Computational Histories and the Post Correspondence Problem

Mark Greenstreet, CpSc 421, Term 1, 2006/07

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

- Post's Correspondence Problem

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 $C_0 = q_0\, w$, $\forall 0 \le i < k.\ C_i \xrightarrow[M]{1} C_{i+1}$; $\forall 0 \le i < k.\ C_i$ 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 $\overline{atm}$ to other problems.

A Machine for $w\#w$

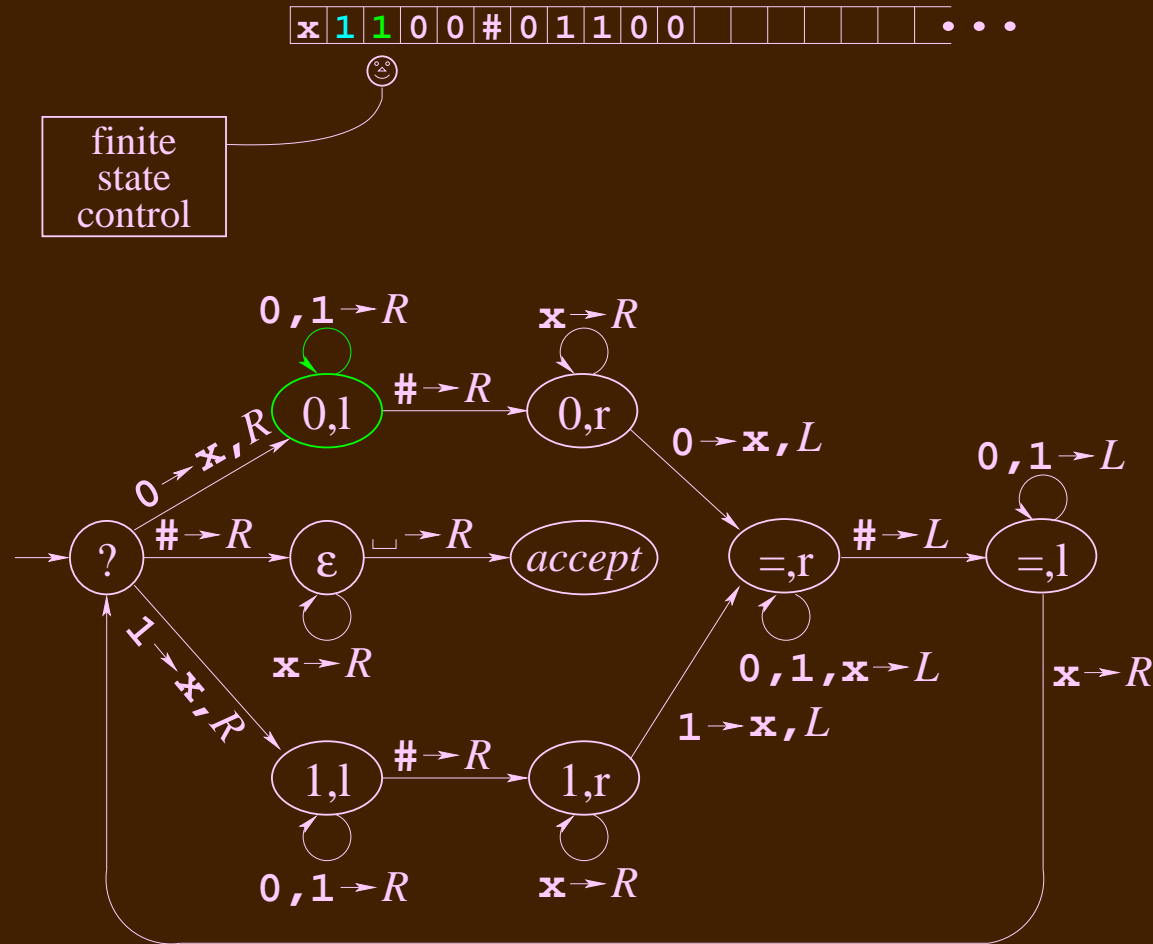
