## Session 20

PDA Corresponding to a CFG

## 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.



## CFG correspond to PDA

- What is the evidence so far:
  - $Pal$, $Pal_{even}$, $Pal_{mark}$?
  - What about languages with a more complex structure?
  - What about natural language?



## PDA corresponding to a 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 $M$



## A PDA for *Pal*

- The language:

  $Pal = \{x \mid x = x^r \in \{a, b\}^* \}$
- The grammar:

  $G_{pal} = (\{P\}, \{0, 1\}, P, P \rightarrow 0P0 \mid 1P1 \mid 1 \mid 0 \mid \Lambda\}$
- Define $M_{pal}$:

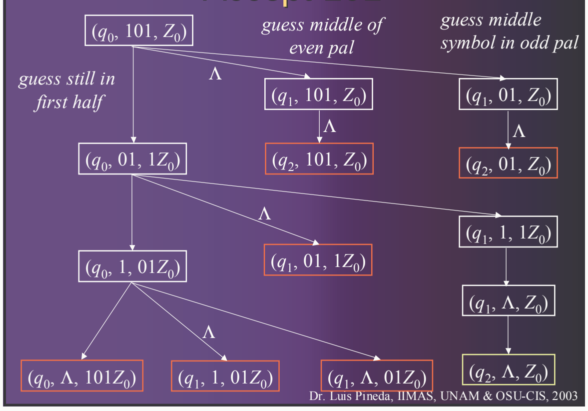
  $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, q_0, Z_0, \{q_2\}, \delta)$



## Transition function for *Pal*

| Id | State | Input | Stack symbol | Move(s) |
|---|---|---|---|---|
| 1 | $q_0$ | 0 | $Z_0$ | $(q_0, 0Z_0), (q_1, Z_0)$ |
| 2 | $q_0$ | 1 | $Z_0$ | $(q_0, 1Z_0), (q_1, Z_0)$ |
| 3 | $q_0$ | 0 | 0 | $(q_0, 00), (q_1, 0)$ |
| 4 | $q_0$ | 1 | 0 | $(q_0, 10), (q_1, 0)$ |
| 5 | $q_0$ | 0 | 1 | $(q_0, 01), (q_1, 1)$ |
| 6 | $q_0$ | 1 | 1 | $(q_0, 11), (q_1, 1)$ |
| 7 | $q_0$ | $\Lambda$ | $Z_0$ | $(q_1, Z_0)$ |
| 8 | $q_0$ | $\Lambda$ | 0 | $(q_1, 0)$ |
| 9 | $q_0$ | $\Lambda$ | 1 | $(q_1, 1)$ |
| 10 | $q_1$ | 0 | 0 | $(q_1, \Lambda)$ |
| 11 | $q_1$ | 1 | 1 | $(q_1, \Lambda)$ |
| 12 | $q_1$ | $\Lambda$ | $Z_0$ | $(q_2, Z_0)$ |
| Other combinations | | | | non |

## Accept 101






## PDA for CFG

- The PDA for *Pal* is too specific: hardwired!
- We need a general PDA that can process any string for any CFG
- We can store the productions in memory, and simulate the behavior of any grammar with a single "universal" PDA!



## PDA corresponding to a CFG

- The strategy:
  - Define a PDA that can simulate a derivation for every string $x$ in $L(G)$ for every CFG
  - For every string $x$ and CFG $G$ simulate the process of deriving $x$ using the productions in $P$ of $G$
- Advantages of a simulation:
  - Test membership of a string in a language
  - Reveal the steps of the derivation!
- Kind of PDA used in the simulation:
  - In general non-deterministic
  - In some (quite important) cases it will be possible to simulate a DPDA



## PDA corresponding to a CFG

- A step in the simulation corresponds to the construction of a portion of the derivation three, and this can achieved in two ways:
  - Top-down: expand the initial symbol $S$ all the way down to the string (leftmost derivations)
  - Bottom-up: build the tree upwards from the string $x$ into the initial symbol $S$ (rightmost derivations)



## PDA corresponding to a CFG

- A step in the simulation corresponds to the construction of a portion of the derivation three, and this can achieved in two ways:
  - Top-down: expand the initial symbol $S$ all the way down to the string (leftmost derivations)
  - Bottom-up: build the tree upwards from the string $x$ into the initial symbol $S$ (rightmost derivations)



## Top-down simulation

- Let $G = (V, \Sigma, S, P)$ be a CFG. The PDA

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

  is defined as follows:
  - $Q = \{q_0, q_1, q_2\}$
  - $A = \{q_2\}$
  - $\Gamma = V \cup \Sigma \cup Z_0$ where $Z_0 \notin V \cup \Sigma$
- Move 0:
  - Initial move to place $S$ on the stack and move to $q_1$:

    $\delta(q_0, \Lambda, Z_0) = \{(q_1, SZ_0)\}$
- Move 3:
  - Final move from $q_1$ to the accepting state $q_2$ when the stack is empty except for $Z_0$:

    $\delta(q_1, \Lambda, Z_0) = \{(q_2, Z_0)\}$

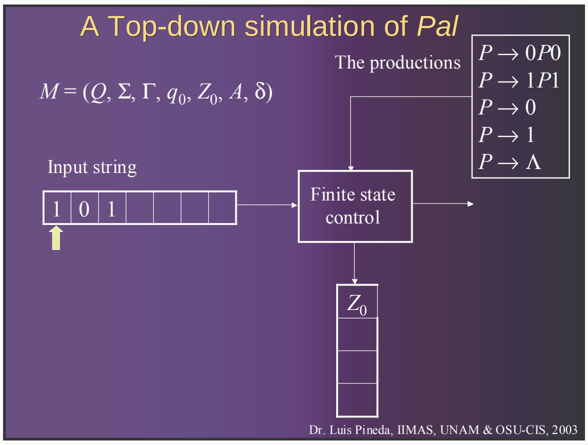
# Top-down simulation

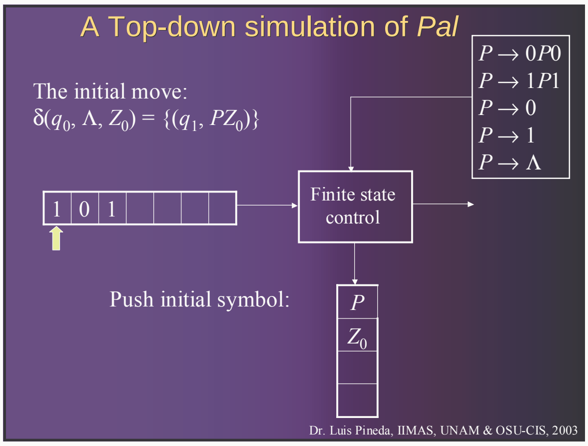
- Move 1:
  - If the left-side variable of a production in $P$ is on the top of the stack, substitute that variable for the string on the right-side of the production (on the top of the stack), without consuming a symbol from the input string:
  - For every $A \in V$,

