Context-Free Grammars

(V, Σ, R, S)

V: non-terminal symbols

Σ: terminals $V \cap \Sigma = \emptyset$

R: productions $R: V \rightarrow (V \cup \Sigma)^*$

S: start symbol

Example

$V = \{ q, f \}$

$\Sigma = \{ 0, 1 \}$

$R = \{ q \rightarrow 11q,\ \ q \rightarrow 00f,\ \ \ f \rightarrow\ 11f,\ \ \ f \rightarrow \varepsilon \}$

$\{ q \rightarrow 11q \mid 00f,\ \ f \rightarrow 11f \mid \varepsilon \}$

$S = q$

Derivation

- If $A \rightarrow B$, then $xAy \Rightarrow xBy$ (xAy *yields* xBy)
- If $s \Rightarrow \cdots \Rightarrow t$, then $s \Rightarrow^* t$.
- x in Σ^* *is generated by* (V,Σ,R,S) if $S \Rightarrow^* x$.
- $G = (V,\Sigma,R,S)$, $\quad L(G) = \{ x \mid S \Rightarrow^* x \}$.

Example

- $G = (\{S\}, \{0,1\}. \{S \rightarrow 0S1 \mid \varepsilon \}, S)$
- ε in $L(G)$ since $S \Rightarrow \varepsilon$
- 01 in $L(G)$ since $S \Rightarrow 0S1 \Rightarrow 01$
- 0011 in $L(G)$ since

  $$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0011$$

- 0^n1^n in $L(G)$ since $S \Rightarrow^* 0^n 1^n$
- $L(G) = \{0^n1^n \mid n \geq 0\}$

Context-Free Language

A language L is *context-free* if

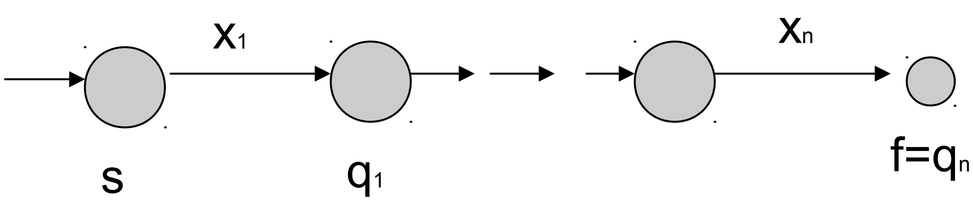
L = L(G) for some CFG G

If L is regular, then L = L(G) for some CFG G

Let L=L(M) for finite automaton M=(Σ, F, Q, δ)

Consider the regular grammar G = (V, Σ, R, S)

V = Q
Σ = Σ
R = { q → xq' | δ(q,x) = q' } U { f → ε | f in F}
S = ί



$$S \Rightarrow x_1q_1 \Rightarrow x_1x_2q_2 \Rightarrow \cdots \Rightarrow x_1\ldots x_nf \Rightarrow x_1\ldots x_n$$

Regular Grammars

A *regular grammar* is a CFG (V, Σ, R, S) where every rule has the form

$$V \rightarrow \Sigma^*(V+\varepsilon)$$

Every regular language is generated by a regular grammar (previous slide)

Regular grammars generate regular languages

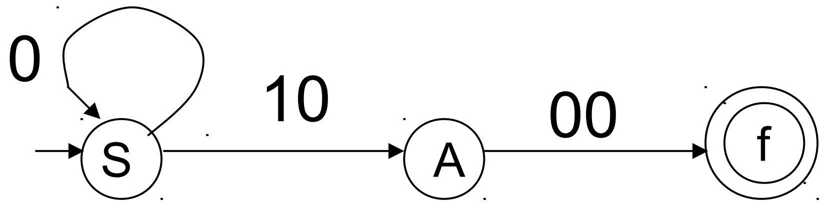
Consider regular grammar $G = (V, \Sigma, R, S)$

Construct a directed graph with vertices $V \cup \{f\}$:

For each rule $A \rightarrow xB$, where x in Σ^* and B in V,
draw edge $A \xrightarrow{x} B$.