`0 1 1 0 0 # 0 1 1 0 0` $\cdots$

finite
state
control
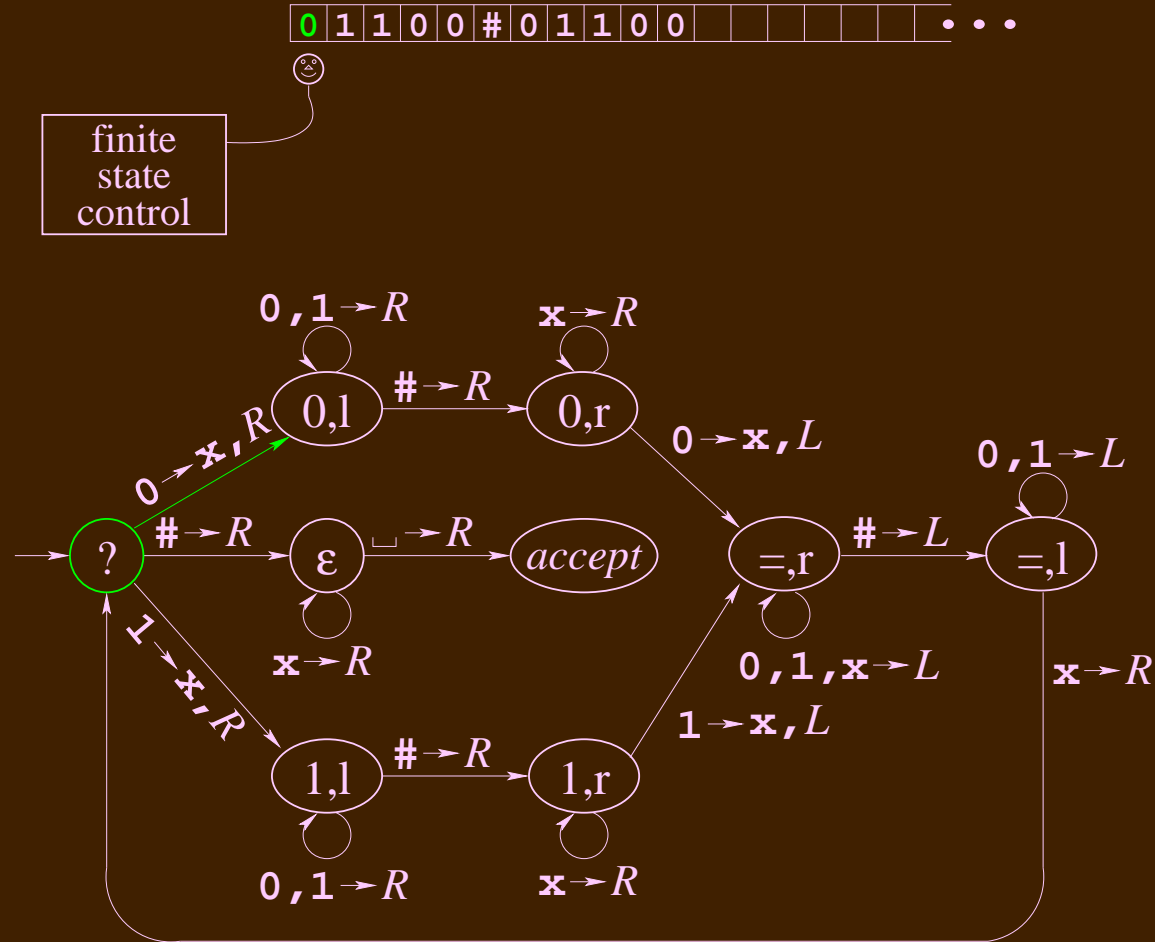
- History: $q_?$01100$01100
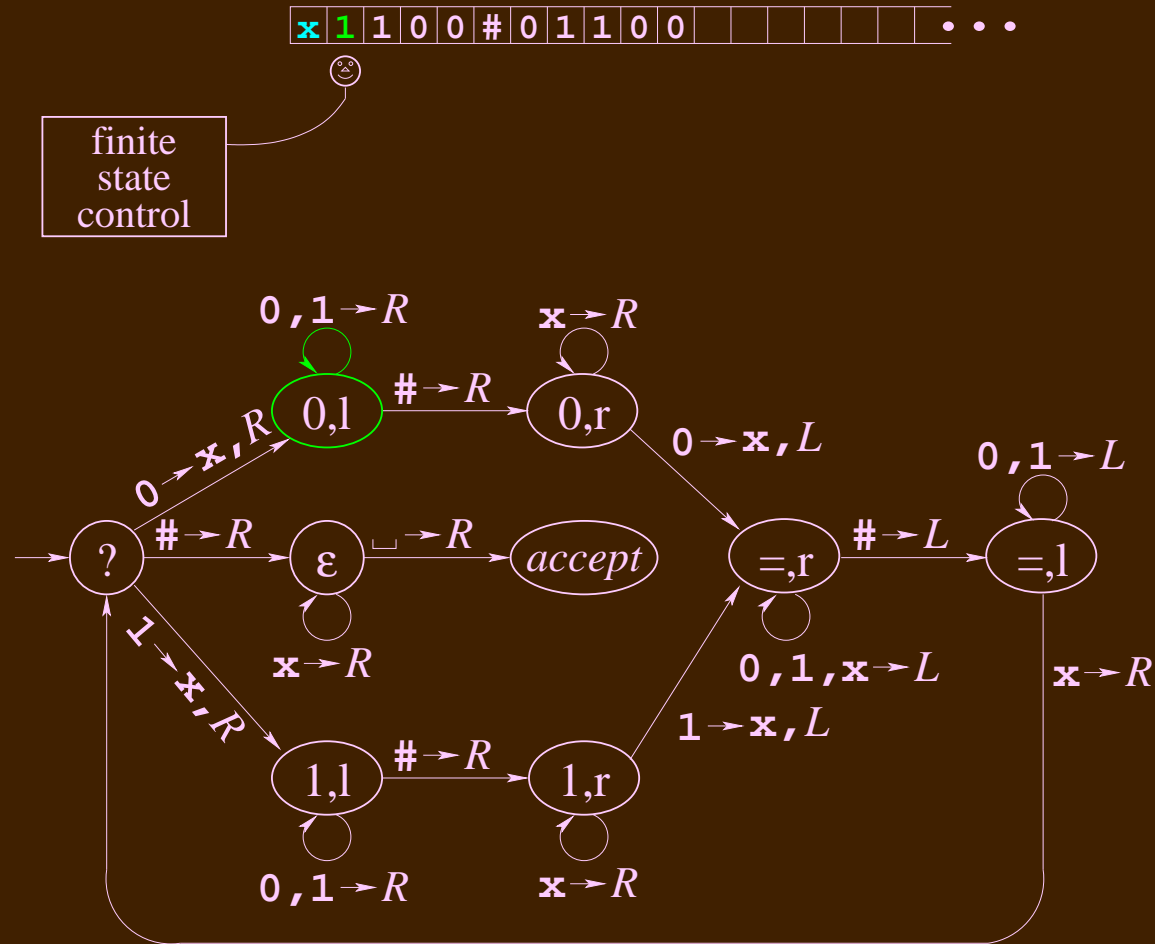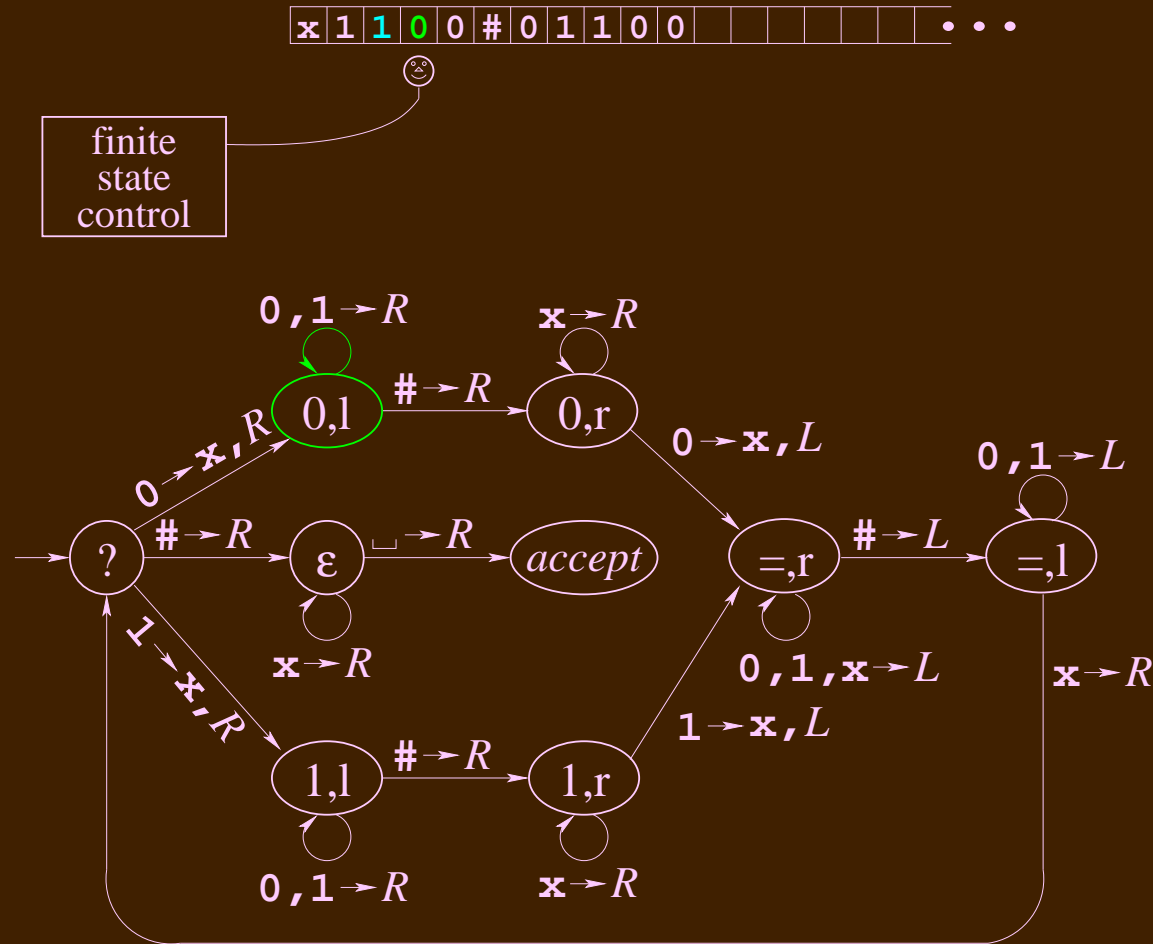
A Machine for $w\#w$

