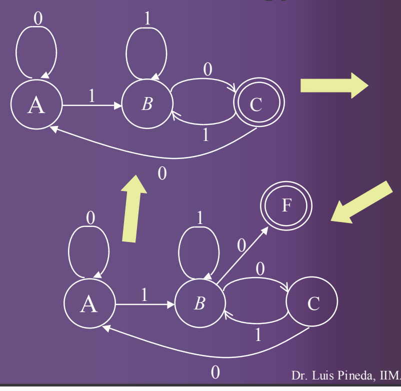
Session 21

CFG Corresponding to a PDA

PDA and CFG

- ✓ There is a PDA M such that $L(M) = L(G)$ for every CFG G
- There is a CFG G such that $L(G) = L(M)$ for every PDA M
- The set of *CFL* generated by CFG (ambiguous or unambiguous) is the set of *CFL* accepted by PDA.

An analogy with FA!



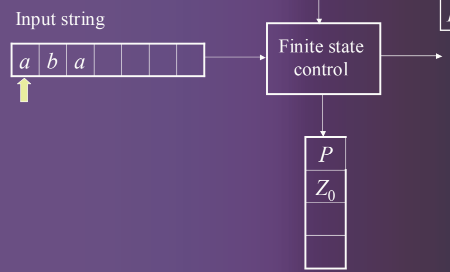
$A \to 0A$
$A \to 1B$
$B \to 0C \mid 0$
$B \to 1B$
$C \to 0A$
$C \to 1B$

A Top-down simulation of *Pal*

$M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$

The productions

$P \to aPa$
$P \to bPb$
$P \to a$
$P \to b$
$P \to \Lambda$



Top-down process of *Pal = aba*

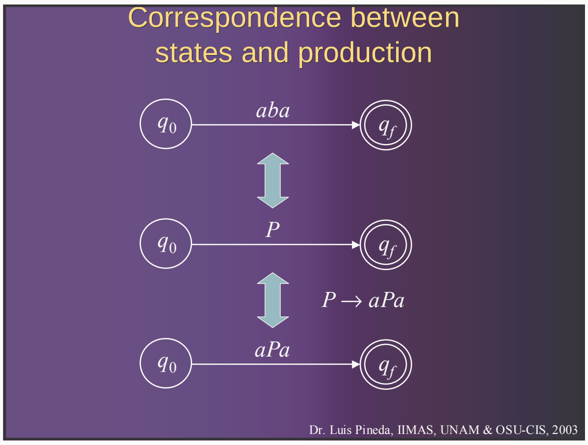
string	state	input	Stack	type: move	Conf.	Production
aba	q_0	Λ	Z_0	0: (q_1, P)	(q_0, aba, Z_0)	
aba	q_1	Λ	P	1: (q_1, aPa)	(q_1, aba, PZ_0)	$P \to aPa$
aba	q_1	a	a	2: (q_1, Λ)	$(q_1, aba, aPaZ_0)$	
ba	q_1	Λ	P	1: (q_1, b)	(q_1, ba, PaZ_0)	$P \to b$
ba	q_1	b	b	2: (q_1, Λ)	(q_1, ba, baZ_0)	
a	q_1	a	a	2: (q_1, Λ)	(q_1, a, aZ_0)	
Λ	q_1	Λ	Z_0	3: (q_2, Z_0)	(q_1, Λ, Z_0)	
Λ	q_2	Λ	Z_0		(q_2, Λ, Z_0)	

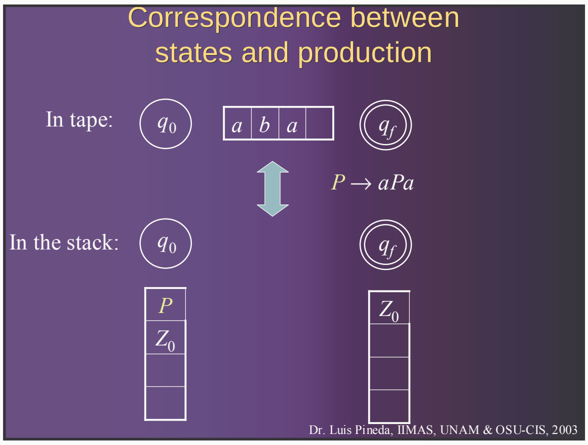
Productions correspond to moves of type 1

