Computational Models - Lecture 5[1]

Handout Mode

Nachum Dershowitz & Yishay Mansour.

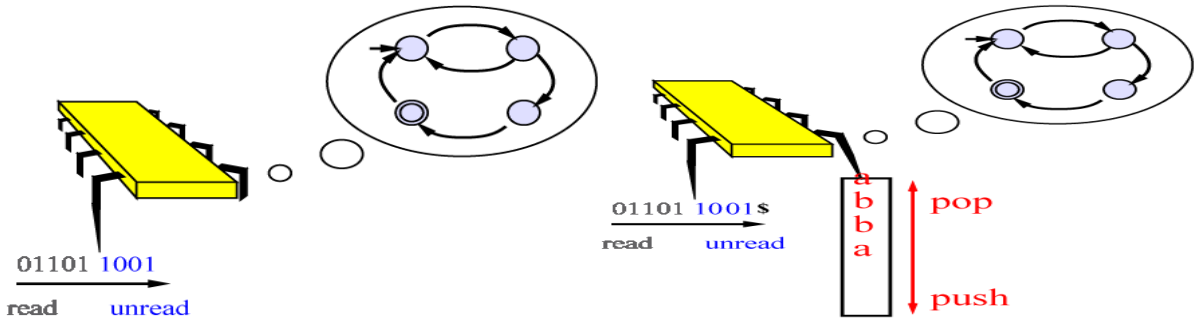
Tel Aviv University.

