Einführung in die Computerlinguistik

Context-Free Grammars (CFG)

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2021

CFG (1)

In contrast to regular grammars, CFGs allow any combination of non-terminal and terminal symbols in the righthand sides of productions.

CFG $G_{telescope}$

- Nonterminals {S, NP, VP, PP, D, N, V, P}
- Terminals {man, girl, the, John, Mary, telescope, with, saw}
- Start symbol S
- Productions

S $\rightarrow$ NP VP	NP $\rightarrow$ D N	N $\rightarrow$ N PP
VP $\rightarrow$ VP PP	VP $\rightarrow$ V NP	PP $\rightarrow$ P NP
N $\rightarrow$ man	N $\rightarrow$ girl	N $\rightarrow$ telescope
D $\rightarrow$ the	NP $\rightarrow$ John	NP $\rightarrow$ Mary
P $\rightarrow$ with	V $\rightarrow$ saw	

CFG (2)

Example continued

$S \rightarrow NP\ VP$	$NP \rightarrow D\ N$	$N \rightarrow N\ PP$	$VP \rightarrow VP\ PP$
$VP \rightarrow V\ NP$	$PP \rightarrow P\ NP$	$N \rightarrow$ man	$N \rightarrow$ girl
$N \rightarrow$ telescope	$D \rightarrow$ the	$NP \rightarrow$ John	$NP \rightarrow$ Mary
$P \rightarrow$ with	$V \rightarrow$ saw		

Sentences one can generate with this grammar:

(1) John saw Mary

(2) John saw the girl

(3) the man with the telescope saw John

(4) John saw the girl with the telescope

…

CFG (3)

CFG

A context-free grammar (CFG) is a tuple $G = \langle N, T, P, S \rangle$ such that

- N and T are disjoint alphabets, the nonterminals and terminals,
- $S \in N$ is the start symbol, and
- P is a set of productions of the form

$$A \to \beta$$

  with $A \in N, \beta \in (N \cup T)^*$.

Any $\beta \in (N \cup T)^*$ with $S \overset{*}{\Rightarrow} \beta$ is called a sentential form.

CFG (4)

CFGs

- CFG $G_{anbn} = \langle \{S\}, \{a, b\}, P, S \rangle$ with productions

  $\mathrm{S} \rightarrow \mathrm{aSb} \quad \mathrm{S} \rightarrow \varepsilon$

  generates the languge $\{a^n b^n \mid n \geq 0\}$

CFG (4)

CFGs

- CFG $G_{anbn} = \langle \{S\}, \{a, b\}, P, S \rangle$ with productions

  $\mathrm{S} \rightarrow \mathrm{aSb} \quad \mathrm{S} \rightarrow \varepsilon$

  generates the languge $\{a^n b^n \mid n \geq 0\}$

- CFG $G_{a,b} = \langle \{S, A, B\}, \{a, b\}, P, S \rangle$ with productions

  $$\begin{array}{lll} \mathrm{S} \rightarrow \mathrm{aB} & \mathrm{S} \rightarrow \mathrm{bA} & \\ \mathrm{A} \rightarrow \mathrm{a} & \mathrm{A} \rightarrow \mathrm{aS} & \mathrm{A} \rightarrow \mathrm{bAA} \\ \mathrm{B} \rightarrow \mathrm{b} & \mathrm{B} \rightarrow \mathrm{bS} & \mathrm{B} \rightarrow \mathrm{aBB} \end{array}$$

  generates the languge $\{w \mid w \in \{a, b\}^+, |w|_a = |w|_b\}$

CFG (5)

Parse tree

A tree t is a parse tree for a CFG $G = \langle N, T, P, S \rangle$ iff

- each node in t is labeled with a $x \in N \cup T \cup \{\epsilon\}$;
- the root label is S;
- if there is a node with label A that has n daughters labeled (from left to right) $x_1, \ldots, x_n$, then $A \rightarrow x_1 \ldots x_n \in P$.
- if a node has label ϵ, it is a leaf and the unique daughter of its mother node.

