PART 2:

Automata:

PDA

Formal Language:

Context-Free Languages

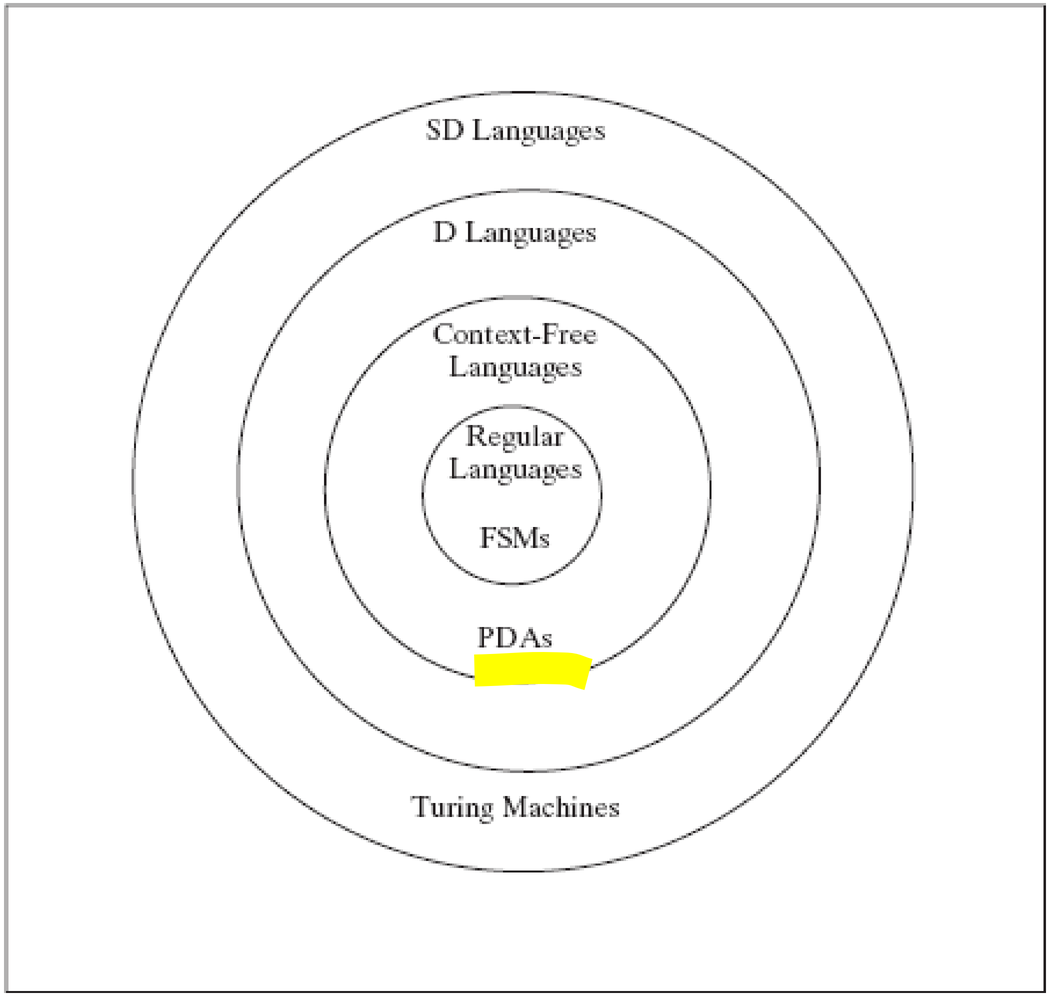
Non-Context-Free Languages

Grammar:

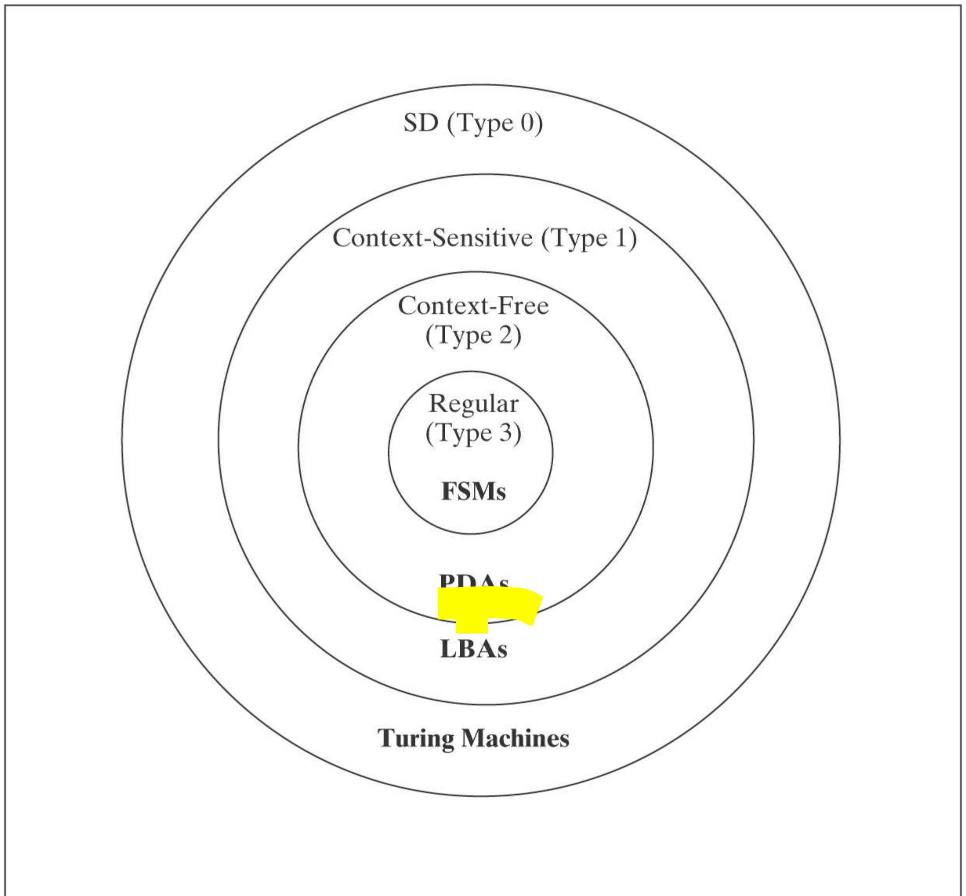
Context-Free Grammars

Pushdown Automata

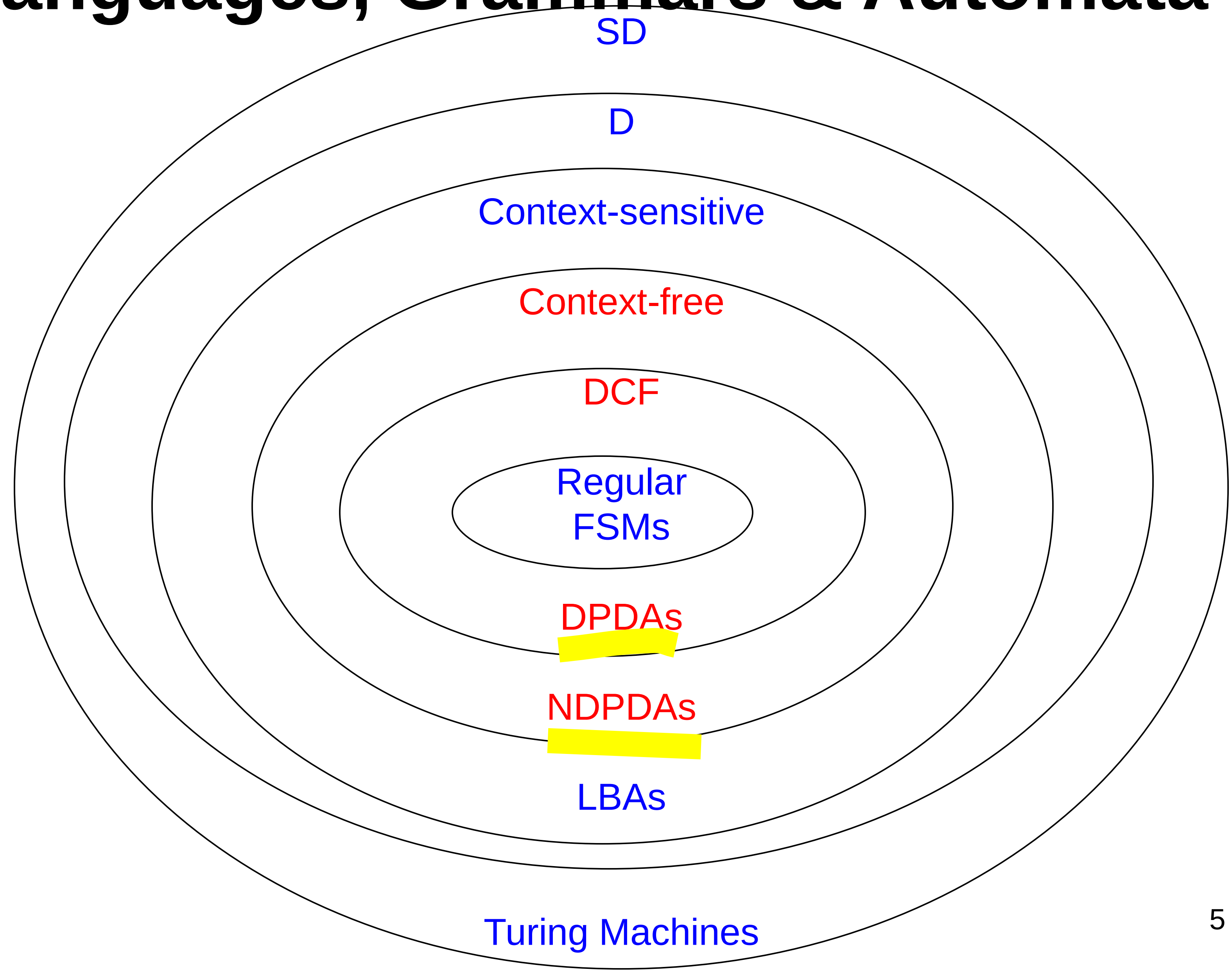
Languages, Grammars & Automata



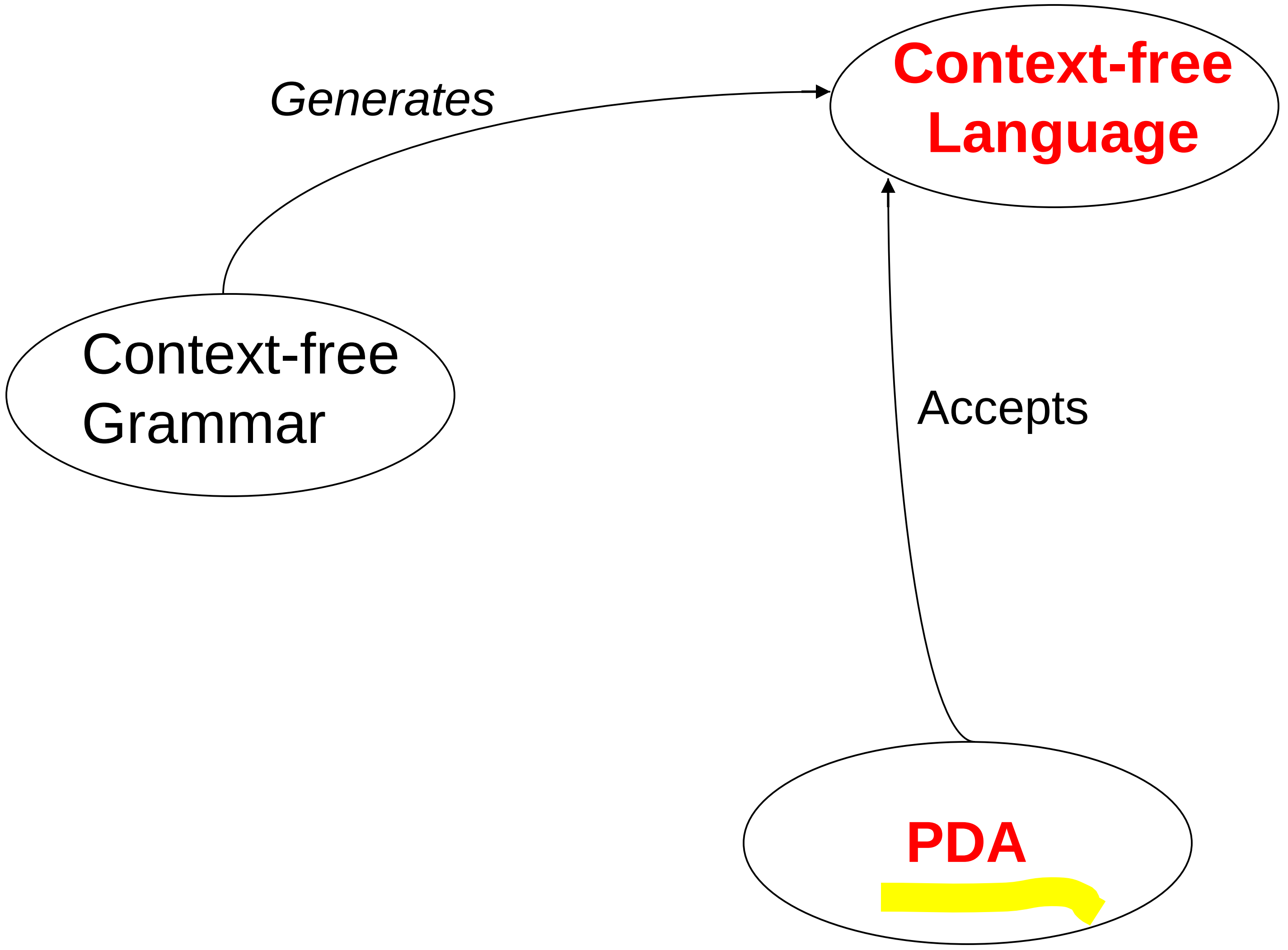
Languages, Grammars & Automata



Languages, Grammars & Automata



Context-free Grammars, Languages, and PDAs



Nondeterministic Pushdown Automaton (NPDA)

NPDA = NDFSM + a single stack (unlimited memory)

Definition of Nondeterministic PDA

NPDA is $M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

K is a finite set of states

Σ is the input alphabet

Γ is the **stack** alphabet

$s \in K$ is the initial state

$A \subseteq K$ is the set of accepting states, and

Δ is the transition **relation**

Definition of Nondeterministic PDA

Δ is the transition relation.

