CS 6041
Theory of Computation

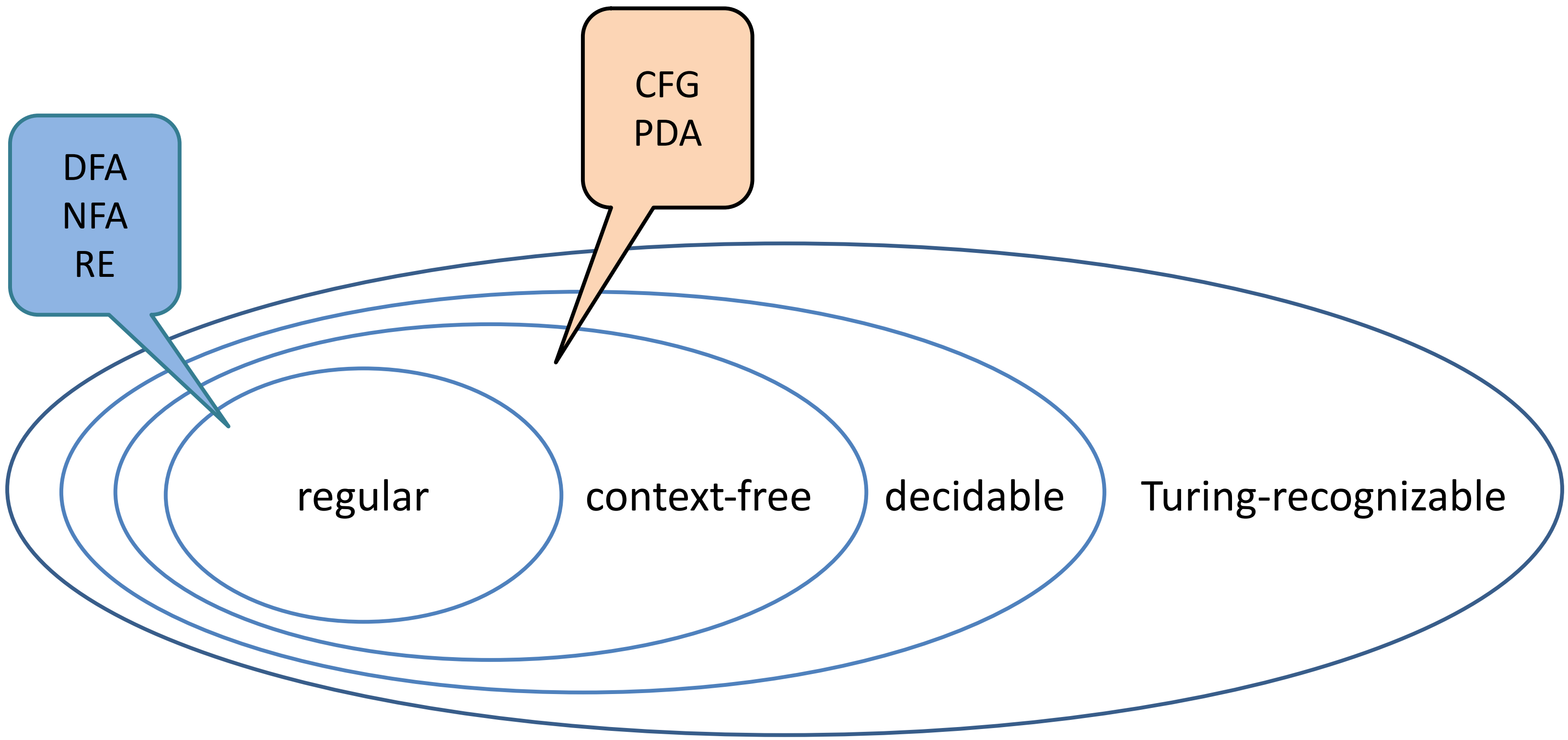
Pushdown Automata

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Pushdown Automata (PDA)



Equivalence of PDA and CFG

- Theorem: A language is context free **if and only if** some pushdown automaton recognizes it

- **A language is CFL $\Longrightarrow$ some PDA recognizes it**

- A language is CFL $\Longleftarrow$ some PDA recognizes it

A language is CFL $\Longrightarrow$ some PDA recognizes it

- Grammar G_1:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

If a PDA can recognize strings in one CFL, done!

