Equivalence between PDAs and CFGs

Theorem 2.20

A language L is context-free if and only if L is accepted by some PDA M.

Proof:

$\Downarrow$ First define a leftmost derivation

$S \overset{*}{\underset{G}{\Rightarrow}} w$

This is a derivation of w in which we always replace the leftmost variable symbol A by some $A \to u \in R$

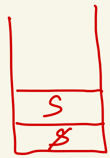
$$S \overset{*}{\Rightarrow} u_1 A v_1 \Rightarrow u_1 u v_1$$

$\in \Sigma^*$

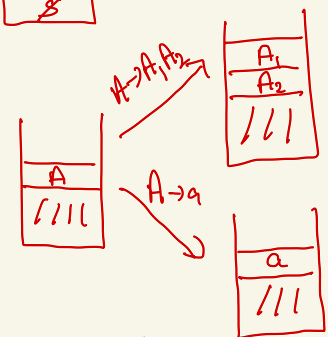
Idea: Given a CFG G s.t. $L = L(G)$ we make a PDA which simulates leftmost derivation of strings in $L(G)$.

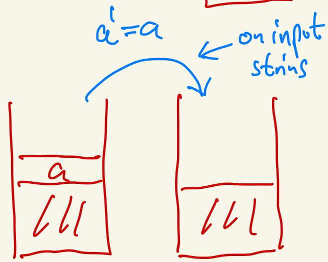
We may assume that G is in Chomsky normal form

1. Place $\$$ on stack and S on top (S is start symbol of G)

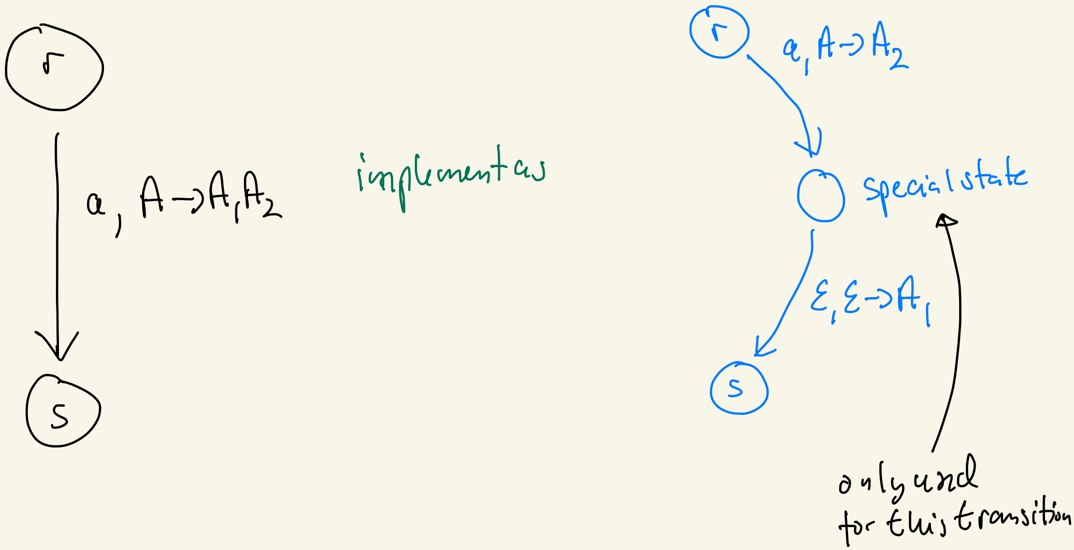


2. Repeat
   - If top symbol on stack is a variable A: select non-deterministically a rule $A \to \beta \in R$ and replace A on stack by β (either A_1A_2 or $a \in \Sigma$)
   - If top of stack is some $a \in \Sigma$: read next input symbol a'; if $a' = a$ remove a from stack. Else reject this branch of the calculation
   - If top of stack is $\$$: enter accept state