It is a finite subset of

$$(K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*) \times (K \times \Gamma^*)$$

state	input or ε	**string** of symbols to **pop** from top of **stack**	state	**string** of symbols to **push** on top of **stack**

Example: NPDA

K =

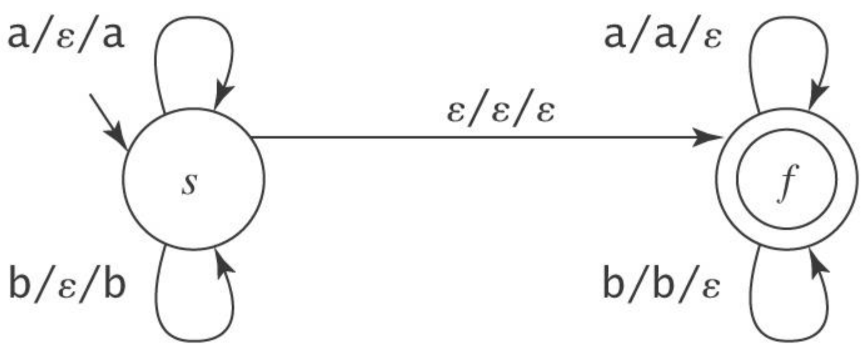
Σ =

Γ =

$s \in K$ =

$A \subseteq K$ =

Δ =

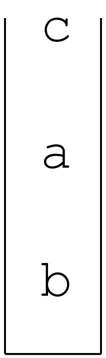


Configurations of NPDA

A configuration of M is an element of $K \times \Sigma^* \times \Gamma^*$.

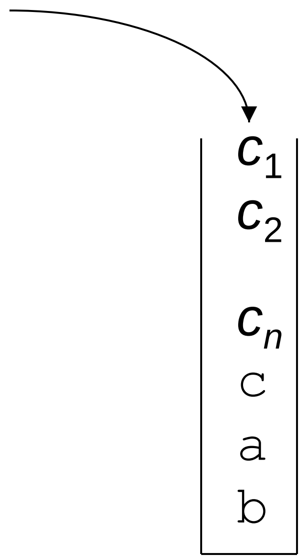
The initial configuration of M is (s, w, ε).

Manipulating the Stack



will be written as cab

If $c_1c_2\ldots c_n$ is pushed onto the stack:



$c_1c_2\ldots c_n$cab

Yields

Let c be any element of $\Sigma \cup \{\varepsilon\}$,
Let γ_1, γ_2 and γ be any elements of Γ^*, and
Let w be any element of Σ^*.

Then:

$(q_1, cw, \gamma_1\gamma) \mid\!-_M (q_2, w, \gamma_2\gamma)$ iff $((q_1, c, \gamma_1), (q_2, \gamma_2)) \in \Delta$.

Let $\mid\!-_M{}^*$ be the *reflexive, transitive closure* of $\mid\!-_M$.

C_1 *yields* configuration C_2 iff $C_1 \mid\!-_M{}^* C_2$

Computations Using NPDA

A *computation* by M is a finite sequence of configurations $C_0, C_1, \ldots, C_n$ for some $n \geq 0$ such that:

- C_0 is an initial configuration,
- C_n is of the form (q, ε, γ), for some state $q \in K_M$ and some string γ in Γ^*, and
- $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$.

Nondeterminism

If M is in some configuration (q_1, s, γ) it is possible that:

- Δ contains exactly one transition that matches.
- Δ contains more than one transition that matches.
- Δ contains no transition that matches.

Accepting

A computation C of M is an *accepting computation* iff:

- $C = (s, w, \varepsilon) \vdash_M^* (q, \varepsilon, \varepsilon)$, and
- $q \in A$.

✓ All the input is read!
✓ The stack is empty!
✓ An accepting state!

M ***accepts*** a string w iff at least one of its computations accepts.

Rejecting

A computation C of M is a *rejecting computation* iff:

- $C = (s, w, \varepsilon) \vdash_M^* (q, w', \alpha)$,
- C is not an accepting computation, and
- M has no moves that it can make from (q, ε, α).

M ***rejects*** a string w iff all of its computations reject.

Other Paths

Other paths may:

- Read all the input and halt in a nonaccepting state,
- Read all the input and halt in an accepting state with the stack not empty,
- Loop forever and never finish reading the input, Reach a dead end where no more input can be read.

So note that it is possible that, on input *w*, *M* neither accepts nor rejects.

Language Accepted by NPDA

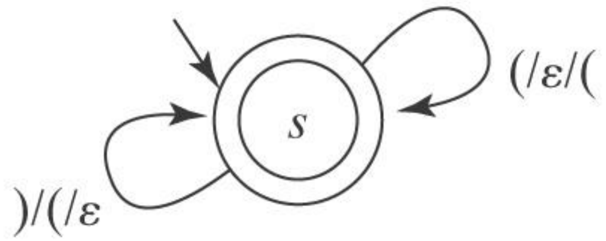
The ***language accepted by M***, denoted ***L(M)***, is the set of all strings accepted by *M*.

Examples & Designing NPDA

Example 12.1

Bal = { w $\in$ {), (}* : the parentheses are balanced}

PDA?