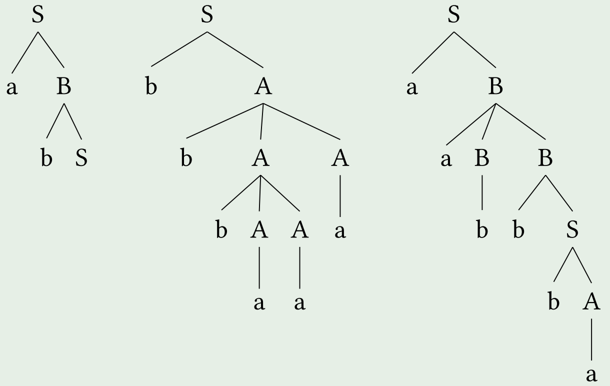
$S \stackrel{*}{\Rightarrow} \alpha$ in G iff there is a parse tree for G with yield α.

CFG (6)

Parse trees for $G_{a,b}$

$S \to aB \quad S \to bA \quad A \to a \quad A \to aS \quad A \to bAA$

$B \to b \quad B \to bS \quad B \to aBB$



CFG (7)

String language, derivation, tree language

- Let $G = \langle N, T, P, S \rangle$ be a CFG. The string language $L(G)$ of G is the set $\{w \in T^* \mid S \stackrel{*}{\Rightarrow} w\}$ where

CFG (7)

String language, derivation, tree language

- Let $G = \langle N, T, P, S \rangle$ be a CFG. The string language $L(G)$ of G is the set $\{w \in T^* \mid S \stackrel{*}{\Rightarrow} w\}$ where
  - for $w, w' \in (N \cup T)^*$: $w \Rightarrow w'$ iff there is a $A \to \beta \in P$ and there are $v, u \in (N \cup T)^*$ such that $w = vAu$ and $w' = v\beta u$.

CFG (7)

String language, derivation, tree language

- Let $G = \langle N, T, P, S \rangle$ be a CFG. The string language $L(G)$ of G is the set $\{w \in T^* \mid S \stackrel{*}{\Rightarrow} w\}$ where
  - for $w, w' \in (N \cup T)^*$: $w \Rightarrow w'$ iff there is a $A \to \beta \in P$ and there are $v, u \in (N \cup T)^*$ such that $w = vAu$ and $w' = v\beta u$.
  - $\stackrel{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$.

CFG (7)

String language, derivation, tree language

- Let $G = \langle N, T, P, S \rangle$ be a CFG. The string language $L(G)$ of G is the set $\{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$ where
  - for $w, w' \in (N \cup T)^*$: $w \Rightarrow w'$ iff there is a $A \rightarrow \beta \in P$ and there are $v, u \in (N \cup T)^*$ such that $w = vAu$ and $w' = v\beta u$.
  - $\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$.

- A derivation of a word $w \in T^*$ is a sequence $S \Rightarrow \alpha_1 \cdots \Rightarrow w$ of derivation steps leading to w.

CFG (7)

String language, derivation, tree language

- Let $G = \langle N, T, P, S \rangle$ be a CFG. The string language $L(G)$ of G is the set $\{w \in T^* \mid S \stackrel{*}{\Rightarrow} w\}$ where
  - for $w, w' \in (N \cup T)^*$: $w \Rightarrow w'$ iff there is a $A \to \beta \in P$ and there are $v, u \in (N \cup T)^*$ such that $w = vAu$ and $w' = v\beta u$.
  - $\stackrel{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$.
- A derivation of a word $w \in T^*$ is a sequence $S \Rightarrow \alpha_1 \cdots \Rightarrow w$ of derivation steps leading to w.
- The tree language is the set of all parse trees with all leaves labelled with $a \in T \cup \{\varepsilon\}$.

CFG (8)

For a single parse tree, there might be more than one corresponding derivation.

Leftmost/rightmost derivation

A derivation is called a

CFG (8)

For a single parse tree, there might be more than one corresponding derivation.

Leftmost/rightmost derivation

A derivation is called a

- leftmost derivation iff, in each derivation step, a production is applied to the leftmost non-terminal of the already derived sentential form.

CFG (8)

For a single parse tree, there might be more than one corresponding derivation.

Leftmost/rightmost derivation

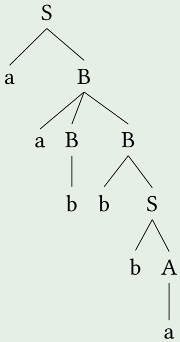
A derivation is called a

- leftmost derivation iff, in each derivation step, a production is applied to the leftmost non-terminal of the already derived sentential form.
- rightmost derivation iff, in each derivation step, a production is applied to the rightmost non-terminal of the already derived sentential form.