For each rule $A \rightarrow x$, where x in Σ^*,
Draw edge $A \xrightarrow{x} f$

Example $G = (\{S,A\}, \{0,1\}, \{S \rightarrow 0S \mid 10A, A \rightarrow 00\}, S)$



Automaton M

$S \Rightarrow^* x$

There is a path from S to f consuming input x

$L(G) = L(M)$

CFL closed under concatenation

Let $A = L(G_A)$ and $B = L(G_B)$,

$G_A = (V_A, \Sigma_A, R_A, S_A)$
$G_B = (V_B, \Sigma_B, R_B, S_B)$

Assume $V_A \cap V_B = \emptyset$.

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

$V = V_A \cup V_B \cup \{S\}$
$\Sigma = \Sigma_A \cup \Sigma_B$
$R = R_A \cup R_B \cup \{S \rightarrow S_A S_B\}$

CFL closed under union

$A = L(G_A)$, $B = L(G_B)$,

$G_A = (V_A, \Sigma_A, R_A, S_A)$
$G_B = (V_B, \Sigma_B, R_B, S_B)$

where $V_A \cap V_B = \emptyset$

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

$V = V_A \cup V_B \cup \{S\}$
$\Sigma = \Sigma_A \cup \Sigma_B$
$R = R_A \cup R_B \cup \{S \rightarrow S_A \mid S_B\}$

CFL closed under Kleene closure

Let L = (G) where G = (V, Σ, R, S)

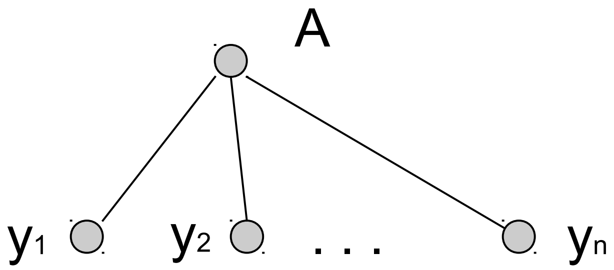
Consider G* =(V, Σ, R*, S),

$$R^* = R \cup \{ S \rightarrow \varepsilon \mid SS\}.$$

Parse Trees

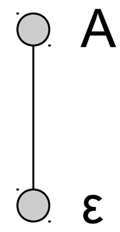
A vertex labeled with a nonterminal is a parse tree

If $A \rightarrow y_1 y_2 \ldots y_n$ is a production, then



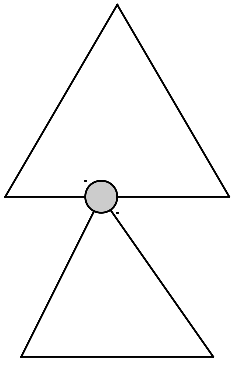
is a parse tree

If $A \rightarrow \varepsilon$ is a production, then



is a parse tree

If a leaf of a parse tree is the root of some other parse tree, then their union is a parse tree

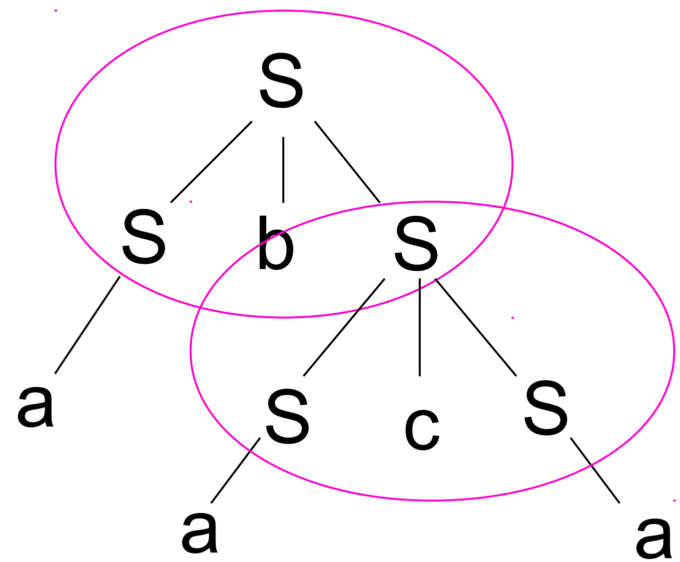


Nothing else is a parse tree

Every derivation has a parse tree

Let G = ({S}, {a, b, c}, R, S), where
R = {S → SbS | ScS | a}

S ⇒ SbS ⇒ SbScS ⇒ abScS ⇒ abSca ⇒ abaca



A parse tree may correspond to multiple derivations

$S \Rightarrow SbS \Rightarrow SbScS \Rightarrow SbSca \Rightarrow abSca \Rightarrow abaca$

