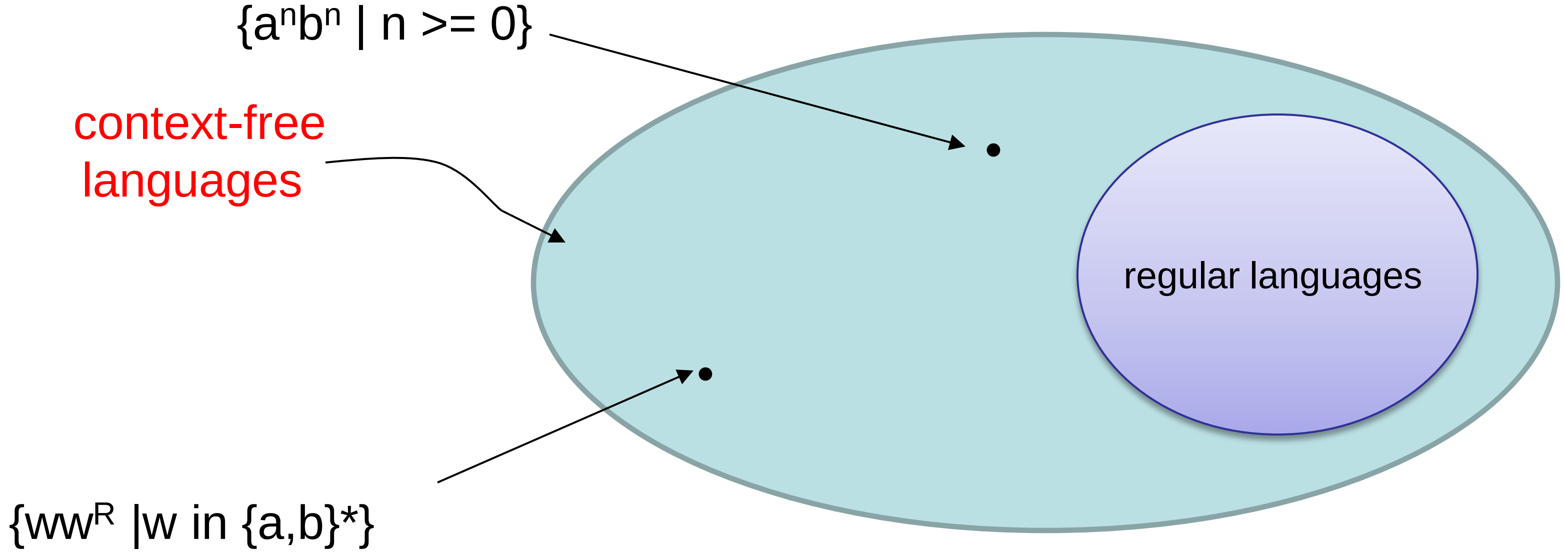
Chapter 2: Context-Free Languages

Context Free Languages

- The class of regular language is a subset of the class of context free languages



Context Free Grammar Definition

- A CFG $G = (V, \Sigma, R, S)$ where $V \cap \Sigma = \emptyset$,
  - V is a finite set of symbols called nonterminals
  - Σ is a finite set of symbols called terminals.
  - R is a finite set of rules, which is a subset of $V \, X \, (V \cup \Sigma)^*$.
    - <nonterminal symbol> $\rightarrow$ a string over terminals and nonterminals.
    - write $A \rightarrow w$ if $(A, w) \in R$.
  - $S \in V$ is the start nonterminal.