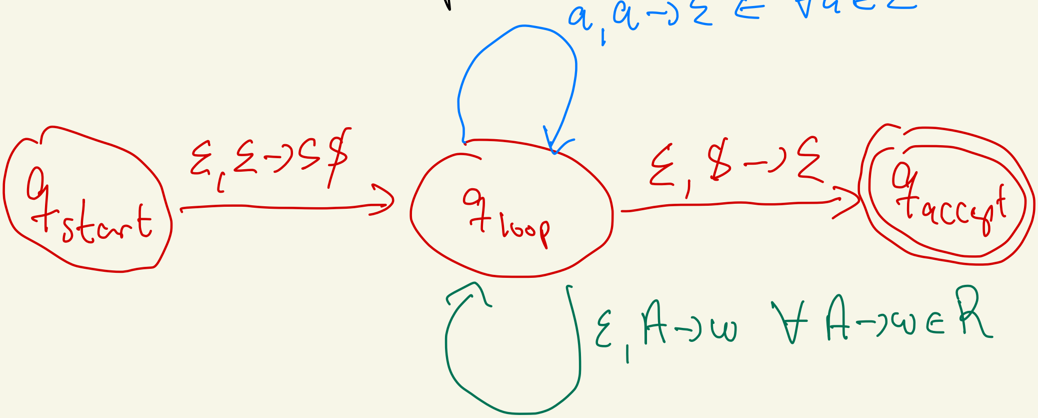




pushing a string on the stack (instead of just one stack symbol)



The PDA M (with special states implicit)



Lemma 2.27

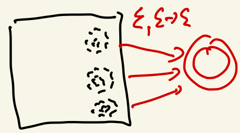
$L = L(M)$ for some PDA M

$\Downarrow$

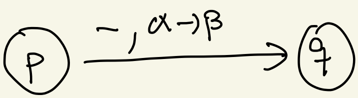
$L = L(G)$ for some context-free grammar G

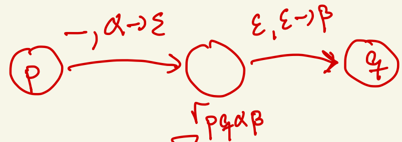
Proof: more complicated

Step 1: Define a restricted PDA

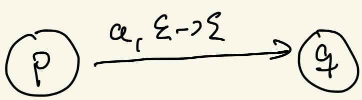


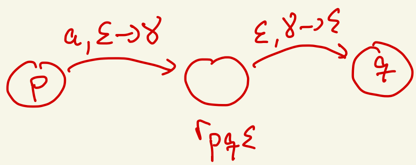
1. One single accept state
2. Always empties stack before accepting a string
3. Every transition either pops one symbol or pushes one symbol





special state for this transition





γ special symbol

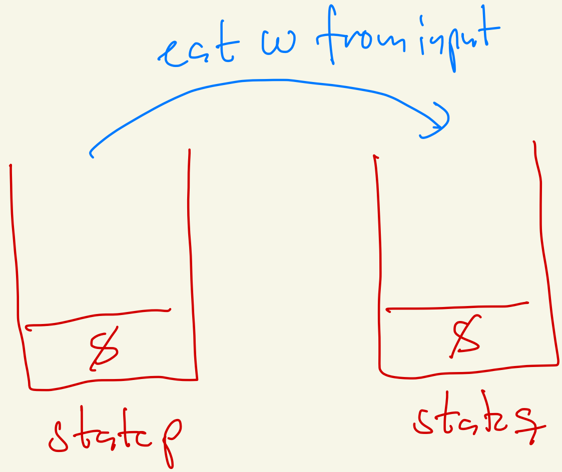
Resulting PDA M' satisfies

$L(M') = L(M)$

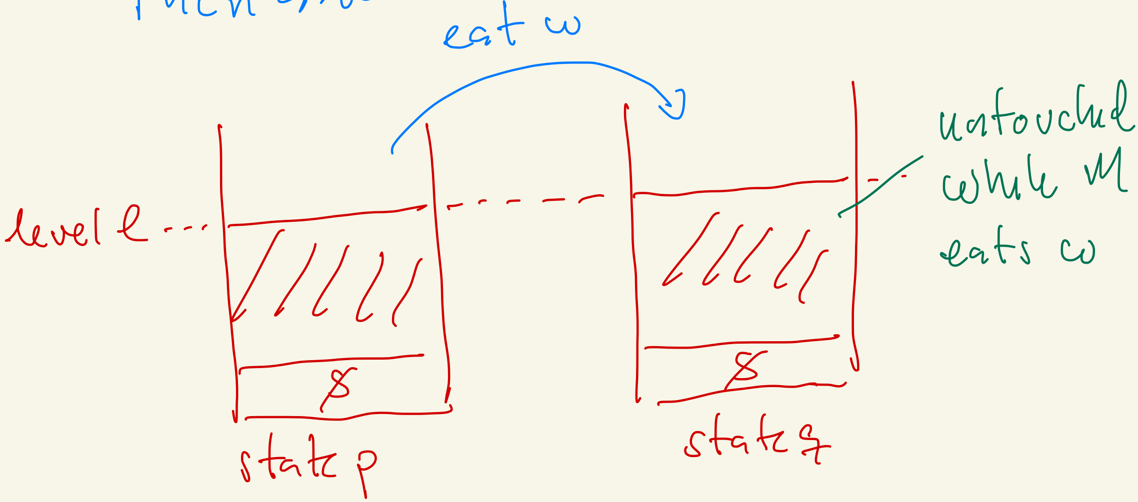
M' has <u>many</u> states but their number only depends on M.