( )

( ) (( ))

((

M = (K, Σ, Γ, Δ, s, A), where:

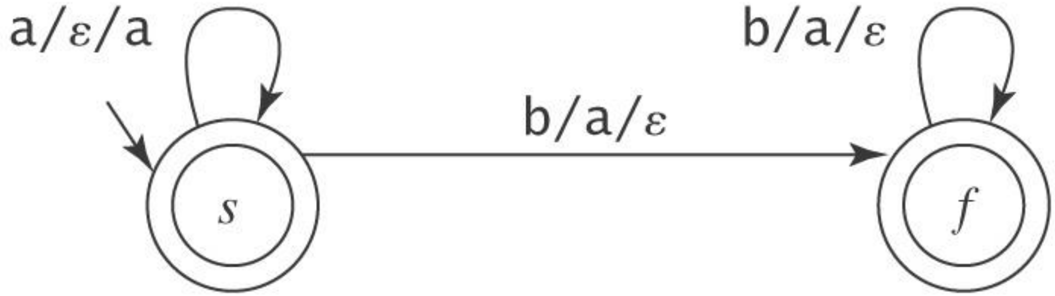
K = {s}	the states
Σ = {(,)}	the input alphabet
Γ = {(}	the stack alphabet
A = {s}	
Δ contains:	
	((s, (, ε), (s, ( ))
	((s,), ( ), (s, ε))

Example 12.2

$A^nB^n = \{a^n b^n : n \geq 0\}$

PDA?

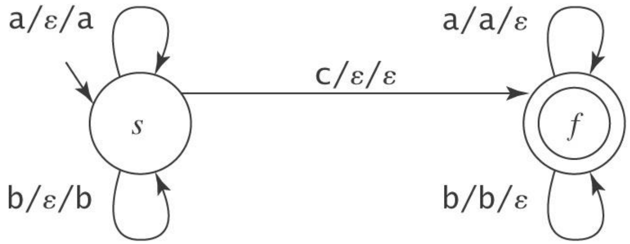
aabb
aaabb
aaa



Example 12.3

$L = \{wcw^R : w \in \{a, b\}^*\}$

PDA?



abcba
aabcbba

$M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

$K = \{s, f\}$	the states
$\Sigma = \{a, b, c\}$	the input alphabet
$\Gamma = \{a, b\}$	the stack alphabet
$A = \{f\}$	the accepting states
Δ contains:	$((s, a, \varepsilon), (s, a))$
	$((s, b, \varepsilon), (s, b))$
	$((s, c, \varepsilon), (f, \varepsilon))$
	$((f, a, a), (f, \varepsilon))$
	$((f, b, b), (f, \varepsilon))$

Example 12.4

L = $\{a^n b^{2n}: n \geq 0\}$

PDA?

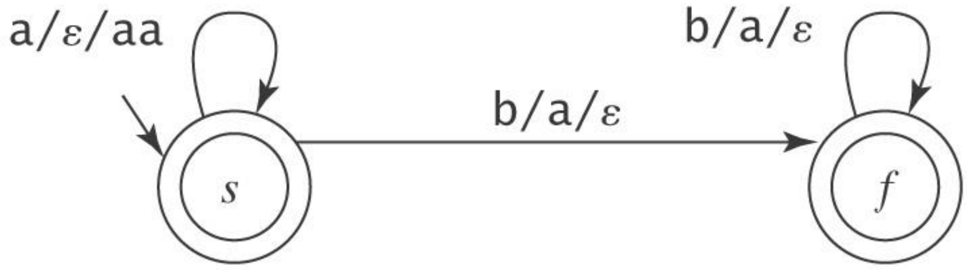


abb
abbbb
aabbb

Deterministic PDA

A PDA M is ***deterministic*** iff:

- Δ_M contains no pairs of transitions that compete with each other, and
- Whenever M is in an accepting configuration it has no available moves.

But many useful PDAs are not deterministic!

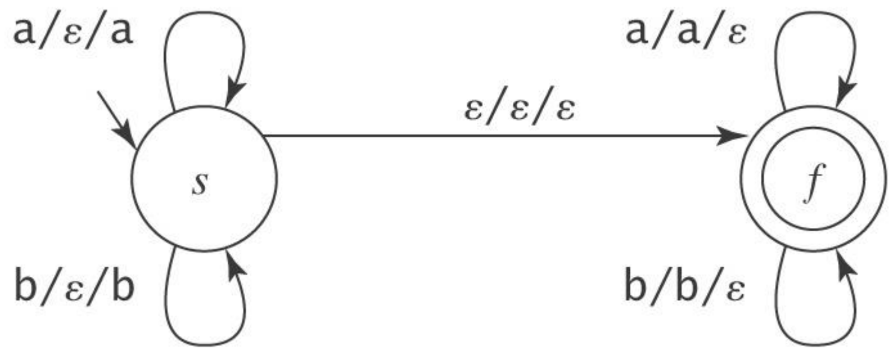
Example 12.5

PalEven $=\{ww^R: w \in \{\mathtt{a}, \mathtt{b}\}^*\}$

CFG:

$S \rightarrow \varepsilon$
$S \rightarrow \mathtt{a}S\mathtt{a}$
$S \rightarrow \mathtt{b}S\mathtt{b}$

NPDA?

