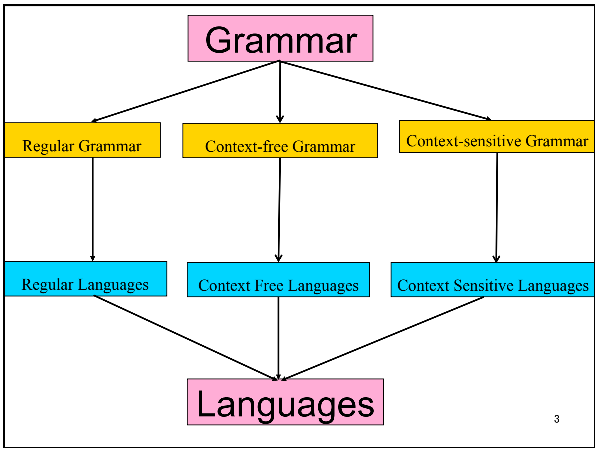
Automata and Languages

Prof. Mohamed Hamada

Software Engineering Lab.
The University of Aizu
Japan

Content



Context Free Languages (CFL)

Definition

L is a **Context Free Language** if and only if there is a context free grammar G=(V,T,S,P) such that:

$$L = L(G) = \{ w \in T^* : S \Rightarrow^* w \}$$

Context Free Languages (CFL)

Why Context Free Languages

Context-free languages allow us to describe languages that are nonregular like $\{ 0^n1^n : n \geq 0\}$.

CFLs are complex enough to give us a model for natural languages (cf. Noam Chomsky) and programming languages. The theory of CFLs is very closely related to the problem of “parsing” a computer program.

Later we will see that CFLs are the languages that can be recognized by automata that have one single stack:

$\{ 0^n1^n : n \geq 0 \}$ is a CFL

$\{ 0^n1^n0^n : n \geq 0 \}$ is not a CFL

5

Context Free Languages (CFL)

Properties of CFL:

If L_1 and L_2 are Context Free Languages then:

1. **The language $L_1 \cup L_2$ is context free**
2. **The language $L_1 . L_2$ is context free**
3. **The language L_1^* and L_2^* are context free**
4. **The language $L_1 \cap L_2$ may be NOT a context free**
5. **The languages $\bar{L}_1$ and $\bar{L}_2$ may be NOT a context free**

6

Context Free Languages (CFL)

Exercise

Consider the Context Free Languages:

$L_1 = \{a^nb^nc^m: n \geq 0, m \geq 0\}$

$L_2 = \{a^nb^mc^m: n \geq 0, m \geq 0\}$

1. **Show that the languages $L_1 \cup L_2$, $L_1 . L_2$ and L_1^* are context free?**

2. **Show that the languages $L_1 \cap L_2$ and $\bar{L}_1$ are NOT context free**

7

Pumping Lemma for CFL

Let L be a context-free language

Then there exists P such that
if $w \in L$ and $|w| \geq P$

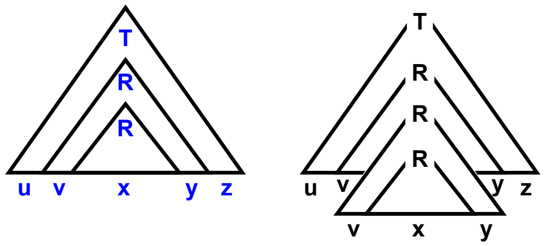
then $w = uvxyz$, where:

1. $|vy| > 0$
2. $|vxy| \leq P$
3. $uv^ixy^iz \in L$ for any $i \geq 0$

8

Pumping Lemma for CFL

Idea: If w is long enough, then any parse tree for w must have a path that contains a variable more than once



9

Pumping Lemma for CFL

Formal Proof: Let b be the maximum number of symbols on the right-hand side of a rule

If the height of a parse tree is h, the length of the string generated is at most: b^h

Let |V| be the number of variables in G

Define $P = b^{|V|+2}$

Let w be a string of length at least P

Let T be the parse tree for w with the smallest number of nodes.