April 24–26, 2017

[1] Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Talk Outline

- Push Down Automata (PDA)
- Equivalence of CFGs and PDAs



- Sipser's book, 2.1, 2.2 & 2.3

- **Midterm: May 26th, 2017 !!!**

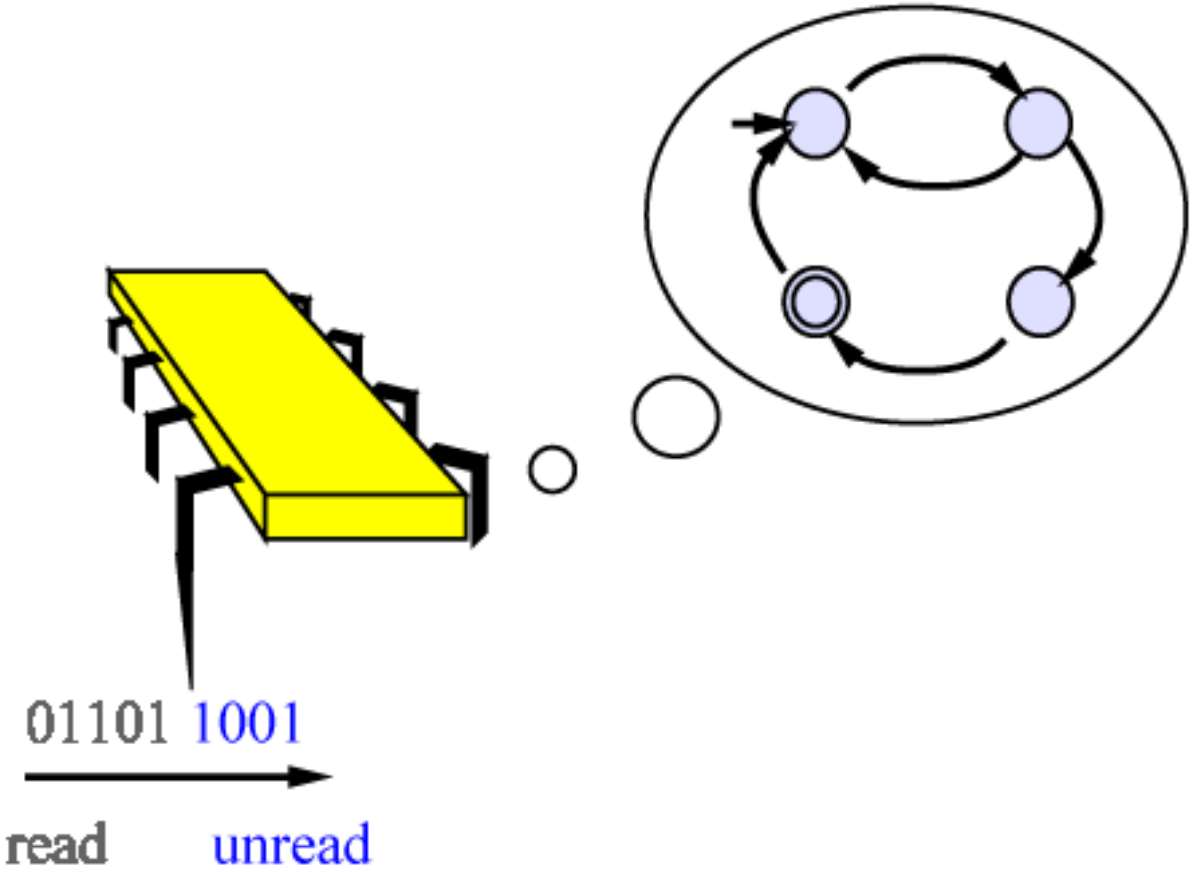
Part I

Push-Down Automata

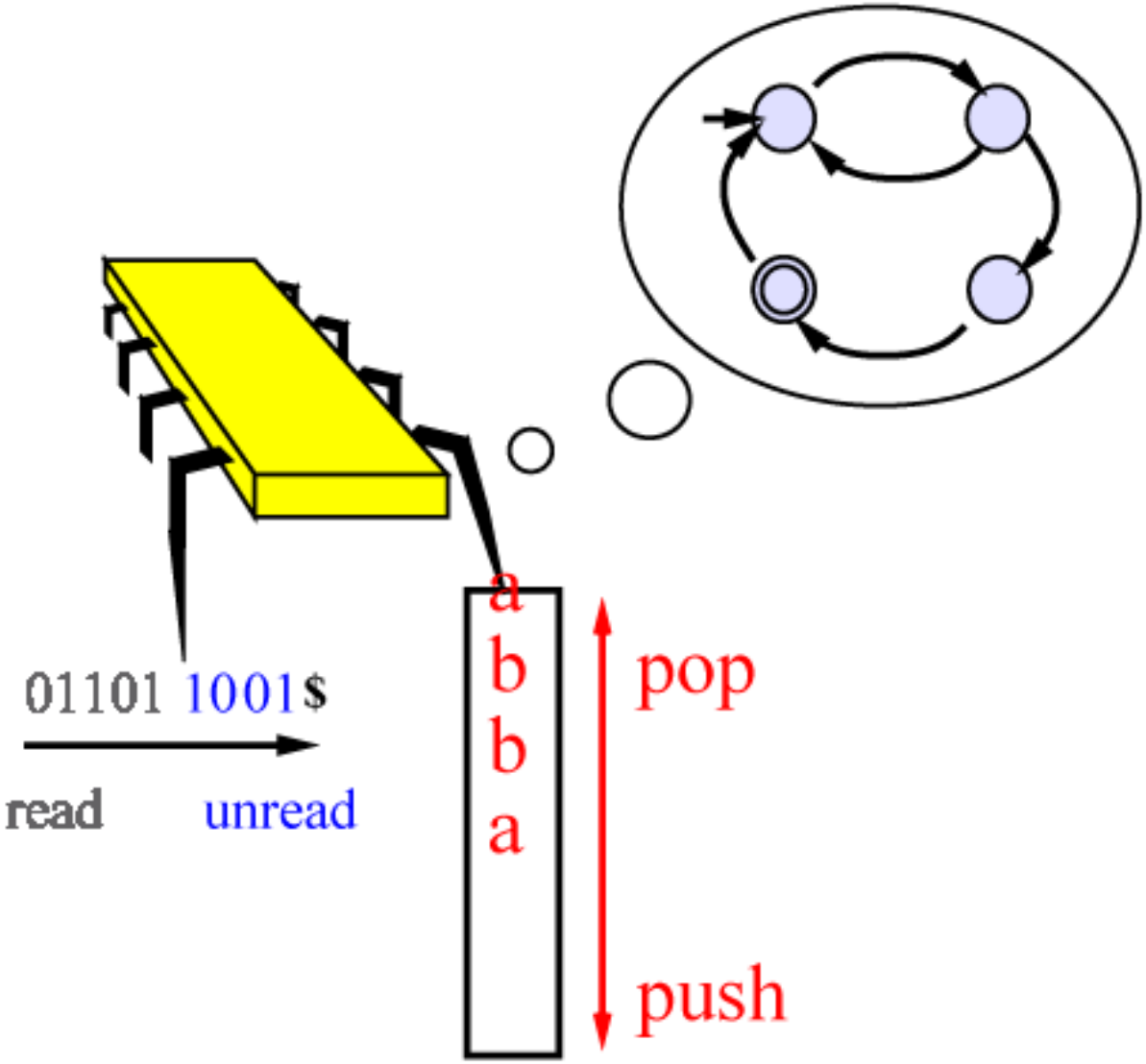
String generators and string acceptors

- Regular expressions are string generators – they tell us how to generate all strings in a language $\mathcal{L}$
- Finite automata (DFA, NFA) are string acceptors – they tell us if a specific string w is in $\mathcal{L}$
- CFGs are string generators
- Are there string acceptors for CFLs?
- YES! Push-down automata

Finite automaton



PushDown automaton



(ignore the '$' sign)

Example 1 — PDA for $\mathcal{L}_1 = \{0^n 1^n : n \geq 0\}$

Informally:

1. Read input symbols
   - 1.1 Push each read 0 on the stack
   - 1.2 Pop a 0 for each read 1
2. `Accept` if stack is empty after last symbol read, and no 0 appears after 1

Recall that $\mathcal{L}_1$ is not regular

Example 2 — PDA for $\mathcal{L}_2 = \{a^i b^j c^k : i = j \vee i = k\}$

Informally:

Read and push a's

Either pop and match with b's or pop and match with c's

A non-deterministic choice

Example 3 – Palindrome

A palindrome is a string w satisfying $w = w^{\mathcal{R}}$.

- "Madam I'm Adam"
- "Dennis and Edna sinned"
- "Red rum, sir, is murder"
- "Able was I ere I saw Elba"
- "In girum imus nocte et consumimur igni" (Latin: "we go into the circle by night, we are consumed by fire".)
- "$\nu\iota\psi o\nu\ \alpha\nu o\mu\eta\mu\alpha\tau\alpha\ \mu\eta\ \mu o\nu\alpha\nu\ o\psi\iota\nu$"
- Palindromes also appear in nature. For example as DNA restriction sites – short genomic strings over $\{A, C, T, G\}$, being cut by (naturally occurring) restriction enzymes.

What the difference from $\{ww^R\}$?

A PDA for Palindromes

Algorithm 1

Input: $x \in \Sigma^*$

1. Start pushing x into stack.
2. At some point, guess that the mid point of x has reached.
3. Pops and compares to input, letter by letter.
4. `Accept` If end of input occurs together with emptying of stack.

- This PDA accepts palindromes of even length over the alphabet (all lengths is an easy modification).
- Again, non-determinism (at which point to make the switch) seems necessary.

PDA – Formal definition

A PDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

- Q is a finite set called the states,
- Σ is a finite set called the input alphabet,
- Γ is a finite set called the stack alphabet,
- $\delta\colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \mapsto \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function,[2]
- $q_0 \in Q$ is the starting state, and
- $F \subseteq Q$ is the set of accepting states.

[2] $X_\varepsilon := X \cup \{\varepsilon\}$.

Formal model of computation

Recall that for $a \in (X_\varepsilon)^*$, we let $d(a) \in X^*$ be a without the ε symbols.

Definition 2 ($\widetilde{\delta}$)

For PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, define $\widetilde{\delta}_P \colon Q \times \Sigma_\varepsilon \times \Gamma^* \mapsto \mathcal{P}(Q \times \Gamma^*)$ by: $(q', s') \in \widetilde{\delta}_P(q, \sigma, s)$, if $\exists a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$ s.t.:

- $(q', b) \in \delta(q, \sigma, a)$
- $s = d(at)$ and $s' = d(bt)$

Namely, reading the symbol σ from the input, can change *configuration* (state and stack value) (q, s) into configuration (q', s').

It might be that

- $|s'| = |s|$: $a, b \neq \varepsilon$ or $a = b = \varepsilon$
- $|s'| > |s|$: $a = \varepsilon$ and $b \neq \varepsilon$
- $|s'| < |s|$: $a \neq \varepsilon$ and $b = \varepsilon$

As usual, we omit the subscript P when clear from the context.

Formal model of computation, cont.

Definition 3

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts $w \in \Sigma^*$, if exist $a_1, \ldots, a_k \in \Sigma_\varepsilon$, $r_0, \ldots, r_k \in Q$ and $s_0, \ldots, s_k \in \Gamma^*$, s.t.

