Context-Free

=

PDA-Recognizable

Recall: We drew PDAs for some context-free languages, and proved that they accepted exactly those specific sets

Today: PDAs are the machine model for context-free grammars.

For regular languages, we presented

- DFAs
- NFAs
- Regular expressions

and showed that the languages accepted by finite automata are exactly those that can be expressed during regexes.

We then showed that some languages are not regular.
Some of these can be generated using a context-free grammar.
Some of those are recognizable using a pushdown automaton.

Claim: The set of languages recognizable by pushdown automata is exactly the set of languages generatable by context-free grammars.

How do we show this?

① Given a CFG generating a language L, construct a PDA which recognizes L.

② Given a PDA recognizing L, construct a CFG which generates L.

Suppose we start with ①. Suppose we are given $G = (NT, T, R, S)$.

Do we have a general idea of the "shape" of the rules in R?

Without this, hard to "uniformly" translate G into a PDA!

Greibach Normal Form:

A CFG $G = (NT, T, R, S)$ is in Greibach Normal form if all production rules in R are of the form

$A \to aB_1 \cdots B_k$, where $k \geq 0$, $A, B_1, \ldots, B_k \in NT$, and $a \in T$.

No rule for generating the empty string ε!

Thm: For any CFG G, there is a CFG G' in Greibach Normal form s.t. $\mathcal{L}(G') = \mathcal{L}(G) \setminus \{\varepsilon\}$.

So, every CFG can be converted to a form where each rule is of the form $A \to aB_1 \cdots B_k$, where $k \geq 0$, $A, B_1, \ldots, B_k \in NT$, and $a \in T$, or of the form $S \to \varepsilon$.

① Suppose we are given such a CFG $G = (NT, T, R, S)$.

We construct a PDA M which accepts $L(G)$ by empty stack.

$M = (Q, \Sigma, \Gamma, \Delta, q_0, \phi)$, where

$Q = \{q_0, q_1\}$, $\Sigma = T$, $\Gamma = NT \cup \{\perp\}$ and

$(q_0, \omega, \perp) \xrightarrow{*} (q, \varepsilon, \perp)$ iff $\omega \in L(G)$

$$\Delta = \{((q_0, \varepsilon, \varepsilon), (q_1, S)),$$

$$\{((q_1, a, A), (q_1, B_1 \cdots B_k)) \mid A \to aB_1 \cdots B_k \in R\},$$

$$\{(q_1, \varepsilon, S), (q_1, \varepsilon)) \mid S \to \varepsilon \in R\}$$

Thm: If a terminal string x is obtained by applying rules in R to the leftmost nonterminal symbol, then M accepts x.

Thm: For any $x, y \in \Sigma^*$, $\gamma \in NT^*$, and $C \in NT$, one can generate $x\gamma$ by n applications of rules in R to C iff $(q_1, xy, C1) \xrightarrow[M]{n} (q_1, y, \gamma 1)$. in leftmost-first order

Proof: By induction on n.

$n = 0$: $C \longrightarrow x\gamma$ in 0 applications of any rule in R
iff $C = x\gamma$ iff $x = \varepsilon$ and $\gamma = C$
iff $(q_1, xy, C1) = (q_1, \varepsilon y, \gamma 1) = (q_1, y, \gamma 1) \xrightarrow[M]{0} (q_1, y, \gamma 1)$.

$n = m+1$: Suppose $C \longrightarrow x\gamma$ in $m+1$ applications of rules from R.
Consider the $m+1^{th}$ rule applied. It must be of the form
$D \longrightarrow c\beta$, where $c \in Tv\{\varepsilon\}$, $D \in NT$, and $\beta \in NT^*$.

Then, there is some $z \in T^*$, and $\alpha \in NT^*$ s.t. $C \xrightarrow{m} zD\alpha$.

Then, $zD\alpha \xrightarrow{1} zc\beta\alpha = x\gamma$. So, $x = zc$, and $\gamma = \beta\alpha$.

