NPDAs Accept
Context-Free Languages
class 11

1

Theorem:

$$\left\{\begin{array}{c}\text{Context-Free}\\ \text{Languages}\\ \text{(Grammars)}\end{array}\right\} = \left\{\begin{array}{c}\text{Languages}\\ \text{Accepted by}\\ \text{NPDAs}\end{array}\right\}$$

2

Proof - Step 1:

$$\left\{\begin{array}{c}\text{Context-Free}\\ \text{Languages}\\ \text{(Grammars)}\end{array}\right\} \subseteq \left\{\begin{array}{c}\text{Languages}\\ \text{Accepted by}\\ \text{NPDAs}\end{array}\right\}$$

Convert any context-free grammar G to a NPDA M with: $L(G) = L(M)$

3

Proof - Step 2:

$$\left\{\begin{array}{c}\text{Context-Free}\\ \text{Languages}\\ \text{(Grammars)}\end{array}\right\} \supseteq \left\{\begin{array}{c}\text{Languages}\\ \text{Accepted by}\\ \text{NPDAs}\end{array}\right\}$$

Convert any NPDA M to a context-free grammar G with: $L(G) = L(M)$

4

Proof - step 1

Converting
Context-Free Grammars
to
NPDAs

5

We will convert any context-free grammar G

to an NPDA automaton M

Such that:

M Simulates leftmost derivations of G

6

Leftmost derivation

$G:$

$$S \Rightarrow \cdots\cdots \Rightarrow \underbrace{\sigma_1\sigma_2\cdots\sigma_k}_{\text{Input processed}}\underbrace{V_1V_2\cdots V_m}_{\text{Stack contents}} \Rightarrow \cdots$$

leftmost variable

$M:$ Simulation of derivation

Input

σ_1	σ_2	$\cdots$	σ_k	σ_{k+1}	$\cdots$

Stack

V_1
V_2
$\vdots$
V_m
$\$$

7

Leftmost derivation

$G:$

$$S \Rightarrow \cdots\cdots\cdots \Rightarrow \sigma_1\sigma_2\cdots\sigma_n$$

string of terminals

$M:$ Simulation of derivation

Input

σ_1	σ_2	$\cdots\cdots$	σ_n

end of input is reached

Stack

$\$$

8

An example grammar:

$S \rightarrow aSTb$

$S \rightarrow b$

$T \rightarrow Ta$

$T \rightarrow \lambda$

What is the equivalent NPDA?

9

Grammar:

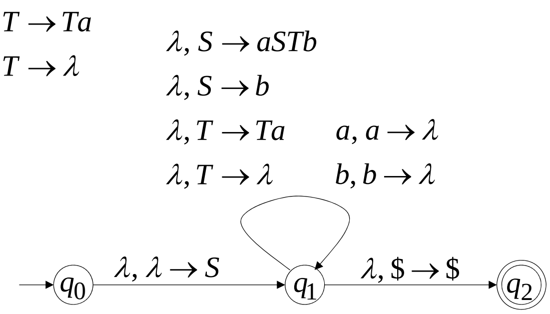
$S \rightarrow aSTb$

$S \rightarrow b$

$T \rightarrow Ta$

$T \rightarrow \lambda$

NPDA:



10

Grammar:

$S \rightarrow aSTb$

$S \rightarrow b$

$T \rightarrow Ta$

$T \rightarrow \lambda$

A leftmost derivation:

$$S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab \Rightarrow abab$$

11

Derivation:

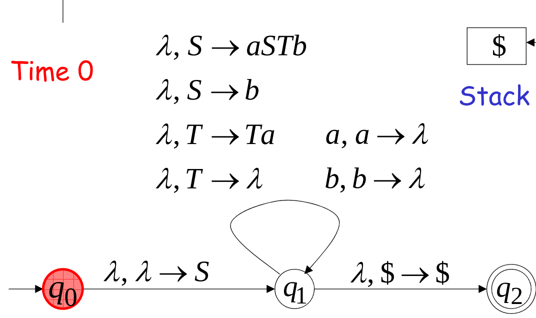
Input

a	b	a	b

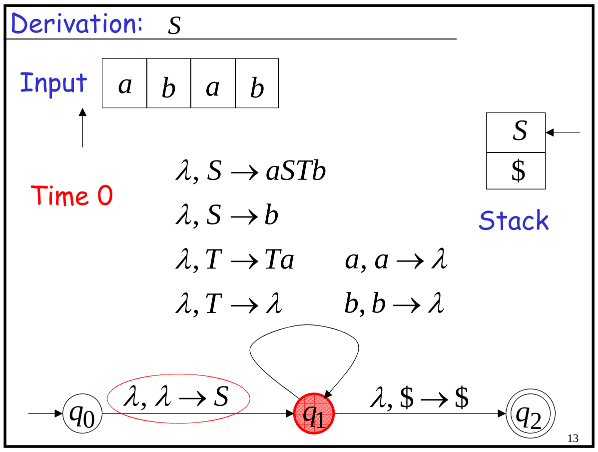
Time 0

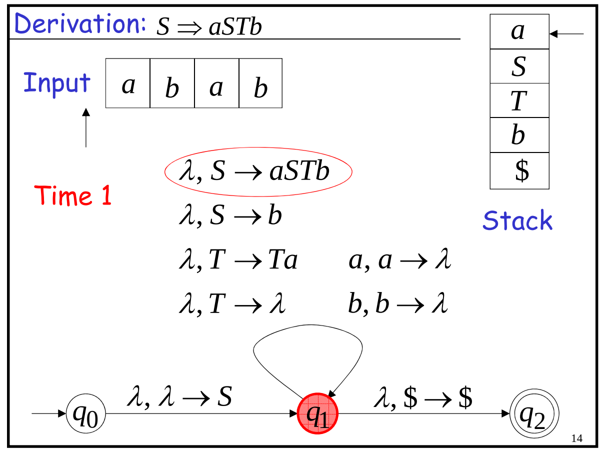
Stack

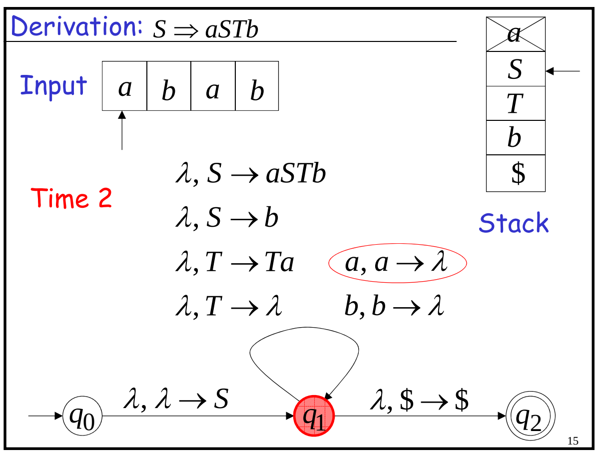
$\$$

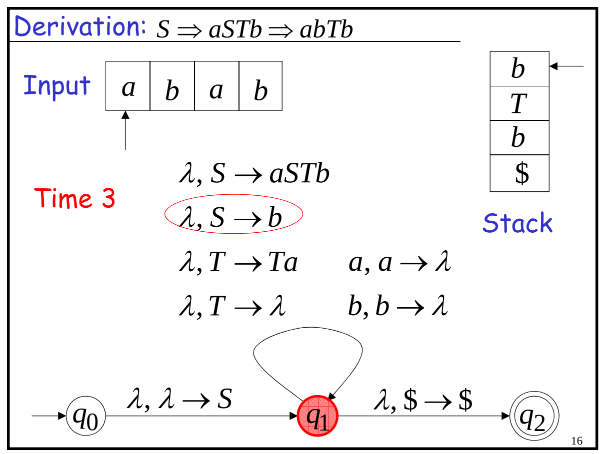


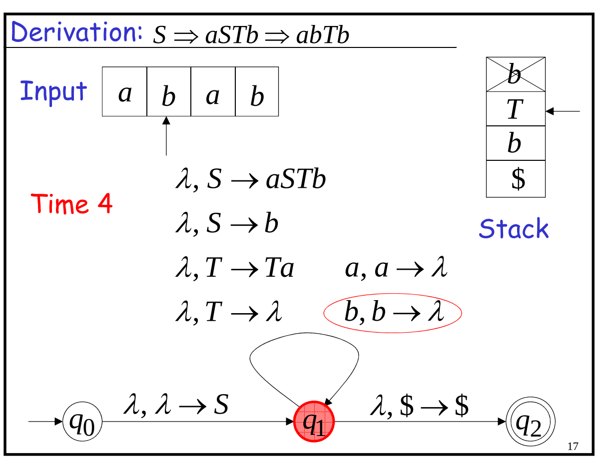
12

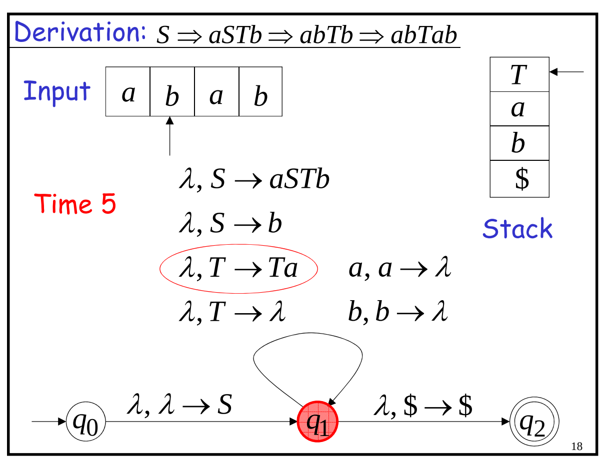

Derivation: S
Input
a b a b
Time 0
$\lambda, S \to aSTb$
$\lambda, S \to b$
$\lambda, T \to Ta$
$\lambda, T \to \lambda$
$a, a \to \lambda$
$b, b \to \lambda$
S
$
Stack
$\lambda, \lambda \to S$
$\lambda, \$ \to \$$
q_0
q_1
q_2
13


Derivation: $S \Rightarrow aSTb$
Input
a b a b
Time 1
$\lambda, S \to aSTb$
$\lambda, S \to b$
$\lambda, T \to Ta$
$\lambda, T \to \lambda$
$a, a \to \lambda$
$b, b \to \lambda$
a
S
T
b
$
Stack
$\lambda, \lambda \to S$
$\lambda, \$ \to \$$
q_0
q_1
q_2
14


Derivation: $S \Rightarrow aSTb$
Input
a b a b
Time 2
$\lambda, S \to aSTb$
$\lambda, S \to b$
$\lambda, T \to Ta$
$\lambda, T \to \lambda$
$a, a \to \lambda$
$b, b \to \lambda$
a
S
T
b
$
Stack
$\lambda, \lambda \to S$
$\lambda, \$ \to \$$
q_0
q_1
q_2
15


Derivation: $S \Rightarrow aSTb \Rightarrow abTb$
Input