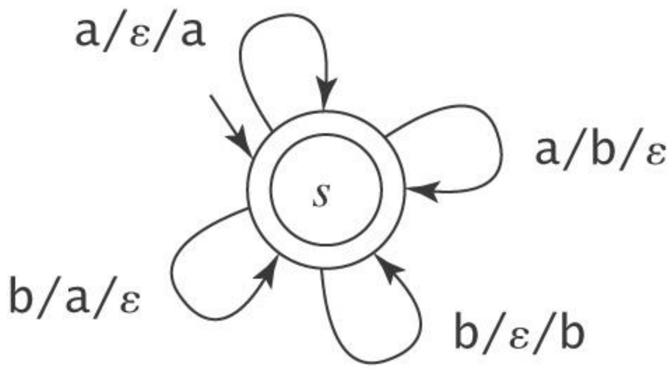


ababbaba
ababba

Example 12.6

L = {$w \in$ {a, b}* : $\#_a(w) = \#_b(w)$}

NPDA?

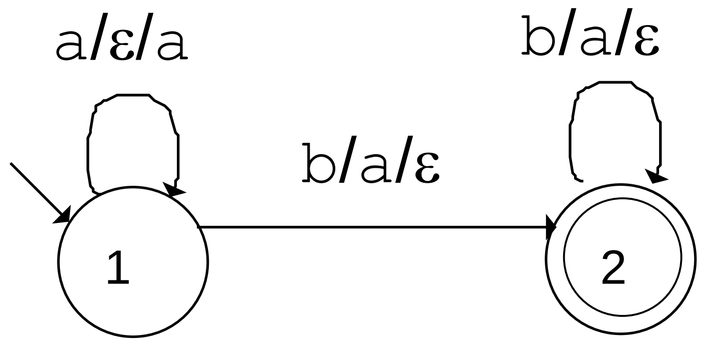


ababbaba
ababba

Example 12.7

L = {$a^m b^n$: $m = n$; $m, n > 0$}

NPDA?



Example 12.7

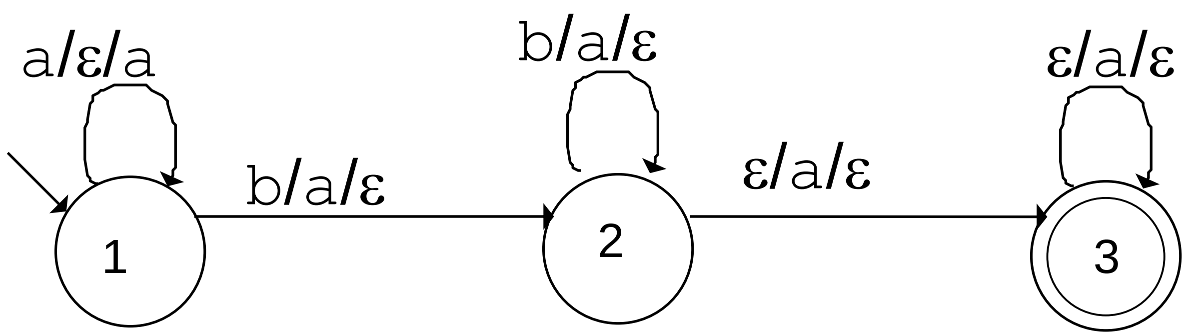
L = $\{a^m b^n : m \neq n;\ m, n > 0\}$

NPDA?

Example 12.7

$L_1 = \{\mathtt{a}^m\mathtt{b}^n : 0 < n < m\}$

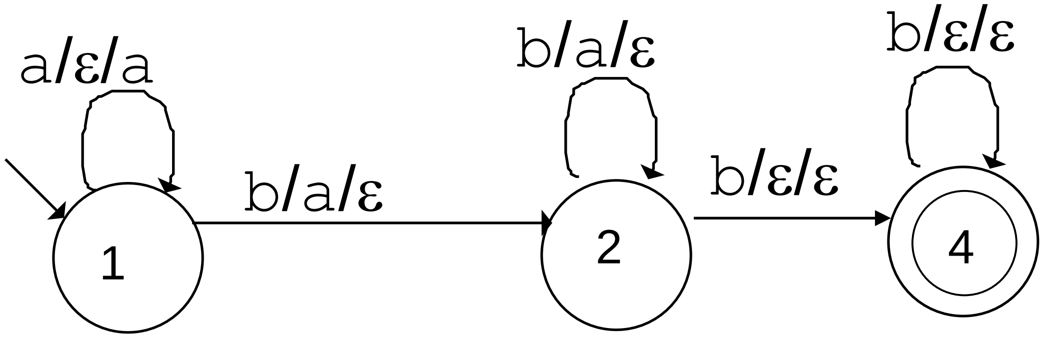
If input is empty but stack is not ($m > n$) (accept):