By IH, $(q_1, zcy, C1) \xrightarrow[M]{m} (q_f, cy, D\alpha 1)$ — (a)

We map each rule in R to a transition in Δ, so $((q_1, c, D), (q_1, \beta)) \in \Delta$, since the $m+1^{th}$ rule was $D \to c\beta$

So, $(q_1, cy, D\alpha 1) \xrightarrow[M]{1} (q_1, y, \beta\alpha 1)$ — (b)

($\beta\alpha = \gamma$)

Combining (a) and (b),

$(q_1, zcy, C1) \xrightarrow[M]{m+1} (q_1, y, \gamma 1)$

Now, suppose $(q_1, xy, C1) \xrightarrow[M]{m+1} (q_1, y, \gamma 1)$.

Let $((q_1, c, D), (q_1, \beta)) \in \Delta$ be the final transition taken.

Then, $x = zc$ for some $z \in \Sigma^*$, $\gamma = \beta\alpha$ for some $\alpha \in \Gamma^*$, and

$(q_1, zcy, C1) \xrightarrow[M]{m} (q_1, cy, D\alpha 1) \xrightarrow[M]{1} (q_1, y, \beta\alpha 1)$.

By IH, one can generate $zD\alpha$ by m applications of rules to C.

By the definition of Δ, $D \to c\beta \in R$.

So, $C \xrightarrow{m} zD\alpha \xrightarrow{1} zc\beta\alpha = x\gamma$.

So $x\gamma$ can be generated from C by m+1 applications of rules.

Thm: $\mathcal{L}(M) = \mathcal{L}(G)$.

Proof: $x \in \mathcal{L}(G)$

iff $S \xrightarrow{*} x$ via some sequence of "leftmost" applications of rules in R

iff $(q_0, x, \perp) \xrightarrow[M]{*} (q, \varepsilon, \perp)$ for some $q \in Q$

iff $(q_0, x, \perp) \xrightarrow[M]{1} (q_1, x, S\perp) \xrightarrow[M]{*} (q_1, \varepsilon, \perp)$

iff $x \in \mathcal{L}(M)$.

earlier theorem

We now look at the other direction.

② Given a PDA recognizing $\mathcal{L}$, construct a CFG which generates $\mathcal{L}$.

Suppose we are given $M = (Q, \Sigma, \Gamma, \Delta, q_0, \emptyset)$
which recognizes $\mathcal{L}$ by empty stack.

Recall that each transition in Δ is of the form $((q, a, \sigma), (q', \sigma'))$

We modify transitions so that each transition either only pushes a symbol onto the stack, or pops one off the stack. (Δ')

- Push+pop: split into multiple transitions
- Neither push nor pop: push arbitrary symbol, pop it off.

We also add a transition which takes M to state f if the input word is read and the stack is empty. (F')

$M' = (Q, \Sigma, \Gamma, \Delta', q_0, \{f\})$ Prove that $\mathcal{L}(M') = \mathcal{L}(M)$.

We now employ a strategy similar to how we constructed an equivalent regular expression given a DFA.

For each pair of states $p, q \in Q$, define $A_{pq} \in NT$.

A_{pq} should generate all strings which take M from state p with empty stack to state q with empty stack.

If M goes from p with empty stack to q with empty stack on some string x, what is the first move of M?