A CFG for some sentences

<Sentence> $\rightarrow$ <noun> <verb> <object>

<noun> $\rightarrow$ Mike | Jean

<verb> $\rightarrow$ likes | sees

<object> $\rightarrow$ flowers | zoo

- Example for the grammar to generate the sentence *Jean likes flowers*:

  <Sentence> $\Rightarrow$ <noun><verb><object> $\Rightarrow$ Jean <verb><object>
  $\Rightarrow$ Jean likes <object> $\Rightarrow$ Jean likes flowers

A CFG for arithmetic expressions

- $E \to E + E \mid E * E \mid (E) \mid a \mid b$
- The start nonterminal: E.
- The set of terminals: {a, b, +, *, (,)}
- The set of nonterminals: {E}
- A derivation for generating a+a*b:

$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \Rightarrow a + a * E$$

$$\Rightarrow a + a * b$$

Another grammar for arithmetic expressions

$E \to E + T \mid T$

$T \to T * F \mid F$

$F \to (E) \mid a \mid b$

A derivation for a + a * b:

$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow x + T \Rightarrow a + T * F$
$\Rightarrow a + F * F \Rightarrow a + a * F \Rightarrow x + a*b$

Derivations and the language of the grammar G: L(G)

- One step derivation:
  - $u \Rightarrow v$ if $u = xAy$, $v = xwy$ and $A \rightarrow w$ in R
- 0 or more steps derivation:
  - $u \Rightarrow^* v$ if $u \Rightarrow u_1 \Rightarrow .... \Rightarrow u_n = v$ $(n \geq 0)$
- $L(G) = \{ w \text{ in } T^* \mid S \Rightarrow^* w \}$.
- A language L is a <u>context-free language</u> if there is a CFG G such that L(G) = L.

Example:

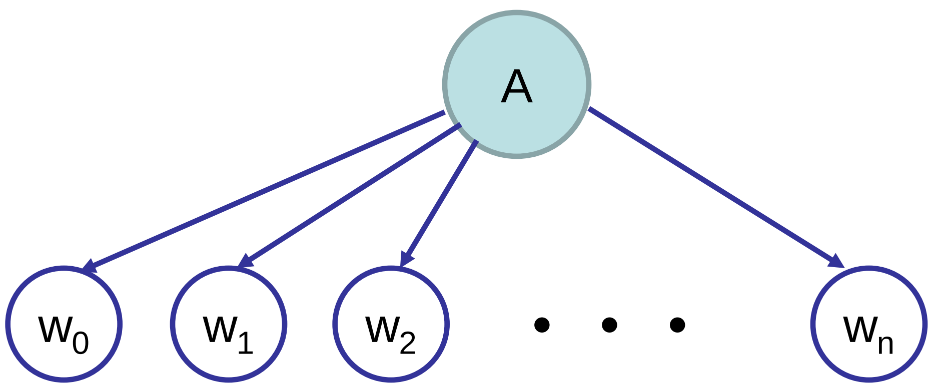
- CFG: $S \to aSb \mid \varepsilon$
- Derivations for generating aabb:

  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aa\varepsilon bb = aabb$

- $L(G) = \{a^n b^n \mid n \geq 0\}$

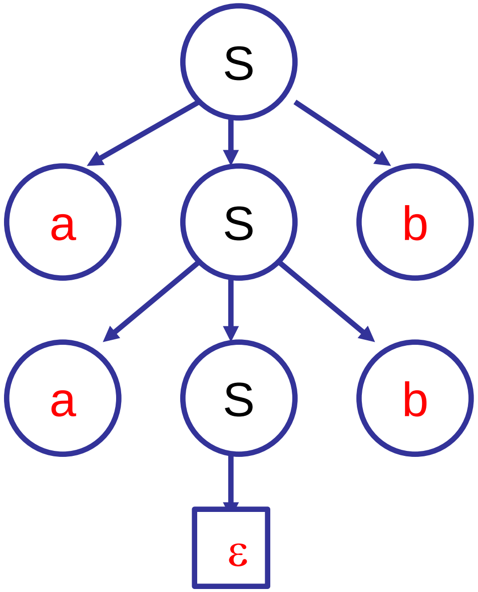
Parse trees

In general, for a rule $A \rightarrow w_0 w_1 \ldots w_n$, each node for w_i is placed as a child of the node labeled with A following the order.