a b a b
Time 3
$\lambda, S \to aSTb$
$\lambda, S \to b$
$\lambda, T \to Ta$
$\lambda, T \to \lambda$
$a, a \to \lambda$
$b, b \to \lambda$
b
T
b
$
Stack
$\lambda, \lambda \to S$
$\lambda, \$ \to \$$
q_0
q_1
q_2
16


Derivation: $S \Rightarrow aSTb \Rightarrow abTb$
Input
a b a b
Time 4
$\lambda, S \to aSTb$
$\lambda, S \to b$
$\lambda, T \to Ta$
$\lambda, T \to \lambda$
$a, a \to \lambda$
$b, b \to \lambda$
b
T
b
$
Stack
$\lambda, \lambda \to S$
$\lambda, \$ \to \$$
q_0
q_1
q_2
17


Derivation: $S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab$
Input
a b a b
Time 5
$\lambda, S \to aSTb$
$\lambda, S \to b$
$\lambda, T \to Ta$
$\lambda, T \to \lambda$
$a, a \to \lambda$
$b, b \to \lambda$
T
a
b
$
Stack
$\lambda, \lambda \to S$
$\lambda, \$ \to \$$
q_0
q_1
q_2
18

Derivation: $S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab \Rightarrow abab$

Input: a b a b

Time 6

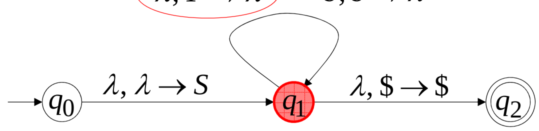
Stack: $\cancel{T}$, a, b, $\$$

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad b, b \rightarrow \lambda$



Derivation: $S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab \Rightarrow abab$