Step 2: Understanding how a PDA M modifies its stack while processing a string w

If



Then also:

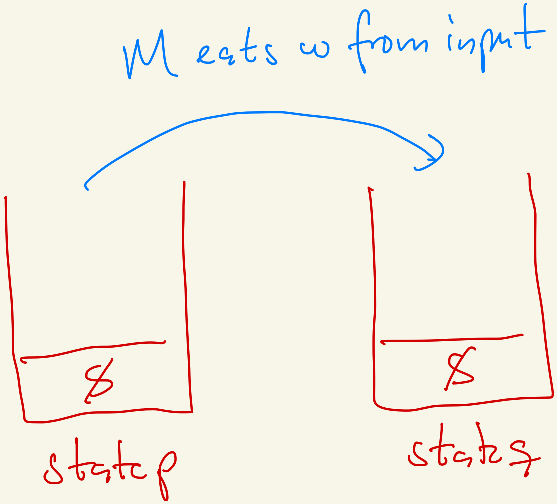


Step 3: The grammar G

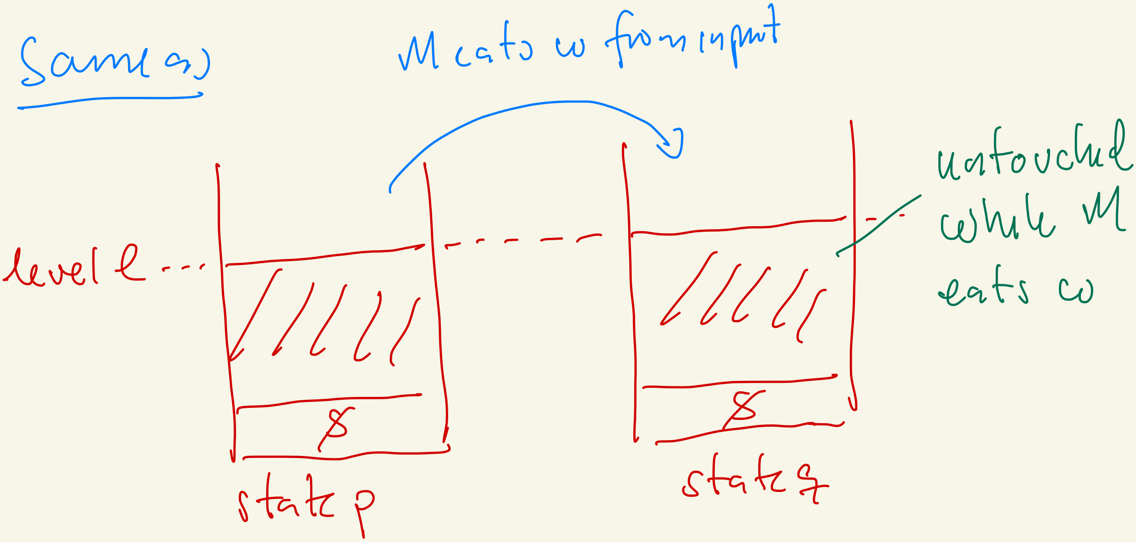
For all choices of states $p, q \in Q(M)$