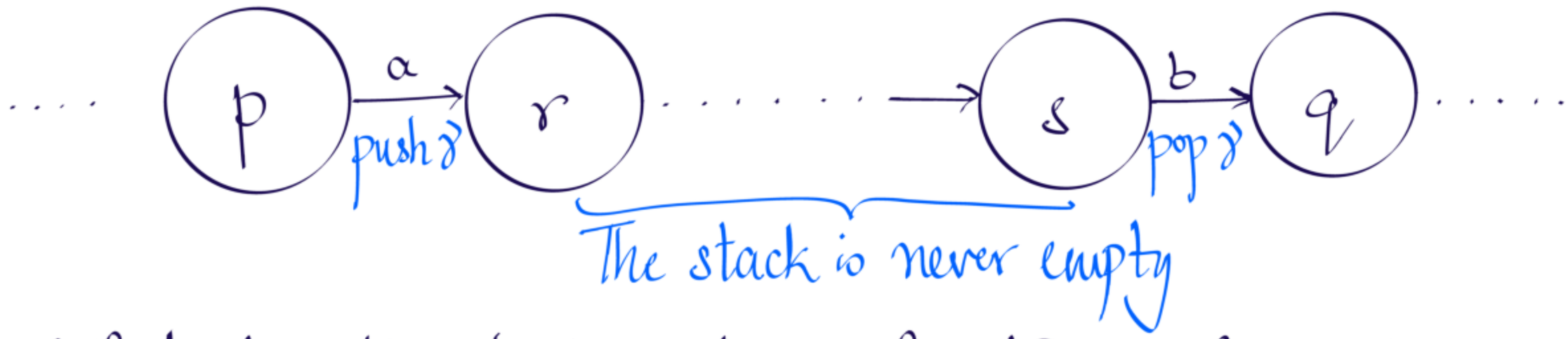
Has to be a push, since we cannot pop from an empty stack!

Similarly, what is the last move? A pop.

We will define the production rules for A_{pq} inductively.

Suppose $x = acwb$ for some w. Two possibilities arise

(a) The symbol that is initially pushed onto the stack, say γ, is only popped off the stack at the end of ω. Then, the operation of M looks as follows:



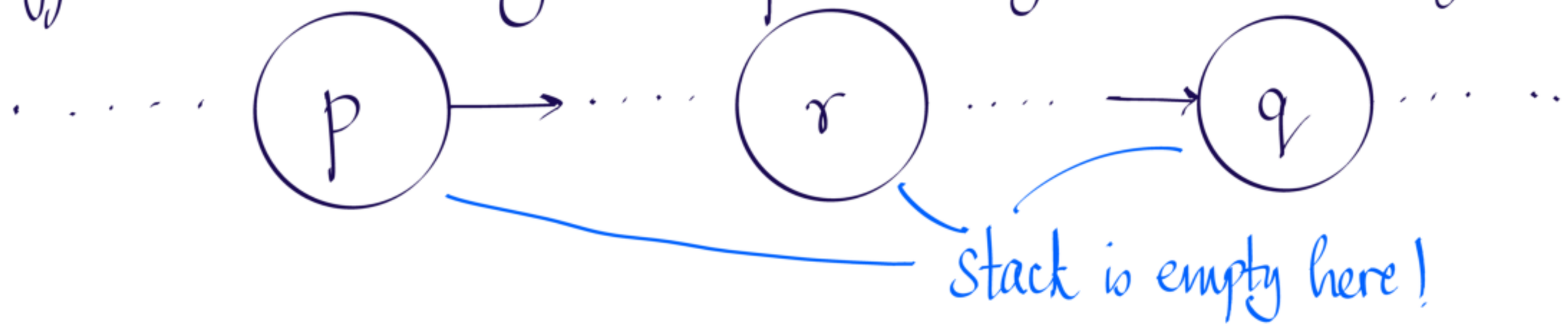
What does the stack contain when the machine enters r? γ

What does the stack contain when the machine exits s ? γ

Whatever else was pushed onto the stack between r and s is popped off by the end of that section of the run!

So, A_{pq} can be replaced by $aA_{rs}b$, i.e. $A_{pq} \rightarrow aA_{rs}b$.

(b) The symbol that is initially pushed onto the stack is popped off the stack midway. The operation of M looks as follows



$x = yz$ s.t. y takes M from (p, empty stack) to (r, empty stack), and z takes M from (r, empty stack) to (q, empty stack)

$A_{pq} \longrightarrow A_{pr} A_{rq}$

Recall: Given a PDA, construct a CFG generating the language of the PDA.