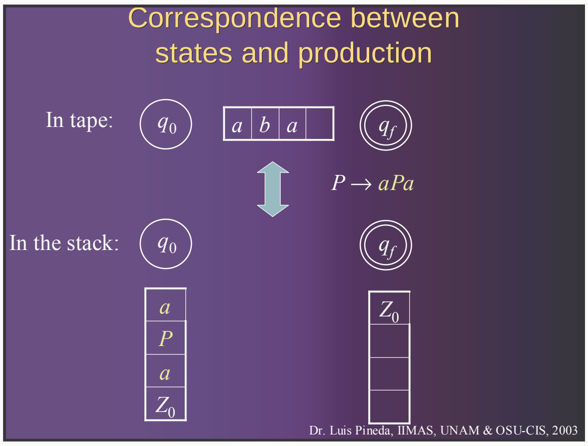
The grammar of the PDA!

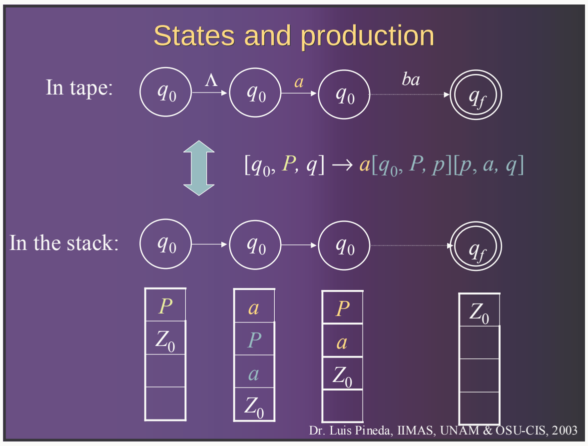
string	state	input	Stack	type: move	Conf.	Production
aba	q_0	Λ	Z_0	0: (q_1, P)	(q_0, aba, Z_0)	
aba	q_1	Λ	P	1: (q_1, aPa)	(q_1, aba, PZ_0)	$P \to aPa$
aba	q_1	a	a	2: (q_1, Λ)	$(q_1, aba, aPaZ_0)$	
ba	q_1	Λ	P	1: (q_1, b)	(q_1, ba, PaZ_0)	$P \to b$
ba	q_1	b	b	2: (q_1, Λ)	(q_1, ba, baZ_0)	
a	q_1	a	a	2: (q_1, Λ)	(q_1, a, aZ_0)	
Λ	q_1	Λ	Z_0	3: (q_2, Z_0)	(q_1, Λ, Z_0)	
Λ	q_2	Λ	Z_0		(q_2, Λ, Z_0)	

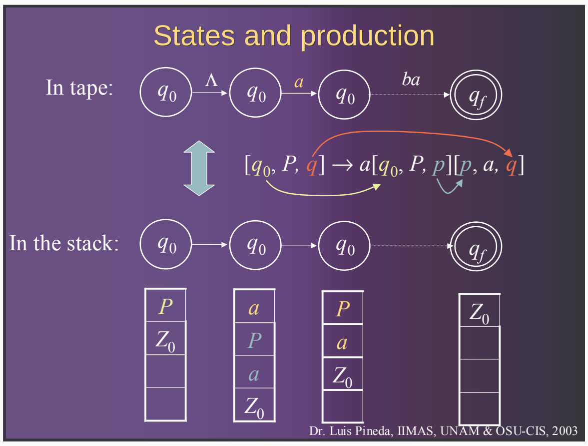
Is it possible to associate a production to each move?