CFG (9)

Derivations

Take the grammar $G_{a,b}$ and the following parse tree:



Leftmost derivation:

$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabB$
$\Rightarrow aabbS \Rightarrow aabbbA \Rightarrow aabbba$

Rightmost derivation:

$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aaBbS$
$\Rightarrow aaBbbA \Rightarrow aaBbba \Rightarrow aabbba$

CFG (10)

For a single word w, there might be more than one parse tree:

Ambiguous CFG

CFG (10)

For a single word w, there might be more than one parse tree:

Ambiguous CFG

- A CFG giving more than one parse tree for some word w is called ambiguous.

  Example: $G_{telescope}$ with w = John saw the man with the telescope

  $G_{a,b}$ with $w = aabbab$

CFG (10)

For a single word w, there might be more than one parse tree:

Ambiguous CFG

- A CFG giving more than one parse tree for some word w is called ambiguous.

  Example: $G_{telescope}$ with $w =$
  John saw the man with the telescope
  $G_{a,b}$ with $w = aabbab$

- A CFL L is called inherently ambiguous if each CFG G with $L = L(G)$ is ambiguous.

  Example:
  $\{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

PDA (1)

A push-down automaton is a FSA with an additional stack. The moves of the automaton depend on

- the current state,
- the next input symbol, and
- the topmost stack symbol.

PDA (1)

A push-down automaton is a FSA with an additional stack. The moves of the automaton depend on

- the current state,
- the next input symbol, and
- the topmost stack symbol.

Each move consists of

- changing state,
- popping the topmost symbol from the stack, and
- pushing a new sequence of symbols on the stack.

PDA (2)

PDA

Example: Automaton that

PDA (2)

PDA

Example: Automaton that

- starts with q_1 and stack $\#$,

PDA (2)

PDA

Example: Automaton that

- starts with q_1 and stack $\#$,
- in q_1: pushes A on the stack for an input symbol a,
- in q_1: leaves stack unchanged and goes to q_2 for an input symbol c,

PDA (2)

PDA

Example: Automaton that

- starts with q_1 and stack $\#$,
- in q_1: pushes A on the stack for an input symbol a,
- in q_1: leaves stack unchanged and goes to q_2 for an input symbol c,
- in q_2: pops an A from the stack for input symbol b,
- in q_2: moves to q_3 if the top of the stack is $\#$

PDA (2)

PDA

Example: Automaton that

- starts with q_1 and stack $\#$,
- in q_1: pushes A on the stack for an input symbol a,
- in q_1: leaves stack unchanged and goes to q_2 for an input symbol c,
- in q_2: pops an A from the stack for input symbol b,
- in q_2: moves to q_3 if the top of the stack is $\#$

The automaton accepts all words that allow to end up in q_3.

PDA (2)

PDA

Example: Automaton that

- starts with q_1 and stack $\#$,
- in q_1: pushes A on the stack for an input symbol a,
- in q_1: leaves stack unchanged and goes to q_2 for an input symbol c,
- in q_2: pops an A from the stack for input symbol b,
- in q_2: moves to q_3 if the top of the stack is $\#$

The automaton accepts all words that allow to end up in q_3.

Language $\{a^n c b^n \mid n \geq 0\}$

PDA (3)

In general, PDAs are non-deterministic, since a given state, input symbol and topmost stack symbol can allow for more than one move.

PDA (3)

In general, PDAs are non-deterministic, since a given state, input symbol and topmost stack symbol can allow for more than one move.

In contrast to FSA, the deterministic version of the automaton is not equivalent to the non-deterministic one: There are languages that are accepted by a non-deterministic PDA but not by any deterministic PDA.

PDA (3)

In general, PDAs are non-deterministic, since a given state, input symbol and topmost stack symbol can allow for more than one move.

In contrast to FSA, the deterministic version of the automaton is not equivalent to the non-deterministic one: There are languages that are accepted by a non-deterministic PDA but not by any deterministic PDA.

CFLs are the languages accepted by (non-deterministic) PDAs.

PDA (4)

PDA

A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ with

PDA (4)

PDA

A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ with

- Q is a finite set of states.

PDA (4)

PDA

A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ with

- Q is a finite set of states.
- Σ is a finite set, the input alphabet.