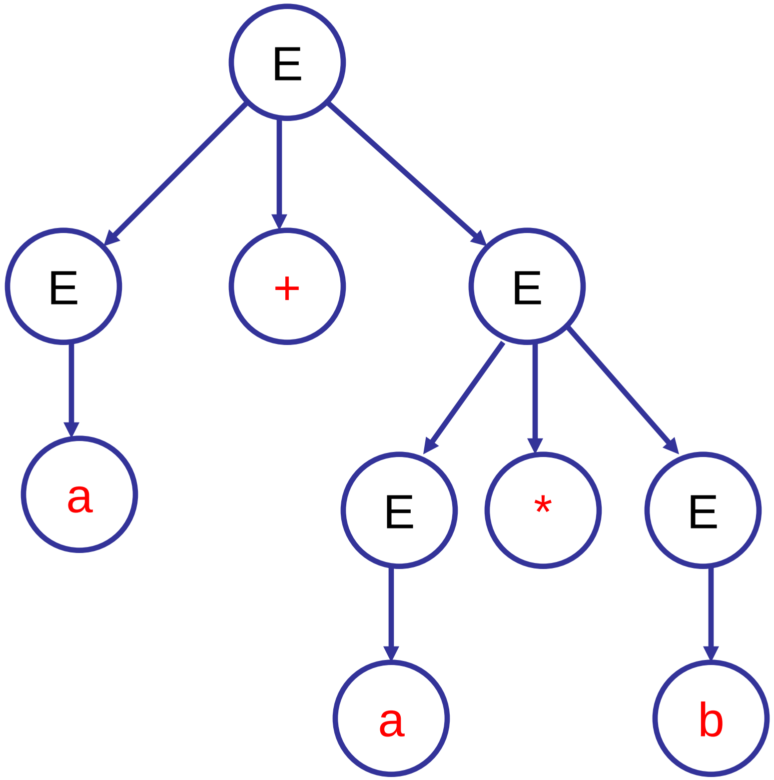


Parse trees (cont'd)

- All derivations can be shown with parse trees.
- The order of rule applications may be lost.



$$E \Rightarrow E + E \Rightarrow E + E * E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * b$$



Leftmost and Rightmost Derivations

- A derivation is a leftmost derivation if at every step the leftmost remaining nonterminal is replaced.
  - Consider $E \Rightarrow E + E \Rightarrow a + E$
- A derivation is a rightmost derivation if at every step the leftmost remaining nonterminal is replaced.
  - $E \Rightarrow E + E \Rightarrow E + a$

Ambiguity

- A string *w* is derived ambiguously in context-free grammar G if it has two or more different leftmost derivations.
- A CFG is ambiguous if it generates some string ambiguously.
- A CFL is inherently ambiguous if it can only be generated by ambiguous grammars.

Ambiguity (cont'd)

- An ambiguous CFG:
  - $E \to E + E \mid E * E \mid (E) \mid a \mid b$
  - For string a + a*b, two leftmost derivations:
    - $E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * b$

    or

    - $E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * b$
- An inherently ambiguous CFL:

  $\{a^n b^m c^m d^n \mid n, m > 0\} \cup \{a^n b^n c^m d^m \mid n, m > 0\}$

Chomsky Normal Form (CNF)

- Every rule in the CFG G is of one of the two forms:
  1) $A \rightarrow a$
  2) $A \rightarrow BC$, $B \neq S$ and $C \neq S$ (S is the start symbol)
  3) Only $S \rightarrow \varepsilon$ is allowed if $\varepsilon \in L(G)$.
- All grammars can be converted into CNF

Closure properties of CFLs

- CFLs are closed under:
  1) Union
  2) Concatenation
  3) Star
- CFLs are NOT closed under intersection or complement

Given two CFGs

- $L_1 = L(G_1)$ where

  $G_1 = (V_1, \Sigma, R_1, S_1)$
- $L_2 = L(G_2)$ where

  $G_2 = (V_2, \Sigma, R_2, S_2)$
- Without loss of generality, we assume that $V_1 \cap V_2 = \varnothing$

General construction of the CFG for the CFL after union

- Let $G = (V, \Sigma, R, S)$ where
  - $V = V_1 \cup V_2 \cup \{ S \}$, (S is a new start symbol)
  - $S \notin V_1 \cup V_2$
  - $R = R_1 \cup R_2 \cup \{ S \to S_1 \mid S_2 \}$