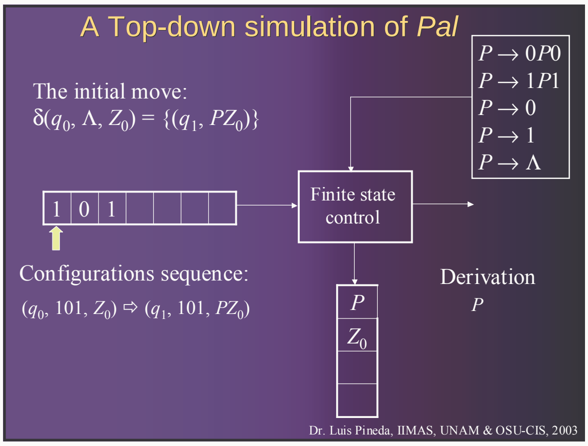
    $\delta(q_1, \Lambda, A) = \{(q_1, \alpha) \mid A \rightarrow \alpha \in P\}$
- Move 2:
  - If there is terminal symbol on the top of stack, such that it is the same symbol being scanned from the input string, pop the symbol from the stack, and consume the symbol in the input string:
  - For every $a \in \Sigma$,

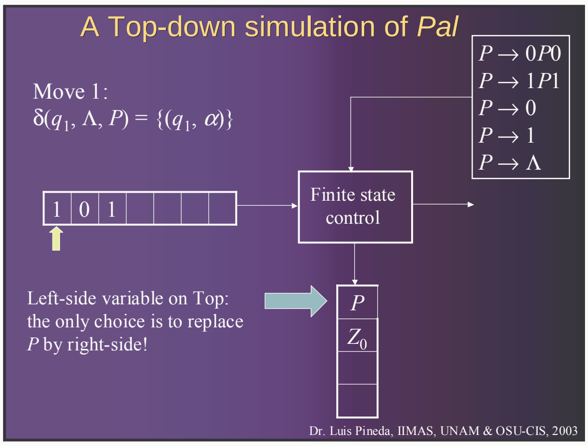
    $\delta(q_1, a, a) = \{(q_1, \Lambda)\}$