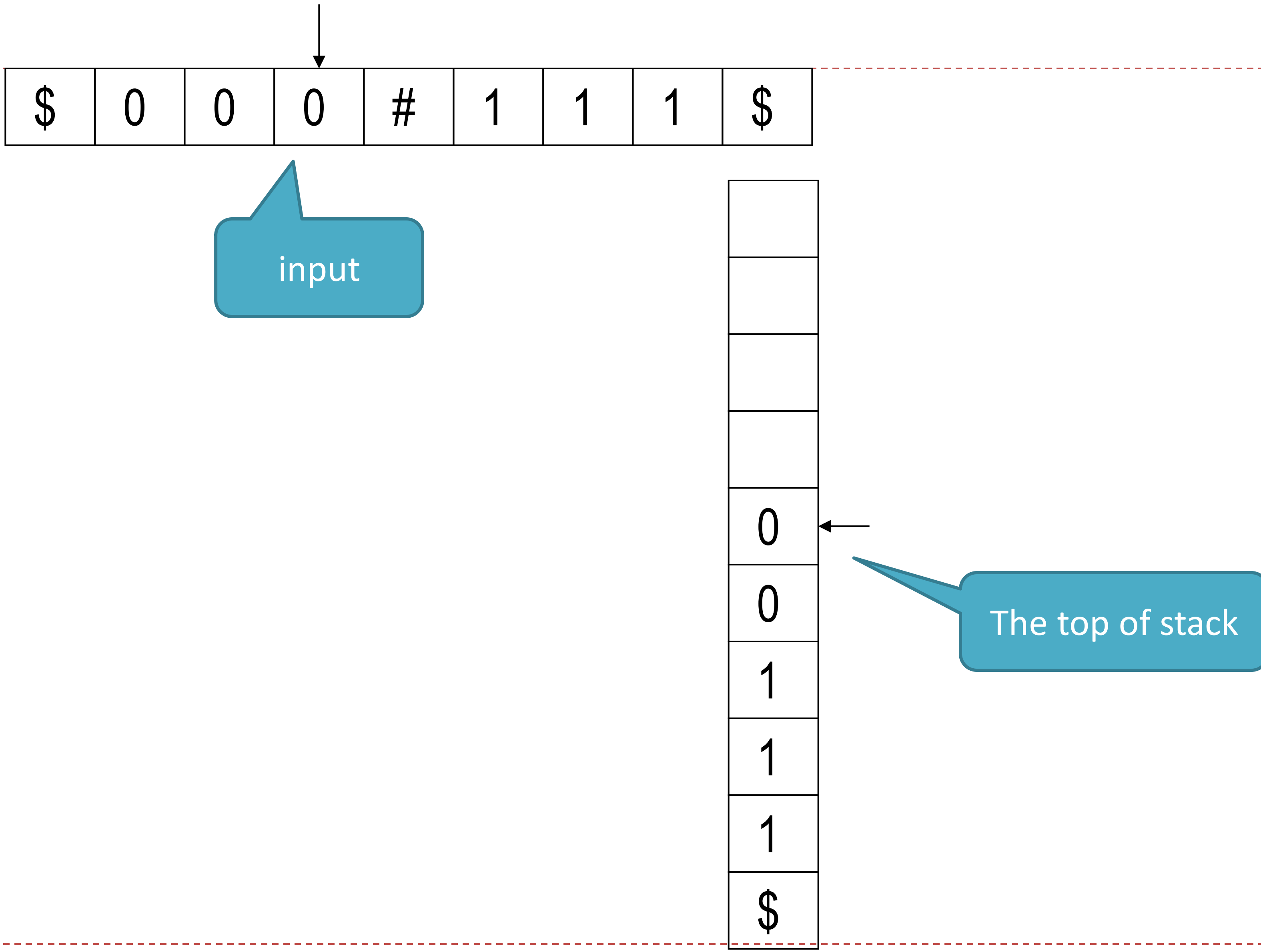
- Derivation: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

How to create a PDA to recognize 000#111

A language is CFL $\Longrightarrow$ some PDA recognizes it

- Details: how PDA recognizes a string of a CFL
  - **Step 1: Compare the input with the top of stack**
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

Step 1: Compare the input with the top of stack

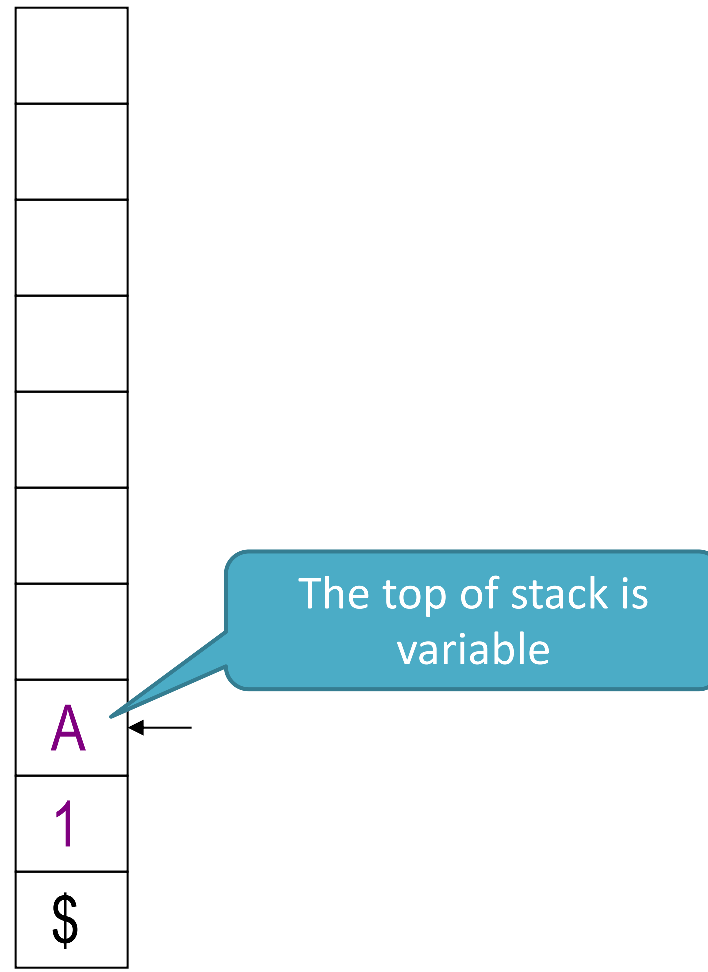


A language is CFL $\Longrightarrow$ some PDA recognizes it

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - **Step 2: If the top of stack is variable, then simulate the derivation**
  - Step 3: If the top of stack is terminal, then do the match