PDA (4)

PDA

A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ with

- Q is a finite set of states.
- Σ is a finite set, the input alphabet.
- Γ is a finite set, the stack alphabet.

PDA (4)

PDA

A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ with

- Q is a finite set of states.
- Σ is a finite set, the input alphabet.
- Γ is a finite set, the stack alphabet.
- $q_0 \in Q$ is the initial state.

PDA (4)

PDA

A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ with

- Q is a finite set of states.
- Σ is a finite set, the input alphabet.
- Γ is a finite set, the stack alphabet.
- $q_0 \in Q$ is the initial state.
- $Z_0 \in \Gamma$ is the initial stack symbol.

PDA (4)

PDA

A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ with

- Q is a finite set of states.
- Σ is a finite set, the input alphabet.
- Γ is a finite set, the stack alphabet.
- $q_0 \in Q$ is the initial state.
- $Z_0 \in \Gamma$ is the initial stack symbol.
- $F \subseteq Q$ is the set of final states.

PDA (4)

PDA

A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ with

- Q is a finite set of states.
- Σ is a finite set, the input alphabet.
- Γ is a finite set, the stack alphabet.
- $q_0 \in Q$ is the initial state.
- $Z_0 \in \Gamma$ is the initial stack symbol.
- $F \subseteq Q$ is the set of final states.
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_{fin}(Q \times \Gamma^*)$ is the transition function. ($\mathcal{P}_{fin}(X)$ is the set of finite subsets of X).

PDA (4)

PDA

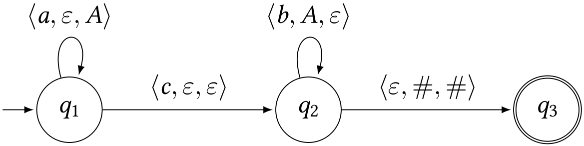
A push-down automaton (PDA) M is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ with

- Q is a finite set of states.
- Σ is a finite set, the input alphabet.
- Γ is a finite set, the stack alphabet.
- $q_0 \in Q$ is the initial state.
- $Z_0 \in \Gamma$ is the initial stack symbol.
- $F \subseteq Q$ is the set of final states.
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_{fin}(Q \times \Gamma^*)$ is the transition function. ($\mathcal{P}_{fin}(X)$ is the set of finite subsets of X).

Equivalently, one can even define δ as
$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \to \mathcal{P}_{fin}(Q \times \Gamma^*)$.

PDA (5)

PDAs can also be drawn as graphs:



An edge label $\langle a, A, B\rangle$ signifies that a is read, A is popped from the stack and B is pushed on the stack.

In other words, $\langle q_2, B\rangle \in \delta(q_1, a, A)$ iff there is an edge from q_1 to q_2 labeled $\langle a, A, B\rangle$.

PDA (6)

Instantaneous description

An instantaneous description of a PDA is a triple (q, w, γ) with

- $q \in Q$ is the current state of the automaton,
- $w \in \Sigma^*$ is the remaining part of the input string, and
- $\gamma \in \Gamma^*$ is the current stack.

PDA (6)

Instantaneous description

An instantaneous description of a PDA is a triple (q, w, γ) with

- $q \in Q$ is the current state of the automaton,
- $w \in \Sigma^*$ is the remaining part of the input string, and
- $\gamma \in \Gamma^*$ is the current stack.

$(q, aw, Z\alpha) \vdash (q', w, \beta\alpha)$ iff $\langle q', \beta \rangle \in \delta(q, a, Z)$ for all $q, q' \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $w \in \Sigma^*$, $Z \in \Gamma$, $\alpha, \beta \in \Gamma^*$.

PDA (6)

Instantaneous description

An instantaneous description of a PDA is a triple (q, w, γ) with

- $q \in Q$ is the current state of the automaton,
- $w \in \Sigma^*$ is the remaining part of the input string, and
- $\gamma \in \Gamma^*$ is the current stack.

$(q, aw, Z\alpha) \vdash (q', w, \beta\alpha)$ iff $\langle q', \beta \rangle \in \delta(q, a, Z)$ for all $q, q' \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $w \in \Sigma^*$, $Z \in \Gamma$, $\alpha, \beta \in \Gamma^*$.

$\overset{*}{\vdash}$ is the reflexive transitive closure of $\vdash$.