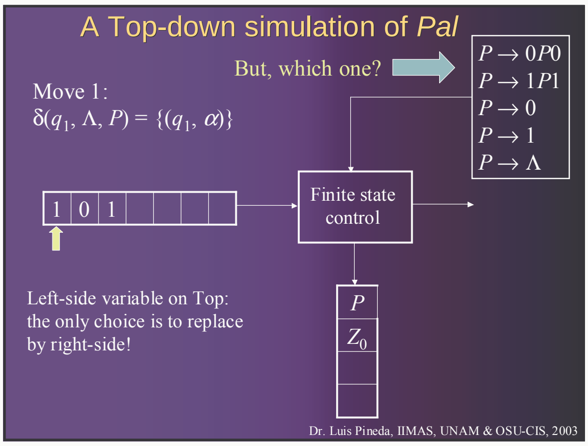


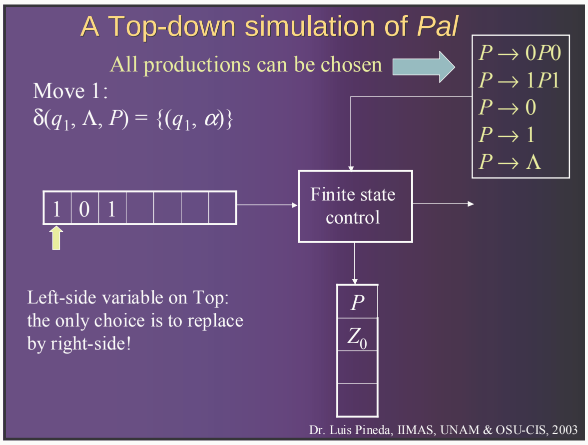


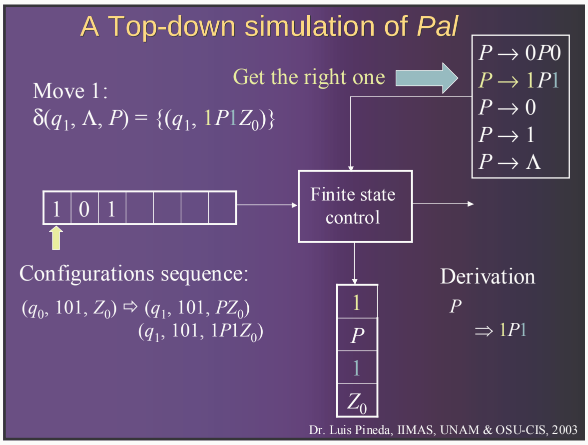



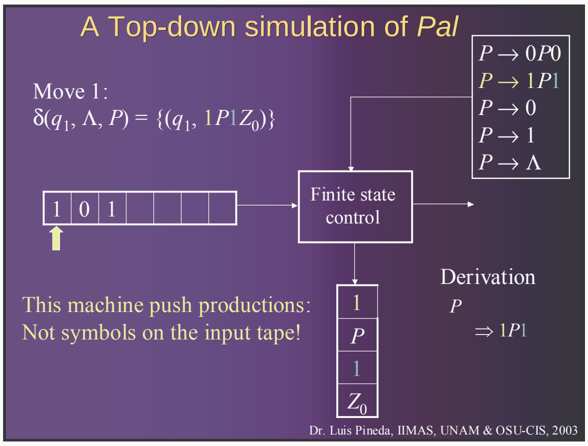



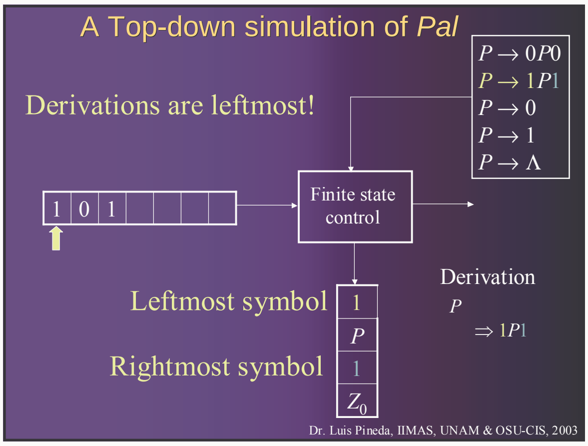



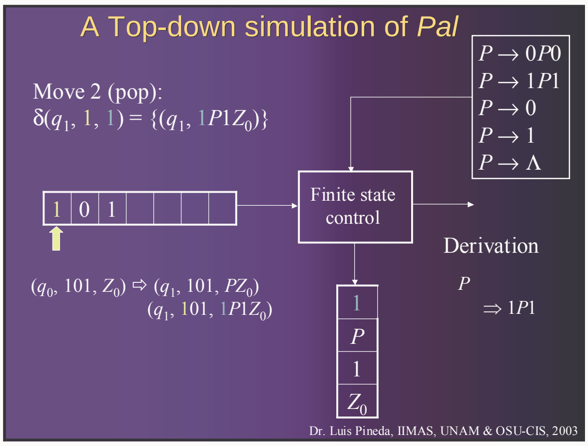



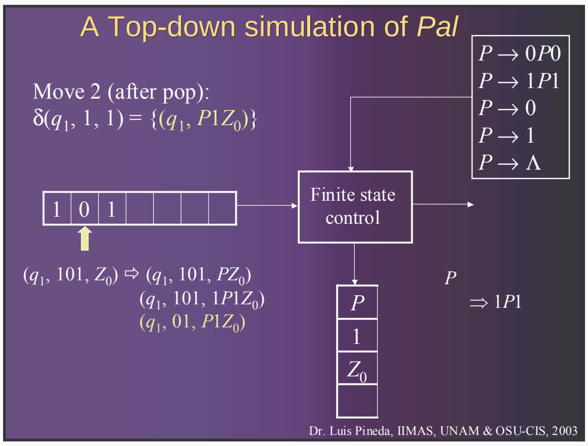

A Top-down simulation of Pal
All productions can be chosen
P → 0P0
P → 1P1
P → 0
P → 1
P → Λ
Move 1:
δ(q1, Λ, P) = {(q1, α)}
1 0 1
Finite state control
P
Z0
Left-side variable on Top:
the only choice is to replace
by right-side!
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Top-down simulation of Pal
Get the right one
P → 0P0
P → 1P1
P → 0
P → 1
P → Λ
Move 1:
δ(q1, Λ, P) = {(q1, 1P1Z0)}
1 0 1
Finite state control
Configurations sequence:
(q0, 101, Z0) ⇨ (q1, 101, PZ0)
(q1, 101, 1P1Z0)
1
P
1
Z0
Derivation
P
⇒ 1P1
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Top-down simulation of Pal
P → 0P0
P → 1P1
P → 0
P → 1
P → Λ
Move 1:
δ(q1, Λ, P) = {(q1, 1P1Z0)}
1 0 1
Finite state control
This machine push productions:
Not symbols on the input tape!
1
P
1
Z0
Derivation
P
⇒ 1P1
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Top-down simulation of Pal
P → 0P0
P → 1P1
P → 0
P → 1
P → Λ
Derivations are leftmost!
1 0 1
Finite state control
Leftmost symbol
Rightmost symbol
1
P
1
Z0
Derivation
P
⇒ 1P1
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Top-down simulation of Pal
P → 0P0
P → 1P1
P → 0
P → 1
P → Λ
Move 2 (pop):
δ(q1, 1, 1) = {(q1, 1P1Z0)}
1 0 1
Finite state control
(q0, 101, Z0) ⇨ (q1, 101, PZ0)
(q1, 101, 1P1Z0)
1
P
1
Z0
Derivation
P
⇒ 1P1
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Top-down simulation of Pal
P → 0P0
P → 1P1
P → 0
P → 1
P → Λ
Move 2 (after pop):
δ(q1, 1, 1) = {(q1, P1Z0)}
1 0 1
Finite state control
(q1, 101, Z0) ⇨ (q1, 101, PZ0)
(q1, 101, 1P1Z0)
(q1, 01, P1Z0)
P
1
Z0
P
⇒ 1P1
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

## A Top-down simulation of *Pal*

Move 1 again: $P \to \alpha$
$\delta(q_1, 0, P) = \{(q_1, \alpha 1 Z_0)\}$

$P \to 0P0$
$P \to 1P1$
$P \to 0$
$P \to 1$
$P \to \Lambda$

Input: 1 | 0 | 1

Finite state control

Stack: $P$, 1, $Z_0$

$(q_1, 101, Z_0) \Rightarrow (q_1, 101, PZ_0)$
$(q_1, 101, 1P1Z_0)$
$(q_1, 01, P1Z_0)$

$P$
$\Rightarrow 1P1$

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

## A Top-down simulation of *Pal*

Move 1 again (after substitution):
$\delta(q_1, \Lambda, P) = \{(q_1, 01Z_0)\}$

$P \to 0P0$
$P \to 1P1$
$P \to 0$
$P \to 1$
$P \to \Lambda$

Input: 1 | 0 | 1

Finite state control

Stack: 0, 1, $Z_0$

$(q_0, 101, Z_0) \Rightarrow (q_1, 101, PZ_0)$
$(q_1, 101, 1P1Z_0)$
$(q_1, 01, P1Z_0)$
$(q_1, 01, 01Z_0)$

$P$
$\Rightarrow 1P1$
$\Rightarrow 101$

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

## A Top-down simulation of *Pal*

Move 2 again (pop):
$\delta(q_1, 0, 0) = \{(q_1, \Lambda)\}$

$P \to 0P0$
$P \to 1P1$
$P \to 0$
$P \to 1$
$P \to \Lambda$

Input: 1 | 0 | 1

Finite state control

Stack: 0, 1, $Z_0$

$(q_0, 101, Z_0) \Rightarrow (q_1, 101, PZ_0)$
$(q_1, 101, 1P1Z_0)$
$(q_1, 01, P1Z_0)$
$(q_1, 01, 01Z_0)$