T must have height at least |V|+2

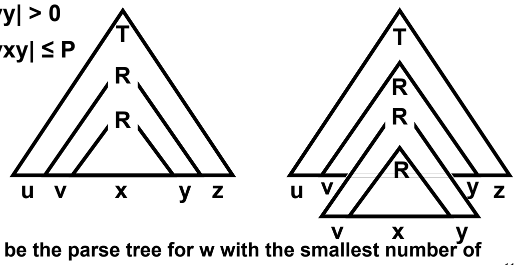
10

Pumping Lemma for CFL

The longest path in T must have ≥ |V|+1 variables

Select R to be the variable that repeats among the lowest |V|+1 variables

1. |vy| > 0
2. |vxy| ≤ P



Let T be the parse tree for w with the smallest number of nodes. T must have height at least |V|+2

11

Pushdown Automata

12

Push Down Automata (PDA)

Pushdown automata are for context-free languages while finite automata are for regular languages.

Big difference though: PDAs have to be nondeterministic (deterministic PDAs are not powerful enough).

PDAs are automata that have a single *stack* as memory.

13

Push Down Automata (PDA)

Definition

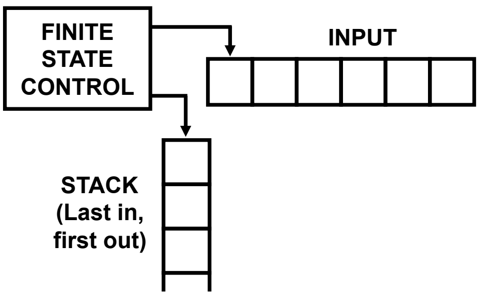
A **Nondeterministic Pushdown Automaton, Acceptor** (**NPDA**) M is defined by a tuple $(Q,\Sigma,\Gamma,\delta,q_0,z,F)$:

- Q is the finite set of internal states
- Σ is the finite input alphabet
- Γ is the finite **stack alphabet**
- $\delta: Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \mathbf{P}(Q \times \Gamma^*)$ is the **transition function** of M, where each δ-value is a finite set
- $q_0 \in Q$ is the starting state of M
- $z \in \Gamma$ is the **stack start symbol**
- $F \subseteq Q$ are the accepting, final states of M

It is the transition function δ that we need to understand…

14

Push Down Automata (PDA)

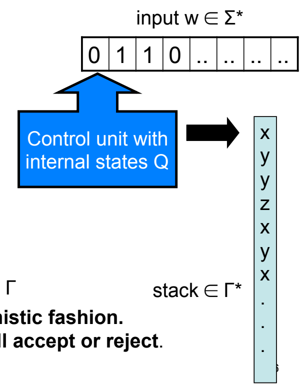


15

Push Down Automata (PDA)

The PDA M reads the input $w \in \Sigma^*$ from left to right.
Depending on
input $w_j \in \Sigma \cup \{\lambda\}$,
stack symbol $s_j \in \Gamma$,
and state $q_k \in Q$
…the PDA M
jumps to a new state $\in Q$,
consume input w_i
removes s_j from the stack
and pushes new elements $\in \Gamma$

This is done in nondeterministic fashion.
After reading w, the PDA will accept or reject.

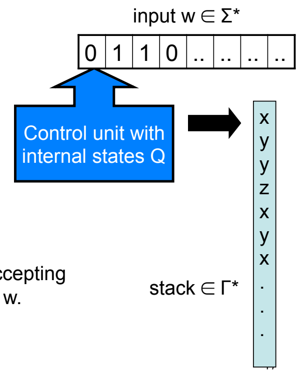


Push Down Automata (PDA)

For given $w_j \in \Sigma \cup \{\lambda\}$, $s_j \in \Gamma$ and $q_k \in Q$, the nondeterminism allows several possibilities for M. (Note the possibility of non-deterministic λ-transitions.)

After the PDA has read input w it can be in different state $\subseteq Q$.

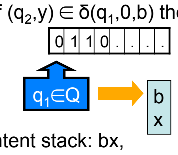
If it is possible to end in an accepting state $\in F \subseteq Q$, then M accepts w.