PDA (7)

There are two alternatives for the definition of the language accepted by a PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$:

Language accepted by an PDA

PDA (7)

There are two alternatives for the definition of the language accepted by a PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$:

Language accepted by an PDA

- The language accepted by M with a final state is

  $L(M) := \{ w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q_f, \epsilon, \gamma) \text{ for a } q_f \in F \text{ and a } \gamma \in \Gamma^* \}$

PDA (7)

There are two alternatives for the definition of the language accepted by a PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$:

Language accepted by an PDA

- The language accepted by M with a final state is

  $L(M) := \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q_f, \epsilon, \gamma)$ for a $q_f \in F$ and a $\gamma \in \Gamma^*\}$
- The language accepted by M with an empty stack is

  $N(M) := \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \epsilon)$ for a $q \in Q\}$

PDA (7)

There are two alternatives for the definition of the language accepted by a PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$:

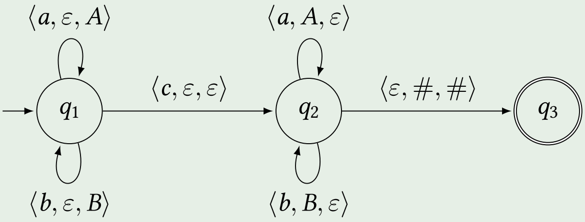
Language accepted by an PDA

- The language accepted by M with a final state is
  $L(M) := \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q_f, \epsilon, \gamma)$ for a $q_f \in F$ and a $\gamma \in \Gamma^*\}$
- The language accepted by M with an empty stack is
  $N(M) := \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \epsilon)$ for a $q \in Q\}$

The two modes of acceptance are equivalent, i.e., for each language L: there is a PDA M_1 with $L = L(M_1)$ iff there is a PDA M_2 with $L = N(M_2)$.

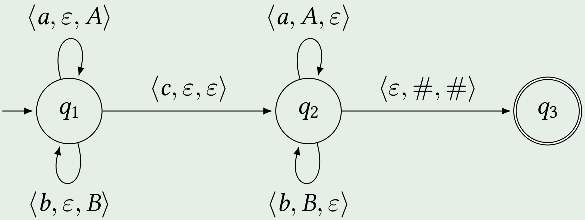
PDA (8)

PDA M for $L(M) = \{wcw^R \mid w \in \{a, b\}^*\}$



PDA (8)

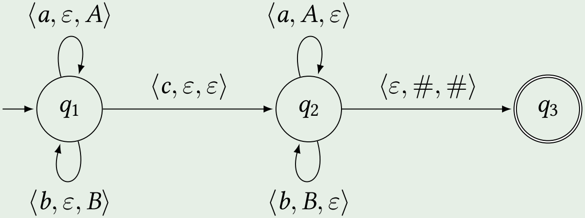
PDA M for $L(M) = \{wcw^R \mid w \in \{a, b\}^*\}$



- $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{\#, A, B\}$.
  $q_0 = q_1$, $Z_0 = \#$, $F = \{q_3\}$.

PDA (8)

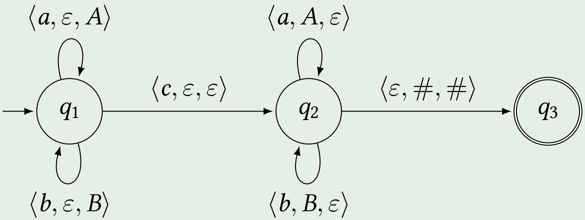
PDA M for $L(M) = \{wcw^R \mid w \in \{a, b\}^*\}$



- $Q = \{q_1, q_2, q_3\}, \Sigma = \{a, b, c\}, \Gamma = \{\#, A, B\}$.
  $q_0 = q_1, Z_0 = \#, F = \{q_3\}$.
- $\delta(q_1, a, \varepsilon) = \{\langle q_1, A\rangle\} \quad \delta(q_1, b, \varepsilon) = \{\langle q_1, B\rangle\}$

PDA (8)

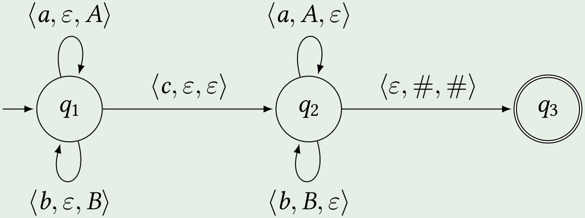
PDA M for $L(M) = \{wcw^R \mid w \in \{a, b\}^*\}$