Modify the PDA so each transition either pushes or pops a symbol, and moves into a special accept state when it accepts with empty stack.

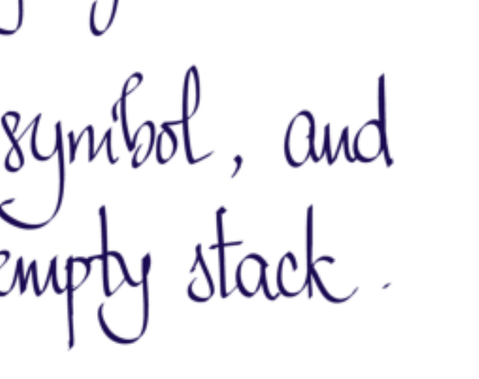
$$M = (Q, \Sigma, \Gamma, \Delta, q_0, \{f\})$$

$A_{pq} \in NT$: Generates all strings which take M from p with empty stack to q with empty stack

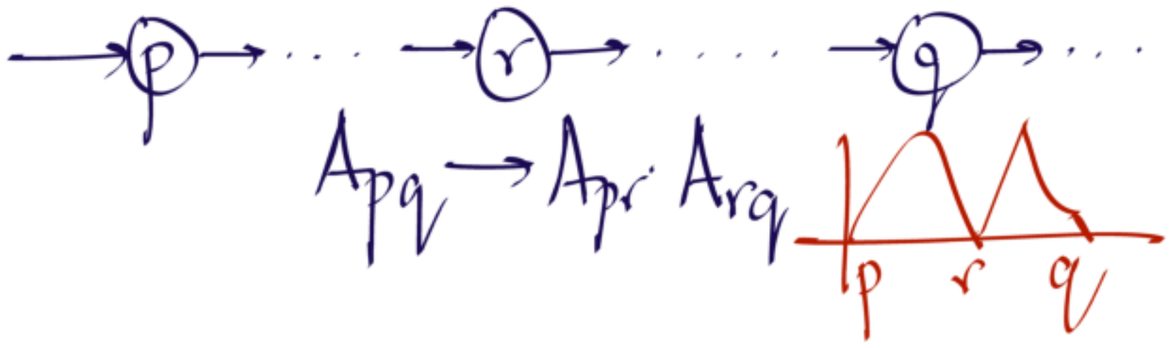
Inductive definition

① If stack is never empty except at p and q:
Symbol pushed on at p is popped off only just before q

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



② If stack is empty midway:



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

Construction: Given a PDA $M = (Q, \Sigma, \Gamma, \Delta, q_0, \{f\})$, we construct $G = (NT, T, R, S)$ as follows:

$$NT = \{A_{pq} \mid p, q \in Q\} \qquad T = \Sigma \qquad S = A_{q_0 f}$$

$$R = \{A_{pp} \to \varepsilon \mid p \in Q\} \cup \{A_{pq} \to A_{pr}A_{rq} \mid p, q, r \in Q\} \cup$$

$$\left\{ A_{pq} \to aA_{rs}b \;\middle|\; \begin{array}{l} p, q, r, s \in Q,\ a, b \in \Sigma \cup \{\varepsilon\},\ t \in \Gamma, \\ ((p, a, \varepsilon), (r, t)) \in \Delta, \\ ((s, b, t), (q, \varepsilon)) \in \Delta \end{array} \right\}$$

We now have to show our initial claim, that, for any $x \in \Sigma^*$,
A_{pq} generates x iff x takes M from p with empty stack
to q with empty stack. We prove each direction separately.

Claim: If A_{pq} generates x, then, x can take M from
p with empty stack to q with empty stack.

Proof: By induction on n, the number of applications of rules
from R in the derivation of x from A_{pq}.

$n = 1$: A derivation with only one rule applied which yields a
string in Σ^* has only one candidate in R, $A_{pp} \to \varepsilon$.
ε takes M from p to p without disturbing the stack.

$n = m+1$: Suppose A_{pq} generates x via $m+1$ applications of rules in R.