$P$
$\Rightarrow 1P1$
$\Rightarrow 101$

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

## A Top-down simulation of *Pal*

Move 2 again (after pop):
$\delta(q_1, 0, 0) = \{(q_1, \Lambda)\}$

$P \to 0P0$
$P \to 1P1$
$P \to 0$
$P \to 1$
$P \to \Lambda$

Input: 1 | 0 | 1

Finite state control

Stack: $\Lambda$, 1, $Z_0$

$(q_0, 101, Z_0) \Rightarrow (q_1, 101, PZ_0)$
$(q_1, 101, 1P1Z_0)$
$(q_1, 01, P1Z_0)$
$(q_1, 01, 01Z_0)$
$(q_1, 1, 1Z_0)$

$P$
$\Rightarrow 1P1$
$\Rightarrow 101$

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

## A Top-down simulation of *Pal*

Move 2 again (pop):
$\delta(q_1, 1, 1) = \{(q_1, \Lambda)\}$

$P \to 0P0$
$P \to 1P1$
$P \to 0$
$P \to 1$
$P \to \Lambda$

Input: 1 | 0 | 1

Finite state control

Stack: $\Lambda$, 1, $Z_0$

$(q_0, 101, Z_0) \Rightarrow (q_1, 101, PZ_0)$
$(q_1, 101, 1P1Z_0)$
$(q_1, 01, P1Z_0)$
$(q_1, 01, 01Z_0)$
$(q_1, 1, 1Z_0)$

$P$
$\Rightarrow 1P1$
$\Rightarrow 101$

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

## A Top-down simulation of *Pal*

Move 2 again (after pop):
$\delta(q_1, 1, 1) = \{(q_1, \Lambda)\}$

$P \to 0P0$
$P \to 1P1$
$P \to 0$
$P \to 1$
$P \to \Lambda$

Input: 1 | 0 | 1

Finite state control

Stack: $\Lambda$, $\Lambda$, $Z_0$

$(q_0, 101, Z_0) \Rightarrow (q_1, 101, PZ_0)$
$(q_1, 101, 1P1Z_0)$
$(q_1, 01, P1Z_0)$
$(q_1, 01, 01Z_0)$
$(q_1, 1, 1Z_0)$
$(q_1, \Lambda, Z_0)$

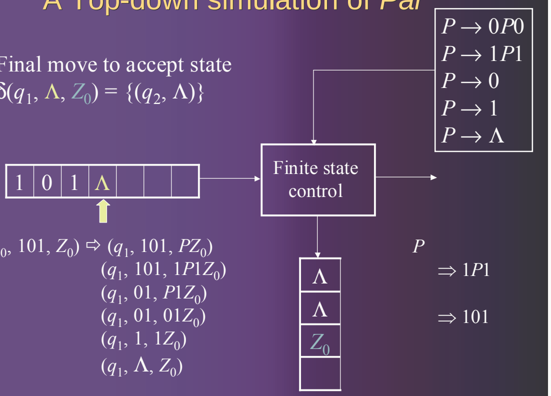
$P$
$\Rightarrow 1P1$
$\Rightarrow 101$

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

## A Top-down simulation of *Pal*

Final move to accept state

$\delta(q_1, \Lambda, Z_0) = \{(q_2, \Lambda)\}$




$(q_0, 101, Z_0) \Rightarrow (q_1, 101, PZ_0)$
$(q_1, 101, 1P1Z_0)$
$(q_1, 01, P1Z_0)$
$(q_1, 01, 01Z_0)$
$(q_1, 1, 1Z_0)$
$(q_1, \Lambda, Z_0)$

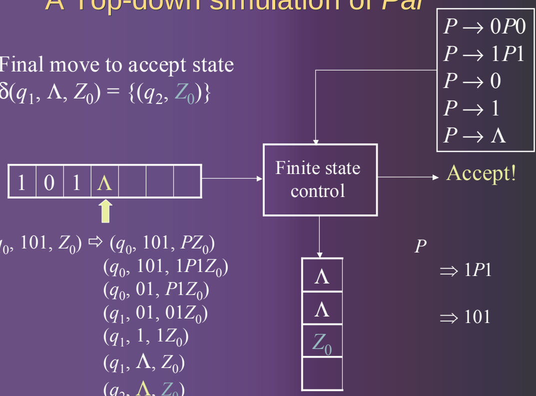
$P$
$\Rightarrow 1P1$
$\Rightarrow 101$



## A Top-down simulation of *Pal*

Final move to accept state

$\delta(q_1, \Lambda, Z_0) = \{(q_2, Z_0)\}$




$(q_0, 101, Z_0) \Rightarrow (q_0, 101, PZ_0)$
$(q_0, 101, 1P1Z_0)$
$(q_0, 01, P1Z_0)$
$(q_1, 01, 01Z_0)$
$(q_1, 1, 1Z_0)$
$(q_1, \Lambda, Z_0)$
$(q_2, \Lambda, Z_0)$

$P$
$\Rightarrow 1P1$
$\Rightarrow 101$



## Top-down process 101

| string | state | input | Stack | move | Conf. | Production |
|---|---|---|---|---|---|---|
| 101 | $q_0$ | $\Lambda$ | $Z_0$ | 0: $(q_1, P)$ | $(q_0, 101, Z_0)$ | |
| 101 | $q_1$ | $\Lambda$ | $P$ | 1: $(q_1, 1P1)$ | $(q_1, 101, PZ_0)$ | $P \to 1P1$ |
| 101 | $q_1$ | 1 | 1 | 2: $(q_1, \Lambda)$ | $(q_1, 101, 1P1Z_0)$ | |
| 01 | $q_1$ | $\Lambda$ | $P$ | 1: $(q_1, 0)$ | $(q_1, 01, P1Z_0)$ | $P \to 0$ |
| 01 | $q_1$ | 0 | 0 | 2: $(q_1, \Lambda)$ | $(q_1, 01, 01Z_0)$ | |
| 1 | $q_1$ | 1 | 1 | 2: $(q_1, \Lambda)$ | $(q_1, 1, 1Z_0)$ | |
| $\Lambda$ | $q_1$ | $\Lambda$ | $Z_0$ | 3: $(q_2, Z_0)$ | $(q_1, \Lambda, Z_0)$ | |
| $\Lambda$ | $q_2$ | $\Lambda$ | $Z_0$ | | $(q_2, \Lambda, Z_0)$ | |



## Top-down simulation CFG

- A step in the simulation corresponds to the construction of a portion of the derivation tree.
- Top-down: expand the initial symbol $S$ all the way down to the string (leftmost derivations)
- The sequence of configuration of the PDA correspond to a leftmost derivation in the grammar!
- A configuration sequence reaching the final state corresponds to a leftmost derivation of a string in the language of the CFG
- If the PDA behaves deterministically (there is only one leftmost derivation) the grammar is unambiguous!