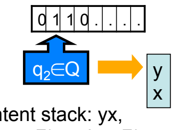
Push Down Automata (PDA)

Transitions

If $(q_2,y) \in \delta(q_1,0,b)$ then the following transition is allowed:



goes to



content stack: bx, with $b \in \Gamma$ and $x \in \Gamma^*$

content stack: yx, with $y \in \Gamma^*$ and $x \in \Gamma^*$

The states of the NPDA can be described by the triplets (q_1,aw,bx) and (q_2,w,yx) respectively and we denote this possible transition by $(q_1,aw,bx) \vdash (q_2,w,yx)$.

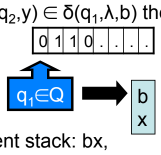
There can be several options according to δ, but it is required that the set $\delta(q_1,a,b)$ is finite.

18

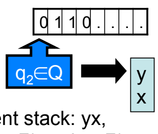
Push Down Automata (PDA)

λ-Transitions

If $(q_2,y) \in \delta(q_1,\lambda,b)$ then the following transition is allowed:



goes to



content stack: bx, with $b \in \Gamma$ and $x \in \Gamma^*$

content stack: yx, with $y \in \Gamma^*$ and $x \in \Gamma^*$

Here the NPDA does not read a input letter and makes a λ-transition (compare λ-transitions for NFA).

In general, if the NPDA is allowed to make several steps we write: $(q,w_1\ldots w_n,x) \vdash^* (q',w_i\ldots w_n,x')$.

19

Push Down Automata (PDA)

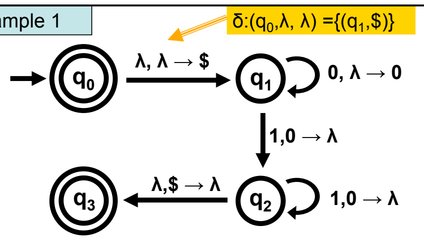
Accepted Language

- Definition: Given a NPDA $M = (Q,\Sigma,\Gamma,\delta,q_0,z,F)$, the language accepted by M is defined by:
  $L(M) = \{ w \in \Sigma^* : (q_0,w,z) \vdash^* (p,\lambda,u) \text{ with } p \in F,\ u \in \Gamma^* \}$.

- Note that the input part has to be empty (λ) in the end.
- The content of the stack does not matter.
- We only require that there is a possible transition.
- The only role of $z \in \Gamma$ is to start with a nonempty stack.

20

PDA: Example 1



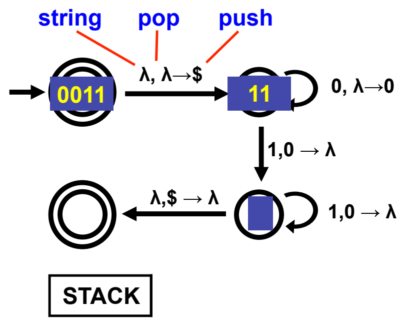
$Q = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{0,1\}$ $\Gamma = \{\$,0,1\}$

$\delta: Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \mathbf{P}(Q \times \Gamma^*)$

$\delta(q_1,1,0) = \{ (q_2, \lambda) \}$ $\delta(q_2,1,1) = \varnothing$

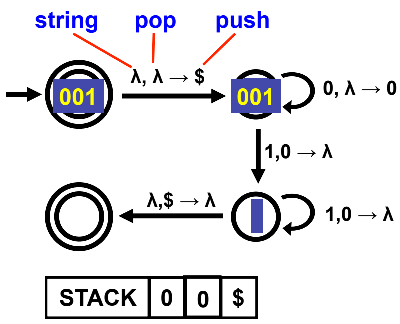
21

PDA: Example 1



22

PDA: Example 1



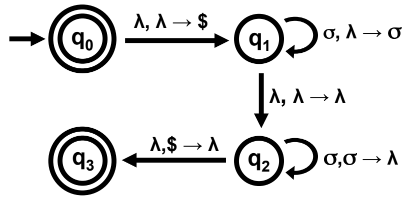
PDA to recognize $L = \{ 0^n1^n \mid n \geq 0 \}$

