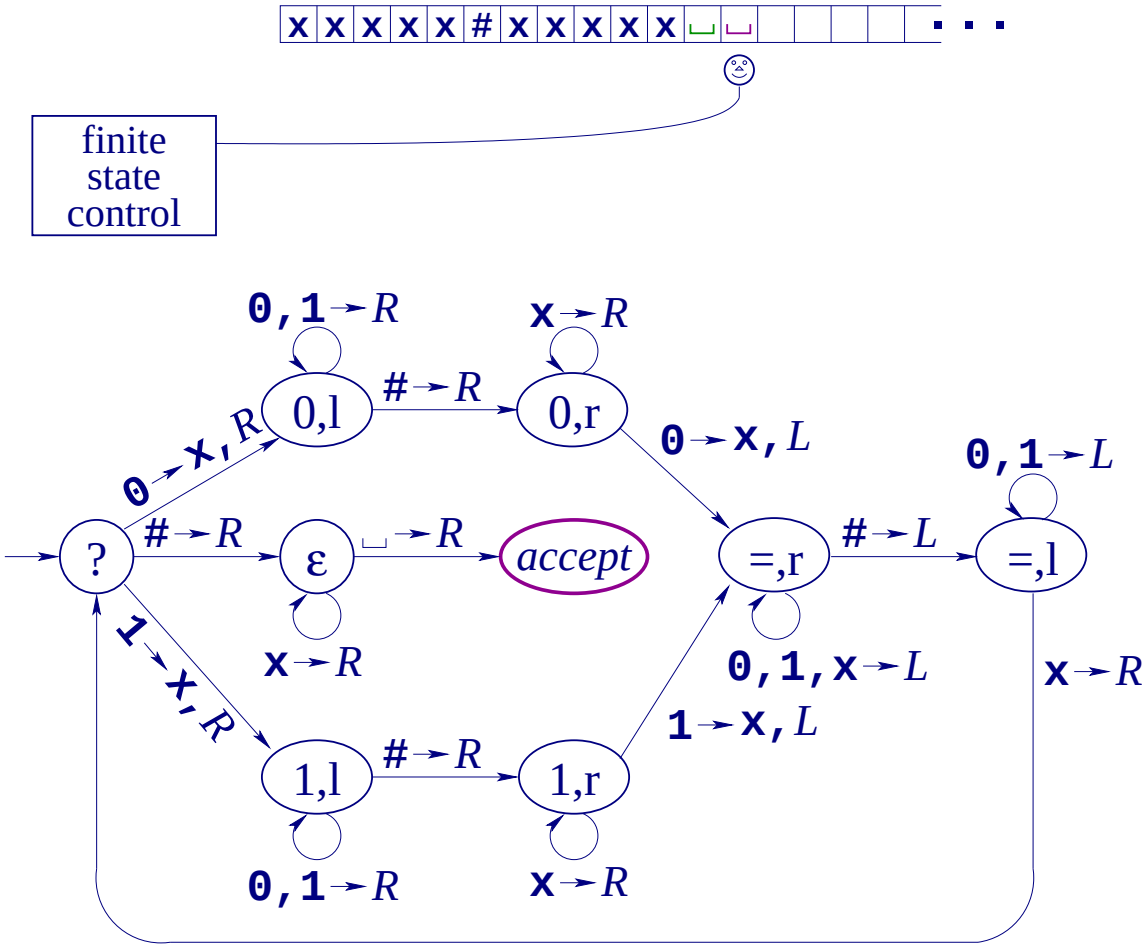
- History: $q_?01100\$01100\ \#xq_{(0,1)}1100\$01100\ \#x1q_{(0,1)}100\$01100$ $\#x11q_{(0,1)}00\$01100$

A Machine for $w\#w$



- History: $q_?01100\$01100\ \#xq_{(0,1)}1100\$01100\ \#x1q_{(0,1)}100\$01100$
  $\#x11q_{(0,1)}00\$01100\quad \ldots\#xxxxx\$xxxxxq_\epsilon\square$

A Machine for $w\#w$



- History: $q_?01100\$01100\ \#xq_{(0,1)}1100\$01100\ \#x1q_{(0,1)}100\$01100$
  $\#x11q_{(0,1)}00\$01100\ \ldots\ \#xxxxx\$xxxxxq_\epsilon\square$ $\#xxxxx\$xxxxx\square q_{accept}\square$

Linear Bounded Automata (LBAs)

- An LBA is like a TM except that it has a bounded amount of tape.
- In particular, the input string is bracketted between a left-endmarker, $\vdash$, and a right-endmarker $\dashv$.
  - An LBA cannot overwrite either of the endmarkers.
  - An LBA cannot move its head past either of the endmarkers.
  - Thus, an LBA can only use as many tape squares as were used to write the input.
- LBAs and decidability
  - The halting problem for LBAs is Turing-decidable (but not LBA decidable).
  - However, there are questions about LBA computation that are not Turing decidable. We'll look at one in the next few slides.