has the same parse tree

Each parse tree corresponds to exactly one leftmost derivation

A *leftmost derivation*

$$S \overset{*}{\Rightarrow}_{L} y$$

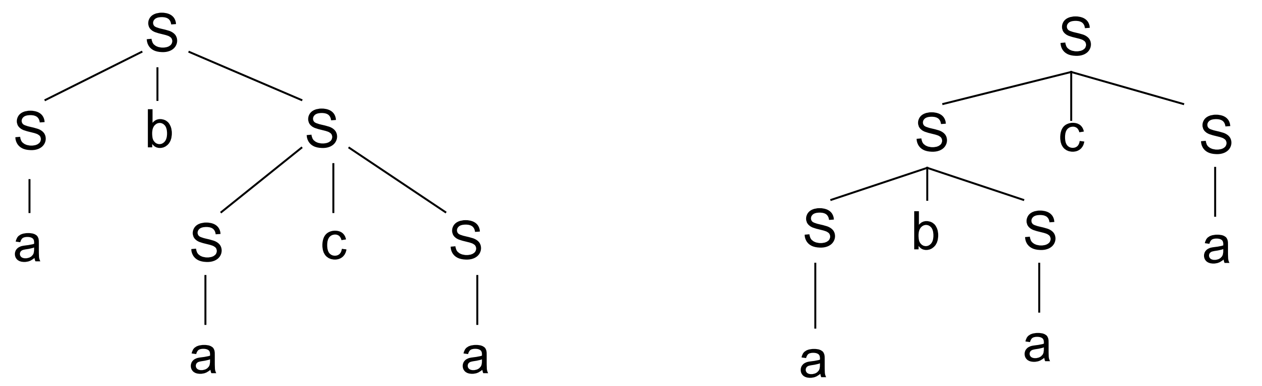
is obtained by applying at each step some production to the leftmost nonterminal symbol

$$S \Rightarrow SbS \Rightarrow abS \Rightarrow abScS \Rightarrow abacS \Rightarrow abaca$$

Ambiguous CFG

A CFG is *ambiguous* if some string has more than one parse tree

$G = (\{S\}, \{a, b, c\}, \{S \rightarrow SbS \mid ScS \mid a\}, S)$
is ambiguous because abaca has two parse trees



Parsing

w in (V U Σ)* is a *left sentential form* if
$S \Rightarrow_L^* w$.

The *leftmost graph* g(G) is defined as follows:

The vertex set is the set of all left sentential forms

There exists directed edge (x, y) if $x \Rightarrow_L y$

A breadth-first construction of g(G) might (termination is an issue) yield a derivation for a string (if one exists)

Greibach normal form

All productions are of the form

$$A \rightarrow a\ x$$

Where a in Σ, and x in V*

If CFL L does not contain the empty string, then L is generated by a CFG in G-NF

A breadth-first construction of g(G) *will (termination is no issue)* yield a derivation for a string (if one exists)

Chomsky normal form

All productions are of the forms

$A \rightarrow BC$, or $A \rightarrow a$

Where a in Σ, and B,C in V

If CFL L does not contain the empty string, then L is generated by a CFG in C-NF

A breadth-first construction of g(G) *will (termination is no issue)* yield a derivation for a string (if one exists)