- $w = d(a_1, \ldots, a_k)$.
- $r_0 = q_0$ and $r_k \in F$.
- $s_0 = \varepsilon$.
- $(r_{i+1}, s_{i+1}) \in \widetilde{\delta}_P(r_i, a_{i+1}, s_i)$, for all $0 \leq i < k$.

Definition 4

The language accepted by PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, denoted $\mathcal{L}(P)$, is the set of all strings $w \in \Sigma^*$ accepted by P.

An equivalent definition

Definition 5

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts $w \in \Sigma^*$, if $\widehat{\delta}_P(q_0, w, \varepsilon) \cap (F \times \Gamma^*) \neq \emptyset$

Definition 6 (ε-environment)

For PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ and $(q, s) \in Q \times \Gamma^*$, let
$E(q, s) = \{(q', s') \in Q \times \Gamma^* : \exists (q_1, s_1), \ldots, (q_k, s_k)$ s.t.
$(q_1, s_1) = (q, s) \wedge (q_k, s_k) = (q', s') \wedge \forall i \in [k-1]: \ (q_{i+1}, s_{i+1}) \in \widetilde{\delta}(q_i, \varepsilon, s_i)\}$.
$E(\mathcal{A} \subseteq Q \times \Gamma^*) = \bigcup_{(q,s) \in \mathcal{A}} E(q, s)$

Definition 7 ($\widehat{\delta}$)

For PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, define $\widehat{\delta}_P : Q \times \Sigma^* \times \Gamma^* \mapsto \mathcal{P}(Q \times \Gamma^*)$ by:

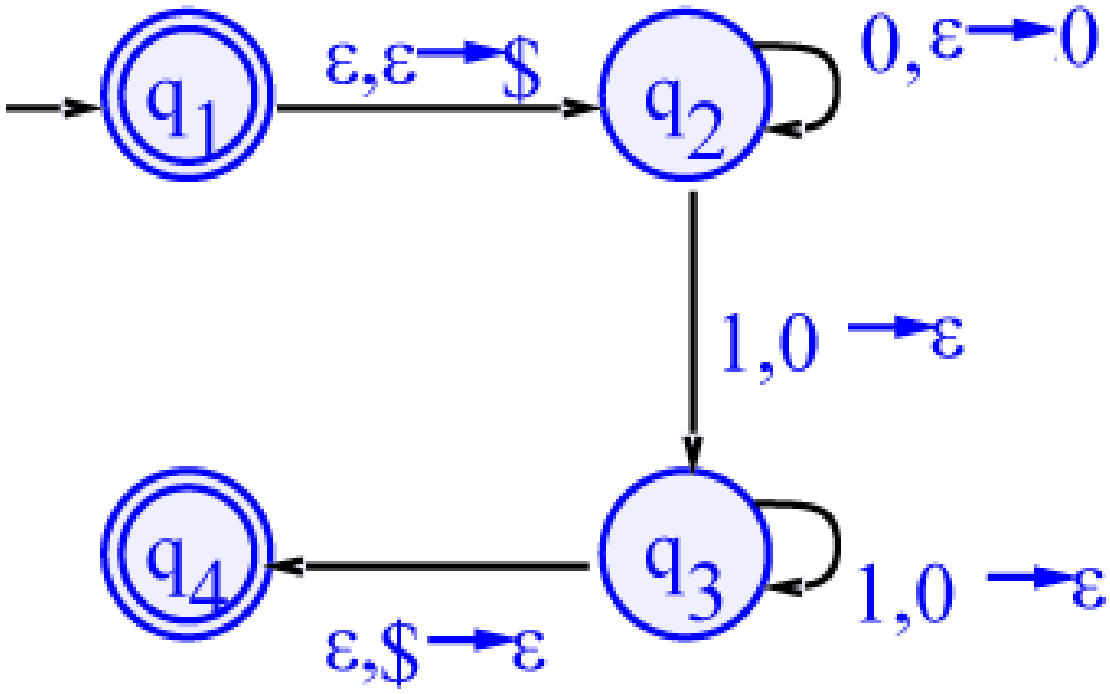
$$\widehat{\delta}_P(q, w, s) = \begin{cases} E(q, s), & w = \varepsilon, \\ E\left(\bigcup_{(q', s') \in \widehat{\delta}_P(q, w_1, \ldots, n-1, s)} \widetilde{\delta}_P(q', w_n, s')\right), & n = |w| \geq 1. \end{cases}$$

Diagram notation

When drawing the automata diagram, we use the following notation

- Transition $a, b \to c$ from state q to q' means $(q', c) \in \delta(q, a, b)$, and informally means the automata
  - read a from input
  - pop b from stack
  - push c onto stack
- Meaning of ε transitions (informally):
  - $a = \varepsilon$: don't read input
  - $b = \varepsilon$: don't pop any symbol
  - $c = \varepsilon$: don't push any symbol

A PDA for $\mathcal{L}_1 = \{0^n 1^n : n \geq 0\}$



Claim 8

$0011 \in L(P)$.

Proof: take

	$w'_1 = \varepsilon$	$w'_2 = 0$	$w'_3 = 0$	$w'_4 = 1$	$w'_5 = 1$	$w'_6 = \varepsilon$
$s_0 = \varepsilon$	$s_1 = \$$	$s_2 = 0\$$	$s_3 = 00\$$	$s_4 = 0\$$	$s_5 = \$$	$s_6 = \varepsilon$
$r_0 = q_1$	$r_1 = q_2$	$r_2 = q_2$	$r_3 = q_2$	$r_4 = q_3$	$r_5 = q_3$	$r_6 = q_4$

A PDA for $\mathcal{L}_1 = \{0^n 1^n : n \geq 0\}$

We want to show that $L(P) = \mathcal{L}_1 = \{0^n 1^n : n \geq 0\}$
What do we need to prove?

Claim 9

- $\widehat{\delta}(q_1, \varepsilon, \varepsilon) = \{(q_1, \varepsilon), (q_2, \$)\}$.
- $\widehat{\delta}(q_1, 0^k, \varepsilon) = \{(q_2, 0^k\$)\}$, for $k \geq 1$.
- $\widehat{\delta}(q_1, 0^k 1^i, \varepsilon) = \{(q_3, 0^{k-i}\$)\}$, for $k > i \geq 1$.
- $\widehat{\delta}(q_1, 0^k 1^k, \varepsilon) = \{(q_3, \$), (q_4, \varepsilon)\}$, for $k \geq 1$.
- $\widehat{\delta}(q_1, w, \varepsilon) = \emptyset$, for $w \notin \{0^k 1^i | k \geq i \geq 0\}$.