What is the first rule that can be applied?

Obviously not $A_{pp} \to \varepsilon$. Either $A_{pq} \to A_{pr}A_{rq}$ or $A_{pq} \to a A_{rs} b$.

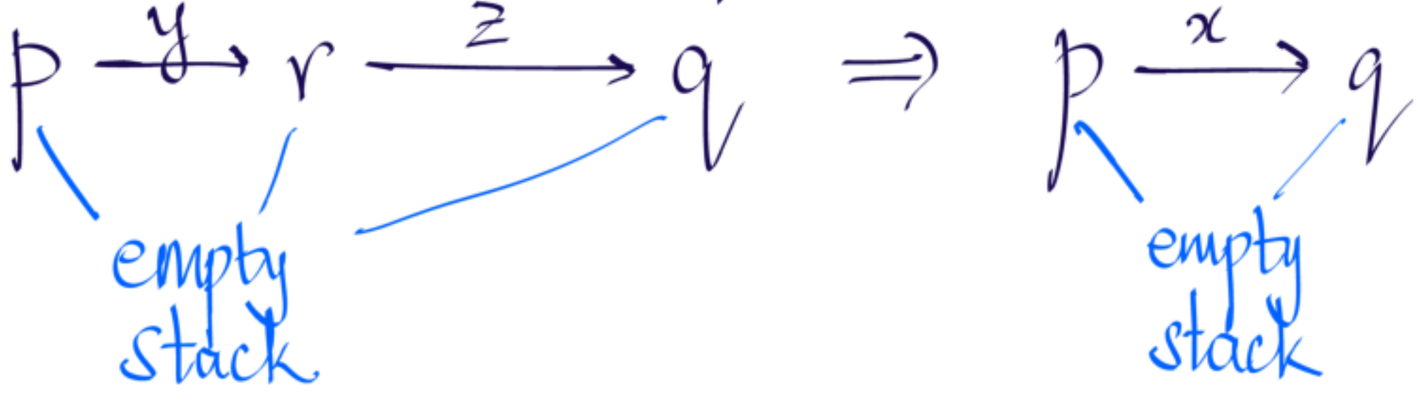
(a) Suppose the rule applied is of the form $A_{pq} \to A_{pr}A_{rq}$.

A_{pr} generates some $y \in \Sigma^*$, and A_{rq} generates some z s.t. $x = yz$.

By IH, y takes M from p to r and maintains empty stack at r

z takes M from r to q and maintains empty stack at q.

Suppose M is in state p, and the stack is empty.



(b) Suppose the rule applied is of the form $A_{pq} \to aA_{rs}b$.