CYK (Cocke, Younger, Kasami) algorithm

Given G in C-NF and w in Σ^*, decide if w in L(G) in $O(|w|^3)$ time

Given $w = a_1 \dots a_n$, define

$V_{ij} = \{ A \text{ in } V \mid A \Rightarrow^* w_{ij} = a_i \dots a_j \}$

Note that w in L(G) if and only if S in V_{1n}

- A in V_{ii} if and only if $A \rightarrow a_i$

- For $j > i$, $A \Rightarrow^* w_{ij}$ if and only if

  $A \rightarrow BC$, $B \Rightarrow^* w_{ik}$, $C \Rightarrow^* w_{k+1j}$

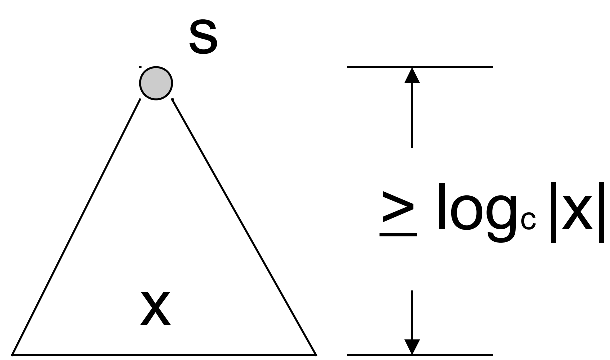
  That is,

$$V_{ij} = \bigcup_{i \le k < j} \{ A \mid A \rightarrow BC,\ B \text{ in } V_{ik},\ C \text{ in } V_{k+1j} \}$$

Pumping Lemma

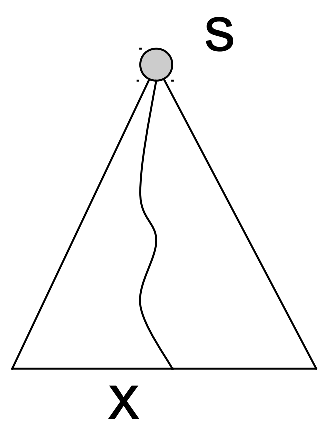
Let $G = (V, \Sigma, R, S)$ be a CFG, and let c be such that for every production $A \rightarrow w$, $|w| \leq c$

Consider a parse tree T for x in $L(G)$,



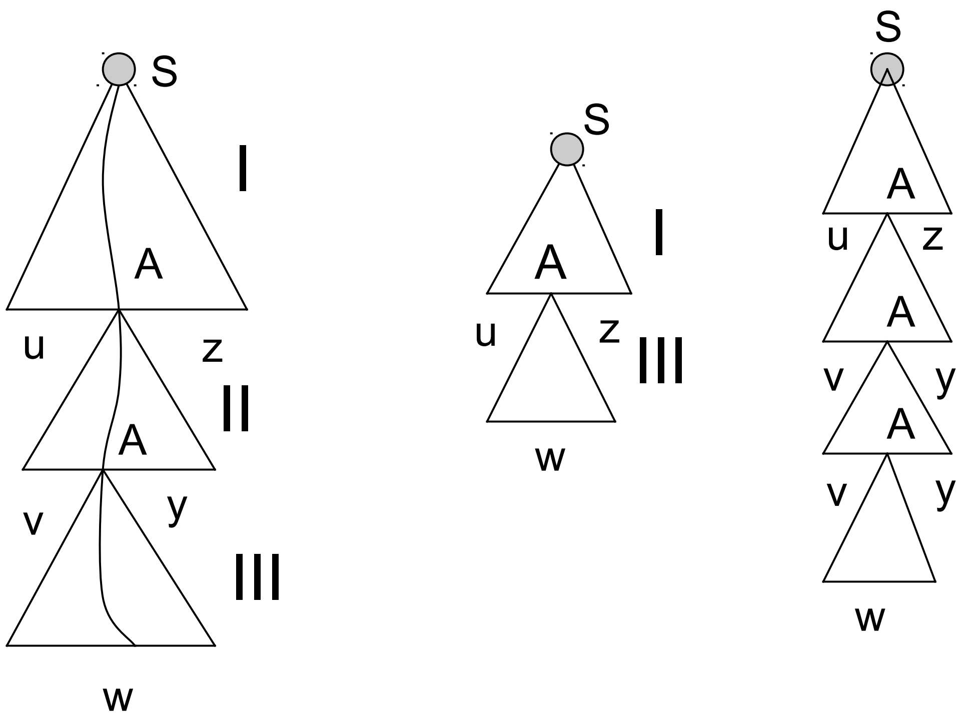
If $|x| > K = c^{|v|+1}$, then T has depth > |V|+1

Hence some path from root to a leaf contains more than |V| variables



Therefore, some nonterminal A appears twice on the path; consider a path with lowest such A