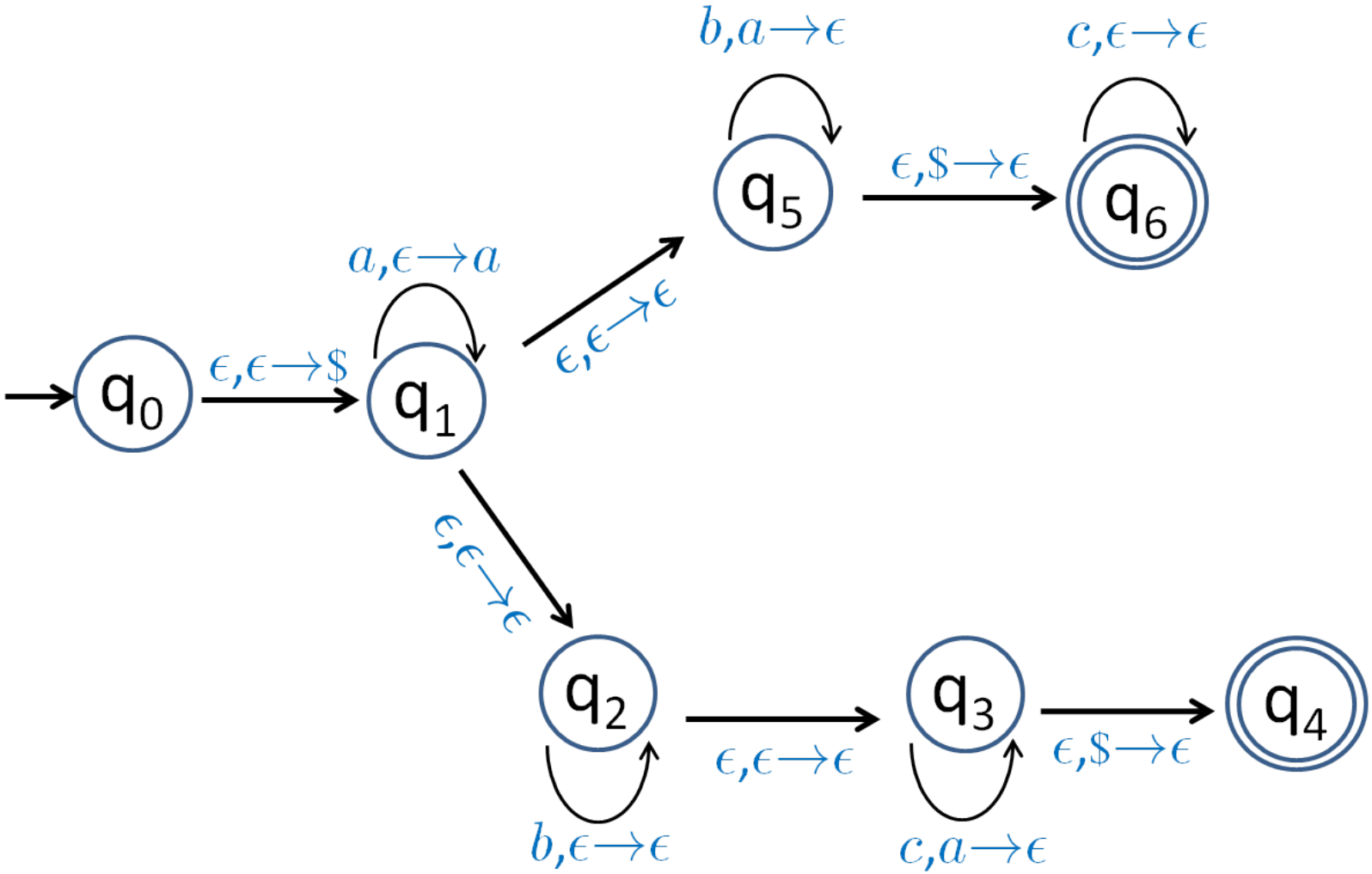
Knowing when stack is empty

It is convenient to be able to know when the stack is empty, but there is no built-in mechanism to do that.

Solution

1. Start by pushing $ onto stack.
2. When you see it again, stack is empty.

A PDA for $\mathcal{L}_2 = \{a^i b^j c^k : i = j \vee i = k\}$

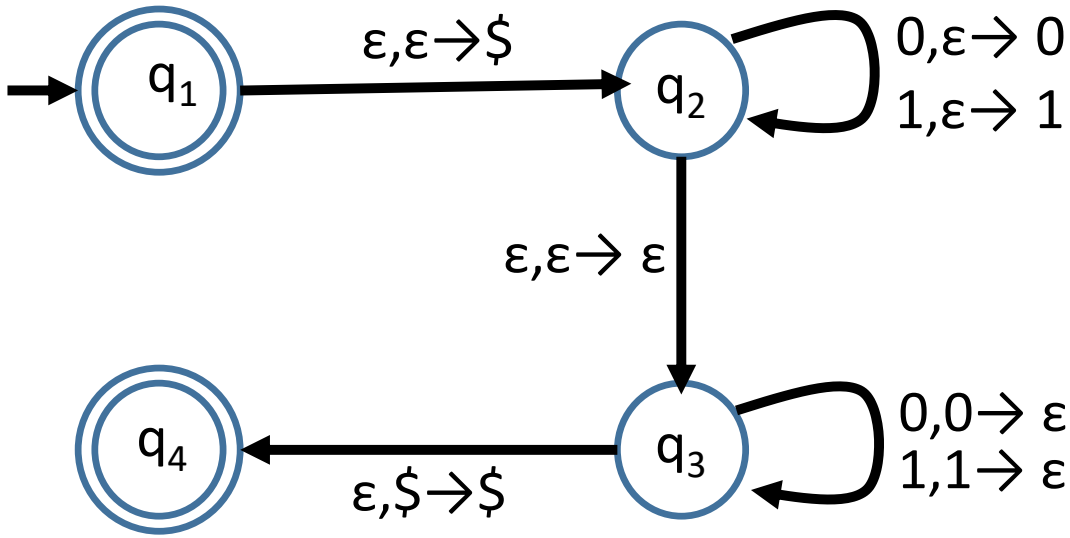


A PDA for $\mathcal{L}_2 = \{a^i b^j c^k : i = j \vee i = k\}$, cont.

- Non-determinism is essential here!
- Unlike finite automata, non-determinism does add power.
- But we saw deterministic algorithm to deicide any CFL (and as we see later, CFLs are exactly the languages decided by PDAs)!
- How to prove that non-determinism adds power?