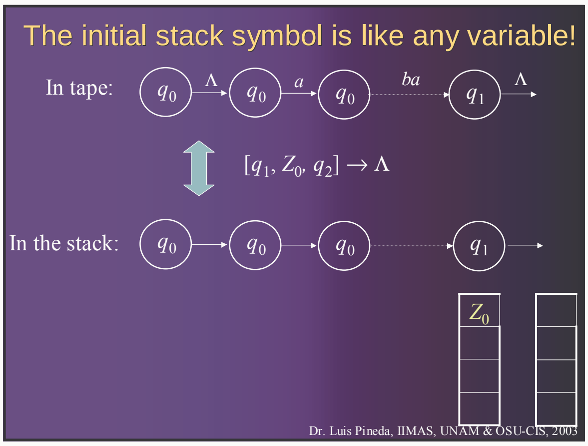

Correspondence between states and production
q0
aba
qf
q0
P
qf
P → aPa
q0
aPa
qf
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


Correspondence between states and production
In tape:
q0
a b a
qf
P → aPa
In the stack:
q0
qf
P
Z0
Z0
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


Correspondence between states and production
In tape:
q0
a b a
qf
P → aPa
In the stack:
q0
qf
a
P
a
Z0
Z0
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


States and production
In tape:
q0 Λ q0 a q0 ba qf
[q0, P, q] → a[q0, P, p][p, a, q]
In the stack:
q0 q0 q0 qf
P
Z0
a
P
a
Z0
P
a
Z0
Z0
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


States and production
In tape:
q0 Λ q0 a q0 ba qf
[q0, P, q] → a[q0, P, p][p, a, q]
In the stack:
q0 q0 q0 qf
P
Z0
a
P
a
Z0
P
a
Z0
Z0
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


The initial stack symbol is like any variable!
In tape:
q0 Λ q0 a q0 ba q1 Λ
[q1, Z0, q2] → Λ
In the stack:
q0 q0 q0 q1
Z0
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Finding a Grammar for a PDA

- Convert the PDA accepting by final state to the corresponding PDA accepting by empty stack
- Provide the set of variable symbols (i.e. Γ) :
  - Symbols correspond to variables in the grammar
  - Go into the stack in moves type 1 (i.e. pop symbol on top of the stack and push right side of production)
  - Match symbols of the input string in Σ in moves of type 2 (i.e. consume input and pop)
- Find the productions that correspond to each state and move of the PDA. These productions have the form $A \rightarrow a\alpha$ where
  - A is the top of the stack
  - a is the symbol in the input string
  - α is the string replacing the top of the stack by the move

Transition table of Pal_{mark}

Id	State	Input	Stack symbol	Move	
1	q_0	a	Z_0	(q_0, aZ_0)	
2	q_0	b	Z_0	(q_0, bZ_0)	
3	q_0	a	a	(q_0, aa)	
4	q_0	b	a	(q_0, ba)	
5	q_0	a	b	(q_0, ab)	
6	q_0	b	b	(q_0, bb)	
7	q_0	c	Z_0	(q_1, Z_0)	
8	q_0	c	a	(q_1, a)	
9	q_0	c	b	(q_1, b)	
10	q_1	a	a	(q_1, Λ)	
11	q_1	b	b	(q_1, Λ)	
12	q_1	Λ	Z_0	(q_2, Z_0)	*FS*
Other combinations				non	

Accepting by empty stack

Id	State	Input	Stack symbol	Move
1	q_0	a	Z_0	(q_0, aZ_0)
2	q_0	b	Z_0	(q_0, bZ_0)
3	q_0	a	a	(q_0, aa)
4	q_0	b	a	(q_0, ba)
5	q_0	a	b	(q_0, ab)
6	q_0	b	b	(q_0, bb)
7	q_0	c	Z_0	(q_1, Z_0)
8	q_0	c	a	(q_1, a)
9	q_0	c	b	(q_1, b)
10	q_1	a	a	(q_1, Λ)
11	q_1	b	b	(q_1, Λ)
12	q_1	Λ	Z_0	(q_1, Λ)
Other combinations				non

In push moves variables count symbols!

Id	State	Input	Stack symbol	Move
1	q_0	a	Z_0	(q_0, AZ_0)
2	q_0	b	Z_0	(q_0, BZ_0)
3	q_0	a	A	(q_0, AA)
4	q_0	b	A	(q_0, BA)
5	q_0	a	B	(q_0, AB)
6	q_0	b	B	(q_0, BB)
7	q_0	c	Z_0	(q_1, Z_0)
8	q_0	c	A	(q_1, A)
9	q_0	c	B	(q_1, B)
10	q_1	a	A	(q_1, Λ)
11	q_1	b	B	(q_1, Λ)
12	q_1	Λ	Z_0	(q_1, Λ)
Other combinations				non

The job of the productions

- For each symbols that is pushed in the first half of the string, there must be a symbol to cancel it with the corresponding symbol in the second half!
- The low of symbols preservation in *Pal*: symbols are never created or destroyed, they just cancel each other!
- The rule to move hypothesis: for each move there is a production!