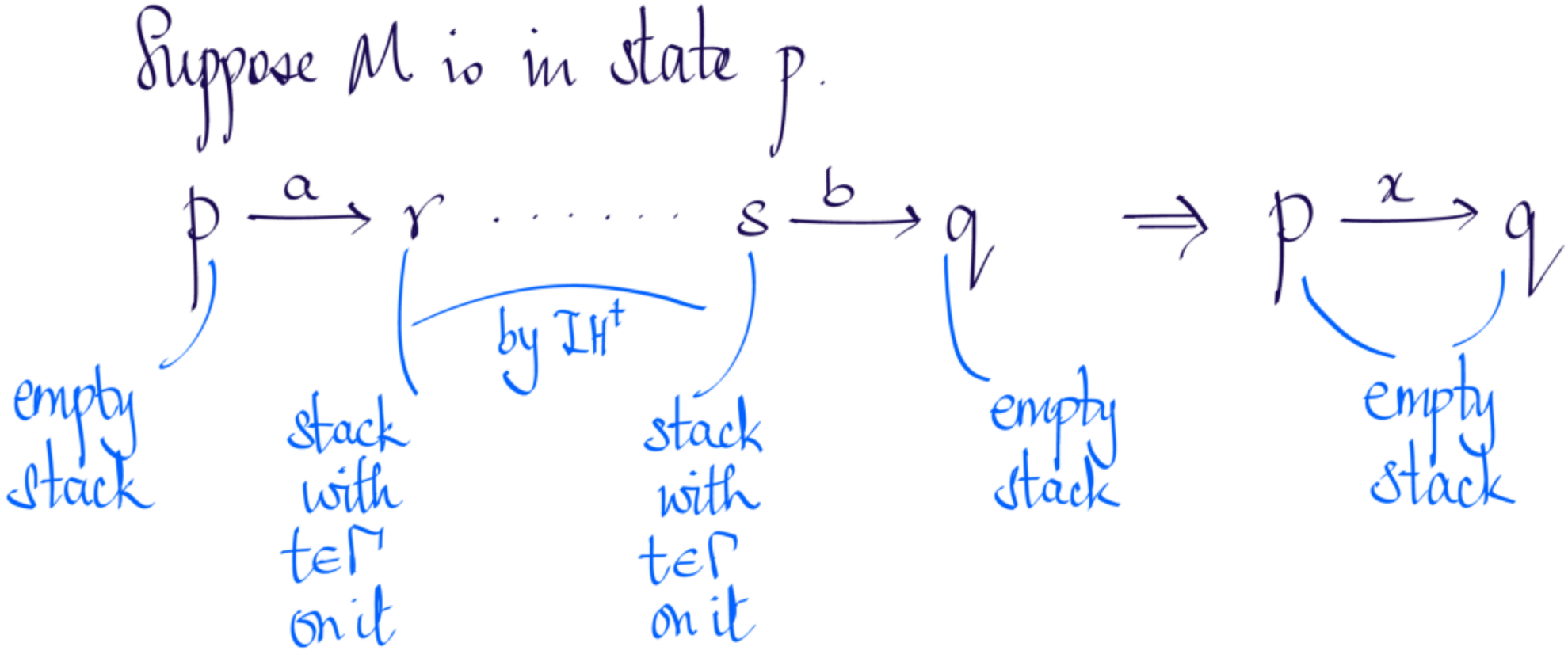
A_{rs} generates some y s.t. $x = ayb$.

By IH, y takes M from r to s with empty stack at both r and s.

Since $A_{pq} \to aA_{rs}b \in R$, we know that there is some $t \in \Gamma$ s.t.

$\{((p, a, \varepsilon), (r, t)), ((s, b, t), (q, \varepsilon))\} \subseteq \Delta$.

Suppose M is in state p.



□

Claim: If x takes M from p with empty stack to q with empty stack, then App generates x by applications of rules in R.

Proof: There is some $n > 0$ s.t. $(p, x, \bot) \xrightarrow{n}_M (q, \varepsilon, \bot)$.

Proof by induction on n.