- Does not seem trivial or immediate.
- Another example: $\mathcal{L} = \{x^n y^n : n \geq 0\} \cup \{x^n y^{2n} : n \geq 0\}$ is accepted by a non-deterministic PDA, but not by a deterministic one. (Proof? Book!)

A PDA for $\mathcal{L}_3 = \{ww^R : w \in \{0,1\}\}$



PDA languages

The Push-Down Automata Languages, $\mathcal{L}_{\text{PDA}}$, is the set of all languages that can be described by some PDA:

- $\mathcal{L}_{\text{PDA}} = \{\mathcal{L}(M) \colon M \text{ is a PDA}\}$

It is immediate that $\mathcal{L}_{\text{PDA}} \supsetneq \mathcal{L}_{\text{DFA}}$: every DFA is just a PDA that ignores the stack.

- $\mathcal{L}_{\text{CFG}} \subseteq \mathcal{L}_{\text{PDA}}$?
- $\mathcal{L}_{\text{PDA}} \subseteq \mathcal{L}_{\text{CFG}}$?
- $\mathcal{L}_{\text{PDA}} = \mathcal{L}_{\text{CFG}}$!!!

Proof in last hour of next class.

Comparing PDA and Finite Automata

- PDA may be deterministic or non-deterministic.
- Unlike finite automata, non-determinism adds power: There are some languages accepted only by non-deterministic PDAs.

Transition function δ looks different than DFA or NFA cases, reflecting stack functionality.

Part II

Equivalence Theorem

The CFG–PDA equivalence theorem

Theorem 10

$\mathcal{L}_{\text{PDA}} = \mathcal{L}_{\text{CFG}}$.

(i.e., A language is context free if and only if some pushdown automata accepts it.)

This time (unlike the regular expression vs. regular languages theorem), both the proof "if" part and of the "only if" part are non trivial.

Proof sketch follows.

CFL $\Longrightarrow$ PDA

Lemma 11

$\mathcal{L}_{\text{CFG}} \subseteq \mathcal{L}_{\text{PDA}}$: any CFL has a PDA that accepts it.

- Let $\mathcal{L}$ be a CFL, and let $G = (V, \Sigma, R, S)$ be a CFG for $\mathcal{L}$
- We build a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, such that on input w it "figures out" if there is a derivation of w using G.

Question 12

How does P figure out which substitution to make?

Answer: It guesses.

Simplifying assumptions

1. In a single move, a PDA can push a whole word (from some fixed set) into the stack (first letter at the top)

   Can we justify it?

2. When deriving a word from a CFL, we always substitute the left most variable

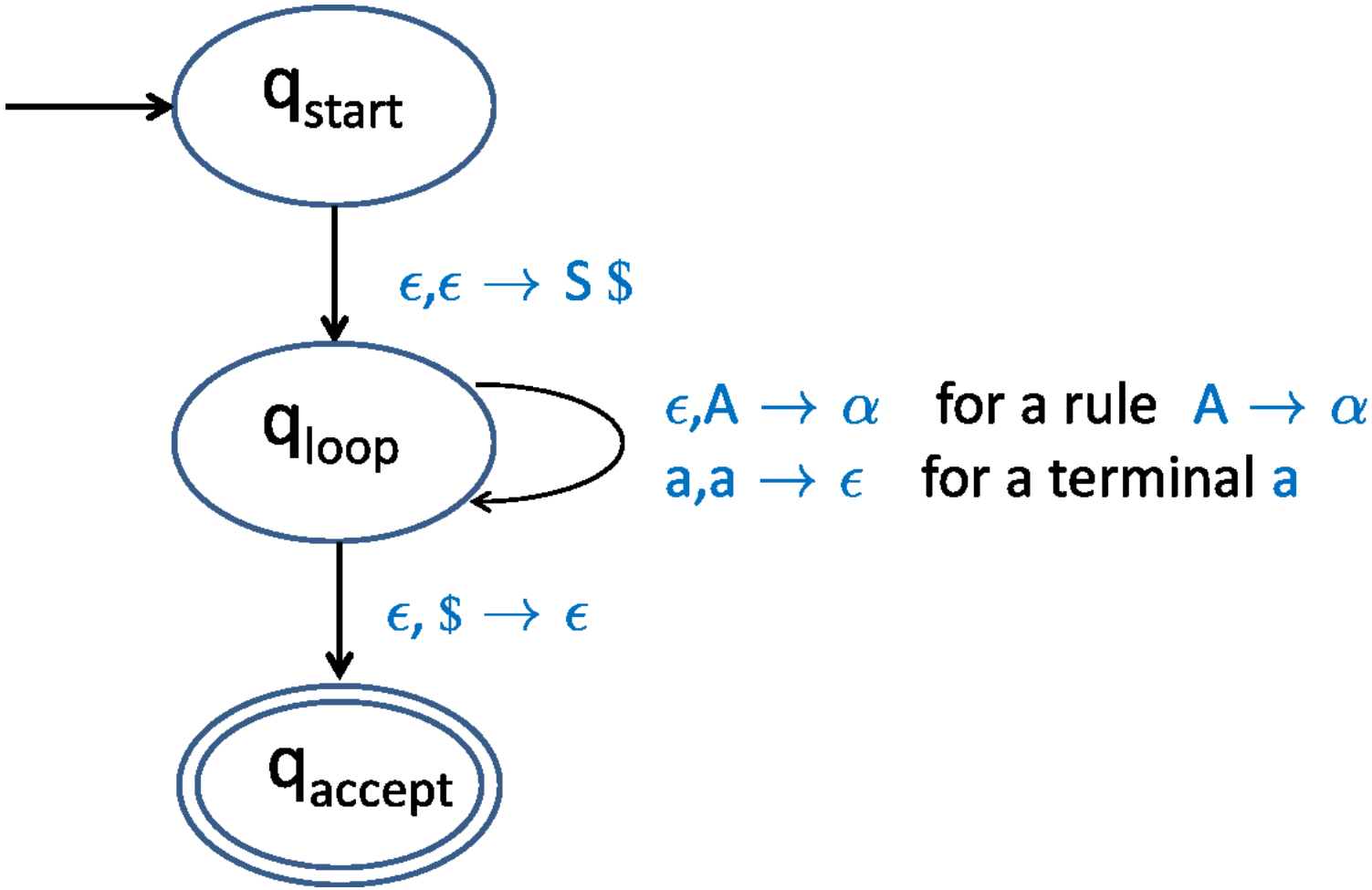
   Does it change the derived language?