Step 2: If the top of stack is variable, then simulate the derivation

\$	0	0	0	#	1	1	1	\$



- Grammar G_1:

$$A \rightarrow 0A1$$

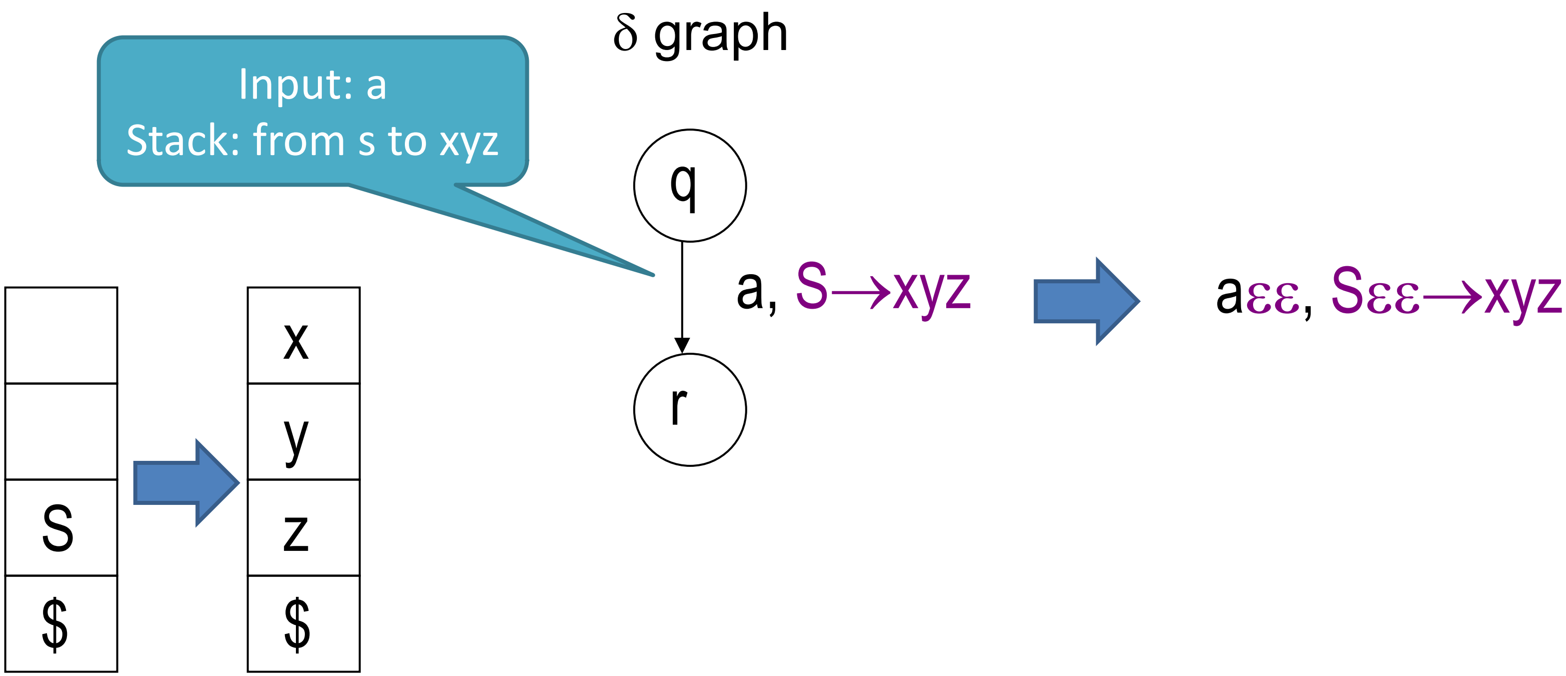
$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1$