Example

- $L_1 = \{ a^n b^n \mid n \geq 0 \}$
  - G_1: $S_1 \rightarrow aS_1b \mid \varepsilon$
- $L_2 = \{ b^n a^n \mid n \geq 0 \}$
  - G_2: $S_2 \rightarrow bS_2a \mid \varepsilon$
- The grammar for $L_1 \cup L_2$
  - Add a new start symbol S and rules $S \rightarrow S_1 \mid S_2$, so the new grammar is:

$$S \rightarrow S_1 \mid S_2$$
$$S_1 \rightarrow aS_1b \mid \varepsilon$$
$$S_2 \rightarrow bS_2a \mid \varepsilon$$

General construction of the CFG for the CFL after concatenation

- Let $G = (V, \Sigma, R, S)$ where
  - $V = V_1 \cup V_2 \cup \{ S \}$,
  - $S \notin V_1 \cup V_2$
  - $R = R_1 \cup R_2 \cup \{ S \rightarrow S_1 S_2 \}$

S is a new start symbol and $S \rightarrow S_1S_2$ is a new rule.

Example

- $L_1 = \{ a^n b^n \mid n \geq 0 \}$, $L_2 = \{ b^n a^n \mid n \geq 0 \}$
- $L_1.L_2 = \{ a^n b^{\{n+m\}} a^m \mid n, m \geq 0 \}$
- The CFG for $L_1.L_2$:
  - Add a new start symbol S and rule $S \rightarrow S_1 S_2$ so the CFG for $L_1.L_2$ is:

$$S \rightarrow S_1 S_2$$
$$S_1 \rightarrow aS_1b \mid \varepsilon$$
$$S_2 \rightarrow bS_2a \mid \varepsilon$$

General construction of the CFG for the CFL after Star

- Let $G = (V, \Sigma, R, S)$ where
  - $V = V_1 \cup \{ S \}$,
  - $S \notin V_1$
  - $R = R_1 \cup \{ S \rightarrow SS_1 \mid \varepsilon\}$

Examples

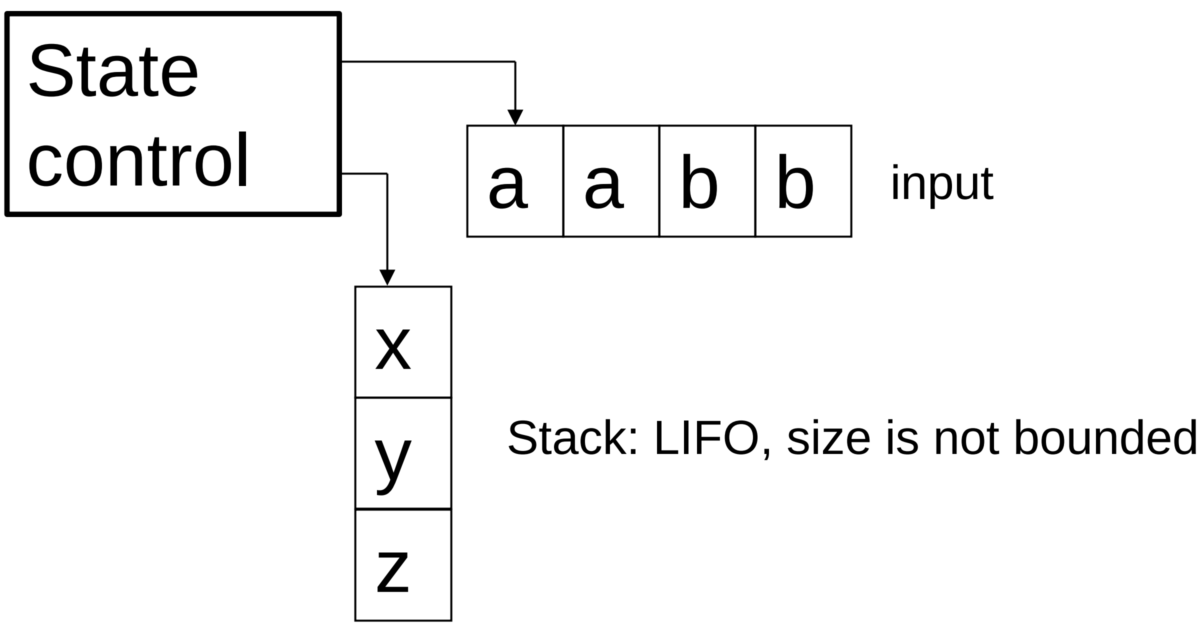
- $L_1 = \{a^n b^n \mid n \geq 0\}$
  - $L_1^* = \{a^{n1}b^{n1} \ldots a^{nk}b^{nk} \mid k \geq 0$ and $n_i \geq 0$ for all i $\}$
- $L_2 = \{ a^{n^2} \mid n \geq 1 \}$
  - $L_2^* = a^*$
- The CFG for L_1^*:
  - Add a new start symbol S and rules $S \to SS_1 \mid \varepsilon$.
  - The CFG for L_1^* is:

$$S \to SS_1 \mid \varepsilon$$

$$S_1 \to aS_1b \mid \varepsilon$$

Push Down Automaton (PDA)

- PDA is a language acceptor model for CFLs.
- Similar to NFA but has an extra component called a stack

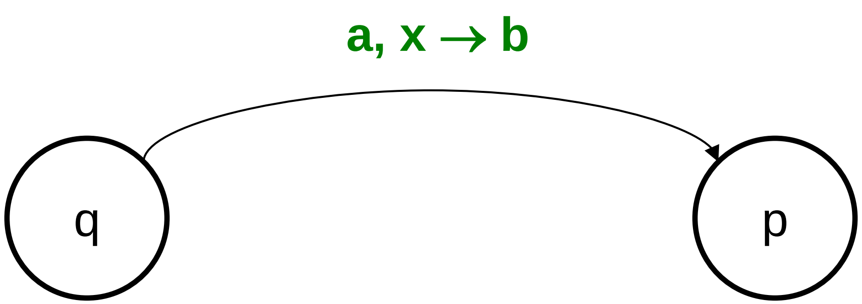


PDA (cont’d)

- In one move, a PDA can :
  - change state,
  - read a symbol from the input tape or ignore it,
  - Write a symbol to the top (push) of the stack and the rest in the stack are “push down”, or
  - Remove a symbol from the top (pop) of the stack and other symbols in the stack are moved up.

PDA (cont'd)

- If read a, transit from state p to state q, pop x from the stack, and push b into the stack, it is showed as



Definition of PDA

- A PDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q0, F)$, where $Q, \Sigma, \Gamma, \delta, F$ are finite sets:
  1. Q is the set of states
  2. Σ is the input alphabet
  3. Γ is the stack alphabet
  4. $\delta: (Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon) \longrightarrow (Q \times \Gamma_\varepsilon)$
  5. $q0 \in Q$ is the start state, and
  6. $F \subseteq Q$ is the set of accept states

Example of a PDA

- PDA for L = $\{0^n1^n \mid n \geq 0\}$

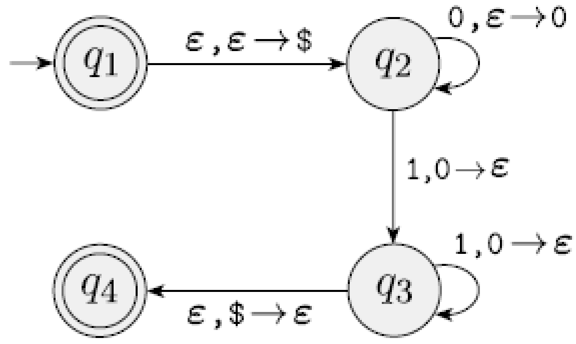


FIGURE 2.15
State diagram for the PDA M_1 that recognizes $\{0^n1^n \mid n \geq 0\}$