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

A Machine for $w\#w$

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

A Machine for $w\#w$

| x | 1 | 1 | 0 | 0 | # | 0 | 1 | 1 | 0 | 0 | | | | | | | $\cdots$ |

finite
state
control

$0,1\to R$

$\mathtt{x}\to R$

$\mathtt{0}\to\mathtt{x},R$

$\#\to R$ (0,l) $\#\to R$ (0,r)

$0\to\mathtt{x},L$

$0,1\to L$

$\#\to R$ (ε) $\sqcup\to R$ (accept)

(?) $=,r$ $\#\to L$ $=,l$

$1\to\mathtt{x},R$

$\mathtt{x}\to R$

$0,1,\mathtt{x}\to L$

$1\to\mathtt{x},L$

$\mathtt{x}\to R$

(1,l) $\#\to R$ (1,r)

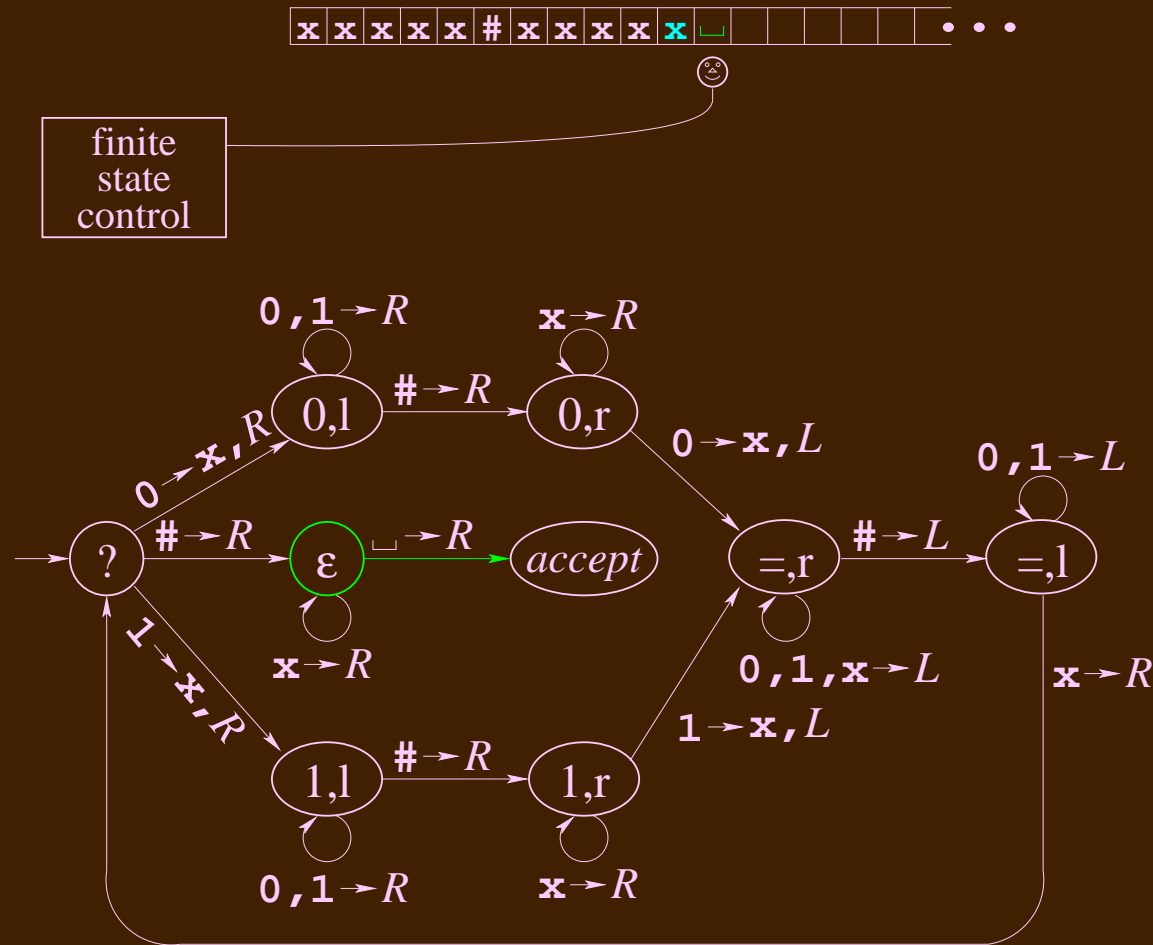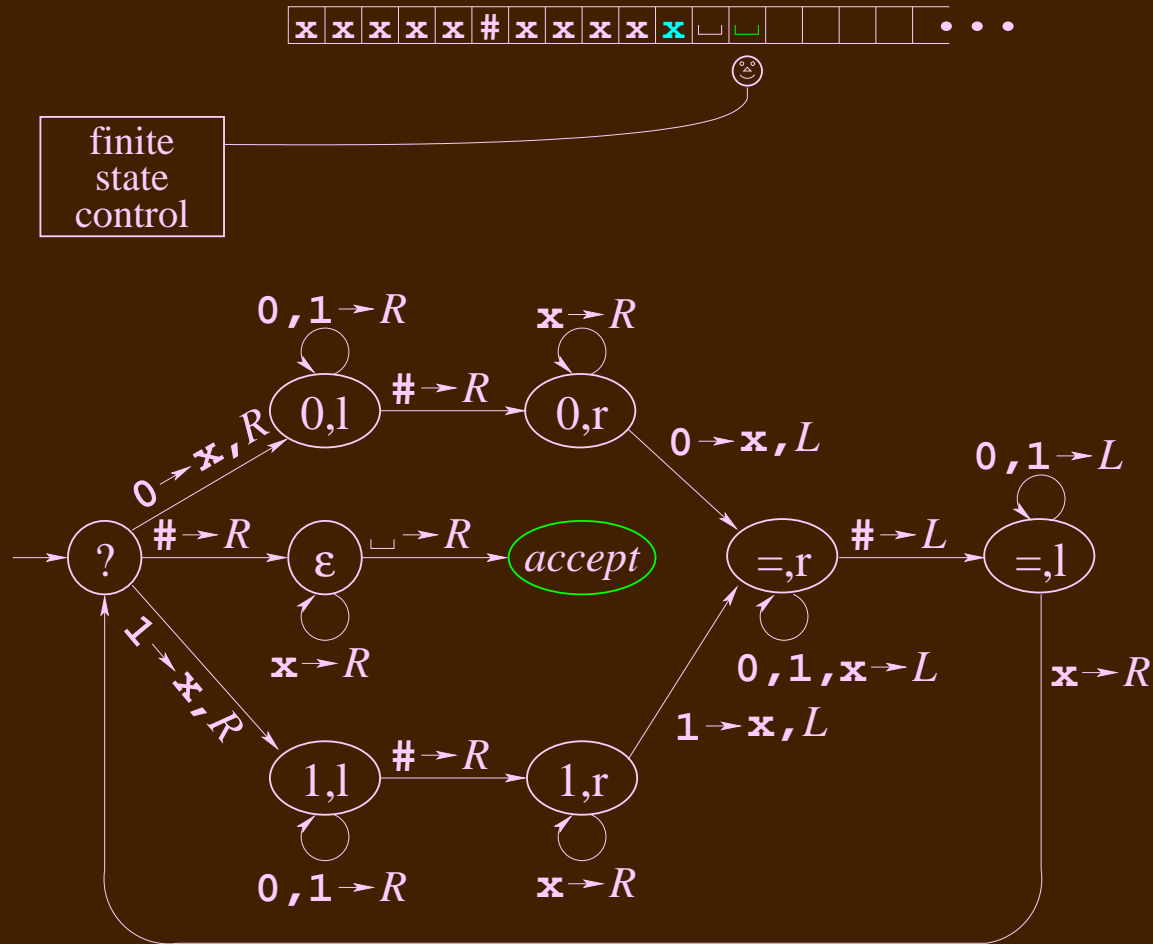$0,1\to R$ $\mathtt{x}\to R$