$n = 0$: $(p, x, \bot) \xrightarrow{0}_M (p, x, \bot) = (q, \varepsilon, \bot)$

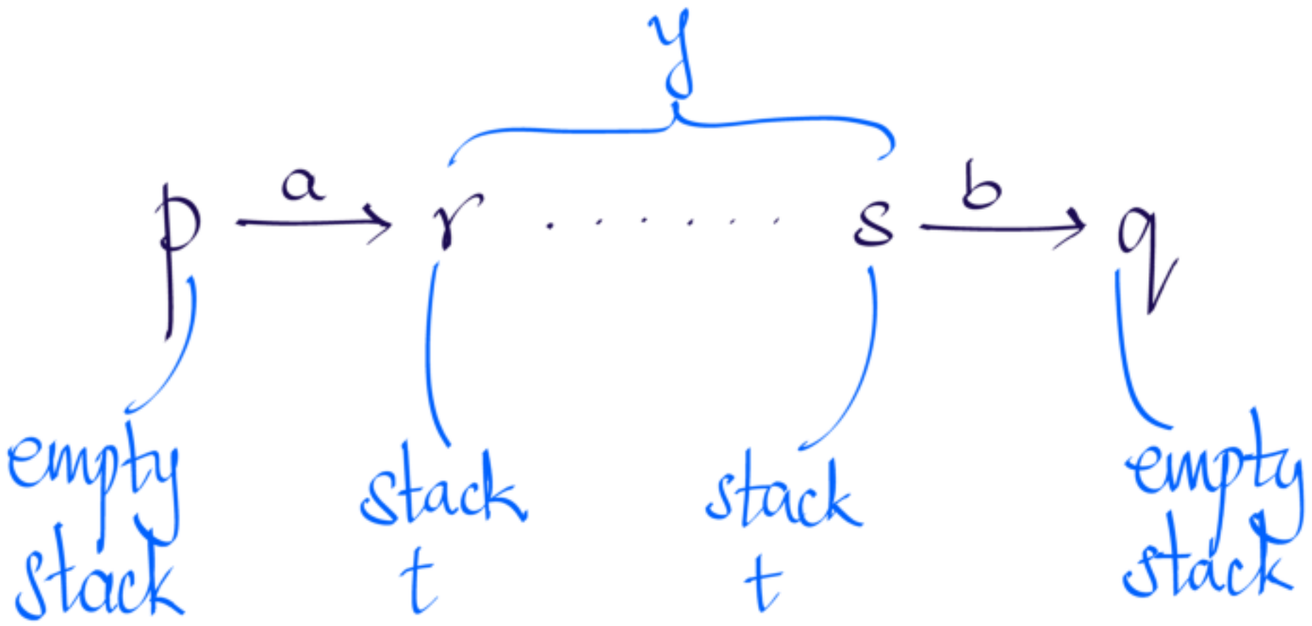
So $p = q$, and $x = \varepsilon$. $App \to \varepsilon$, so done.

$n = m + 1$: Two cases arise

(i) The stack is not empty anytime except at p and q:

The same symbol must be pushed and popped at the "ends"

Suppose we push a $t \in \Gamma$ to start with, and pop it off at the end.

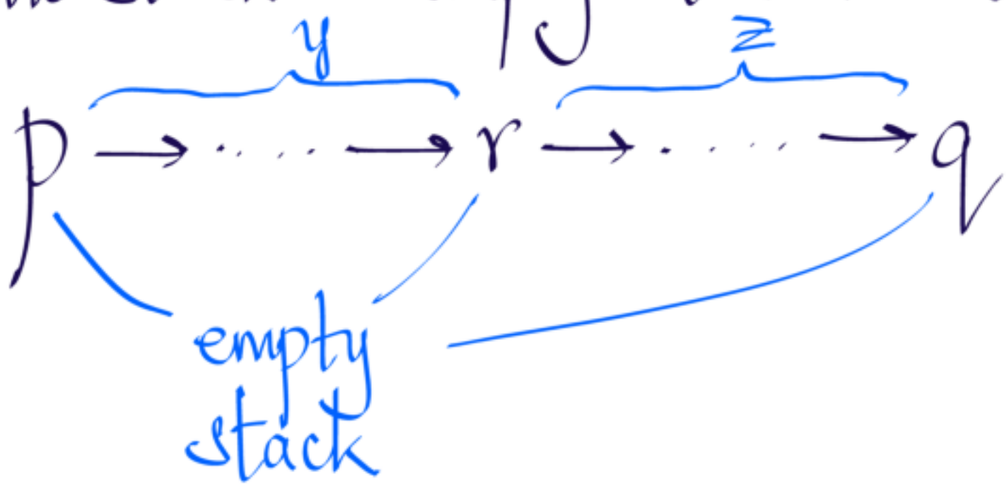


$x = ayb$, where

$A_{rs} \to y$ by IH

So $A_{pq} \to aA_{rs}b$.

(ii) The stack is empty at some midway point:



$x = yz$, where

$A_{pr} \to y$ } by IH

$A_{rq} \to z$ }

So $A_{pq} \to A_{pr}A_{rq}$