Input: a b a b

Time 7

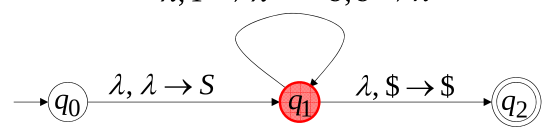
Stack: $\cancel{a}$, b, $\$$

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad b, b \rightarrow \lambda$



Derivation:

Input: a b a b

Time 8

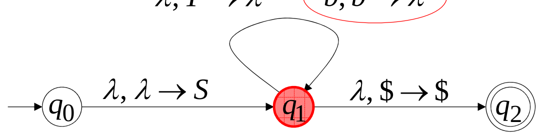
Stack: $\cancel{b}$, $\$$

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad b, b \rightarrow \lambda$



Derivation:

Input: a b a b

Time 9

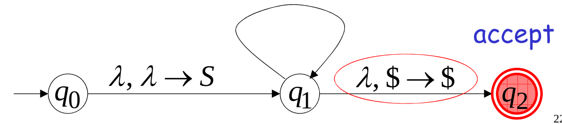
Stack: $\$$

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad b, b \rightarrow \lambda$



In general:

Given any grammar G

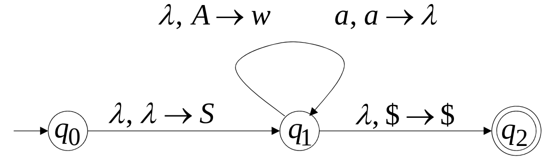
We can construct a NPDA M

With $L(G) = L(M)$

Constructing NPDA M from grammar G :

For any production $A \rightarrow w$ $\quad$ For any terminal a

$\lambda, A \rightarrow w$ $\quad a, a \rightarrow \lambda$



Grammar G generates string w

if and only if

NPDA M accepts w

$\Downarrow$

$L(G) = L(M)$

25

Therefore:

For any context-free language
there is a NPDA
that accepts the same language

$$\left\{\begin{array}{c}\text{Context-Free}\\\text{Languages}\\\text{(Grammars)}\end{array}\right\} \subseteq \left\{\begin{array}{c}\text{Languages}\\\text{Accepted by}\\\text{NPDAs}\end{array}\right\}$$

26

Proof - step 2

Converting
NPDAs
to
Context-Free Grammars

27

For any NPDA M

we will construct

a context-free grammar G with

$L(M) = L(G)$

28

Intuition: The grammar simulates the machine

A derivation in Grammar G :

terminals variables

$$S \Rightarrow \cdots \Rightarrow abc\ldots ABC\ldots \Rightarrow \ldots \Rightarrow abc\ldots$$

Input processed Stack contents

Current configuration in NPDA M

29

Some Necessary Modifications

Modify (if necessary) the NPDA so that:

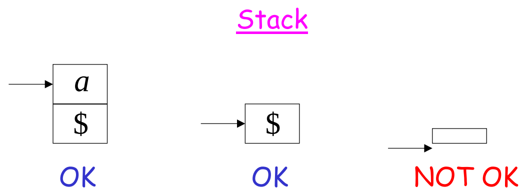
1) The stack is never empty

2) It has a single final state
and empties the stack when it accepts a string

3) Has transitions in a special form

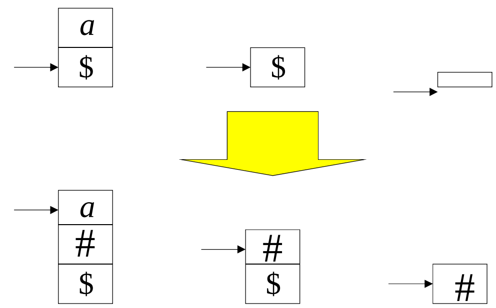
30

1) Modify the NPDA so that
the stack is never empty



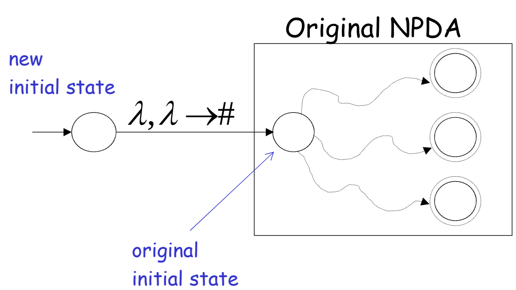
31

Introduce the new symbol # to denote the bottom of the stack



32

At the beginning push # into the stack

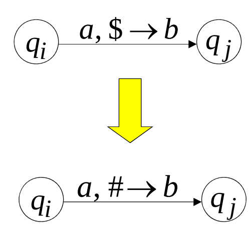


33

In transitions:

replace every instance of $ with #

Example:



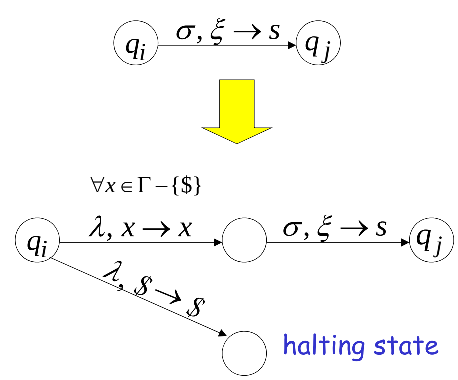
34

Convert all transitions so that:

if the automaton attempts to pop or replace $ it will halt

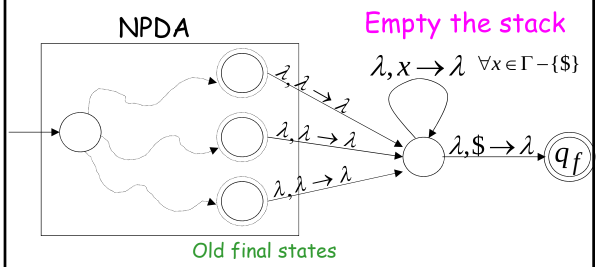
35

Convert transitions as follows:



36

2) Modify the NPDA so that
it empties the stack
and has a unique final state



37

3) modify the NPDA so that
transitions have the following forms:

$$q_i \xrightarrow{\sigma,\, B \to \lambda} q_j$$

OR

$$q_i \xrightarrow{\sigma,\, B \to CD} q_j$$

B, C, D : stack symbols

38

Convert:

$$q_i \xrightarrow{\sigma,\, \lambda \to y} q_j$$

$$\Downarrow$$

$$q_i \xrightarrow{\sigma,\, \tau \to y\tau} q_j$$

$\forall \tau \in \Gamma - \{\$\}$

39

Convert:

symbols

$$q_i \xrightarrow{\sigma,\, A \to B} q_j$$

$$\Downarrow$$

$$q_i \xrightarrow{\sigma,\, A \to XB} \bigcirc \xrightarrow{\sigma,\, X \to \lambda} q_j$$

$X \in \Gamma - \{\$\}$

40

Convert: $|y| \geq 2$

symbols

$$q_i \xrightarrow{\sigma,\, A \to By} q_j$$

$$\Downarrow$$

Convert recursively

$$q_i \xrightarrow{\sigma,\, A \to y} \bigcirc \xrightarrow{\sigma,\, X \to BX} q_j$$

$\forall X \in \Gamma - \{\$\}$

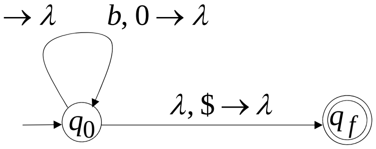
41

Example of a NPDA in correct form:

$$L(M) = \{w : \ n_a = n_b\}$$

$\$$: initial stack symbol

$a, \$ \to 0\$$	$b, \$ \to 1\$$
$a, 0 \to 00$	$b, 1 \to 11$
$a, 1 \to \lambda$	$b, 0 \to \lambda$



42

The Grammar Construction

In grammar G:

Stack symbol

Variables: $(q_i B q_j)$

states

Terminals:
Input symbols of NPDA

43

For each transition

$$q_i \xrightarrow{a,\ B \to \lambda} q_j$$

We add production $(q_i B q_j) \to a$

44

For each transition

$$q_i \xrightarrow{a,\ B \to CD} q_j$$

We add productions

$$(q_i B q_k) \to a(q_j C q_l)(q_l D q_k)$$

For all possible states q_k, q_l
in the automaton

45

Stack bottom symbol

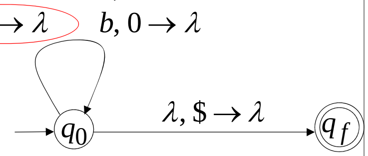
Start Variable: $(q_o \$ q_f)$

Start state final state

46

Example:

$a, \$ \to 0\$$ $\quad b, \$ \to 1\$$
$a, 0 \to 00$ $\quad b, 1 \to 11$
$a, 1 \to \lambda$ $\quad b, 0 \to \lambda$

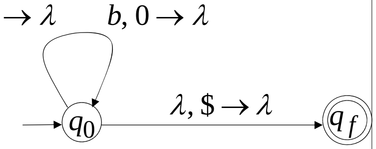


Grammar production: $(q_0 1 q_0) \to a$

47

Example:

$a, \$ \to 0\$$ $\quad b, \$ \to 1\$$
$a, 0 \to 00$ $\quad b, 1 \to 11$
$a, 1 \to \lambda$ $\quad b, 0 \to \lambda$



Grammar productions:

$$(q_0 \$ q_0) \to b(q_0 1 q_0)(q_0 \$ q_0) \mid b(q_0 1 q_f)(q_f \$ q_0)$$
$$(q_0 \$ q_f) \to b(q_0 1 q_0)(q_0 \$ q_f) \mid b(q_0 1 q_f)(q_f \$ q_f)$$

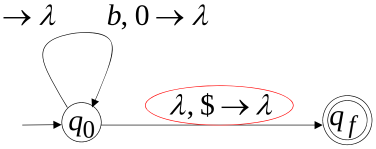
48

Example:

$a, \$ \rightarrow 0\$ \quad b, \$ \rightarrow 1\$$

$a, 0 \rightarrow 00 \quad b, 1 \rightarrow 11$

$a, 1 \rightarrow \lambda \quad b, 0 \rightarrow \lambda$



Grammar production: $(q_0 \$ q_f) \rightarrow \lambda$

49

Resulting Grammar: $(q_0 \$ q_f)$: start variable

$$(q_0 \$ q_0) \rightarrow b(q_0 1 q_0)(q_0 \$ q_0) \mid b(q_0 1 q_f)(q_f \$ q_0)$$
$$(q_0 \$ q_f) \rightarrow b(q_0 1 q_0)(q_0 \$ q_f) \mid b(q_0 1 q_f)(q_f \$ q_f)$$

$$(q_0 1 q_0) \rightarrow b(q_0 1 q_0)(q_0 1 q_0) \mid b(q_0 1 q_f)(q_f 1 q_0)$$
$$(q_0 1 q_f) \rightarrow b(q_0 1 q_0)(q_0 1 q_f) \mid b(q_0 1 q_f)(q_f 1 q_f)$$

$$(q_0 \$ q_0) \rightarrow a(q_0 0 q_0)(q_0 \$ q_0) \mid a(q_0 0 q_f)(q_f \$ q_0)$$
$$(q_0 \$ q_f) \rightarrow a(q_0 0 q_0)(q_0 \$ q_f) \mid a(q_0 0 q_f)(q_f \$ q_f)$$

50

$$(q_0 0 q_0) \rightarrow a(q_0 0 q_0)(q_0 0 q_0) \mid a(q_0 0 q_f)(q_f 0 q_0)$$
$$(q_0 0 q_f) \rightarrow a(q_0 0 q_0)(q_0 0 q_f) \mid a(q_0 0 q_f)(q_f 0 q_f)$$

$$(q_0 1 q_0) \rightarrow a$$
$$(q_0 0 q_0) \rightarrow b$$

$$(q_0 \$ q_f) \rightarrow \lambda$$

51

Derivation of string $abba$

$$(q_0 \$ q_f) \Rightarrow a(q_0 0 q_0)(q_0 \$ q_f) \Rightarrow$$
$$ab(q_0 \$ q_f) \Rightarrow$$
$$abb(q_0 1 q_0)(q_0 \$ q_f) \Rightarrow$$
$$abba(q_0 \$ q_f) \Rightarrow abba$$

52

In general:

$$(q_i A q_j) \stackrel{*}{\Rightarrow} w$$

if and only if

the NPDA goes from q_i to q_j
by reading string w and
the stack doesn't change below A
and then A is removed from stack

53

Therefore:

$$(q_0 \$ q_f) \stackrel{*}{\Rightarrow} w$$

if and only if

w is accepted by the NPDA

54

Therefore:

For any NPDA
there is a context-free grammar
that accepts the same language

$$\left\{\begin{array}{c}\text{Context-Free}\\ \text{Languages}\\ \text{(Grammars)}\end{array}\right\} \supseteq \left\{\begin{array}{c}\text{Languages}\\ \text{Accepted by}\\ \text{NPDAs}\end{array}\right\}$$

55

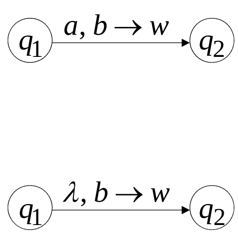
Deterministic PDA

DPDA

56

Deterministic PDA: DPDA

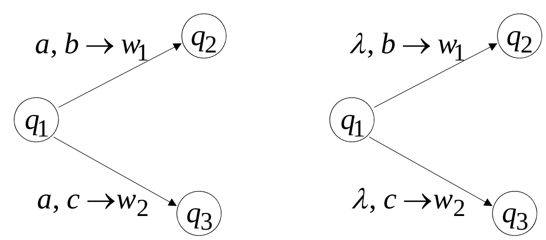
Allowed transitions:



(deterministic choices)

57

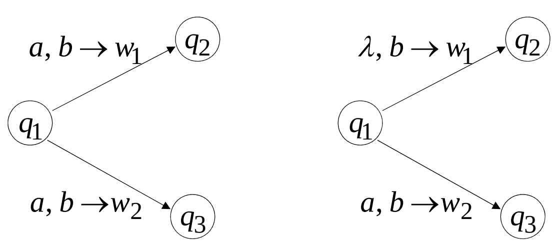
Allowed transitions:



(deterministic choices)

58

Not allowed:

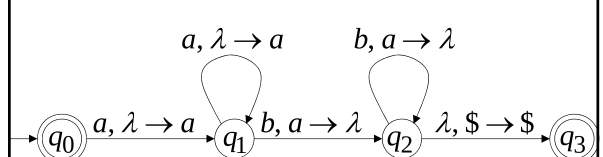


(non deterministic choices)

59

DPDA example

$$L(M) = \{a^n b^n : n \geq 0\}$$



60

The language $L(M) = \{a^n b^n : n \geq 0\}$

is **deterministic context-free**

61

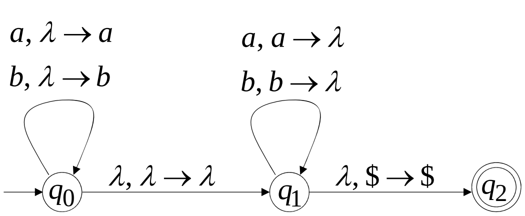
Definition:

A language L is **deterministic context-free** if there exists some DPDA that accepts it

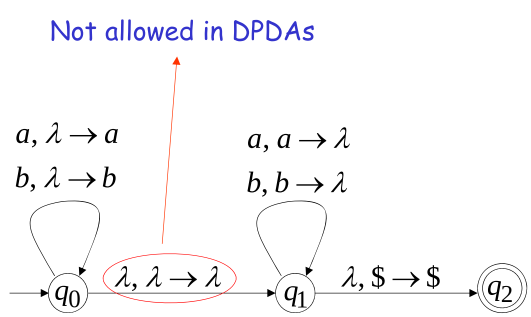
62

Example of Non-DPDA (NPDA)

$$L(M) = \{ww^R\}$$



63



64

NPDAs

Have More Power than

DPDAs

65

It holds that:

$\left\{ \text{Deterministic Context-Free Languages (DPDA)} \right\} \subseteq \left\{ \text{Context-Free Languages NPDAs} \right\}$

Since every DPDA is also a NPDA

66

We will actually show:

$$\underset{L \notin}{\left\{\begin{array}{c}\text{Deterministic} \\ \text{Context-Free} \\ \text{Languages} \\ \text{(DPDA)}\end{array}\right\}} \subset \underset{L \in}{\left\{\begin{array}{c}\text{Context-Free} \\ \text{Languages} \\ \text{(NPDA)}\end{array}\right\}}$$

We will show that there exists a context-free language L which is not accepted by any DPDA

67

The language is:

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

We will show:

- L is context-free
- L is **not** deterministic context-free

68

$$L = \{a^n b^n\} \cup \{a^n b^{2n}\}$$

Language L is context-free

Context-free grammar for L :