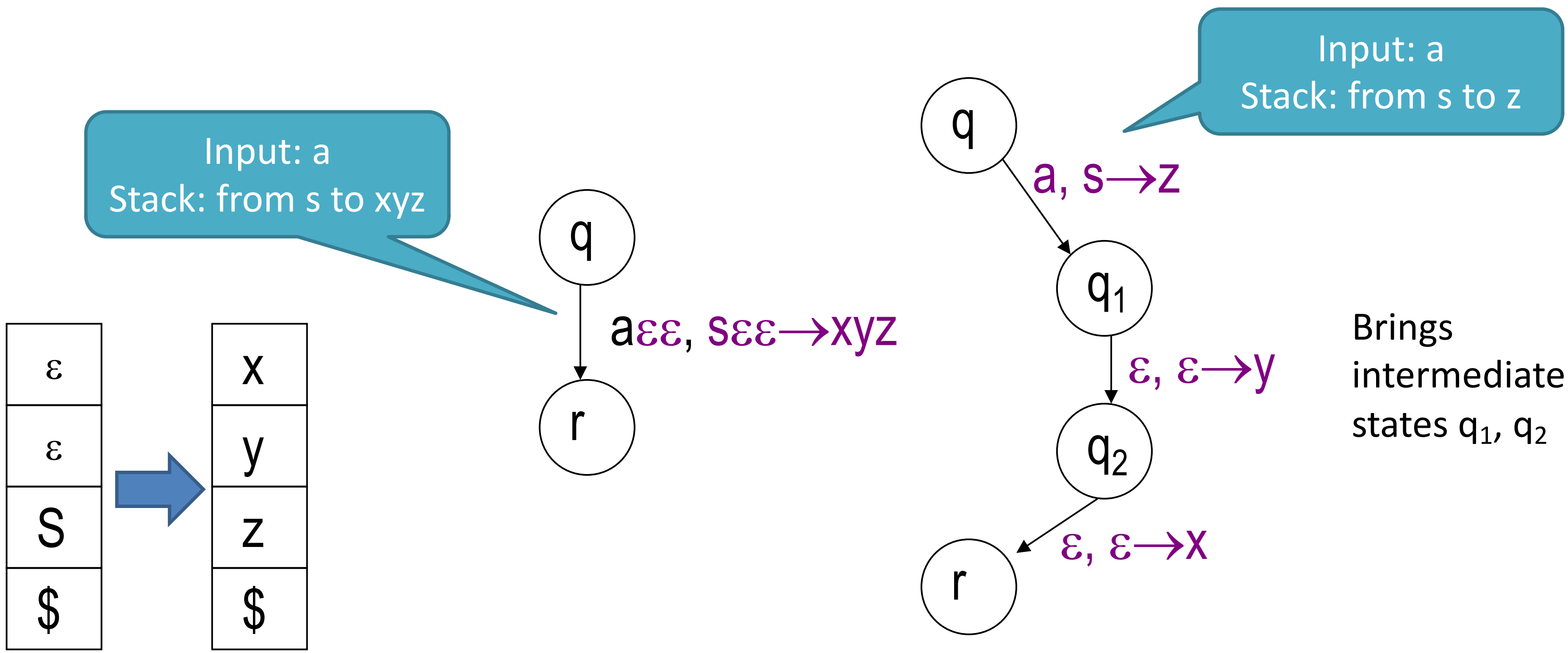
Step 2: If the top of stack is variable, then simulate the derivation

- Simulate the derivation $S \Rightarrow xyz$ based on rule $S \rightarrow xyz$



Step 2: If the top of stack is variable, then simulate the derivation

- Simulate the derivation $S \Rightarrow xyz$ based on rule $S \rightarrow xyz$

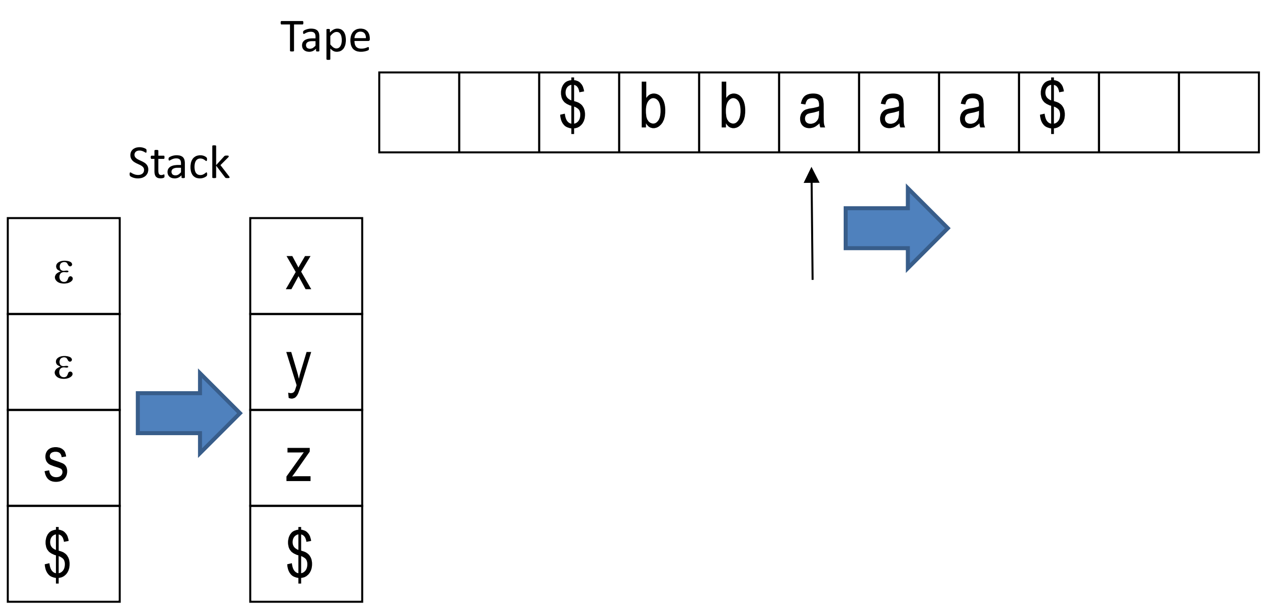


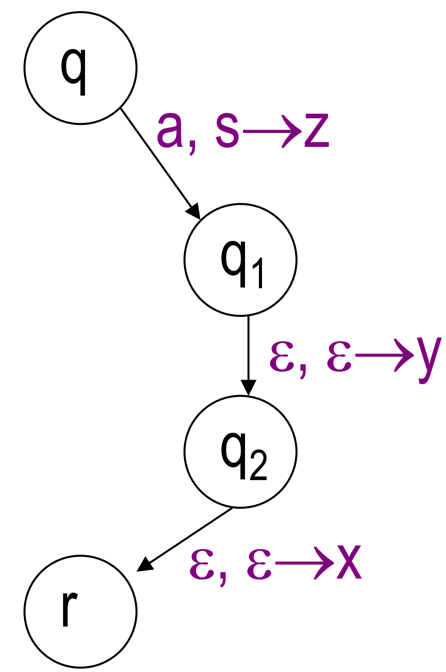
Step 2: If the top of stack is variable, then simulate the derivation