E_{LBA} is Undecidable

- $E_{LBA} = \{B \mid B$ describes an LBA and $L(B) = \emptyset\}$.
- Let $VC_{M,w} = \{x \mid x$ is a valid, computational history in which M accepts $w\}$.
- For any M and w, we construct an LBA, B, such that $L(B) \neq \emptyset$ iff M accepts w.
  - B first checks to make sure that its input tape matches $q_0 w \# (\Gamma^* Q \Gamma^* \#)^* \Gamma^* q_{accept} \Gamma^*$.
    This is a regular language; so, it's straightforward to make a B that does this check. If the input fails this check, B rejects.
  - Then, B checks each pair of consecutive configurations to make sure that they correspond to actions of M.
  - If each pair is valid, B accepts, else B rejects.

Checking pairs of configurations:

- Mark the first symbol of the tape.
  Scan to the first symbol of the next configuration and mark it.
  Return to the first symbol on the tape.
  while(true) {
  if(the symbol under the head is in Q)
  check a move for the head at the left end of the tape.
  else if(the symbol to the right of the marked one is in Q)
  check the move for this position.
  else make sure the marked symbol on the next
  configuration matches this one.
  if(the marked symbol on the left is $\#$)
  if(the right configuration is the last one)
  accept.
  else move markers to compare next pair of configurations.
  }

- reject if any of these checks fail.

Checking Histories: Example

A history for the machine that recognizes $w\$w$:

$$q_?01100\$01100\#xq_{(0,1)}1100\$01100\#xxq_{(0,1)}100\$01100$$
$$\#xxxq_{(0,1)}00\$01100\ldots\#xxxxx\$xxxxxq_\epsilon\square$$
$$\#xxxxx\$xxxxx\square q_{accept}\square$$

Reducing $\overline{A_{TM}}$ to E_{LBA}

- Assume that $M_{E_{LBA}}$ is a TM that decides E_{LBA}.
- $M_{\overline{A_{TM}}}$ does the following on input $M\#w$:
  - Construct the description of B as described above.
  - Runs $M_{E_{LBA}}$ with B's description as input.
  - If $M_{E_{LBA}}$ accepts then accept: $w \notin L(M)$.
  - If $M_{E_{LBA}}$ rejects then reject: $w \in L(M)$.
  - Note that LBA gets enough tape to decide $w \in L(M)$ because the input includes the entire computational history.
- $\overline{A_{TM}}$ is not decidable. Thus, E_{LBA} is not decidable either.

An Undecidable Problem for CFLs

- Let G be a CFG. $L(G) = \Sigma^*$ is undecidable.
- Proof: given M and w we construct a PDA, P, that generates all strings that are not valid computational histories for M running on w and accepting. If $L(G) = \Sigma^*$, then $w \notin L(M)$.
- We have to modify the computational history slightly: we'll write it $C_0 \# C_1^{\mathcal{R}} \# C_2 \# C_3^{\mathcal{R}} \ldots C_k$ (or $C_k^{\mathcal{R}}$ if k is odd.
- The CFG is the union of three cases:
  - The string is not of the form $q_0 w \#(\Gamma^* Q \Gamma^* \#)^* \Gamma^* q a \Gamma^*$. This is a regular language.
  - There is some i such that C_{i+1} is not a valid successor of C_i. P pushes the string C_i onto its stack and pops it off while reading C_{i+1}, checking the validity of the computation.
    - If i is even, then it checks the reverse of the configuration string.
    - Else i is odd and it checks the forward version.

The Post Correspondence Problem

- Given a set, P of pairs of strings:

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

  where each $t_i, b_i \in \Sigma^*$,

- Question: Does there exist a sequence $i_1, i_2, \ldots i_n$ such that:

$$t_{i_1} t_{i_2} \cdots t_{i_n} = b_{i_1} b_{i_2} \cdots b_{i_n}$$

  Note: the same pair can occur multiple times, i.e. there can be $j \neq m$ s.t. $i_j = i_m$.

Post Correspondence is undecidable

- Proof by computational histories.
- Sketch:
  - Start with a pair that has the initial configuration for a TM on the bottom and an empty string on top.
  - Include pairs in P whose top strings match the current configuration, and whose bottom strings build the next configuration.
  - A bunch of details to:
    - Account for moving the tape head.
    - Extend the tape with blanks when needed.
    - Force the first pair of a solution to be the one that gives the initial configuration.
    - . . .
- We'll go through the full proof in the next lecture.

This coming week (and beyond)

- Reading
  - Today: Sipser 5.2
  - Nov. 12 (Wednesday): Sipser 5.3
  - Nov. 14 (Friday): Sipser 7.1
  - Nov. 17 (Friday): Sipser 7.2
- Homework
  - (Today): HW 8 due.
  - Nov. 14 (Friday): HW 10 goes out.
  - Nov. 17 (a week from Today): HW 9 due.