The grammar: first version

Id	State	Input	Stack	Move	Productions
1	q_0	a	Z_0	(q_0, AZ_0)	$Z_0 \rightarrow aAZ_0$
2	q_0	b	Z_0	(q_0, BZ_0)	$Z_0 \rightarrow bBZ_0$
3	q_0	a	A	(q_0, AA)	$A \rightarrow aAA$
4	q_0	b	A	(q_0, BA)	$A \rightarrow bBA$
5	q_0	a	B	(q_0, AB)	$B \rightarrow aAB$
6	q_0	b	B	(q_0, BB)	$B \rightarrow bBB$
7	q_0	c	Z_0	(q_1, Z_0)	$Z_0 \rightarrow cZ_0$
8	q_0	c	A	(q_1, A)	$A \rightarrow cA$
9	q_0	c	B	(q_1, B)	$B \rightarrow cB$
10	q_1	a	A	(q_1, Λ)	$A \rightarrow a\Lambda$
11	q_1	b	B	(q_1, Λ)	$B \rightarrow b\Lambda$
12	q_1	Λ	Z_0	(q_1, Λ)	$Z_0 \rightarrow \Lambda$
Other combinations				non	

Top-down simulation

- We consider some typical moves:

Id	State	Input	Stack symbol	Move
1	q_0	a	Z_0	(q_0, AZ_0)
3	q_0	a	A	(q_0, AA)

- If we see an a (or a b) in the first half of the string, we push a variable matching such input, such that we will be able to pop the corresponding a in the second part!
- Conversely, to generate such a in the first part of the string and its corresponding a in the second we need rules of the form:

$$Z_0 \rightarrow aAZ_0 \text{ and } A \rightarrow aAA$$

Top-down simulation

- We consider some typical moves:

Id	State	Input	Stack symbol	Move
7	q_0	c	Z_0	(q_1, Z_0)
8	q_0	c	A	(q_1, A)

- If we see a c, we need to consume it and change state to process the second part of the string!
- Conversely, to generate such c we need rules of the form:

$$Z_0 \rightarrow cZ_0 \text{ and } A \rightarrow cA$$

Top-down simulation

- We consider some typical moves:

Id	State	Input	Stack symbol	Move
10	q_1	a	A	(q_1, Λ)

- If we see an a (or a b) in the second part, we need to pop it
- Conversely, to generate terminal a's we need a rule of the form:

$$A \rightarrow a \qquad \text{(i.e. } A \rightarrow a\Lambda)$$

Top-down simulation

- We consider some typical moves:

Id	State	Input	Stack symbol	Move
12	q_1	Λ	Z_0	(q_1, Λ)

- If we see Λ in the second part, we need to accept (by empty stack!)
- Conversely, to generate such Λ in the second half we need a rule of the form:

$$Z_0 \rightarrow \Lambda \qquad \text{(i.e. } Z_0 \rightarrow \Lambda\Lambda)$$

A leftmost derivation

- Derive *aca*:

$Z_0 \Rightarrow aAZ_0$ (by 1: $Z_0 \rightarrow aAZ_0$)

$\Rightarrow acAZ_0$ (by 8: $A \rightarrow cA$)

$\Rightarrow acaZ_0$ (by 10: $A \rightarrow a\Lambda$)

$\Rightarrow aca\Lambda = aca$ (by 12: $Z_0 \rightarrow \Lambda$)

A leftmost derivation

- However this grammar over-generates:

$Z_0 \Rightarrow aAZ_0$ (by 1: $Z_0 \rightarrow aAZ_0$)

$\Rightarrow aaZ_0$ (by 10: $A \rightarrow a\Lambda$)

$\Rightarrow aa\Lambda$ (by 12: $Z_0 \rightarrow \Lambda$)

  – but *aa* is not in this language!
  – Despite that *aa* is not accepted by the PDA:

$$(q_0, aa, Z_0) \Rightarrow (q_0, a, AZ_0)$$
$$\Rightarrow (q_0, \Lambda, AAZ_0)$$

- This grammar is missing something!