Decompose T into subtrees I, II, III

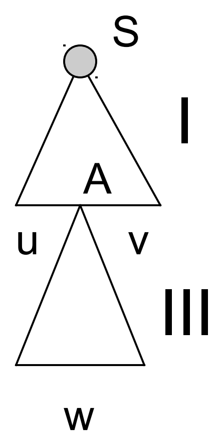


Correspondingly, $x = uvwyz$

and $uv^i wy^i z$ is in L(G) for any $i \geq 0$

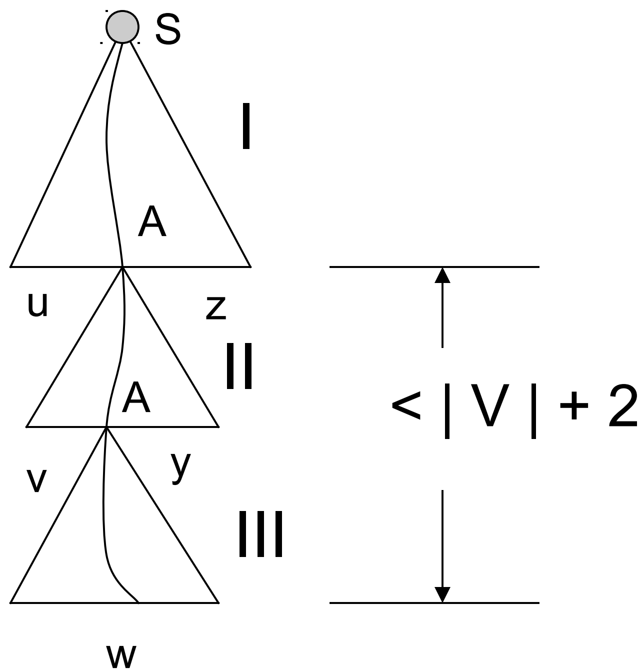
If T is a parse tree for x having minimum number of nodes, then $vy \neq \varepsilon$

If $vy = \varepsilon$, then the following is a parse tree for x



having fewer nodes than T

A is a lowest repeated nonterminal



Subtree II u III has at most K leaves and depth < |V|+2
(otherwise A is not the lowest repeated nonterminal)

Pumping Lemma:

For any CFL L, there exists a constant K such that all x in L with $|x| > K$ can be expressed as $x = uvwyz$ such that

(1) $vy \neq \varepsilon$,

(2) for any $i \geq 0$, $uv^i w y^i z$ is in L,

(3) $|vwy| < K$

CFL not closed under intersection

$A = \{a^m b^m c^n \mid m, n \geq 0\}$

$B = \{a^m b^n c^n \mid m, n \geq 0\}$

A and B are CFL, but

$A \cap B = \{a^n b^n c^n \mid n \geq 0\}$ is not

CFL not closed under complement

Deterministic Pushdown Automata

$(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Q: *states* (finite set)

Σ: *input alphabet* (finite set)

Γ: *stack symbols* (finite set)

$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$ (*transition function)*

q_0: start state

z_0: initial stack top symbol

F: accepting states

Pushdown Automata (nondeterministic)

$\delta(q,a,x) = \{(q',\beta),\ldots\}$

q: current state

a: input symbol (ε transitions allowed)

x: current stack top

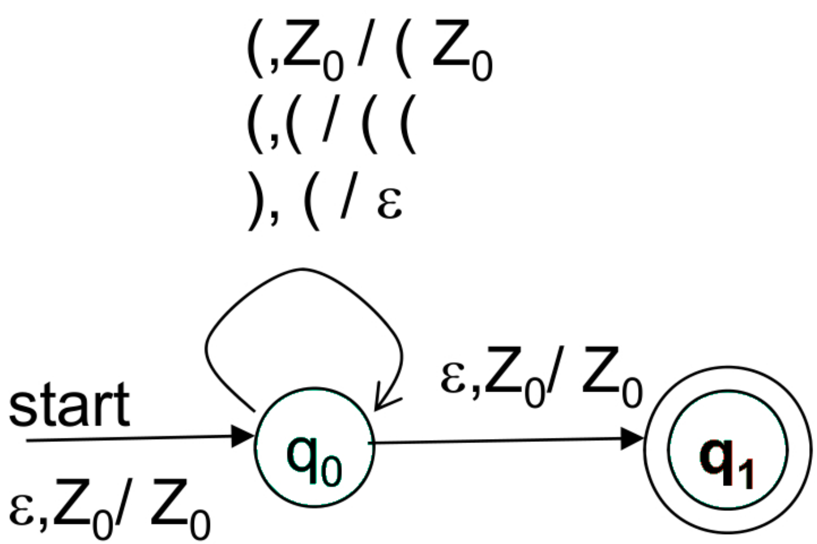
q': next state

β: replacement for x (top at left, $\beta = \varepsilon$ for pop)