- $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{\#, A, B\}$.
  $q_0 = q_1$, $Z_0 = \#$, $F = \{q_3\}$.
- $\delta(q_1, a, \varepsilon) = \{\langle q_1, A \rangle\}$ $\quad \delta(q_1, b, \varepsilon) = \{\langle q_1, B \rangle\}$
- $\delta(q_1, c, \varepsilon) = \{\langle q_2, \varepsilon \rangle\}$

PDA (8)

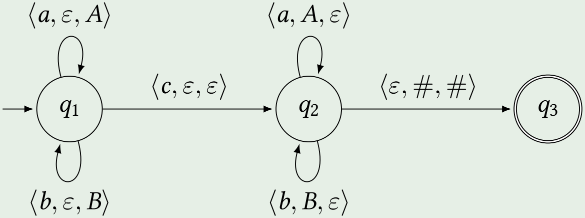
PDA M for $L(M) = \{wcw^R \mid w \in \{a, b\}^*\}$



- $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{\#, A, B\}$.
  $q_0 = q_1$, $Z_0 = \#$, $F = \{q_3\}$.
- $\delta(q_1, a, \varepsilon) = \{\langle q_1, A\rangle\}$ $\quad \delta(q_1, b, \varepsilon) = \{\langle q_1, B\rangle\}$
- $\delta(q_1, c, \varepsilon) = \{\langle q_2, \varepsilon\rangle\}$
- $\delta(q_2, a, A) = \{\langle q_2, \epsilon\rangle\}$ $\quad \delta(q_2, b, B) = \{\langle q_2, \epsilon\rangle\}$

PDA (8)

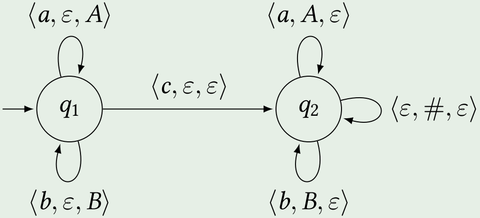
PDA M for $L(M) = \{wcw^R \mid w \in \{a, b\}^*\}$



- $Q = \{q_1, q_2, q_3\}, \Sigma = \{a, b, c\}, \Gamma = \{\#, A, B\}.$
  $q_0 = q_1, Z_0 = \#, F = \{q_3\}.$
- $\delta(q_1, a, \varepsilon) = \{\langle q_1, A\rangle\} \quad \delta(q_1, b, \varepsilon) = \{\langle q_1, B\rangle\}$
- $\delta(q_1, c, \varepsilon) = \{\langle q_2, \varepsilon\rangle\}$
- $\delta(q_2, a, A) = \{\langle q_2, \epsilon\rangle\} \quad \delta(q_2, b, B) = \{\langle q_2, \epsilon\rangle\}$
- $\delta(q_2, \epsilon, \#) = \{\langle q_3, \#\rangle\}$

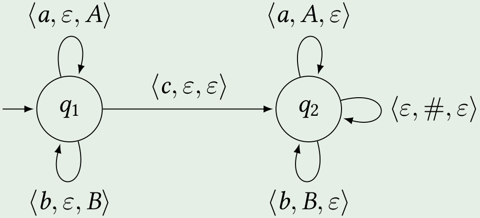
PDA (9)

PDA M for $N(M) = \{wcw^R \mid w \in \{a, b\}^*\}$



PDA (9)

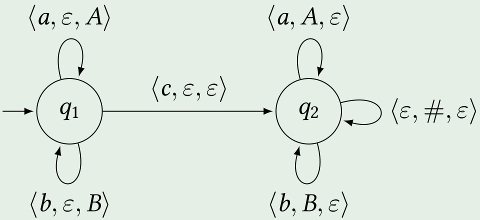
PDA M for $N(M) = \{wcw^R \mid w \in \{a,b\}^*\}$