● History: $q_?01100\$01100\ \#xq_{(0,1)}1100\$01100\ \#xxq_{(0,1)}100\$01100$

$\#xxxq_{(0,1)}00\$01100$

A Machine for $w\#w$

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

  $\#xxxq_{(0,1)}00\$01100 \ \ldots\#xxxxx\$xxxxxq_\epsilon\square$

A Machine for $w\#w$

$$x\,x\,x\,x\,x\,\#\,x\,x\,x\,x\,x$$

finite
state
control

$0,1 \to R$

$\mathbf{x} \to R$

$0,l \xrightarrow{\#\to R} 0,r$

$0 \to \mathbf{x}, L$

$0,1 \to L$

$0 \to \mathbf{x}, R$

$? \xrightarrow{\#\to R} \varepsilon \xrightarrow{\sqcup \to R} accept$

$=,r \xrightarrow{\#\to L} =,l$

$\mathbf{x} \to R$

$0,1,\mathbf{x} \to L$

$1 \to \mathbf{x}, R$

$1 \to \mathbf{x}, L$

$\mathbf{x} \to R$

$1,l \xrightarrow{\#\to R} 1,r$

$0,1 \to R$

$\mathbf{x} \to R$

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

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

E_{LBA} is Undecidable

- Recall, $E_{LBA} = \{B \mid B$ 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.

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

- Assume that $M_{E_{LBA}}$ is a TM that decides E_{LBA}.

- $M_{A_{TM}}$ 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 computatioon its stack and pops.
    - If i is even, then it checks the reverse of the configuration string.
    - Else i is odd and it checks the forward version.

Reading List:

- Today: Sipser, 5.2.

- Nov. 20: Sipser, 5.3

- Nov. 22: Sipser, 6.1

- Nov. 24: Sipser, 6.2

- Nov. 27: Sipser, 6.2 (cont., final exam cut-off)

- Nov. 29: The GHz race is over, and what it means for you.

- Dec.   1: Everything else about complexity theory.