Transition diagram

$V = Q$; $q_0 = \rightarrow\bigcirc$, $f = \circledcirc$ for $f \in F$)

$$E = \{\, q \xrightarrow{a,\, x\, /\, \beta} q' \mid (q', \beta) \in \delta(q, a, x) \,\}$$



Instantaneous Description

(q, w, α)

q: current state

w: remaining input

α: stack contents *(top at left)*

$Id \rightarrow Id'$ if instantaneous description Id could change to Id' in one move of the PDA

$Id \rightarrow^{*} Id'$ (zero or more moves)

The language L(P) of PDA P

w such that $(q_0,w,z_0) \longrightarrow^* (f,\varepsilon,\alpha)$ for some final state f

$(q_0, ((())())(), Z_0) \longrightarrow (q_0, (())())(), (Z_0) \longrightarrow (q_0, ())())(), ((Z_0) \longrightarrow$
$(q_0,))())(), (((Z_0) \longrightarrow (q_0,)())(), ((Z_0) \longrightarrow (q_0, ())(), (Z_0) \longrightarrow$
$(q_0,))(), ((Z_0) \longrightarrow (q_0,)(), (Z_0) \longrightarrow (q_0, (), Z_0) \longrightarrow$
$(q_0,), (Z_0) \longrightarrow (q_0, \varepsilon, Z_0) \longrightarrow (q_1, \varepsilon, Z_0)$ *accept*

w such that $(q_0,w,z_0) \longrightarrow^* (q,\varepsilon,\varepsilon)$ *for some state q*

Accept: empty stack or final state

P' simulates P:

- When P accepts, P' empties its stack
- P' avoids emptying stack prematurely

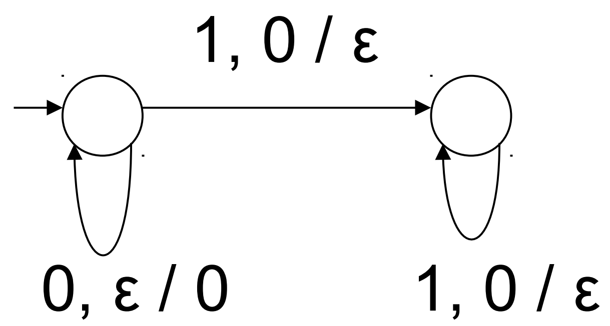
  (use special stack-bottom marker)

P simulates P':

- If P' would accept by empty stack...
- ...then P moves to accepting state

Accept by empty stack

$L = \{0^n 1^n \mid n \geq 0\}$



initialization: empty stack
special notation: push

CFG (V,Σ,R,S) to PDA
(top-down)

$(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Q: $\{q\}$

Σ: Σ

Γ: $V \cup \Sigma$

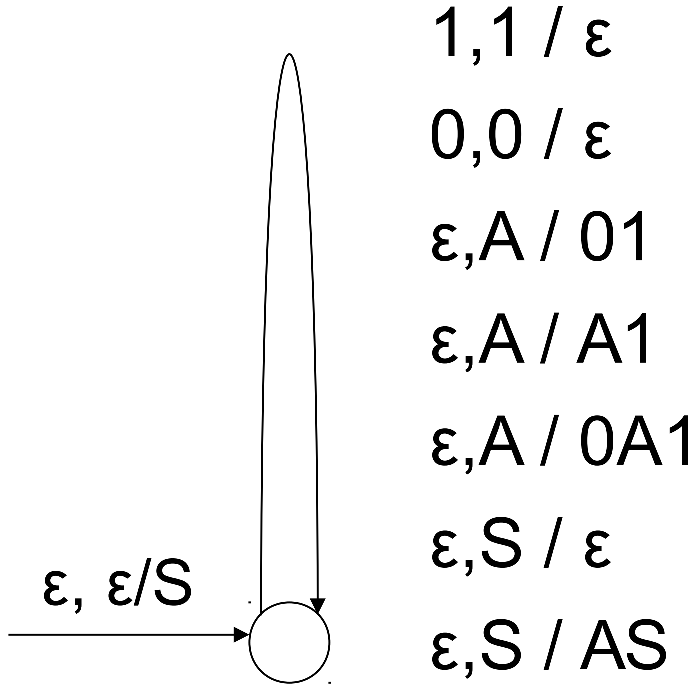
$\delta(q,\varepsilon,A) = \{(q,\alpha) \mid A \rightarrow \alpha \text{ in } R\}$ *rewrite variables (store on stack)*