- $Q = \{q_1, q_2\}, \Sigma = \{a, b, c\}, \Gamma = \{\#, A, B\}$.
  $q_0 = q_1, Z_0 = \#, F = \emptyset$.
- $\delta(q_1, a, \varepsilon) = \{\langle q_1, A\rangle\} \quad \delta(q_1, b, \varepsilon) = \{\langle q_1, B\rangle\}$
- $\delta(q_1, c, \varepsilon) = \{\langle q_2, \varepsilon\rangle\}$
- $\delta(q_2, a, A) = \{\langle q_2, \epsilon\rangle\} \quad \delta(q_2, b, B) = \{\langle q_2, \epsilon\rangle\}$

PDA (9)

PDA M for $N(M) = \{wcw^R \mid w \in \{a, b\}^*\}$



- $Q = \{q_1, q_2\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{\#, A, B\}$.
  $q_0 = q_1$, $Z_0 = \#$, $F = \emptyset$.
- $\delta(q_1, a, \varepsilon) = \{\langle q_1, A\rangle\}$ $\quad \delta(q_1, b, \varepsilon) = \{\langle q_1, B\rangle\}$
- $\delta(q_1, c, \varepsilon) = \{\langle q_2, \varepsilon\rangle\}$
- $\delta(q_2, a, A) = \{\langle q_2, \epsilon\rangle\}$ $\quad \delta(q_2, b, B) = \{\langle q_2, \epsilon\rangle\}$
- $\delta(q_2, \epsilon, \#) = \{\langle q_2, \epsilon\rangle\}$

PDA (10)

DPDA

A PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ is a deterministic PDA (DPDA) iff

- for all $q \in Q, Z \in \Gamma, a \in \Sigma \cup \{\epsilon\}$: $|\delta(q, a, Z)| \leq 1$, and
- for all $q \in Q, Z \in \Gamma$: if $\delta(q, \epsilon, Z) \neq \emptyset$, then $\delta(q, a, Z) = \emptyset$ for all $a \in \Sigma$.

The preceding example was a DPDA.

PDA (10)

DPDA

A PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ is a deterministic PDA (DPDA) iff

- for all $q \in Q, Z \in \Gamma, a \in \Sigma \cup \{\epsilon\}$: $|\delta(q, a, Z)| \leq 1$, and
- for all $q \in Q, Z \in \Gamma$: if $\delta(q, \epsilon, Z) \neq \emptyset$, then $\delta(q, a, Z) = \emptyset$ for all $a \in \Sigma$.

The preceding example was a DPDA.

The class of languages accepted by DPDAs is smaller than the class accepted by (non-deterministic) PDAs.

Example of a language that requires a non-determinstic PDA: $\{ww^R \mid w \in \{a, b\}^*\}$.

PDA and CFG (1)

For each CFL L, there is a PDA M with $L = N(M)$.

PDA and CFG (1)

For each CFL L, there is a PDA M with $L = N(M)$.

Construction: Assume that $\epsilon \notin L$. $L = L(G)$ for a CFG $G = \langle N, T, P, S \rangle$ in Greibach-Normal Form (GNF).
This means that all productions have the form $A \rightarrow a\gamma$ with $A \in N, a \in T, \gamma \in N^*$.
Every CFG can be transformed into an equivalent CFG in GNF.

PDA and CFG (1)

For each CFL L, there is a PDA M with $L = N(M)$.

Construction: Assume that $\epsilon \notin L$. $L = L(G)$ for a CFG
$G = \langle N, T, P, S \rangle$ in Greibach-Normal Form (GNF).
This means that all productions have the form $A \to a\gamma$ with
$A \in N, a \in T, \gamma \in N^*$.
Every CFG can be transformed into an equivalent CFG in GNF.

Equivalent PDA:
$M = \langle \{q\}, T, N, \delta, q, S, \emptyset \rangle$ with $\langle q, \gamma \rangle \in \delta(q, a, A)$ iff $A \to a\gamma \in P$.

PDA and CFG (1)

For each CFL L, there is a PDA M with $L = N(M)$.

Construction: Assume that $\epsilon \notin L$. $L = L(G)$ for a CFG $G = \langle N, T, P, S \rangle$ in Greibach-Normal Form (GNF).
This means that all productions have the form $A \to a\gamma$ with $A \in N, a \in T, \gamma \in N^*$.
Every CFG can be transformed into an equivalent CFG in GNF.

Equivalent PDA:
$M = \langle \{q\}, T, N, \delta, q, S, \emptyset \rangle$ with $\langle q, \gamma \rangle \in \delta(q, a, A)$ iff $A \to a\gamma \in P$.

The automaton simulates leftmost derivations in G.