- E.g., we need derivation S $\Rightarrow$ xyz

  Just Pop 'S', Push 'z','y','x'

  Input header just move by one 'a'

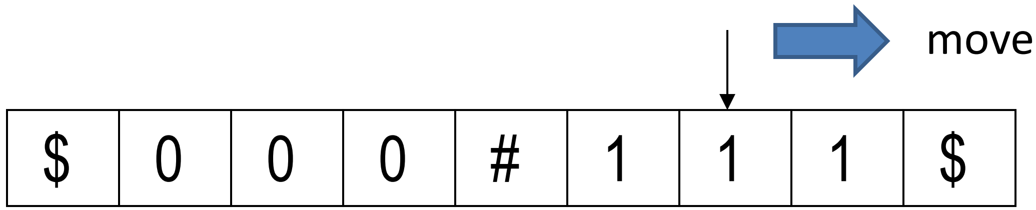




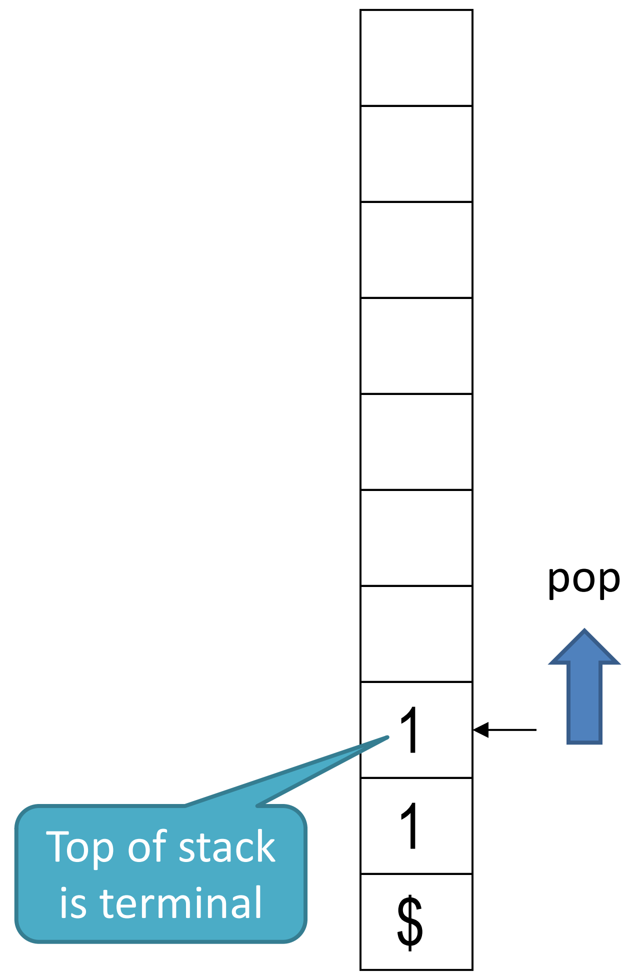
A language is CFL $\Longrightarrow$ some PDA recognizes it

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - **Step 3: If the top of stack is terminal, then do the match**

Step 3: If the top of stack is terminal, then do the match



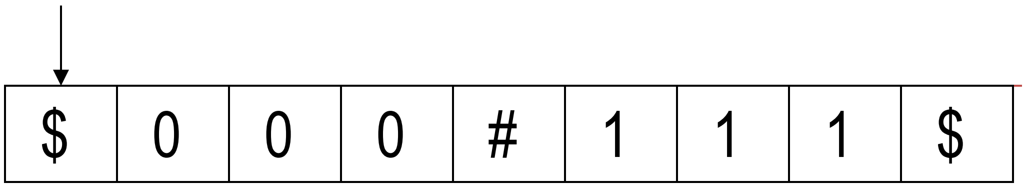
- Compare the current input and top element on stack, if match succeeds:
  - The header move forward
  - The stack pop one element

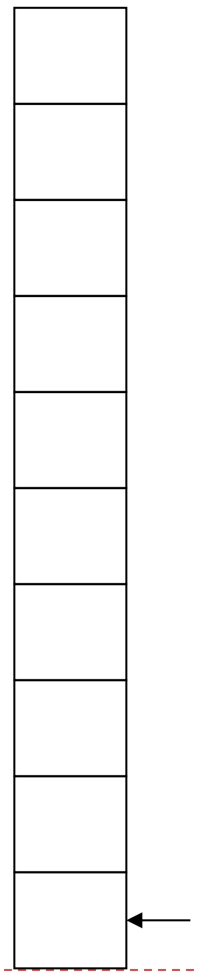


A language is CFL $\Longrightarrow$ some PDA recognizes it

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

Example: how PDA recognizes the input 000#111





- Grammar G_1:

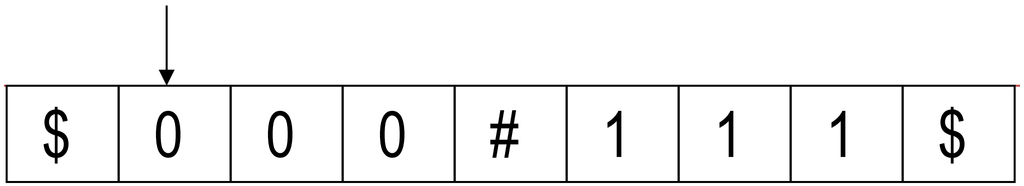
$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation :

Example: how PDA recognizes the input 000#111



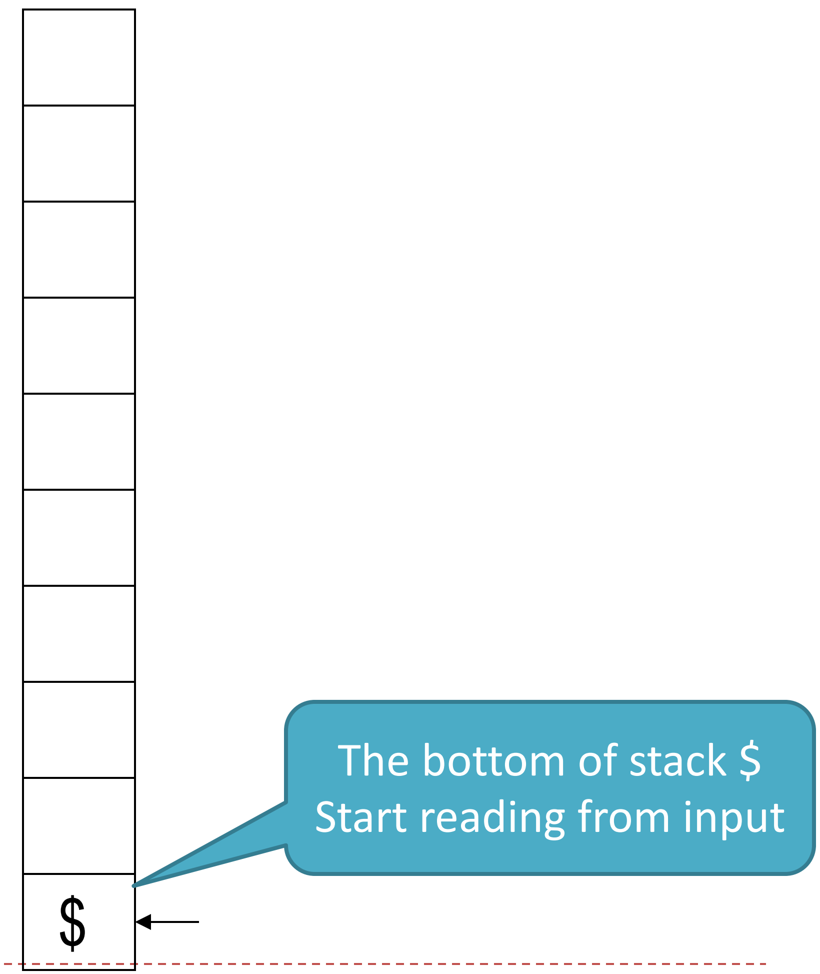
- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation :



Example: how PDA recognizes the input 000#111

$	0	0	0	#	1	1	1	$

- Grammar G_1:

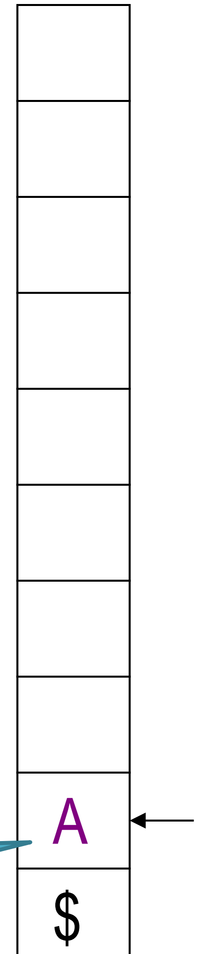
$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : A

Push the start variable A



Example: how PDA recognizes the input 000#111

$	0	0	0	#	1	1	1	$

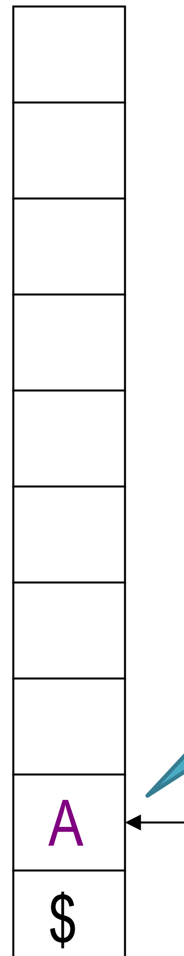
- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1$



Step 2: The top of stack is variable, so simulate the derivation

Example: how PDA recognizes the input 000#111

$	0	0	0	#	1	1	1	$

During the derivation, the input is always ε; head stays still

- Gra

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- Derivation : $A \Rightarrow 0A1$