$\delta(q,a,a) = \{(q,\varepsilon) \mid a \text{ in } \Sigma\}$ *match input (with derivation on stack)*

q_0: q

z_0: S

F: not applicable; *accept by empty stack*



$$G = (\{S,A\}, \{0,1\}, \{S \rightarrow AS|\varepsilon, A \rightarrow 0A1|A1|01\}, S)$$

PDA to CFG

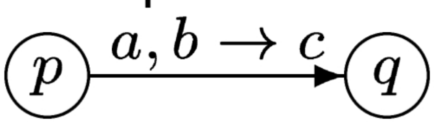
$P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

F: $\{q_a\}$ (if not, then $\delta(q,\varepsilon,\varepsilon) = \{(q_a,\varepsilon)\}$ for all q in F)

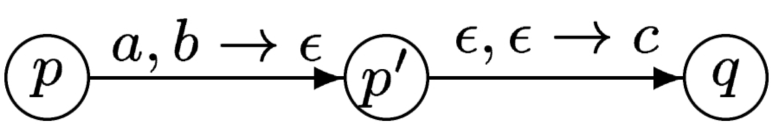
P empties its stack before accepting (modify δ if necessary)

δ: a *single* symbol is either pushed or popped

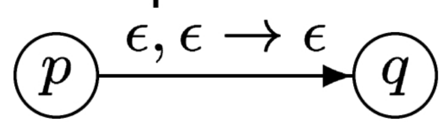
Replace



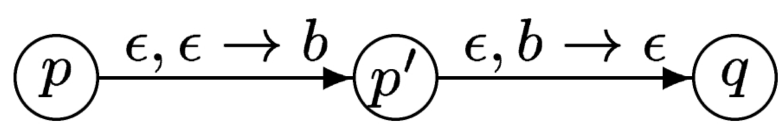
by



Replace



by



G has (for each pair p,q of states of P) a variable A_{pq} that generates all strings which can take P from p with an empty stack to q with an empty stack

G has start symbol $A_{q_0\, q_a}$; the language of G is therefore the language accepted by P

On input w, P must first push (since each move is either push or pop, and the stack is initially empty) and P must have last move pop (because the stack returns to empty)

During the computation on w, either:

a) The first symbol pushed is the last popped

b) The stack is emptied during the computation (when the first symbol is popped)

Case a: the computation of P on input w is simulated by $A_{pq} \rightarrow a\, A_{rs}\, b$ where a is the input symbol read at the first move (from state p), b is the symbol read at the last move (to state q), r is the state following p, and s is the spate preceding q

Case b: the computation of P on input w is simulated by $A_{pq} \rightarrow A_{pr}\, A_{rq}$ where r is the state where the stack becomes empty

BNF Notations

- Variables: left-hand-side or delimited by < >
- Terminals: boldface, underlined, or quoted
- ::= or = abbreviates $\rightarrow$
- . . . abbreviates "one or more" (as does +)

  (replace β... with variable V, add productions V $\rightarrow$ Vβ|β)
- * abbreviates zero or more

  (replace β* with variable V', add productions V' $\rightarrow$ V'β|ε)
- Symbols delimited by [ ] are optional

  (replace [β] with variable V'', add productions V'' $\rightarrow$ β|ε)
- Symbols delimited by { } or ( ) are treated as a unit

  (replace {β} with variable V''', add productions V''' $\rightarrow$ β)

Syntax diagrams