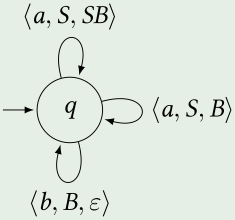
PDA and CFG (2)

From CFG to PDA

Take the CFG $G_{anbn} = \langle \{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow ab\}, S \rangle$.

Equivalent CFG in GNF:
$G'_{anbn} = \langle \{S, B\}, \{a, b\}, \{S \rightarrow aSB, S \rightarrow aB, B \rightarrow b\}, S \rangle$.

Equivalent PDA: $M = \langle \{q\}, \{a, b\}, \{S, B\}, \delta, q, S, \emptyset \rangle$ with acceptance with empty stack:



PDA and CFG (3)

For each PDA M with $L = N(M)$: L is a context-free language.

PDA and CFG (3)

For each PDA M with $L = N(M)$: L is a context-free language.

Construction of equivalent CFG for given PDA
$M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$:

- nonterminals: S and all $[q_1, Z, q_2]$ with $q_1, q_2 \in Q$, $Z \in \Gamma$.
- productions: $S \rightarrow [q_0, Z_0, q]$ for every $q \in Q$, and
  $[q, A, q_{m+1}] \rightarrow a[q_1, B_1, q_2], \ldots, [q_m, B_m, q_{m+1}]$ for $q, q_1, \ldots, q_{m+1} \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $A, B_1, \ldots, B_m \in \Gamma$ such that $\langle q_1, B_1 \ldots B_m \rangle \in \delta(q, a, A)$
  $[q, A, q_1] \rightarrow a$ if $\langle q_1, \epsilon \rangle \in \delta(q, a, A)$.

PDA and CFG (3)

For each PDA M with $L = N(M)$: L is a context-free language.

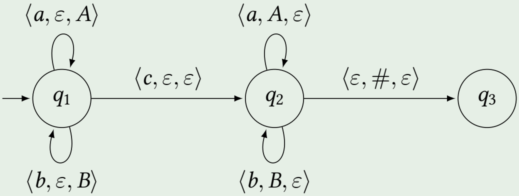
Construction of equivalent CFG for given PDA
$M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$:

- nonterminals: S and all $[q_1, Z, q_2]$ with $q_1, q_2 \in Q$, $Z \in \Gamma$.
- productions: $S \to [q_0, Z_0, q]$ for every $q \in Q$, and
  $[q, A, q_{m+1}] \to a[q_1, B_1, q_2], \ldots, [q_m, B_m, q_{m+1}]$ for $q, q_1, \ldots, q_{m+1} \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $A, B_1, \ldots, B_m \in \Gamma$ such that $\langle q_1, B_1 \ldots B_m \rangle \in \delta(q, a, A)$
  $[q, A, q_1] \to a$ if $\langle q_1, \epsilon \rangle \in \delta(q, a, A)$.

$[q_1, A, q_2] \overset{*}{\Rightarrow} w$ iff $(q_1, w, A) \overset{*}{\vdash} (q_2, \epsilon, \epsilon)$.

PDA and CFG (4)

From PDA to CFG



Productions of the equivalent CFG (only useful productions listed):

$S \rightarrow [q_1, \#, q_3] \quad [q_1, \#, q_3] \rightarrow c[q_2, \#, q_3] \quad [q_2, \#, q_3] \rightarrow \varepsilon$

$[q_1, \#, q_3] \rightarrow a[q_1, A, q_2][q_2, \#, q_3] \quad [q_1, \#, q_3] \rightarrow b[q_1, B, q_2][q_2, \#, q_3]$

$[q_1, A, q_2] \rightarrow a[q_1, A, q_2][q_2, A, q_2] \quad [q_1, A, q_2] \rightarrow b[q_1, B, q_2][q_2, A, q_2]$

$[q_1, B, q_2] \rightarrow a[q_1, A, q_2][q_2, B, q_2] \quad [q_1, A, q_2] \rightarrow b[q_1, B, q_2][q_2, B, q_2]$

$[q_1, A, q_2] \rightarrow c[q_2, A, q_2] \quad [q_1, B, q_2] \rightarrow c[q_2, B, q_2]$

$[q_2, A, q_2] \rightarrow a \quad [q_2, B, q_2] \rightarrow b$

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation.* Addison Wesley.