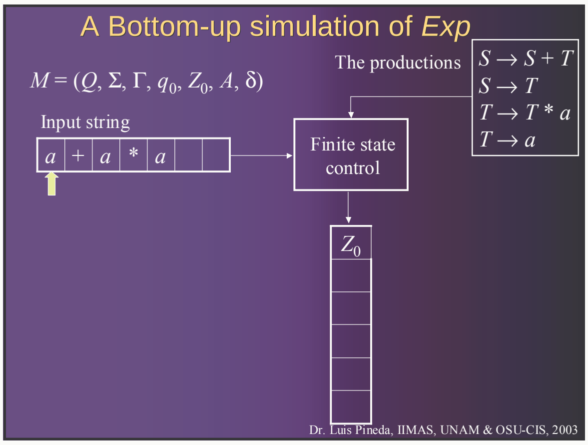
## PDA corresponding to a CFG

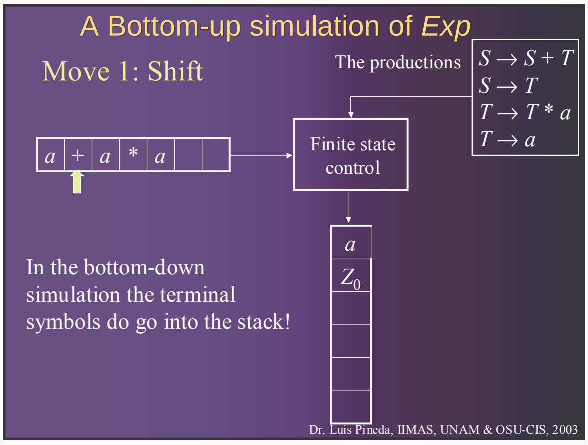
- A step in the simulation corresponds to the construction of a portion of the derivation tree, and this can achieved in two ways:
  - ✓ Top-down: expand the initial symbol $S$ all the way down to the string (leftmost derivations)
  - – Bottom-up: build the tree upwards from the string $x$ into the initial symbol $S$ (rightmost derivations)

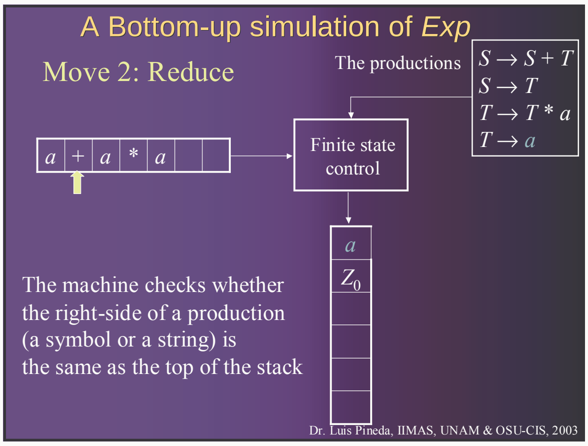


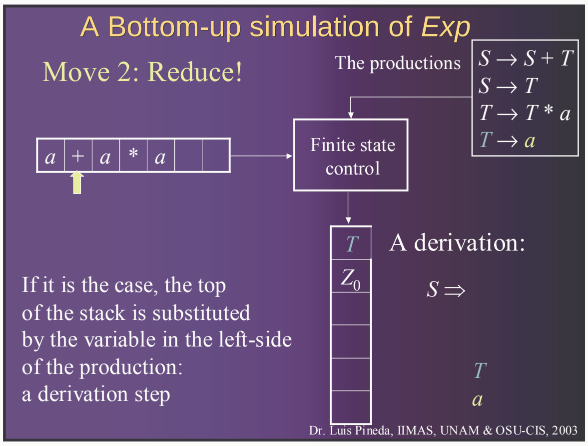
## PDA corresponding to a CFG

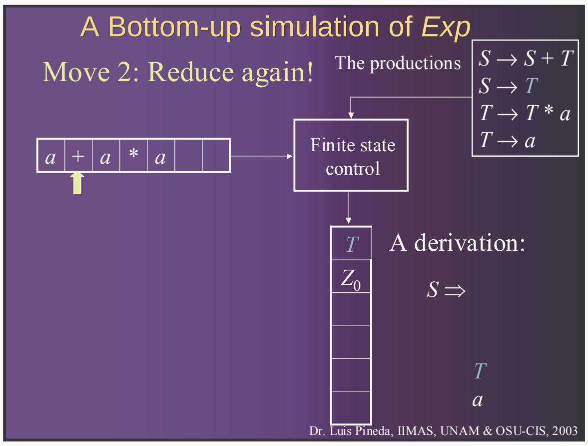
- A step in the simulation corresponds to the construction of a portion of the derivation tree, and this can achieved in two ways:
  - ✓ Top-down: expand the initial symbol $S$ all the way down to the string (leftmost derivations)
  - – Bottom-up: build the tree upwards from the string $x$ into the initial symbol $S$ (rightmost derivations)

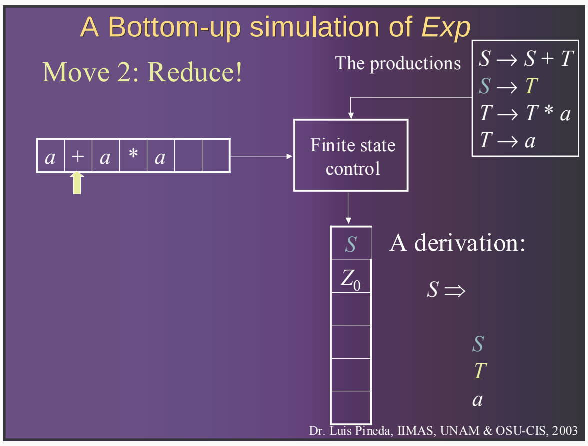

A Bottom-up simulation of *Exp*
The productions
$S \rightarrow S + T$
$S \rightarrow T$
$T \rightarrow T * a$
$T \rightarrow a$
$M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$
Input string
a + a * a
Finite state control
$Z_0$
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