Informal description of P

Algorithm 13 (P)

1. Push $S\$$ on stack
2. While top of the stack t is not $\$$:
3. If t is variable A,
   (*non-deterministically*) select rule $A \to \alpha$ and substitute t with α.
4. If t is a terminal a,
   read next input and compare; `Reject` if different.
5. `Accept` if end of input and stack is empty.

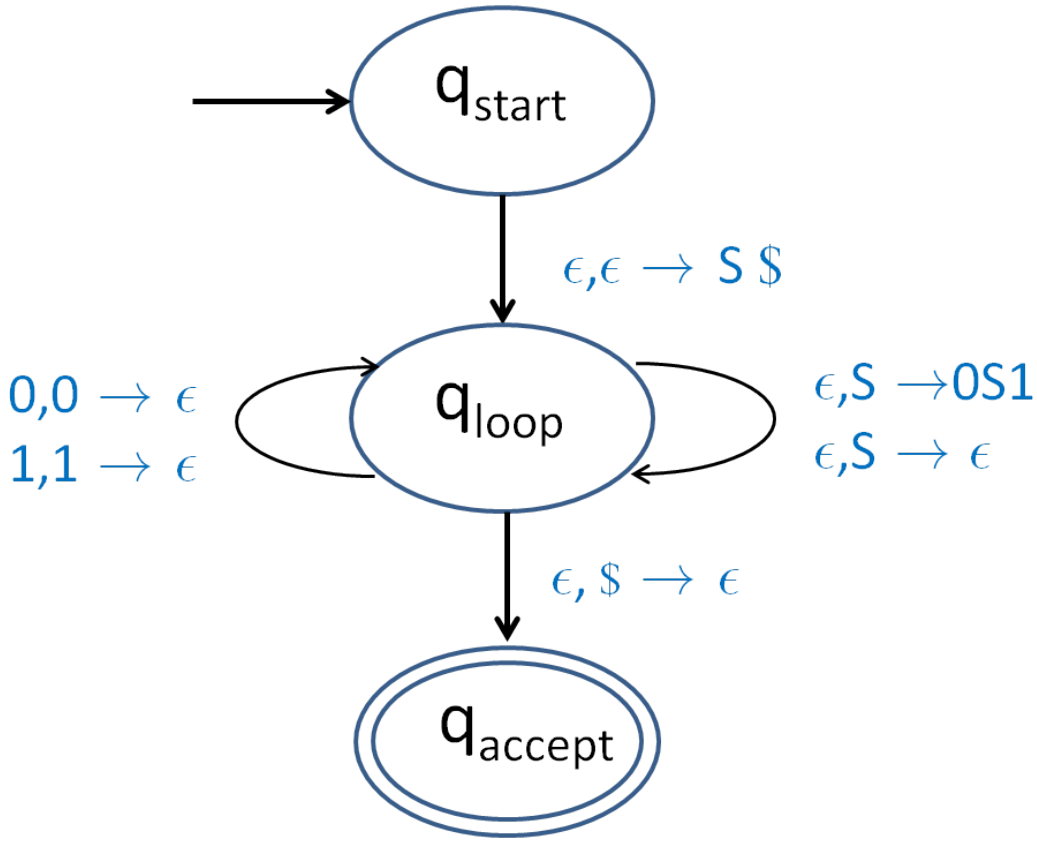
State Diagram for P



Example

consider the CFG:
$S \to 0S1|\varepsilon$.

The related PDA:



Claim: $\mathcal{L}(P) = \mathcal{L}(G)$

Claim 14

$S \xrightarrow{*} \alpha$ iff $\alpha = \alpha_1\alpha_2$ such that $(q_{loop}, \alpha_2\$) \in \widehat{\delta}(q_{loop}, \alpha_1, \$)$.

Does the above yields that $\mathcal{L}(P) = \mathcal{L}(G)$?

Note that $\alpha_1 \in \Sigma^*$

$S \stackrel{*}{\to} \alpha \implies \alpha = \alpha_1\alpha_2$ **such that** $(q_{loop}, \alpha_2\$) \in \widehat{\delta}(q_{loop}, \alpha_1, S\$)$

Proof by induction on the number of derivations steps used to yield α from S.

- Single derivation step: hence there is a rule $S \to \alpha$. Thus $(q_{loop}, \alpha\$) \in \widehat{\delta}(q_{loop}, \varepsilon, S\$)$, and the proof follows for $\alpha_1 = \varepsilon$ and $\alpha_2 = \alpha$.
- Assume $S \stackrel{*}{\to} \alpha$ in $k > 1$ derivation steps, and let α' be the string derived by the first $(k-1)$ steps.
- By i.h $\alpha' = \alpha'_1\alpha'_2$ such that $(q_{loop}, \alpha'_2\$) \in \widehat{\delta}(q_{loop}, \alpha'_1, S\$)$
- Write $\alpha'_2 = w_1 A w_2$ where A is the left most variable in α'_2.
- k'th derivation step replaces this occurrence of A with a string s (?)
- It is easy to see that $(q_{loop}, sw_2\$) \in \widehat{\delta}(q_{loop}, \alpha'_1 w_1, S\$)$.
- To complete the proof take $\alpha_1 = \alpha'_1 w_1$ and $\alpha_2 = sw_2$.

$\alpha = \alpha_1\alpha_2$ **such that** $(q_{loop}, \alpha_2\$) \in \widehat{\delta}(q_{loop}, \alpha_1, \$) \implies S \xrightarrow{*} \alpha$

Proof by induction on the number of steps used by P to process α_1.

- Single step: $\alpha_1 = \varepsilon$ and $\alpha_2 = S\$$, and the proof follows since $S \xrightarrow{*} S$.
- Assume α_1 was processed in $k > 1$ steps. Let α_1' and α_2' be input string read and stack value *before* last step:

  i.e., By definition, $(q_{loop}, \alpha_2\$) \in \widehat{\delta}(q_{loop}, a, \alpha_2'\$)$, for $a = \varepsilon$ if $\alpha_1 = \alpha_1'$, and last letter of α_1 o/w.
- $(q_{loop}, \alpha_2'\$) \in \widehat{\delta}(q_{loop}, \alpha_1', \$)$.
- By i.h $S \xrightarrow{*} \alpha' = \alpha_1'\alpha_2'$.
- If k'th move of P is reading an input character, then $\alpha_1\alpha_2 = \alpha_1'\alpha_2'$, and therefore $S \xrightarrow{*} \alpha_1\alpha_2$
- O/w, $\alpha_1' = \alpha_1$, $\alpha_2' = Aw$ and $\alpha_2 = sw$ for some rule $A \to s \in R$
- Hence $S \xrightarrow{*} \alpha_1\alpha_2$

PDA $\Longrightarrow$ CFL

Lemma 15

$\mathcal{L}_{\text{PDA}} \subseteq \mathcal{L}_{\text{CFG}}$.