23

PDA: Example 2

EVEN-LENGTH **PALINDROMES**

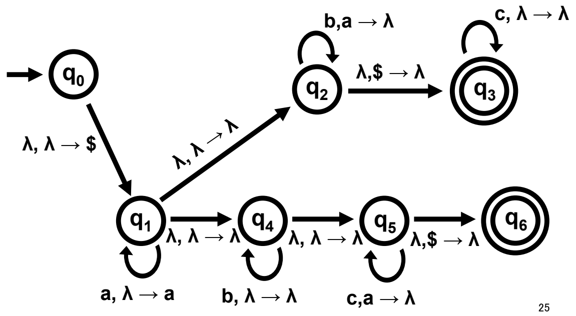
$\Sigma = \{a, b, c, \ldots, z\}$



24

PDA: Example 3

Build a PDA to recognize
L = { $a^i b^j c^k$ | i, j, k ≥ 0 and (i = j or i = k) }



Push Down Automata (PDA)

Example 4

•How to make a NPDA that accepts {w∈{a,b}* : #a = #b}?
•Central idea push/pull counting symbols 0 (for a) and 1 (for b) on the stack as you read the string w.
•Answer: NPDA M = $(\{q_0,q_f\},\{a,b\},\{0,1,z\},\delta,q_0,z,\{q_f\})$ with:
•$\delta(q_0,\lambda,z) = \{(q_f,z)\}$
$\delta(q_0,a,z) = \{(q_0,0z)\}$
$\delta(q_0,b,z) = \{(q_0,1z)\}$
$\delta(q_0,a,0) = \{(q_0,00)\}$
$\delta(q_0,b,0) = \{(q_0,\lambda)\}$
$\delta(q_0,a,1) = \{(q_0,\lambda)\}$
$\delta(q_0,b,1) = \{(q_0,b,11)\}$

Processing baab gives:
$(q_0,baab,z) \vdash (q_0,aab,1z)$
$\vdash (q_0,ab,z)$
$\vdash (q_0,b,0z)$
$\vdash (q_0,\lambda,z)$
$\vdash (q_f,\lambda,z)$.

Push Down Automata (PDA)

Exercise

•How to recognize ww^R∈{a,b}* ?
•Idea: Read w and put it on the stack (first in, last out).
At the half-way point, start checking the remaining
input string w^R against the stack content w^R.

•Crucial observation: We have to guess the half-way point.

PDA $\Longleftrightarrow$ CFL

A language L is context-free if and only if there is a non-deterministic pushdown automaton M that recognizes L.

Two step proof:
Theorem 1: Given a CFG G, we can construct a NPDA M_G such that $L(G)=L(M_G)$.
Theorem 2: Given a PDA M, we can construct a CFG G such that $L(G)=L(M_G)$.

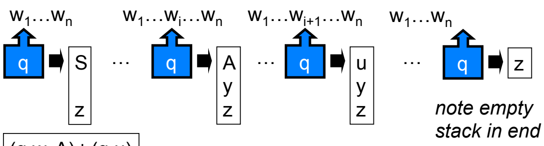
Central idea: Use Greibach normal form for the CFG and put the variables on the stack while reading letters.

PDA ⟵ CFL

Proof Idea

- Without loss of generality, we assume that the CFG is in Greibach normal form, for which we know that the derivation of $w_1 \ldots w_n$ has to look like (assuming $A \to w_i u$):
- $S \Rightarrow \ldots \Rightarrow w_1 \ldots w_{i-1} Ay \Rightarrow w_1 \ldots w_i uy \Rightarrow \ldots \Rightarrow w_1 \ldots w_n$ with $y,u \in V^*$.