G will contain a variable A_{pq}

A_{pq} will generate all strings $w \in \Sigma^*$ such that

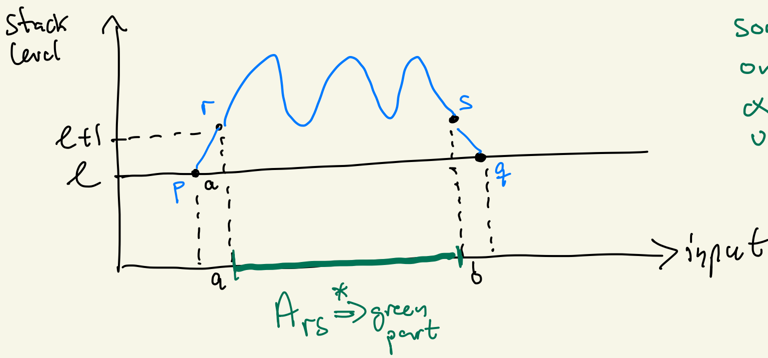


Same as



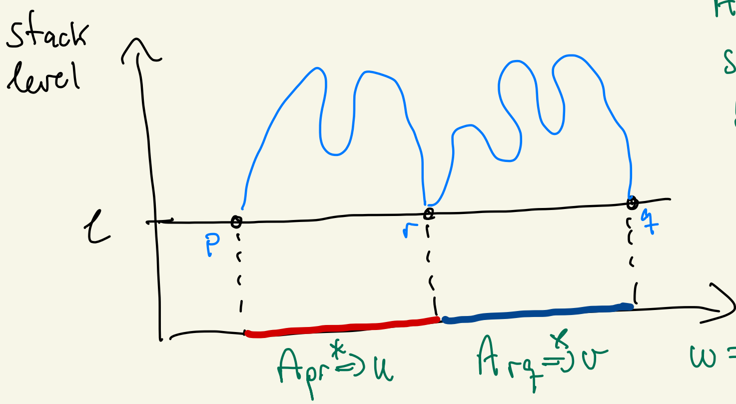
2 possible scenarios when M eats w and goes from state p with stack level ℓ to state q with stack level ℓ and never goes below level ℓ on stack

a) M's stack is at level ℓ initially and after reading w but above level ℓ in between



First step of M is the eat some $a \in \Sigma_\varepsilon$, push some $\alpha \in \Gamma$ on the stack and go to state r. α remains on the stack until the last step when M goes from state s to state q while eating $b \in \Sigma_\varepsilon$ and popping α

b) After reading a proper prefix u of w, M is again down to level ℓ on its stack



Again M starts by pushing some $\alpha \in \Gamma$ on the stack, but this time α is popped again before we reach the state just before q (and w is read)

$w = uv$

Definition of $G = (V, \Sigma, R, S)$

$$V = \{ A_{pq} \mid p, q \in Q(M) \}$$

$$S = A_{q_0 q_{accept}}$$

Rules:

- $\forall p, q, r, s \in Q(M)\ \forall t \in \Gamma\ \forall a, b \in \Sigma_\varepsilon$:

  If $p \xrightarrow{a,\, \varepsilon \to t} r$ and $s \xrightarrow{b,\, t \to \varepsilon} q$

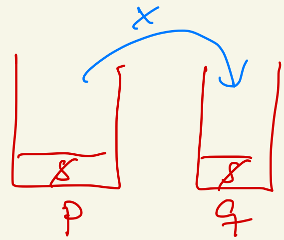
  are transitions of M, then add

  $A_{pq} \to a A_{rs} b$ to R

- $\forall p, q, r \in Q(M)$ add $A_{pq} \to A_{pr} A_{rq}$ to R

- $\forall p \in Q$ add $A_{pp} \to \varepsilon$ to R

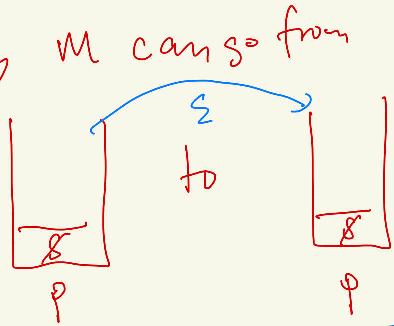
Claim 2.30 $A_{pq} \overset{*}{\Rightarrow} x \Rightarrow$



Proof induction over # steps in the derivation

1. step : Then $p = q$ and $A_{pp} \to \varepsilon$ so $x \in \Sigma^*$

clearly M can go from



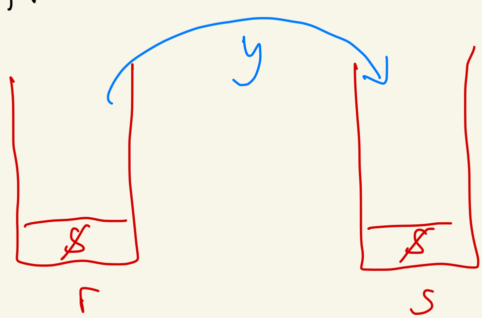
without reading anything

$\boxed{\text{hypothesis : claim true when} \leq k \text{ steps in the derivation}}$

Consider a derivation with k+1 steps and look at the first rule used in the derivation

