DPDA: Deterministic PDA

At any moment, at most one move is possible:

$$\delta(q, a, b) = \{(p, c)\} \text{ or } \Theta$$

Deterministic PDA: DPDA

Allowed transitions: for a in Σ, b in Γ

$$q_1 \xrightarrow{a, b \to c} q_2$$

or

$$q_1 \xrightarrow{\varepsilon, b \to c} q_2$$

or

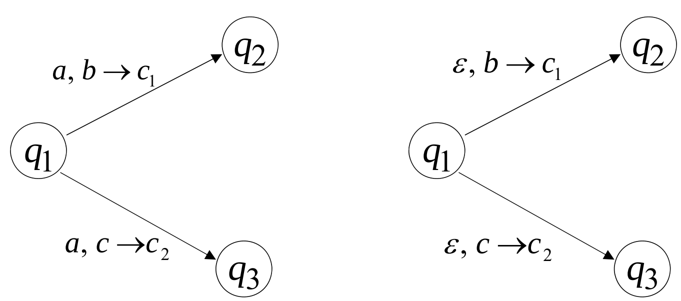
$$q_1 \xrightarrow{a, \varepsilon \to c} q_2$$

or

$$q_1 \xrightarrow{\varepsilon, \varepsilon \to c} q_2$$

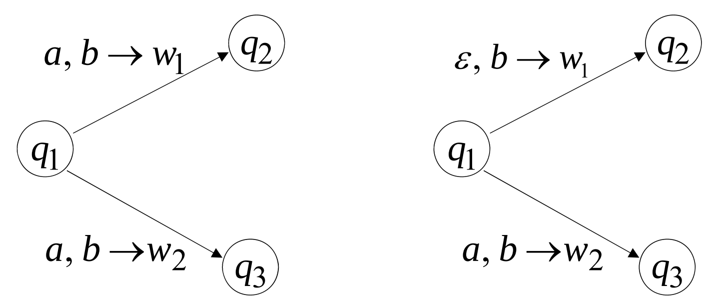
but only one of them is possible.

Allowed transitions:



(deterministic choices)

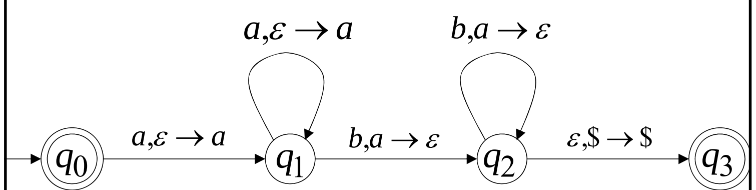
Not allowed:



(non deterministic choices)

DPDA example

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



Assume the stack initially has \$

Definition:

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

Example:

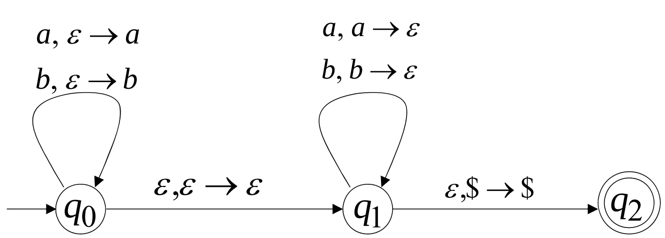
The language $L(M) = \{a^n b^n : n \geq 0\}$

is **deterministic context-free**

6

Example of Non-DPDA (PDA)

$$L(M) = \{ww^R : \ w \in \{a,b\}^*\}$$



Assume the stack initially has \$

7

PDAs Have More Power than DPDAs

It holds that:

$$\left\{ \begin{array}{c} \text{Deterministic} \\ \text{Context-Free} \\ \text{Languages} \\ \text{(DPDA)} \end{array} \right\} \subseteq \left\{ \begin{array}{c} \text{Context-Free} \\ \text{Languages} \\ \text{PDAs} \end{array} \right\}$$

Since every DPDA is also a PDA

8

We will actually show:

$$\left\{\begin{array}{c}\text{Deterministic}\\ \text{Context-Free}\\ \text{Languages}\\ \text{(DPDA)}\end{array}\right\} \subset \left\{\begin{array}{c}\text{Context-Free}\\ \text{Languages}\\ \text{(PDA)}\end{array}\right\}$$

$L \notin$ $\qquad$ $L \in$

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

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

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

Language L is context-free

Context-free grammar for L :

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

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

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

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)

Proof: Assume for contradiction that

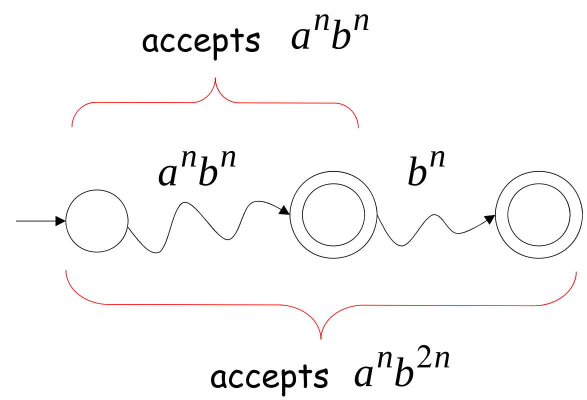
$$L = \{a^n b^n\} \cup \{a^n b^{2n}\}$$

is deterministic context free

Therefore:

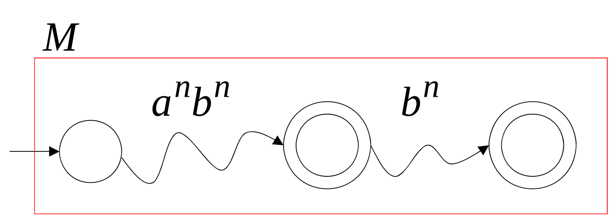
there is a DPDA M that accepts L

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

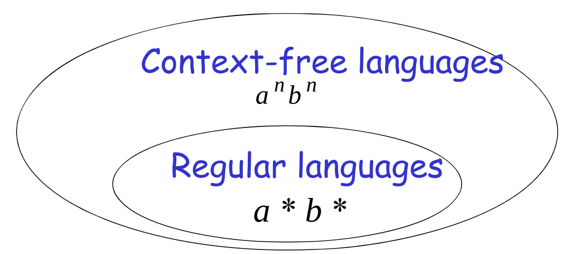


DPDA M with $L(M) = \{a^n b^n\} \cup \{a^n b^{2n}\}$

Such a path exists due to determinism



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



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

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

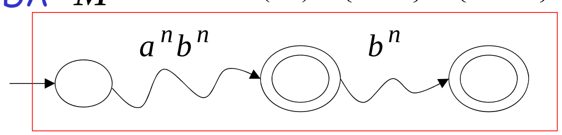
We will construct a PDA that accepts:

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

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

which is a contradiction!

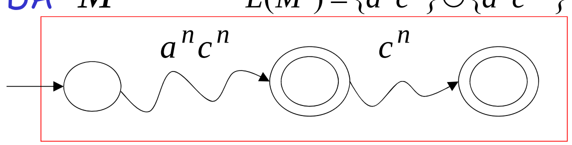
DPDA M $\quad L(M) = \{a^n b^n\} \cup \{a^n b^{2n}\}$



Modify M

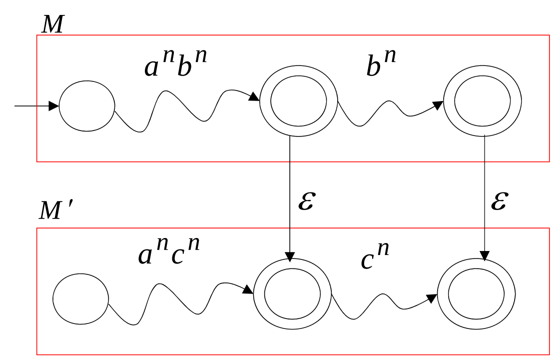
Replace b with c

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



A PDA that accepts $L \cup \{a^n b^n c^n\}$

Connect the final states of M with the final states of M'



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

it is context-free

Contradiction!

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

21

Therefore:

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

Is not deterministic context free

There is **no** DPDA that accepts it

End of Proof

22

Claim: DCFL is not closed under union and intersection.

DCFL: $L_1 = \{a^n b^n\}$ $L_2 = \{a^n b^{2n}\}$

Non-DCFL: $L = L_1 \cup L_2$

DCFL: $L_3 = \{a^n b^n c^m\}$ $L_4 = \{a^n b^m c^m\}$

Non-CFL: $L = L_3 \cap L_4$

23

Claim: DCFL is closed under complement. That is, if L is a DCFL, then so is $\overline{L}$.

Proof Idea: Given a DPDA M for L, we wish to swap accept/non-accept states of M to accept $\overline{L}$.

Three Problems:

1. M hangs before reading all symbols of w.
2. M loops before reading all symbols of w.
3. M accepts w by $\delta(q, \varepsilon, b) = \{(p, \varepsilon)\}$ where p is an accept state but q is not.

24

Three Problems:

1. M hangs before reading all symbols of w.

Solution:

Introduce states q_{init} and q_{dead}. A new initial stack symbol \$: $\delta(q_{init}, \varepsilon, \varepsilon) = \{(q_0, \$)\}$

Add $\delta(q_{dead}, a, \varepsilon) = \{(q_{dead}, \varepsilon)\}$ for any symbol a.

If adding $\delta(q, a, b) = \{ (q_{dead}, \varepsilon) \}$ doesn't cause non-deterministic choices, add it.

If $\delta(q, a, b)$ is defined for some stack symbol b, then define $\delta(q, a, \$) = \{ (q_{dead}, \varepsilon) \}$.