0
A
1
$

POP A
PUSH 1, A, 0

Finish the derivation:
$A \Rightarrow 0A1$

Example: how PDA recognizes the input 000#111

$	0	0	0	#	1	1	1	$

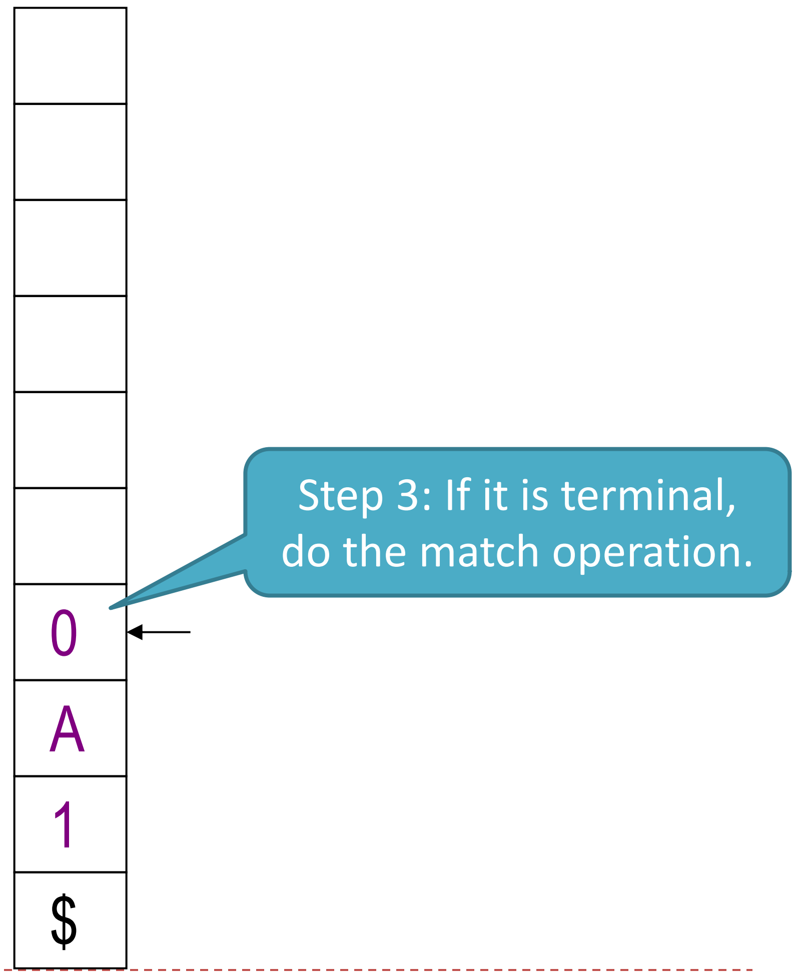
- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1$



Example: how PDA recognizes the input 000#111

\$	0	0	0	#	1	1	1	\$

- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

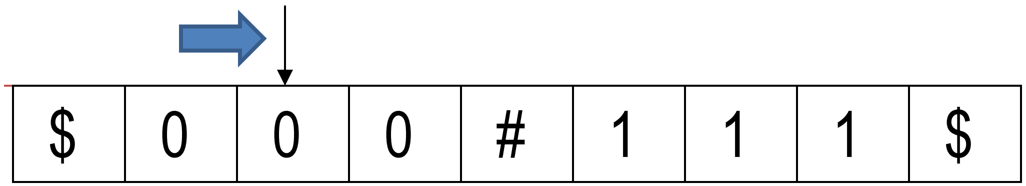
$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1$

A
1
\$

If the match succeeds, pop up the top element in stack and move hand forward.

Example: how PDA recognizes the input 000#111



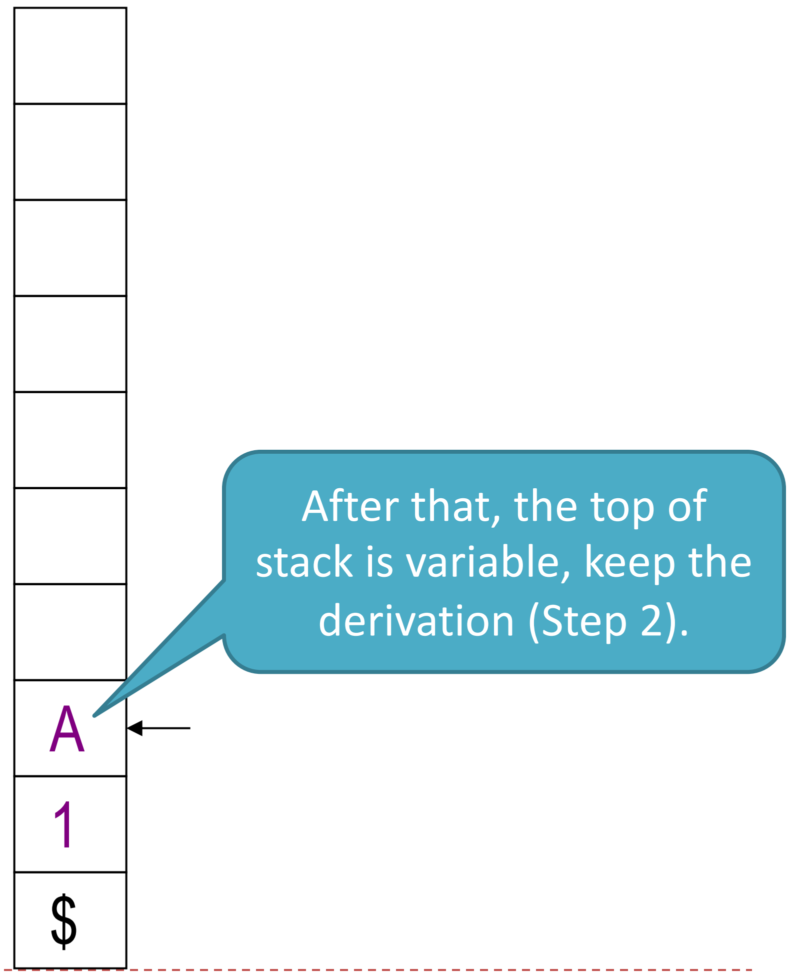
- Grammar G_1:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- Derivation : $A \Rightarrow 0A1$