Example 12.7

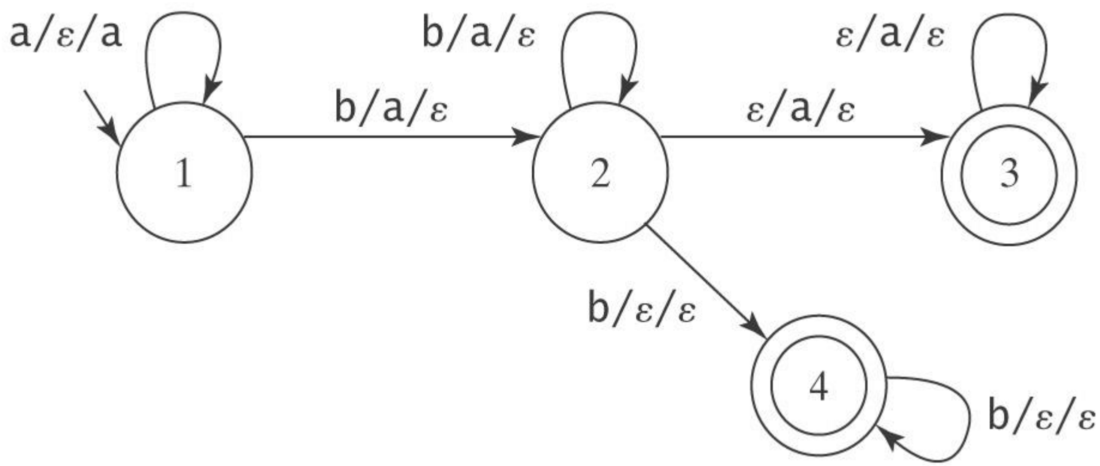
$L_2 = \{\mathtt{a}^m\mathtt{b}^n : 0 < m < n\}$

If stack is empty but input is not ($m < n$) (accept):



Example 12.7

L = {$a^m b^n$: $m \neq n$; $m, n > 0$} = $L_1 \cup L_2$



aab

aaabbbbb

aabb

Example 12.8

$L = A^nB^nC^n = \{a^nb^nc^n : n \geq 0\}$.

NPDA?

Example 12.8

$L = \neg A^n B^n C^n$

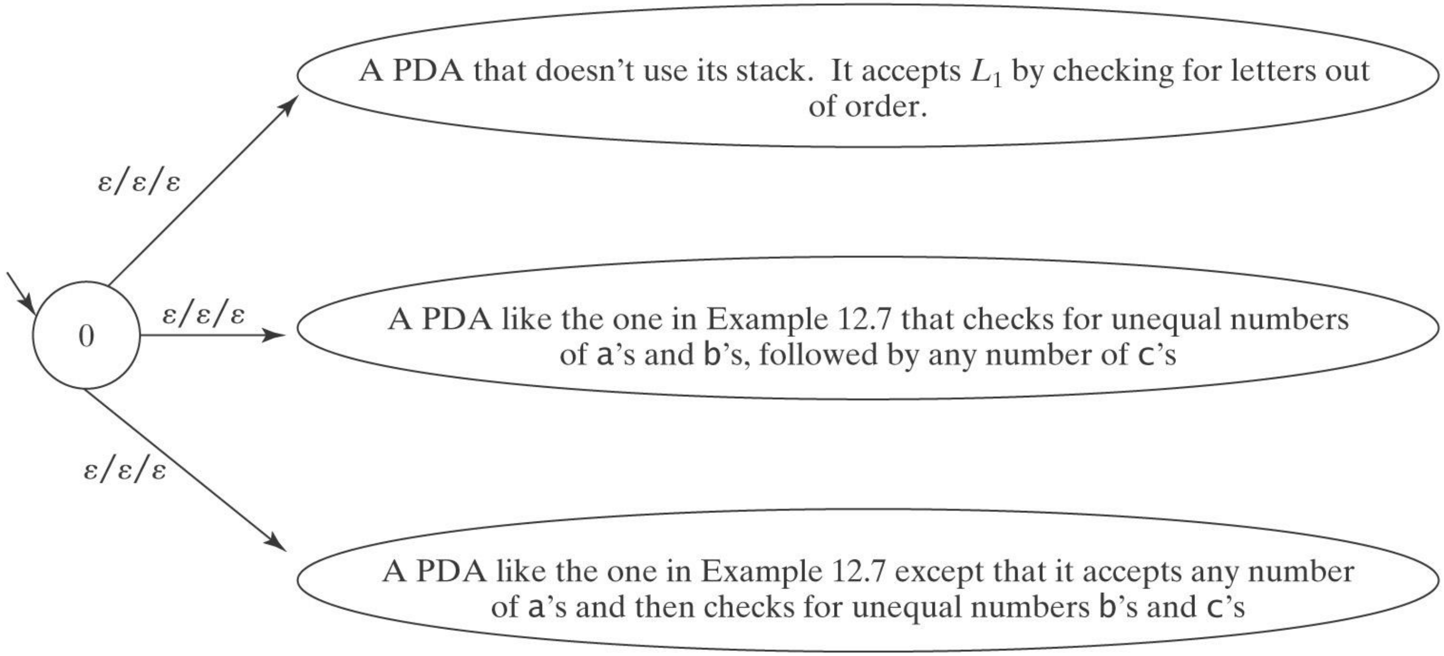
NPDA?