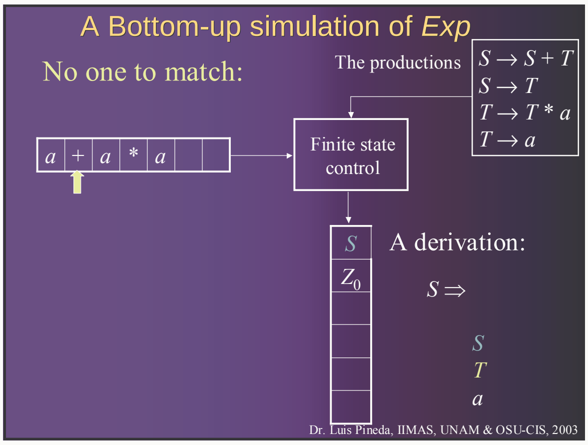

A Bottom-up simulation of *Exp*
Move 1: Shift
The productions
$S \rightarrow S + T$
$S \rightarrow T$
$T \rightarrow T * a$
$T \rightarrow a$
a + a * a
Finite state control
a
$Z_0$
In the bottom-down simulation the terminal symbols do go into the stack!
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

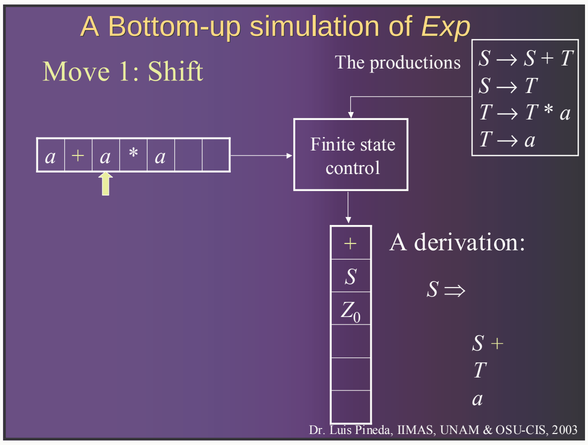

A Bottom-up simulation of *Exp*
Move 2: Reduce
The productions
$S \rightarrow S + T$
$S \rightarrow T$
$T \rightarrow T * a$
$T \rightarrow a$
a + a * a
Finite state control
a
$Z_0$
The machine checks whether the right-side of a production (a symbol or a string) is the same as the top of the stack
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

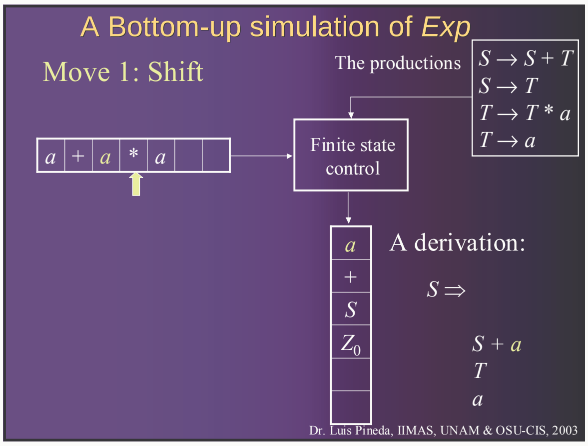

A Bottom-up simulation of *Exp*
Move 2: Reduce!
The productions
$S \rightarrow S + T$
$S \rightarrow T$
$T \rightarrow T * a$
$T \rightarrow a$
a + a * a
Finite state control
T
$Z_0$
A derivation:
$S \Rightarrow$
T
a
If it is the case, the top of the stack is substituted by the variable in the left-side of the production: a derivation step
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

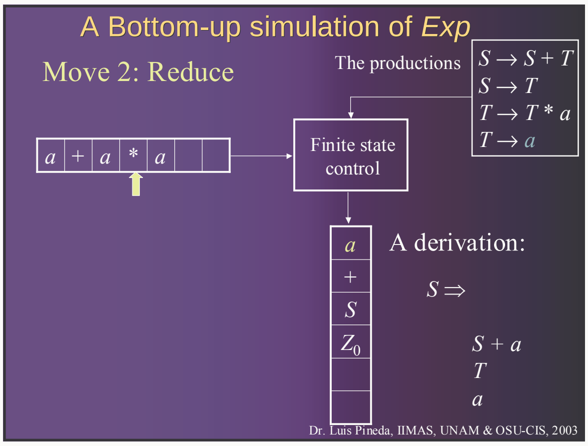

A Bottom-up simulation of *Exp*
Move 2: Reduce again!
The productions
$S \rightarrow S + T$
$S \rightarrow T$
$T \rightarrow T * a$
$T \rightarrow a$
a + a * a
Finite state control
T
$Z_0$
A derivation:
$S \Rightarrow$
T
a
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

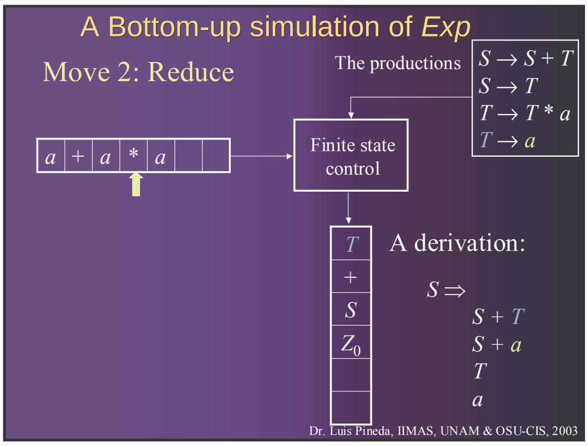

A Bottom-up simulation of *Exp*
Move 2: Reduce!
The productions
$S \rightarrow S + T$
$S \rightarrow T$
$T \rightarrow T * a$
$T \rightarrow a$
a + a * a
Finite state control
S
$Z_0$
A derivation:
$S \Rightarrow$
S
T
a
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