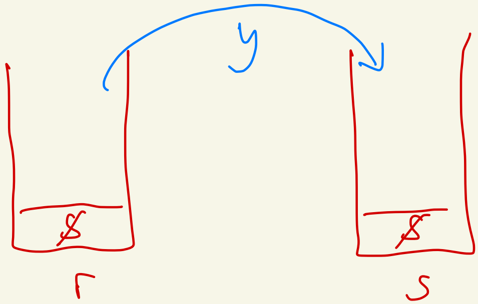
If $A_{pq} \to a A_{rs} b$ then $x = ayb$ when $A_{rs} \overset{*}{\Rightarrow} y$

last derivation has k steps, so by induction

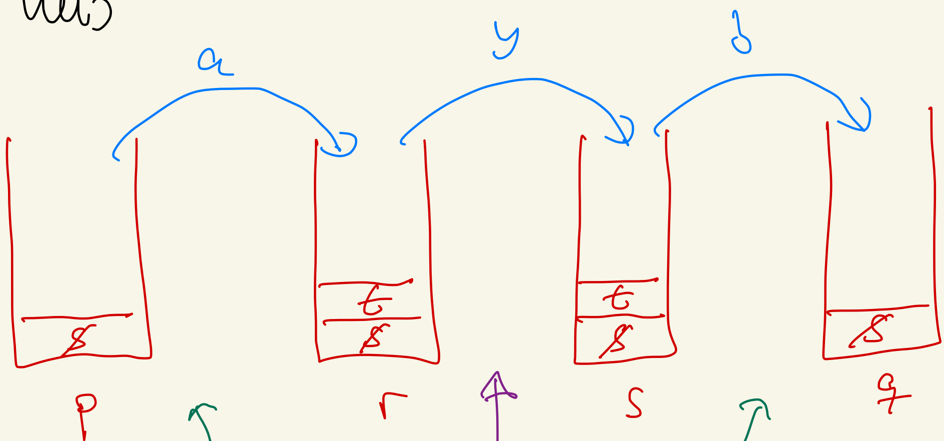


If $A_{pq} \to a A_{rs} b$ then $x = ayb$ where $A_{rs} \overset{*}{\Rightarrow} y$

last derivation has k steps, so by induction



Then



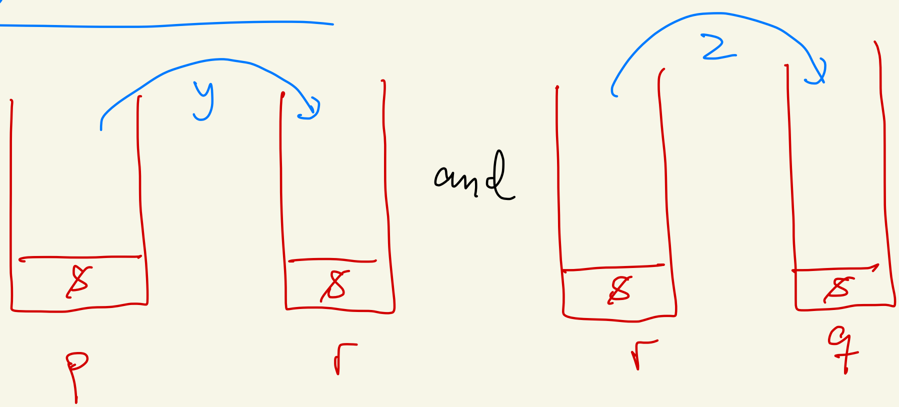
by induction and step 2

follow from the fact that $A_{pq} \to a A_{rs} b$ was added to R

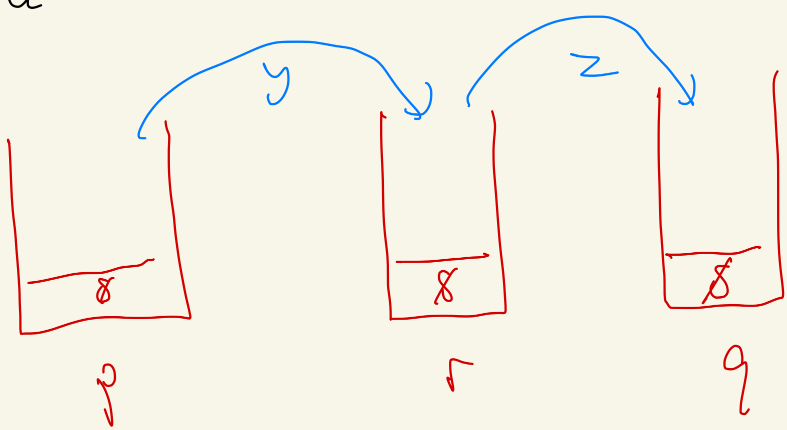
If $A_{pq} \to A_{pr} A_{rq}$ then

$x = yz$ when $\left.\begin{array}{l} A_{pr} \overset{*}{\Rightarrow} y \\ A_{rq} \overset{*}{\Rightarrow} z \end{array}\right\}$ in $\leq k$ steps

by induction:

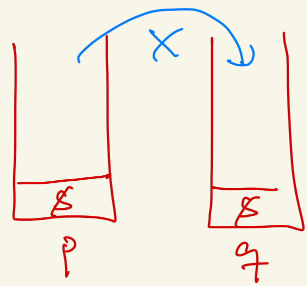


Hence



Claim 2.31

(□)



$\Rightarrow A_{pq} \overset{*}{\Rightarrow} x$

Proof by induction over # of steps in M's computation

0 steps: M cannot read anything in 0 steps

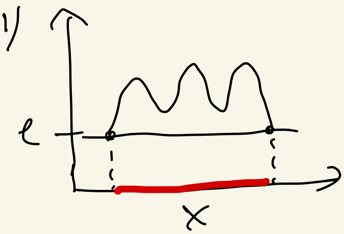
so $p = q$ and $x = \varepsilon$

By definition of G $A_{pp} \to \varepsilon$ so base case ok

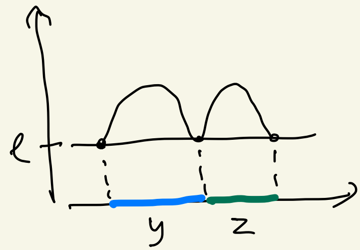
Assume (□) holds if M takes $\leq k$ steps

Consider a computation with $k+1$ steps

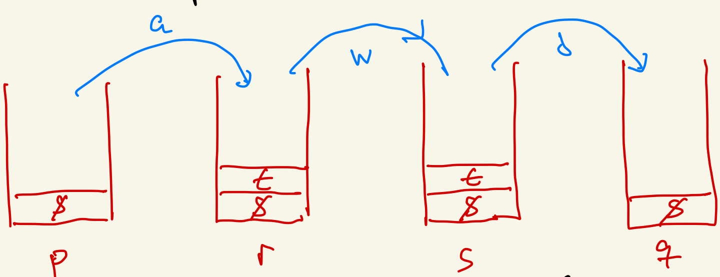
2 cases 1)



2)



1) M computes like this



so $x = awb$ and

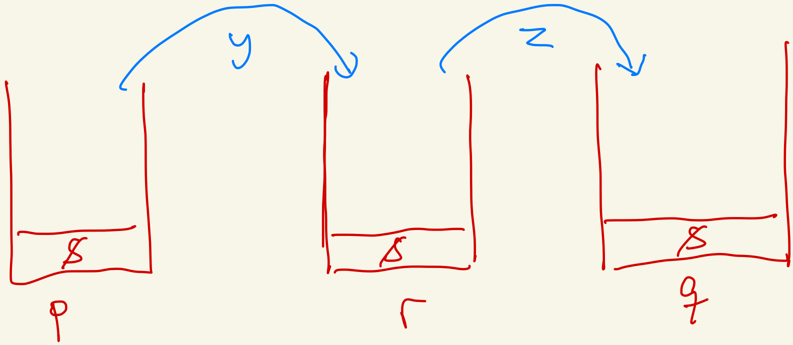
$p \xrightarrow{a,\varepsilon \to t} r \qquad s \xrightarrow{b,t \to \varepsilon} q$

so $A_{pq} \to a A_{rs} b \in R$

Hence $A_{pq} \to a A_{rs} b \underset{\text{induction}}{\overset{*}{\Rightarrow}} awb = x$

2)

M's computation is of the form



By induction we have

$$A_{pr} \overset{*}{\Rightarrow} y \quad \text{and} \quad A_{rq} \overset{*}{\Rightarrow} z$$

So

$$A_{pq} \to A_{pr} A_{rq} \overset{*}{\Rightarrow} y A_{rq} \overset{*}{\Rightarrow} yz = x$$

Claims 2.30 + 2.31 imply that

$$S = A_{q_0 q_{accept}} \overset{*}{\Rightarrow} w \iff M \text{ accepts } w$$