L is the union of two languages:

- $\{w \in \{\texttt{a}, \texttt{b}, \texttt{c}\}^*$: the letters are out of order$\}$
- $\{\texttt{a}^i\texttt{b}^j\texttt{c}^k : i, j, k \geq 0 \text{ and } (i \neq j \text{ or } j \neq k)\}$

Example 12.8

NPDA for L = $\neg A^nB^nC^n$:



Equivalence of NPDAs and CFGs

NPDA = CFG

CFL = CFG = NPDA

Equivalence of PDAs and CFGs

Theorem 12.1 Given a CFG G, there exists a NPDA M such that L(G) = L(M).

Proof Idea:

Proof by Construction

Equivalence of PDAs and CFGs

Theorem 12.2 Given a NPDA M, there exists a CFG G such that L(G) = L(M).

Proof Idea:

Proof by Construction

Context-Free Languages

A language is ***context-free*** iff it is accepted by **some NPDA**.

CFL = NPDA

Equivalence of PDAs and CFGs

Theorem 12.3 A language is context-free iff it is accepted by some NPDA.

Proof Idea:

- For every CFG there exists an equivalent NPDA.
- For every NPDA there exists an equivalent CFG.

Deterministic PDA (DPDA)

Deterministic PDAs

A PDA *M* is *deterministic* iff:

- Δ_M contains no pairs of transitions that compete with each other, and
- Whenever *M* is in an accepting configuration it has no available moves.

Definition of Deterministic PDA

DPDA is $M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

K is a finite set of states

Σ is the input alphabet

Γ is the stack alphabet

$s \in K$ is the initial state

$A \subseteq K$ is the set of accepting states, and

Δ is the transition **function**. It is a finite subset of

$$(K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*) \times (K \times \Gamma^*)$$

state	input or ε	string of symbols to pop from top of stack	state	string of symbols to push on top of stack

Example: DPDA

K =

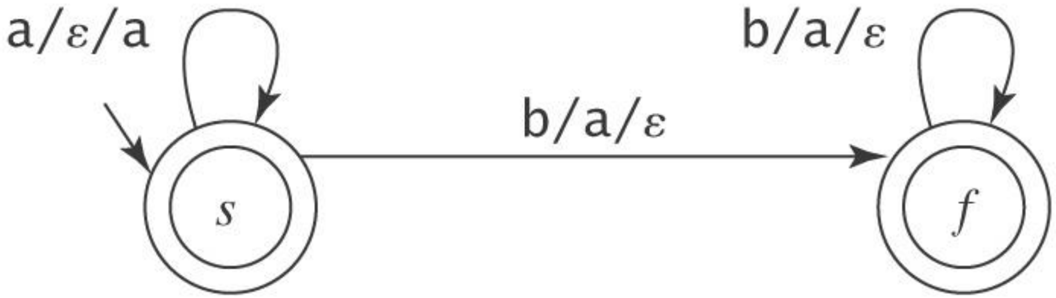
Σ =

Γ =

$s \in K$ =

$A \subseteq K$ =

Δ =



Non-Equivalence of NPDA and DPDA

NPDA ≠ DPDA

!

- ✓ DPDA is weaker than NPDA!
- ✓ DPDA accepts a class of languages DCFLs strictly between the RLs and the CFLs!
- ✓ DCFL = Unambiguous CFL!

Alternative Equivalent & Not Equivalent Definitions of a NPDA

Alternative Equivalent Definitions of a NPDA

- Pop and Push?
  - ➢ any string.
  - ➢ only a single symbol?
- Accept?
  - ➢ if the input is consumed and in an accepting state and the stack is empty.
  - ➢ if the input is consumed and in an accepting state (regardless of the stack content)?
  - ➢ if the input is consumed (regardless of the final state) and the stack is empty?

✓ All of these alternatives are equivalent!

Alternative NOT Equivalent Definitions of a NPDA

- NPDA = NDFSM + **a stack**
- FSM plus **a queue** (instead of stack)?
  - ✓ Tag system (Post machine)
  - ✓ = TM!
- FSM plus **two stacks**?
  - ✓ = TM!

Comparing RL and CFL

Regular Languages	**Context-Free Languages**
regular exprs or regular grammars	context-freegrammars
recognize	parse
DFSMs	NPDAs

Reading Assignment

Chapter 12:

Sections
12.1
12.2
12.3
12.4
12.5
12.6

In-Class Exercises

Chapter 12:

1 – c & j
4