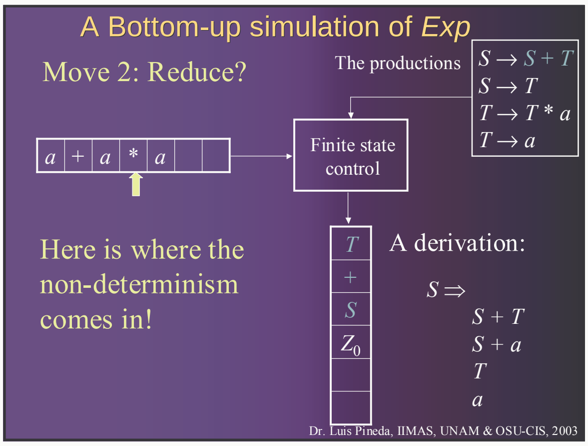

A Bottom-up simulation of Exp
No one to match:
The productions
S → S + T
S → T
T → T * a
T → a
a + a * a
Finite state control
S
Z0
A derivation:
S ⇒
S
T
a
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

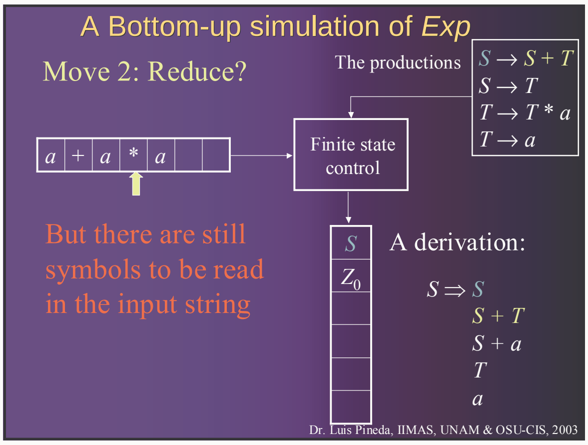

A Bottom-up simulation of Exp
Move 1: Shift
The productions
S → S + T
S → T
T → T * a
T → a
a + a * a
Finite state control
+
S
Z0
A derivation:
S ⇒
S +
T
a
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

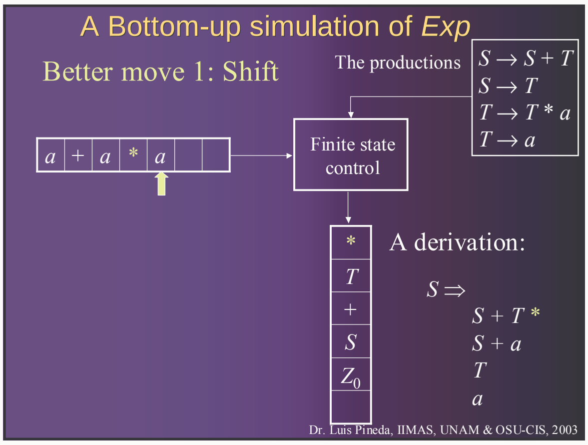

A Bottom-up simulation of Exp
Move 1: Shift
The productions
S → S + T
S → T
T → T * a
T → a
a + a * a
Finite state control
a
+
S
Z0
A derivation:
S ⇒
S + a
T
a
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

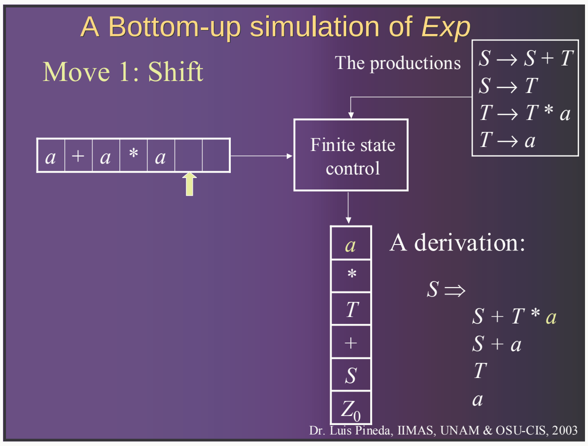

A Bottom-up simulation of Exp
Move 2: Reduce
The productions
S → S + T
S → T
T → T * a
T → a
a + a * a
Finite state control
a
+
S
Z0
A derivation:
S ⇒
S + a
T
a
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

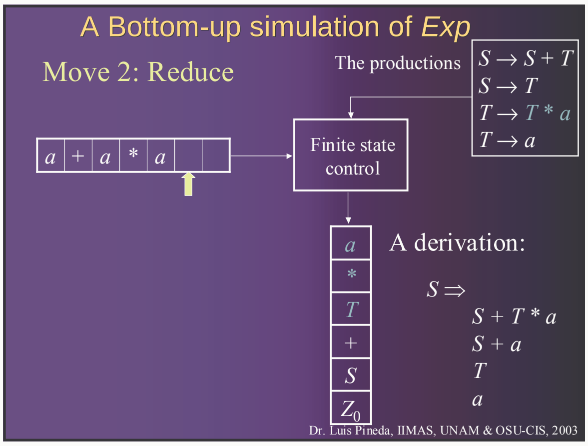

A Bottom-up simulation of Exp
Move 2: Reduce
The productions
S → S + T
S → T
T → T * a
T → a
a + a * a
Finite state control
T
+
S
Z0
A derivation:
S ⇒
S + T
S + a
T
a
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

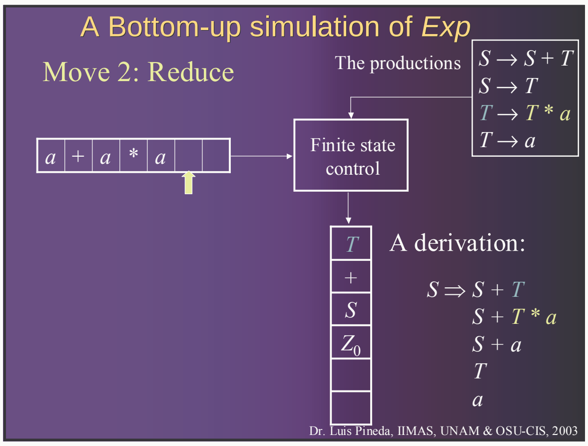

A Bottom-up simulation of Exp
Move 2: Reduce?
The productions
S → S + T
S → T
T → T * a
T → a
a + a * a
Finite state control
Here is where the non-determinism comes in!
T
+
S
Z0
A derivation:
S ⇒
S + T
S + a
T
a
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