$S \rightarrow S_1 \mid S_2 \qquad \{a^n b^n\} \cup \{a^n b^{2n}\}$

$S_1 \rightarrow aS_1b \mid \lambda \qquad \{a^n b^n\}$

$S_2 \rightarrow aS_2bb \mid \lambda \qquad \{a^n b^{2n}\}$

69

Theorem:

The language $L = \{a^n b^n\} \cup \{a^n b^{2n}\}$ is **not** deterministic context-free

(there is **no** DPDA that accepts L)

70

Proof: Assume for contradiction that

$$L = \{a^n b^n\} \cup \{a^n b^{2n}\}$$

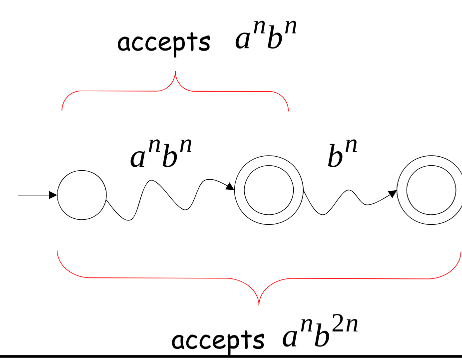
is deterministic context free

Therefore:

there is a DPDA M that accepts L

71

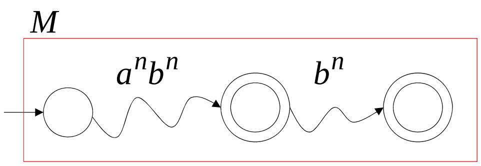
DPDA M with $L(M) = \{a^n b^n\} \cup \{a^n b^{2n}\}$



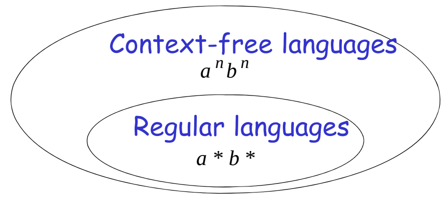
72

DPDA M with $L(M) = \{a^n b^n\} \cup \{a^n b^{2n}\}$

Such a path exists because of the determinism



Fact 1: The language $\{a^n b^n c^n\}$ is not context-free



(we will prove this at a later class using pumping lemma for context-free languages)

Fact 2: The language $L \cup \{a^n b^n c^n\}$ is not context-free

$$(L = \{a^n b^n\} \cup \{a^n b^{2n}\})$$

(we can prove this using pumping lemma for context-free languages)

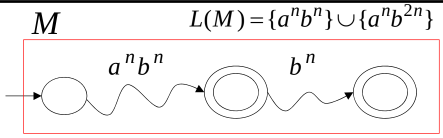
We will construct a NPDA that accepts:

$$L \cup \{a^n b^n c^n\}$$

$$(L = \{a^n b^n\} \cup \{a^n b^{2n}\})$$

which is a contradiction!

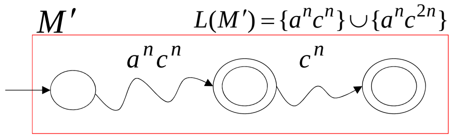
M $L(M) = \{a^n b^n\} \cup \{a^n b^{2n}\}$



Modify M

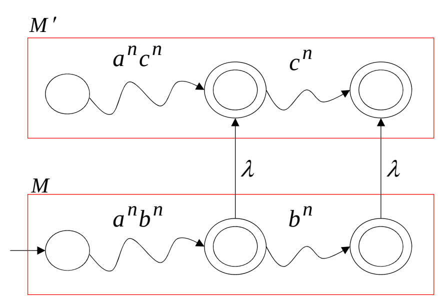
Replace b with c

M' $L(M') = \{a^n c^n\} \cup \{a^n c^{2n}\}$



The NPDA that accepts $L \cup \{a^n b^n c^n\}$

Connect final states of M' with final states of M



Since $L \cup \{a^n b^n c^n\}$ is accepted by a NPDA

it is context-free

Contradiction!

(since $L \cup \{a^n b^n c^n\}$ is not context-free)

Therefore:

Not deterministic context free

$$L = \{a^n b^n\} \cup \{a^n b^{2n}\}$$

There is **no** DPDA that accepts

End of Proof