If a PDA accepts a language then it is context free.

We prove the lemma by constructing a CFG G for a language $\mathcal{L}$ accepted by a PDA P

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$. We assume wlg. that:

- A single accepting state $q_a \in F$.
- P empties the stack before accepting
- Each transition either pops or pushes

Can we justify the above?

Proof idea

- Suppose string x takes P from state p with empty stack to state q with empty stack.
- First move that touches the stack must be a push, last must be a pop.
- *Either stack is empty only at start and finish*:

  Simulate by $A_{pq} \rightarrow aA_{rs}b$, where a, b are first and last symbols in x, r is state that p can reach in a step and s is state that can reach q in a step.
- Or *stack was empty at some point in between*:

  Simulate by $A_{pq} \rightarrow A_{pr}A_{rq}$ where r is intermediate state and P has empty stack.

Defining $G = (V, \Sigma, R, S)$

- $V = \{A_{pq} : p, q \in Q\}$

  Idea: A_{pq} will generate all strings that take P from p with an empty stack, to q with an empty stack

- $S = A_{q_0, q_a}$

- Initially $R = \emptyset$ and

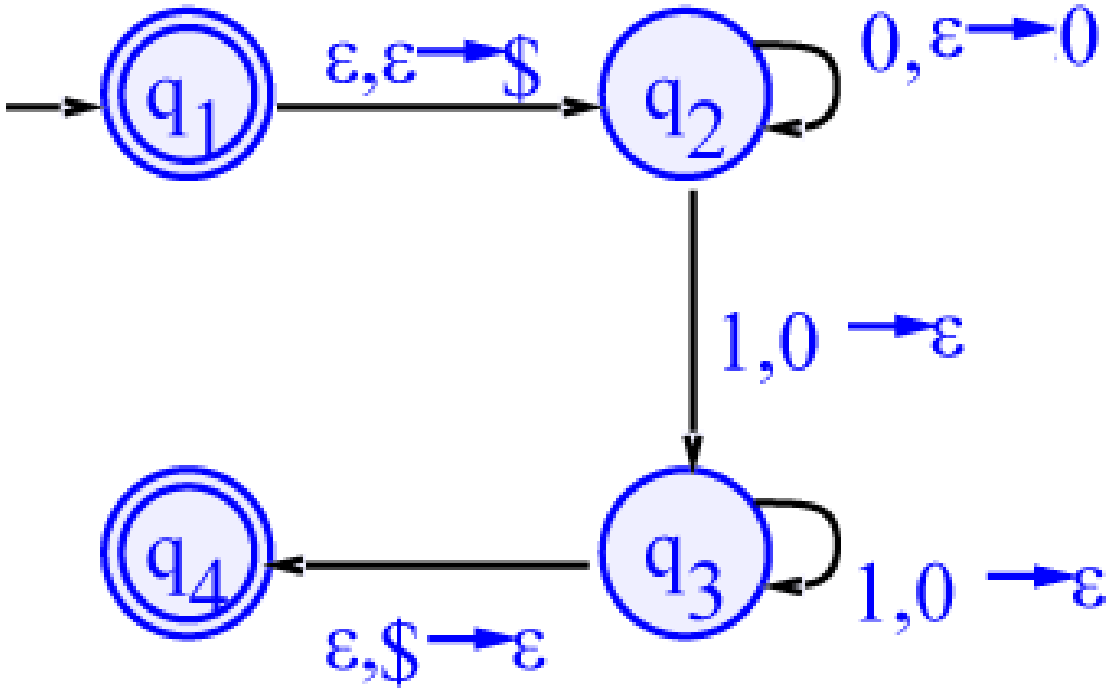
  1. Add $\{A_{pq} \to A_{p,r}A_{r,q} : p, q, r \in Q\}$ to R
  2. Add $\{A_{qq} \to \varepsilon : q \in Q\}$ to R
  3. For all $p, r, s, q \in Q$, $a, b \in \Sigma_\varepsilon$ and $\gamma \in \Gamma$ such that