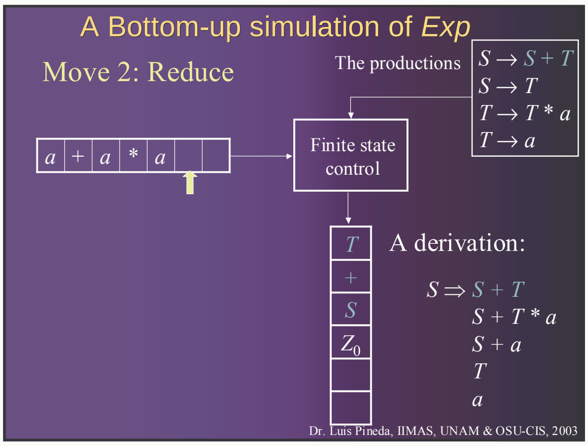

A Bottom-up simulation of Exp
Move 2: Reduce?
The productions
$S \to S + T$
$S \to T$
$T \to T * a$
$T \to a$
a + a * a
Finite state control
But there are still symbols to be read in the input string
S
$Z_0$
A derivation:
$S \Rightarrow S$
$S + T$
$S + a$
$T$
$a$
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Bottom-up simulation of Exp
Better move 1: Shift
The productions
$S \to S + T$
$S \to T$
$T \to T * a$
$T \to a$
a + a * a
Finite state control
*
T
+
S
$Z_0$
A derivation:
$S \Rightarrow$
$S + T *$
$S + a$
$T$
$a$
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Bottom-up simulation of Exp
Move 1: Shift
The productions
$S \to S + T$
$S \to T$
$T \to T * a$
$T \to a$
a + a * a
Finite state control
a
*
T
+
S
$Z_0$
A derivation:
$S \Rightarrow$
$S + T * a$
$S + a$
$T$
$a$
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Bottom-up simulation of Exp
Move 2: Reduce
The productions
$S \to S + T$
$S \to T$
$T \to T * a$
$T \to a$
a + a * a
Finite state control
a
*
T
+
S
$Z_0$
A derivation:
$S \Rightarrow$
$S + T * a$
$S + a$
$T$
$a$
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003


A Bottom-up simulation of Exp
Move 2: Reduce
The productions
$S \to S + T$
$S \to T$
$T \to T * a$
$T \to a$
a + a * a
Finite state control
T
+
S
$Z_0$
A derivation:
$S \Rightarrow S + T$
$S + T * a$
$S + a$
$T$
$a$
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

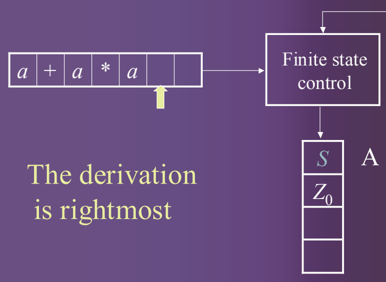

A Bottom-up simulation of Exp
Move 2: Reduce
The productions
$S \to S + T$
$S \to T$
$T \to T * a$
$T \to a$
a + a * a
Finite state control
T
+
S
$Z_0$
A derivation:
$S \Rightarrow S + T$
$S + T * a$
$S + a$
$T$
$a$
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

## A Bottom-up simulation of *Exp*

Move 2: Reduce

The productions

$S \rightarrow S + T$
$S \rightarrow T$
$T \rightarrow T * a$
$T \rightarrow a$




The derivation is rightmost

A derivation:

$$S \Rightarrow S + T$$
$$S + T * a$$
$$S + a$$
$$T$$
$$a$$

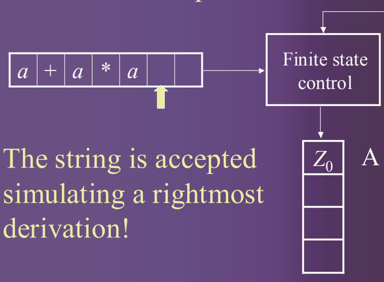


## A Bottom-up simulation of *Exp*

Final Move: Pop *S*

The productions

$S \rightarrow S + T$
$S \rightarrow T$
$T \rightarrow T * a$
$T \rightarrow a$




The string is accepted simulating a rightmost derivation!

A derivation:

$$S \Rightarrow S + T$$
$$S + T * a$$
$$S + a$$
$$T$$
$$a$$



## Bottom-up process *a + a * a*

| Id | Move | Production | Stack | Unread Input |
|---|---|---|---|---|
| 1 | | | $Z_0$ | $a + a * a$ |
| 2 | Shift | | $aZ_0$ | $+ a * a$ |
| 3 | reduce | $T \rightarrow a$ | $TZ_0$ | $+ a * a$ |
| 4 | reduce | $S \rightarrow T$ | $SZ_0$ | $+ a * a$ |
| 5 | Shift | | $+SZ_0$ | $a * a$ |
| 6 | Shift | | $a+SZ_0$ | $* a$ |
| 7 | reduce | $T \rightarrow a$ | $T+SZ_0$ | $* a$ |
| 8 | Shift | | $*T+SZ_0$ | a |
| 9 | Shift | | $a*T+SZ_0$ | – |
| 10 | reduce | $T \rightarrow T * a$ | $T+SZ_0$ | – |
| 11 | reduce | $S \rightarrow S + T$ | $SZ_0$ | – |
| 12 | (pop S) | | $Z_0$ | – |
| 13 | accept | | | |



## Bottom-up simulation

- A step in the simulation corresponds to the construction of a portion of the derivation tree.
- Bottom-up: build the tree upwards from the string *x* into the initial symbol *S*
- The sequence of configuration of the PDA correspond to a rightmost derivation in the grammar!
- A configuration sequence reaching the final state corresponds to a rightmost derivation of a string in the language of the CFG
- If the PDA behaves deterministically (there is only one rightmost derivation) the grammar is unambiguous!



## PDA corresponding to a CFG

- A step in the simulation corresponds to the construction of a portion of the derivation three, and this can achieved in two ways:
  - ✓ Top-down: expand the initial symbol *S* all the way down to the string (leftmost derivations)
  - ✓ Bottom-up: build the tree upwards from the string *x* into the initial symbol *S* (rightmost derivations)



## PDA corresponding to a 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*