Preventing overgeneration

- Production rules are “bound” to states
  - Rule 1: Transition from state q_0 back to q_0
  - Rule 10: Transition from state q_1 back to q_1
  - Transition from q_0 to q_1: moves 7, 8 or 9

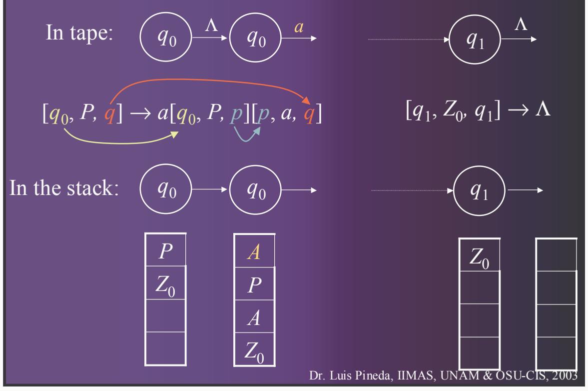
$$
\begin{array}{ll}
Z_0 \Rightarrow aAZ_0 & \text{(by 1)} \\
\quad \ldots & \text{(eventually 7, 8 or 9)} \\
\quad \Rightarrow aaZ_0 & \text{(by 10)} \\
\quad \Rightarrow aa\Lambda & \text{(by 12)}
\end{array}
$$

  - To use production 10, one of productions 7, 8 or 9 must have been used before in the derivation!
- To block over generation we need to codify in the productions of the grammar the corresponding states of the PDA!

The rules of the over-generation case!

Id	State	Input	Stack	Move	Productions
1	q_0	a	Z_0	(q_0, AZ_0)	$Z_0 \rightarrow aAZ_0$
2	q_0	b	Z_0	(q_0, BZ_0)	$Z_0 \rightarrow bBZ_0$
3	q_0	a	A	(q_0, AA)	$A \rightarrow aAA$
4	q_0	b	A	(q_0, BA)	$A \rightarrow bBA$
5	q_0	a	B	(q_0, AB)	$B \rightarrow aAB$
6	q_0	b	B	(q_0, BB)	$B \rightarrow bBB$
7	q_0	c	Z_0	(q_1, Z_0)	$Z_0 \rightarrow cZ_0$
8	q_0	c	A	(q_1, A)	$A \rightarrow cA$
9	q_0	c	B	(q_1, B)	$B \rightarrow cB$
10	q_1	a	A	(q_1, Λ)	$A \rightarrow a$
11	q_1	b	B	(q_1, Λ)	$B \rightarrow b$
12	q_1	Λ	Z_0	(q_1, Λ)	$Z_0 \rightarrow \Lambda$
Other combinations				non	

Codifying the states into the variables!



The grammar

Id	State	Input	Stack	Move	Productions
0					$S \rightarrow [q_0, Z_0, q]$
1	q_0	a	Z_0	(q_0, AZ_0)	$[q_0, Z_0, q] \rightarrow a[q_0, A, p][p, Z_0, q]$
2	q_0	b	Z_0	(q_0, BZ_0)	$[q_0, Z_0, q] \rightarrow b[q_0, B, p][p, Z_0, q]$
3	q_0	a	A	(q_0, AA)	$[q_0, A, q] \rightarrow a[q_0, A, p][p, A, q]$
4	q_0	b	A	(q_0, BA)	$[q_0, A, q] \rightarrow b[q_0, B, p][p, A, q]$
5	q_0	a	B	(q_0, AB)	$[q_0, B, q] \rightarrow a[q_0, A, p][p, B, q]$
6	q_0	b	B	(q_0, BB)	$[q_0, B, q] \rightarrow b[q_0, B, p][p, B, q]$
7	q_0	c	Z_0	(q_1, Z_0)	$[q_0, Z_0, q] \rightarrow c[q_1, Z_0, q]$
8	q_0	c	A	(q_1, A)	$[q_0, A, q] \rightarrow c[q_1, A, q]$
9	q_0	c	B	(q_1, B)	$[q_0, B, q] \rightarrow c[q_1, B, q]$
10	q_1	a	A	(q_1, Λ)	$[q_1, A, q_1] \rightarrow a$
11	q_1	b	B	(q_1, Λ)	$[q_1, B, q_1] \rightarrow b$
12	q_1	Λ	Z_0	(q_1, Λ)	$[q_1, Z_0, q_1] \rightarrow \Lambda$
Other combinations				non	