     3.1 $(r, \gamma) \in \delta(p, a, \varepsilon)$ and

     3.2 $(q, \varepsilon) \in \delta(s, b, \gamma)$

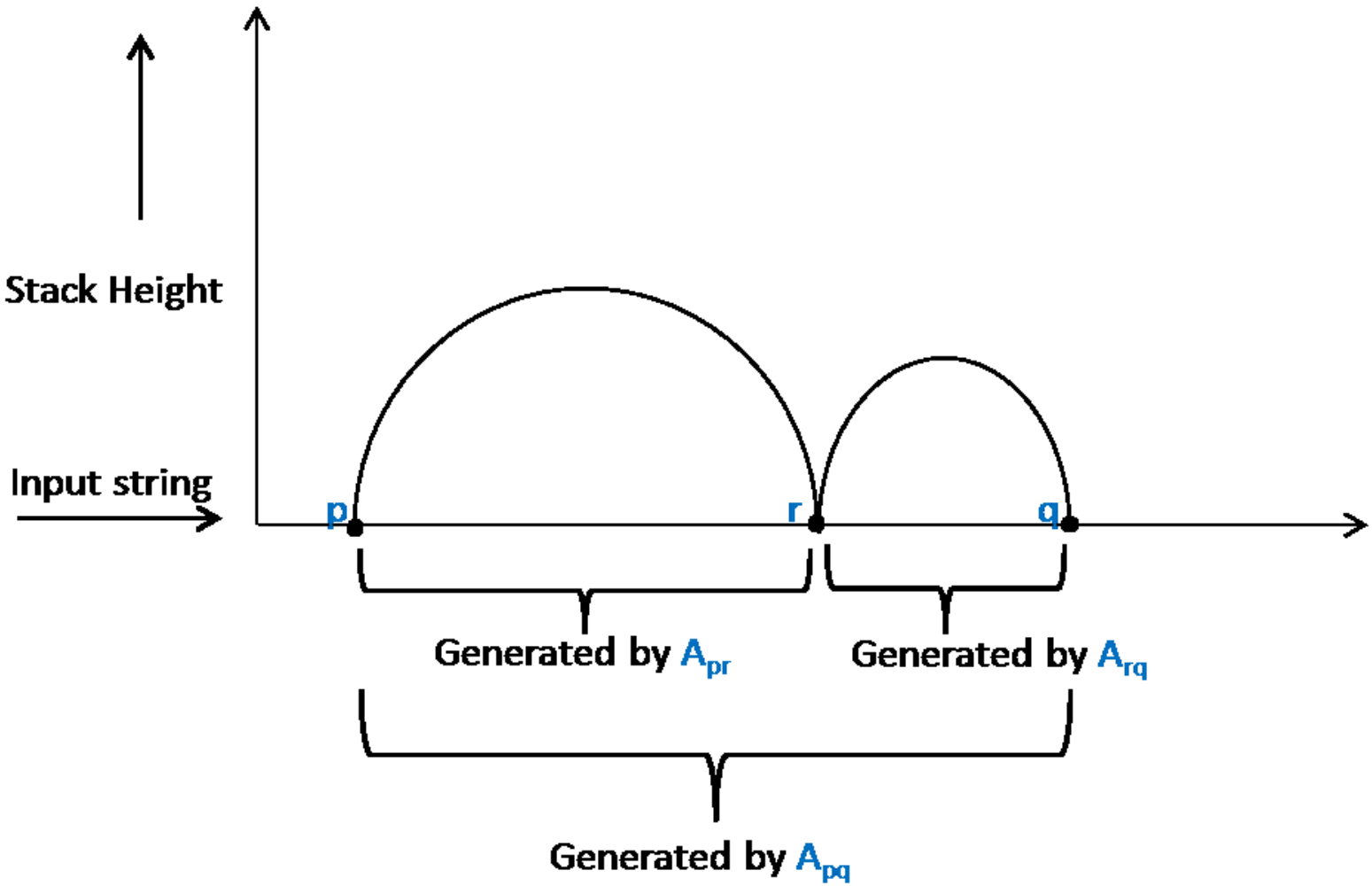
     add $A_{pq} \to aA_{r,s}b$ to R

Example PDA to CFG

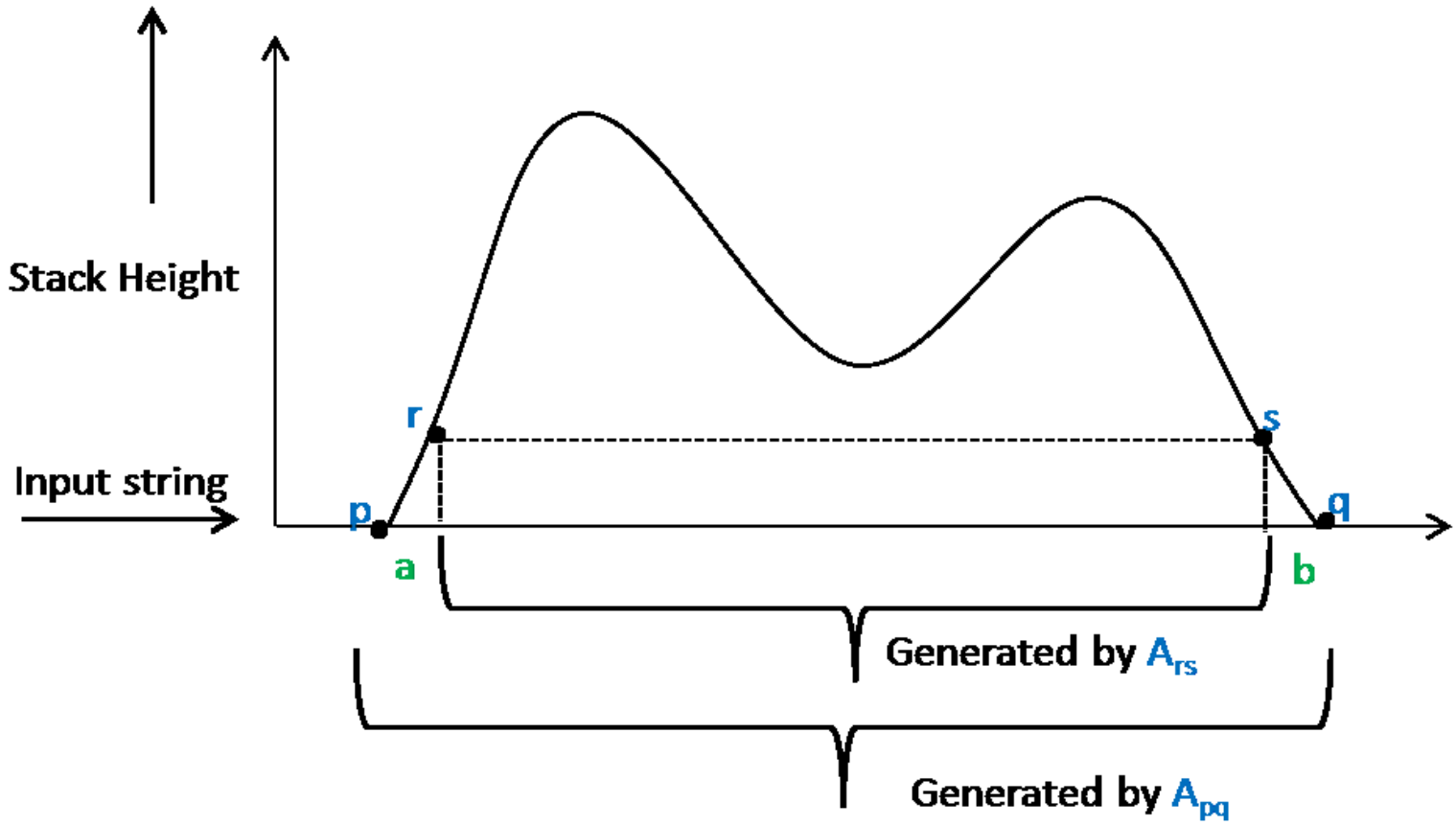


$A_{q_1,q_4} \rightarrow A_{q_2,q_3}$
$A_{q_2,q_3} \rightarrow 0A_{q_2,q_3}1$
$A_{q_2,q_3} \rightarrow 0A_{q_2,q_2}1.$
$A_{q_2,q_2} \rightarrow \varepsilon.$

PDA Computation corresponding to $A_{pq} \rightarrow A_{p,r}A_{r,q}$



PDA Computation corresponding to $A_{pq} \rightarrow aA_{r,s}b$



Claim: $\mathcal{L}(G) = \mathcal{L}(P)$

Claim 16

$A_{pq} \xrightarrow{*} w \in \Sigma^*$ iff $(q, \varepsilon) \in \widehat{\delta}(p, w, \varepsilon)$

Proof by induction on the number of derivation rules/ transitions