- Idea for PDA implementation: when reading the input string $w_1 \ldots w_n$ from left to right on put possible $u \in V^*$ on stack:



$(q,w_i,A) \vdash (q,u)$

PDA ⟵ CFL

Proof details

- Given a CFG $G=(V,T,S,P)$ in Greibach normal form, the NPDA $M=(\{q_0,q_1,q_f\},T,V \cup \{z\},\delta,q_0,z,\{q_f\})$ accepts $L(G)$:

- **$\delta(q_0,\lambda,z) = \{(q_1,Sz)\}$** (start with S on the stack)
- $(q_1,u) \in \delta(q_1,a,A)$ for all $A \to au$ rules
- **$\delta(q_1,\lambda,z) = \{(q_f,z)\}$**

PDA ⟵ CFL

Example

$S \to \lambda \mid a \mid b \mid aSa \mid bSb$ $\xrightarrow{\text{GNF}}$ *$S \to \lambda \mid a \mid b \mid aSA \mid bSB$*
$A \to a, \; B \to b$

$\delta(q_0,\lambda,z) = \{(q_1,Sz)\}$

$(q_1, \lambda) \in \delta(q_1, \lambda,S)$	$(q_1, SA) \in \delta(q_1, a,S)$
$(q_1, \lambda) \in \delta(q_1, a,S)$	$(q_1, SB) \in \delta(q_1, b,S)$
$(q_1, \lambda) \in \delta(q_1, b,S)$	$(q_1, \lambda) \in \delta(q_1, a,A)$
	$(q_1, \lambda) \in \delta(q_1, b,B)$

$\delta(q_1,\lambda,z) = \{(q_f,z)\}$

PDA ⟶ CFL

- To prove that for every NPDA there is a corresponding CFG we use the same ideas as for the previous proofs.

- The proof uses the assumption that the NPDA has only one accepting state q_f that is entered when the stack is empty, and all transitions are of the form (q,a,A) (q',λ) or (q',BC).
- This assumption can be made without loss of generality.

PDA ⟶ CFL

Given PDA P = (Q, Σ, Γ, δ, q, F)

Construct a CFG G = (V, Σ, R, S) such that L(G)=L(P)

First, simplify P to have the following form:

(1) It has a single accept state, q_{accept}

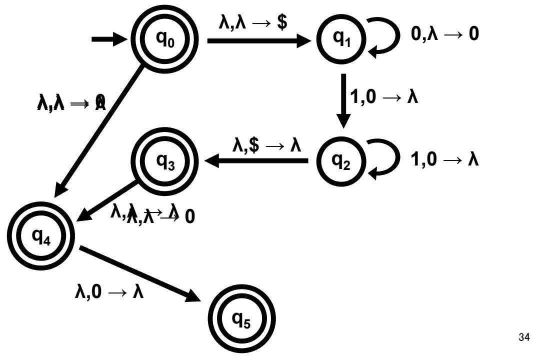
(2) It empties the stack before accepting

(3) Each transition either pushes a symbol or pops a symbol, but not both at the same time

33

PDA ⟶ CFL

SIMPLIFY



34

PDA ⟶ CFL

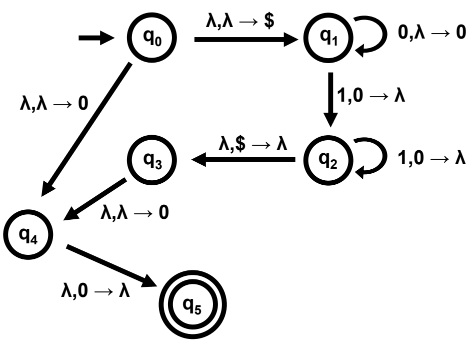
Idea: for each pair of states p and q in P, the grammar will have a variable A_{pq} that generates all strings that can take P from p with an empty stack to q with an empty stack

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

$S = A_{q_0 q_{accpet}}$

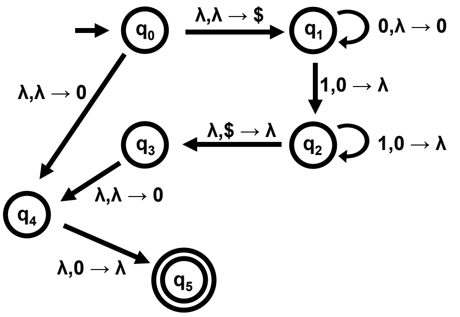
35

PDA ⟶ CFL



36

PDA ⟶ CFL



What strings does $A_{q_0q_1}$ generate? none

What strings does $A_{q_1q_2}$ generate? $\{0^n1^n \mid n > 0\}$

What strings does $A_{q_1q_3}$ generate? none

37

PDA ⟶ CFL

A_{pq} generates all strings that take p with an empty stack to q with an empty stack