Accepting and generating *bacab*

- The moves:

$$
\begin{array}{l}
(q_0, bacab, Z_0) \Rightarrow (q_0, acab, BZ_0) \\
\qquad \Rightarrow (q_0, cab, ABZ_0) \\
\qquad \Rightarrow (q_1, ab, ABZ_0) \\
\qquad \Rightarrow (q_1, b, BZ_0) \\
\qquad \Rightarrow (q_1, \Lambda, Z_0) \\
\qquad \Rightarrow (q_1, \Lambda, \Lambda)
\end{array}
$$

Accepting and generating *bacab*

- The derivation:

$$S \Rightarrow [q_0, Z_0, q] \qquad (0)$$

Accepting and generating *bacab*

- The derivation:

$$\begin{aligned} S &\Rightarrow [q_0, Z_0, q] && (0)\\ &\Rightarrow b[q_0, B, p][p, Z_0, q] && (2) \end{aligned}$$

Production 2: $[q_0, Z_0, q] \rightarrow b[q_0, B, p][p, Z_0, q]$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{aligned} S &\Rightarrow [q_0, Z_0, q] && (0)\\ &\Rightarrow b[q_0, B, p][p, Z_0, q] && (2)\\ &\Rightarrow ba[q_0, A, r][r, B, p][p, Z_0, q] && (5) \end{aligned}$$

Production 5: $[q_0, B, q] \rightarrow a[q_0, A, p][p, B, q]$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{aligned} S &\Rightarrow [q_0, Z_0, q] && (0)\\ &\Rightarrow b[q_0, B, p][p, Z_0, q] && (2)\\ &\Rightarrow ba[q_0, A, r][r, B, p][p, Z_0, q] && (5)\\ &\Rightarrow bac[q_1, A, r][r, B, p][p, Z_0, q] && (8) \end{aligned}$$

Production 8: $[q_0, A, q] \rightarrow c[q_1, A, q]$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{aligned} S &\Rightarrow [q_0, Z_0, q] && (0)\\ &\Rightarrow b[q_0, B, p][p, Z_0, q] && (2)\\ &\Rightarrow ba[q_0, A, r][r, B, p][p, Z_0, q] && (5)\\ &\Rightarrow bac[q_1, A, r][r, B, p][p, Z_0, q] && (8)\\ &\Rightarrow baca[r, B, p][p, Z_0, q] && (10) \end{aligned}$$

Production 10: $[q_1, A, q_1] \rightarrow a$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{aligned} S &\Rightarrow [q_0, Z_0, q] && (0)\\ &\Rightarrow b[q_0, B, p][p, Z_0, q] && (2)\\ &\Rightarrow ba[q_0, A, q_1][q_1, B, p][p, Z_0, q] && (5)\\ &\Rightarrow bac[q_1, A, q_1][q_1, B, p][p, Z_0, q] && (8)\\ &\Rightarrow baca[q_1, B, p][p, Z_0, q] && (10) \end{aligned}$$

Production 10: r gets bound to q_1

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{aligned} S &\Rightarrow [q_0, Z_0, q] && (0)\\ &\Rightarrow b[q_0, B, p][p, Z_0, q] && (2)\\ &\Rightarrow ba[q_0, A, q_1][q_1, B, p][p, Z_0, q] && (5)\\ &\Rightarrow bac[q_1, A, q_1][q_1, B, p][p, Z_0, q] && (8)\\ &\Rightarrow baca[q_1, B, p][p, Z_0, q] && (10)\\ &\Rightarrow bacab[p, Z_0, q] && (11) \end{aligned}$$

Production 11: $[q_1, B, q_1] \rightarrow b$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{array}{lll} S \Rightarrow & [q_0, Z_0, q] & (0) \\ \Rightarrow & b[q_0, B, q_1][q_1, Z_0, q] & (2) \\ \Rightarrow & ba[q_0, A, q_1][q_1, B, q_1][q_1, Z_0, q] & (5) \\ \Rightarrow & bac[q_1, A, q_1][q_1, B, q_1][q_1, Z_0, q] & (8) \\ \Rightarrow & baca[q_1, B, q_1][q_1, Z_0, q] & (10) \\ \Rightarrow & bacab[q_1, Z_0, q] & (11) \end{array}$$