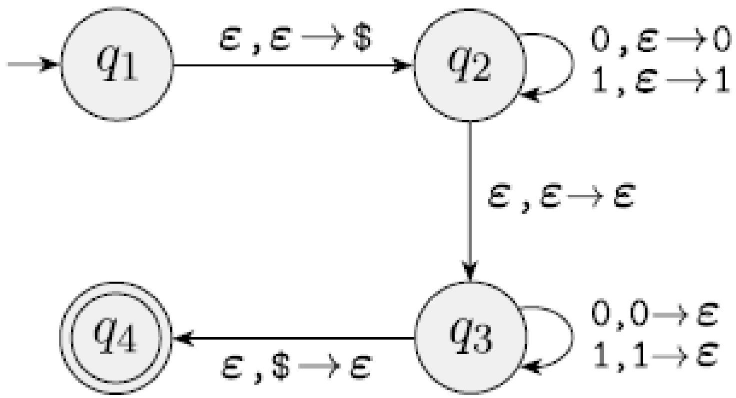
Initially place a special symbol $ on the stack and then pop it at the end before acceptance.

This figure is taken from the book *Introduction to Theory of Computation, Michael Sipser,* page 115.

Definition of L(M)

- The language that PDA M accepts:
  - L(M) = $\{w \in \Sigma^* \mid M \text{ accepts } w\}$

Example



What is the language of the above PDA?

This figure is taken from the book *Introduction to Theory of Computation, Michael Sipser,* page 116.

Equivalence with CFGs

- For every CFG G there is a PDA M such that $L(G) = L(M)$
- For every PDA M there is a CFG G such that $L(M) = L(G)$

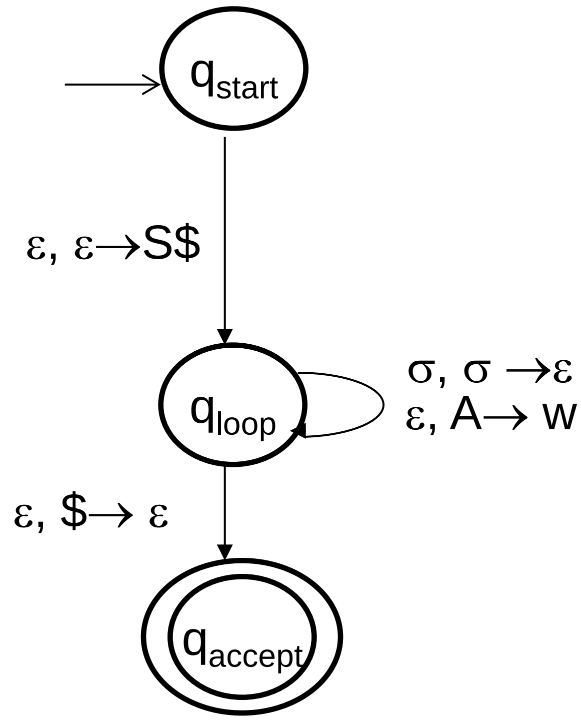
CFG → PDA

- Given CFG $G = (V, \Sigma, R, S)$
  - Let PDA $M = (Q, \Sigma, \Sigma \cup V \cup \{\$\}, \delta, q_{start}, \{q_{accept}\})$
  - $Q = \{q_{start}, q_{loop}, q_{accept}\}$

1. $((q_{start}, \varepsilon, \varepsilon), (q_{loop}, S\$)) \in \delta$
2. For each rule $A \rightarrow w$,

   $((q_{loop}, \varepsilon, A), (q_{loop}, w)) \in \delta$
3. For each symbol $\sigma \in \Sigma$

   $((q_{loop}, \sigma, \sigma), (q_{loop}, \varepsilon)) \in \delta$
4. $((q_{loop}, \varepsilon, \$), (q_{accept}, \varepsilon)) \in \delta$