Let x be such a string

- P's first move on x must be a push
- P's last move on x must be a pop

Two possibilities:

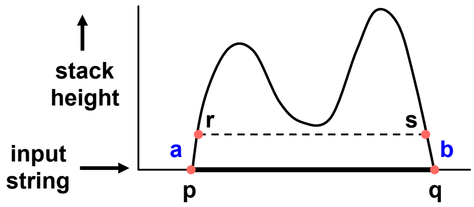
1. The symbol popped at the end is the one pushed at the beginning

2. The symbol popped at the end is not the one pushed at the beginning

38

PDA ⟶ CFL

1. The symbol popped at the end is the one pushed at the beginning



δ(p,a,z) = {(r,H)}

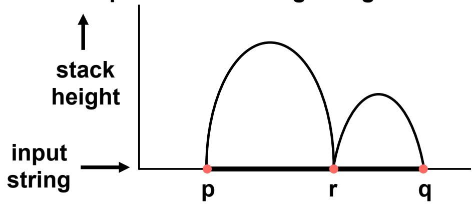
$A_{pq} \to aA_{rs}b$

δ(s,b,H) = {(q,z)}

39

PDA ⟶ CFL

2. The symbol popped at the end is not the one pushed at the beginning



δ(p,a,z) = {(n,H)}

δ(m,b,E) = {(q,z)}

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

40

PDA ⟹ CFL

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

$S = A_{q_0 q_{accpet}}$

For each $p,q,r,s \in Q$, $t \in \Gamma$ and $a,b \in \Sigma_\lambda$

If $(r,t) \in \delta(p,a,\lambda)$ and $(q, \lambda) \in \delta(s,b,t)$

Then add the rule $A_{pq} \to aA_{rs}b$

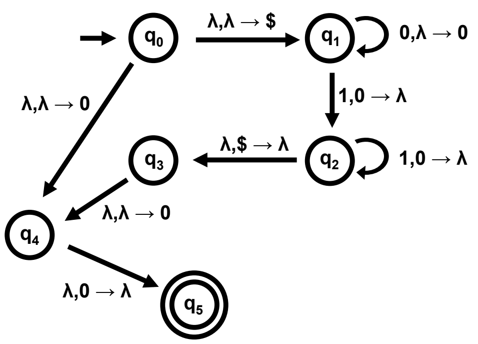
For each $p,q,r \in Q$,

add the rule $A_{pq} \to A_{pr}A_{rq}$

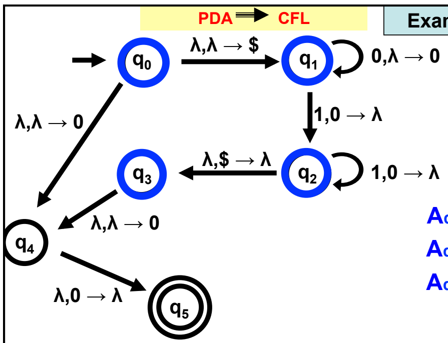
For each $p \in Q$,

add the rule $A_{pp} \to \lambda$

PDA ⟹ CFL Example



PDA ⟹ CFL Example



$A_{qq} \to \lambda$

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

$A_{q_0q_3} \to \lambda A_{q_1q_2}\lambda$

$A_{q_1q_2} \to 0A_{q_1q_2}1$

$A_{q_1q_2} \to 0A_{q_1q_1}1$

What strings does $A_{q_0q_1}$ generate? none

What strings does $A_{q_1q_2}$ generate? $\{0^n1^n \mid n > 0\}$

What strings does $A_{q_1q_3}$ generate? none