Production 11: p gets bound to q_1

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{array}{lll} S \Rightarrow & [q_0, Z_0, q] & (0) \\ \Rightarrow & b[q_0, B, q_1][q_1, Z_0, q] & (2) \\ \Rightarrow & ba[q_0, A, q_1][q_1, B, q_1][q_1, Z_0, q] & (5) \\ \Rightarrow & bac[q_1, A, q_1][q_1, B, q_1][q_1, Z_0, q] & (8) \\ \Rightarrow & baca[q_1, B, q_1][q_1, Z_0, q] & (10) \\ \Rightarrow & bacab[q_1, Z_0, q] & (11) \\ \Rightarrow & bacab\Lambda & (12) \end{array}$$

Production 12: $[q_1, Z_0, q_1] \rightarrow \Lambda$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{array}{lll} S \Rightarrow & [q_0, Z_0, q_1] & (0) \\ \Rightarrow & b[q_0, B, q_1][q_1, Z_0, q_1] & (2) \\ \Rightarrow & ba[q_0, A, q_1][q_1, B, q_1][q_1, Z_0, q_1] & (5) \\ \Rightarrow & bac[q_1, A, q_1][q_1, B, q_1][q_1, Z_0, q_1] & (8) \\ \Rightarrow & baca[q_1, B, q_1][q_1, Z_0, q_1] & (10) \\ \Rightarrow & bacab[q_1, Z_0, q_1] & (11) \\ \Rightarrow & bacab & (12) \end{array}$$

Production 12: q gets bound to q_1

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Accepting and generating *bacab*

- The derivation:

$$\begin{array}{lll} S \Rightarrow & [q_0, Z_0, q_1] & (0) \\ \Rightarrow & b[q_0, B, q_1][q_1, Z_0, q_1] & (2) \\ \Rightarrow & ba[q_0, A, q_1][q_1, B, q_1][q_1, Z_0, q_1] & (5) \\ \Rightarrow & bac[q_1, A, q_1][q_1, B, q_1][q_1, Z_0, q_1] & (8) \\ \Rightarrow & baca[q_1, B, q_1][q_1, Z_0, q_1] & (10) \\ \Rightarrow & bacab[q_1, Z_0, q_1] & (11) \\ \Rightarrow & bacab & (12) \end{array}$$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

PDA and CFG

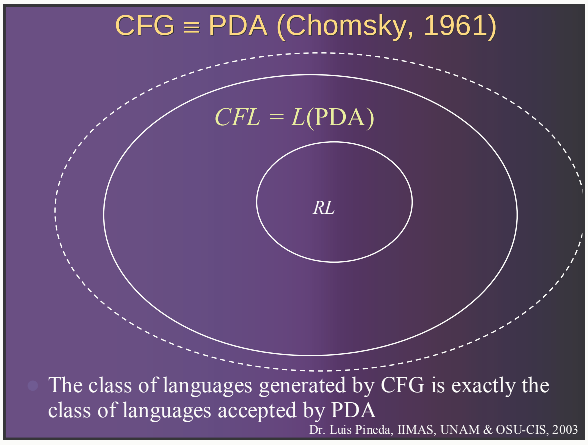
- ✓ There is a PDA M such that $L(M) = L(G)$ for every CFG G
- ✓ There is a CFG G such that $L(G) = L(M)$ for every PDA M
- The set of *CFL* generated by CFG (ambiguous or unambiguous) is the set of *CFL* accepted by PDA.

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

PDA and CFG

- ✓ There is a PDA M such that $L(M) = L(G)$ for every CFG G
- ✓ There is a CFG G such that $L(G) = L(M)$ for every PDA M
- ✓ The set of *CFL* generated by CFG (ambiguous or unambiguous) is the set of *CFL* accepted by PDA.

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


CFG ≡ PDA (Chomsky, 1961)
CFL = L(PDA)
RL
The class of languages generated by CFG is exactly the class of languages accepted by PDA
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003