Example

S → aTb | b
T → Ta | ε

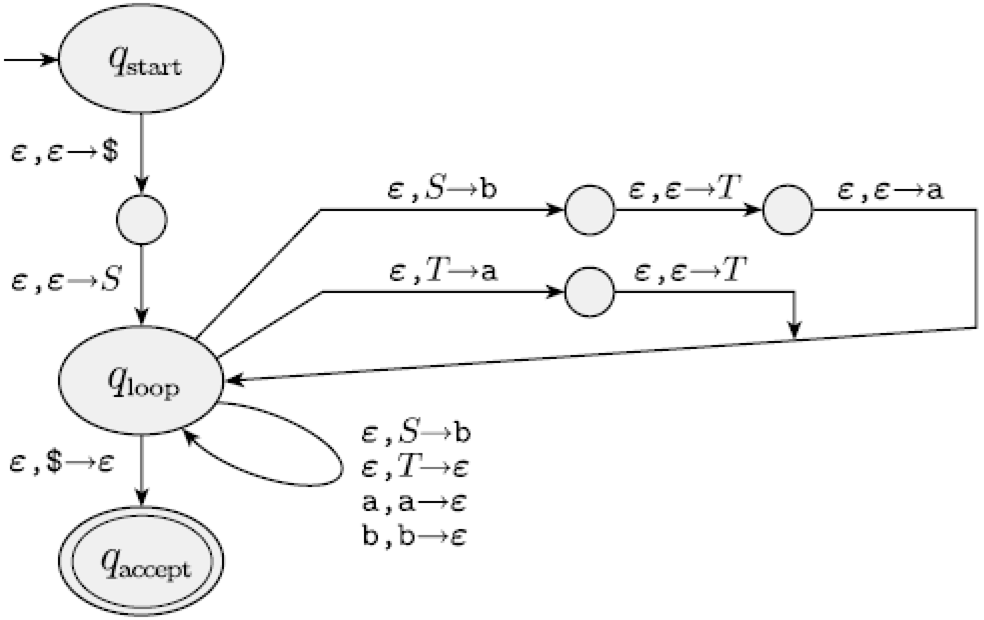


FIGURE 2.26
State diagram of P_1

This figure is taken from the book *Introduction to Theory of Computation, Michael Sipser,* page 120.

PDA P → CFG G

- First, we simplify our task by modifying P slightly to give it the following three features:
  1. It has a single accept state, q_{accept}.
  2. It empties its stack before accepting.
  3. Each transition either pushes a symbol onto the stack (a *push* move) or pops one off the stack (a *pop* move), but it does not do both at the same time.

PDA P $\rightarrow$ CFG G (cont'd)

- For P = $\{Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\}\}$, to construct G:
- The variables of G are $\{A_{pq} \mid p, q \in Q\}$.
- The start variable is $Aq_0q_{qaccept}$

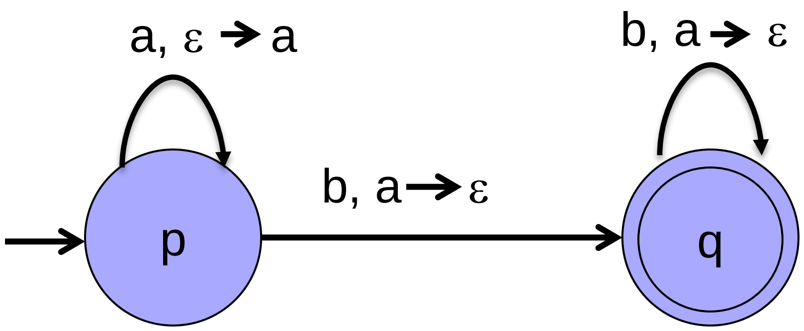
  Type 1: For each p, q, r, s $\in$ Q, u $\in \Gamma$, and a, b $\in \Sigma_\varepsilon$, if ((p, a, ε), (r, u)) is in δ and ((s, b, u), (q, ε)) is in δ, put the rule $A_{pq} \rightarrow aA_{rs}b$ in G.

  Type 2: For each p, q, r $\in$ Q, put the rule $A_{pq} \rightarrow A_{pr}A_{rq}$ in G.

  Type 3: Finally, for each p $\in$ Q, put the rule $A_{pp} \rightarrow \varepsilon$ in G.