Three Problems:

2. M loops before reading all symbols of w.

Solution:

If $\delta(q, \varepsilon, b)$ leads to a loop, then change it as $\delta(q, \varepsilon, b) = \{ (q_{dead}, \varepsilon) \}$ if all states in the loop are non-accept states; otherwise, $\delta(q, \varepsilon, b) = \{ (q_{accept}, \varepsilon) \}$.

Three Problems:

3. M accepts w by $\delta(q, \varepsilon, b) = \{(p, \varepsilon)\}$ where p is an accept state but q is not.

Solution: Modify the original DPDA so that the states are divided into

Reading states: $\delta(q, a, \varepsilon) = \{(p, c)\}$

Non-reading states: $\delta(q, \varepsilon, b) = \{(p, c)\}$

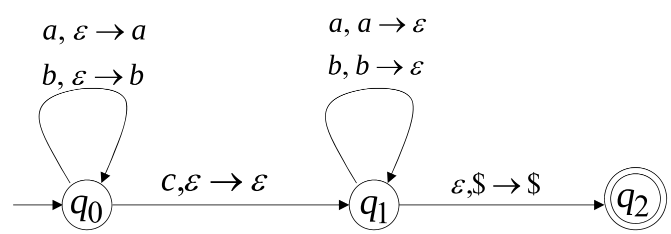
If both a and b are not ε in $\delta(q, a, b)$, then introduce q_b and replace $\delta(q, a, b) = \{ (p, c) \}$ by $\delta(q, \varepsilon, b) = \{ (q_b, \varepsilon) \}$ and $\delta(q_b, a, \varepsilon) = \{ (p, c) \}$.

Only reading states can be accept states.

When swapping states, only on reading states.

Example of DPDA

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



Assume the stack initially has \$

How to accept its complement?

Deterministic CF in Practice

- LR(k) grammar: it generates deterministic CFL, where L means "the input is read from Left to right", and R(k) means "Right most derivation decided by the first k input symbols".
- Most programming languages are specified by LR(k), where k ≤ 2.
- LR(k) Parser: an efficient algorithm to decide if a word is in L(G), where G is a LR(k) grammar.

Example: an LR(1) grammar

1) $E \to E+T$
2) $E \to T$
3) $T \to T*F$
4) $T \to F$
5) $F \to (E)$
6) $F \to a$

Rightmost derivation:
$E \Rightarrow E+T \Rightarrow E+F \Rightarrow E+a \Rightarrow T+a \Rightarrow T*F+a \Rightarrow T*a+a \Rightarrow F*a+a \Rightarrow a*a+a$

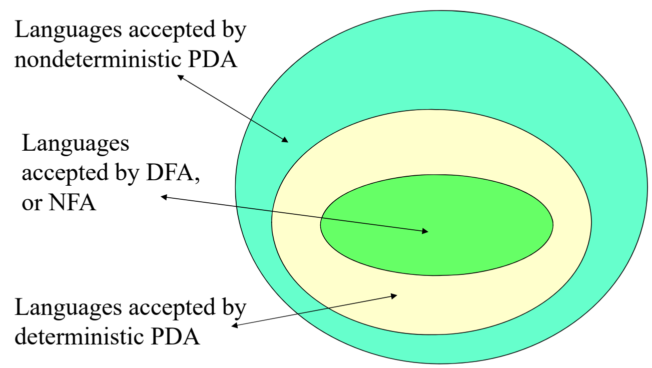
In the LR(k) parser, a stack is used to store the derivation steps backward.

Two actions on stack:
1. shift: move a symbol from input to the stack;
2. reduce: replace the right-hand side of a rule in the top of the stack by its left-hand side.

Actions of LR(1)-Parser -- Example

stack	input	action
	a*a+a$	shift
a	*a+a$	reduce by F→a
F	*a+a$	reduce by T→F
T	*a+a$	shift
T*	a+a$	shift
T*a	+a$	reduce by F→a
T*F	+a$	reduce by T→T*F
T	+a$	reduce by E→T
E	+a$	shift
E+	a$	shift
E+a	$	reduce by F→a
E+F	$	reduce by T→F
E+T	$	reduce by E→E+T
E	$	accept

A Hierarchy of Languages



Closure Properties

Operations	Regular	Context-free	Deterministic CF
union	yes	yes	no
concatenation	yes	yes	no
star	yes	yes	no
intersection	yes	no	no
complement	yes	no	yes
reversal	yes	yes	no
shuffle	yes	no	no

33

Reversal Operation

Given $w = a_1a_2\ldots a_n$, define $w^R = a_n \ldots a_2a_1$
Let $L^R = \{ w^R \mid w \text{ in } L \}$.

Claim: If L is regular, so is L^R

Claim: If L is context-free, so is L^R.

Claim: $L_1 = \{ ww^R \mid w \text{ in } (0+1)^* \}$ is context-free, but not regular.

34