Example

- Let M be the PDA for $\{a^n b^n \mid n > 0\}$
  - M = $\{\{p, q\}, \{a, b\}, \{a\}, \delta, p, \{q\}\}$, where
  - $\delta = \{((p, a, \varepsilon),(p, a)), ((p, b, a), (q, \varepsilon)), ((q, b, a),(q, \varepsilon))\}$



Example (cont'd)

- CFG, $G = (V, \{a, b\}, A_{pq}, R)$, A_{pq} is the start variable
  - $V = \{A_{pp}, A_{pq}, A_{qp}, A_{qq}\}$.
- R contains the following rules:
  - Type 1:
    - $A_{pq} \rightarrow aA_{pp}b$
    - $A_{pq} \rightarrow aA_{pq}b$
  - Type 2:
    - $A_{pp} \rightarrow A_{pp} A_{pp} \mid A_{pq} A_{qp}$
    - $A_{pq} \rightarrow A_{pp} A_{pq} \mid A_{pq} A_{qq}$
    - $A_{qp} \rightarrow A_{qp} A_{pp} \mid A_{qq} A_{qp}$
    - $A_{qq} \rightarrow A_{qp} A_{pq} \mid A_{qq} A_{qq}$
  - Type 3:
    - $A_{pp} \rightarrow \varepsilon$
    - $A_{qq} \rightarrow \varepsilon$

We can discard all rules containing the variables A_{qq} and A_{qp}. And we can also simplify the rules containing A_{pp} and get the grammar with just two rules $A_{pq} \rightarrow ab$ and $A_{pq} \rightarrow aA_{pq}b$.

Non-context free languages

- Pumping lemma for context-free languages:
  - If *A* is an infinite context-free language, then there is a number *p* (the pumping length) where, if s is any string in *A* of length at least *p*, then s may be divided into five pieces $s = uvxyz$ satisfying the conditions:

1. $|vy| > 0$,
2. $|vxy| <= p$, and
3. For each $i \geq 0$, $uv^ixy^iz \in A$.

Some non context-free languages

- The following languages are not context-free.
  1. $\{a^n b^n c^n \mid n \geq 0\}$.
  2. $\{ww \mid w \text{ in } \{a, b\}^*\}$
  3. $\{a^{n^2} \mid n \geq 0\}$
  4. $\{w \text{ in } \{a, b, c\}^* \mid w$ has equal a’s, b’s and c’s$\}$.

Prove $L=\{a^n b^n c^n \mid n \geq 0\}$ is not a CFL

- Assume L is a CFL. L is infinite.
- Let $w = a^p b^p c^p$, where p is the pumping length

$$|vy| > 0$$

$$|w| = 3p \geq p$$

$$|vxy| \leq p$$

$$W = \underbrace{a \ldots\ldots a}_{p}\underbrace{b \ldots\ldots b}_{p}\underbrace{c \ldots\ldots c}_{p}$$

Example (contd.)

Case 1:

- Both *v* and *y* contain only one type of alphabet symbols, such that v does not contain both *a*'s and *b*'s or both *b*'s and *c*'s and the same holds for *y*. Two possibilities are shown below.

$$a \underbrace{\ldots\ a}_{v}\ \underbrace{b \ldots\ b}_{y/v}\ \underbrace{c \ldots}_{y}\ c$$

- In this case the string uv^2xy^2z cannot contain equal number of *a*'s, *b*'s and *c*'s. Therefore, $uv^2xy^2z \notin L$.

Example (cont’d)

Case 2:

– Either *v* or *y* contain more than one type of alphabet symbols. Two possibilities are shown below.

$$a\underbrace{\ldots a \ldots\ a\ b}_{v}\ \underbrace{b\ b\ldots}_{y/v}\ \underbrace{b\ c\ \ldots\ c}_{y}$$

– In this case the string uv^2xy^2z may contain equal number of the three alphabet symbols but won’t contain them in the correct order. Therefore,

$uv^2xy^2z \notin$ L.

CFL is not closed under intersection or complement

- Let $\Sigma = \{a, b, c\}$. Both L and L' are CFLs
  - L = {w over Σ | w has equal a's and b's}
  - L' = {w over Σ | w has equal b's and c's}.
- $L \cap L'$ = {w over Σ | w has equal a's, b's and c's}, it is not a CFL.
- Because of CFLs are closed under Union and the DeMorgan's law, we can see that CFLs are not closed under complement either.