Example: how PDA recognizes the input 000#111

\$	0	0	0	#	1	1	1	\$

- Grammar G_1:

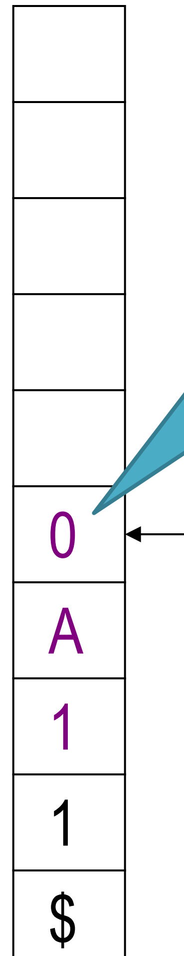
$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

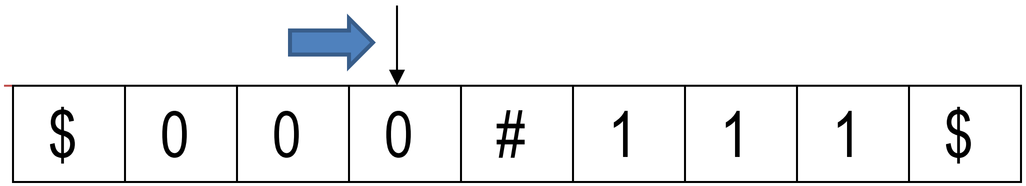
- Derivation : $A \Rightarrow 0A1$

$\Rightarrow 00A11$



After derivation, the top of stack becomes terminal again, keep the match with input (Step 3).

Example: how PDA recognizes the input 000#111



- Grammar G_1:

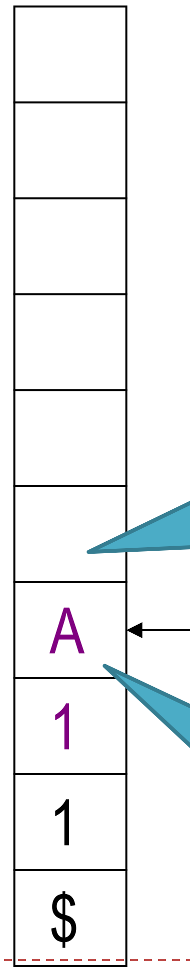
$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1$

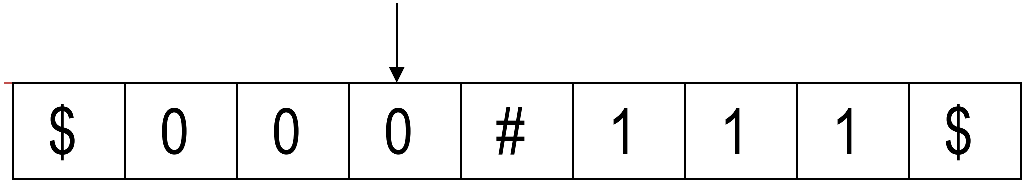
$$\Rightarrow 00A11$$



If the match succeeds, pop up the top element in stack and move hand forward.

After that, the top of stack is variable again, keep the derivation (Step 2).

Example: how PDA recognizes the input 000#111



- Grammar G_1:

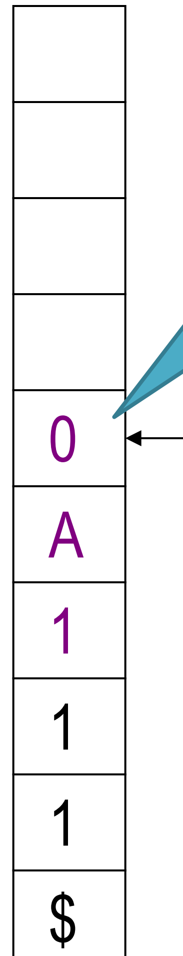
$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

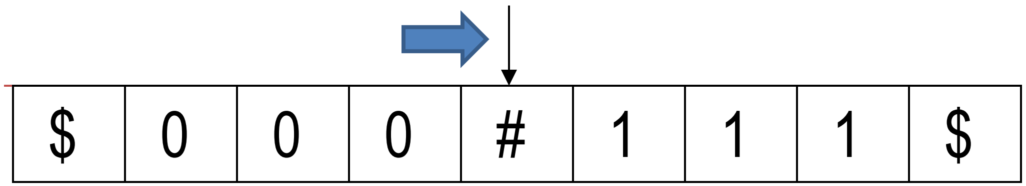
- Derivation : $A \Rightarrow 0A1$

$$\Rightarrow 00A11 \Rightarrow 000A111$$



After derivation, the top of stack becomes terminal again, keep the match with input (Step 3).

Example: how PDA recognizes the input 000#111

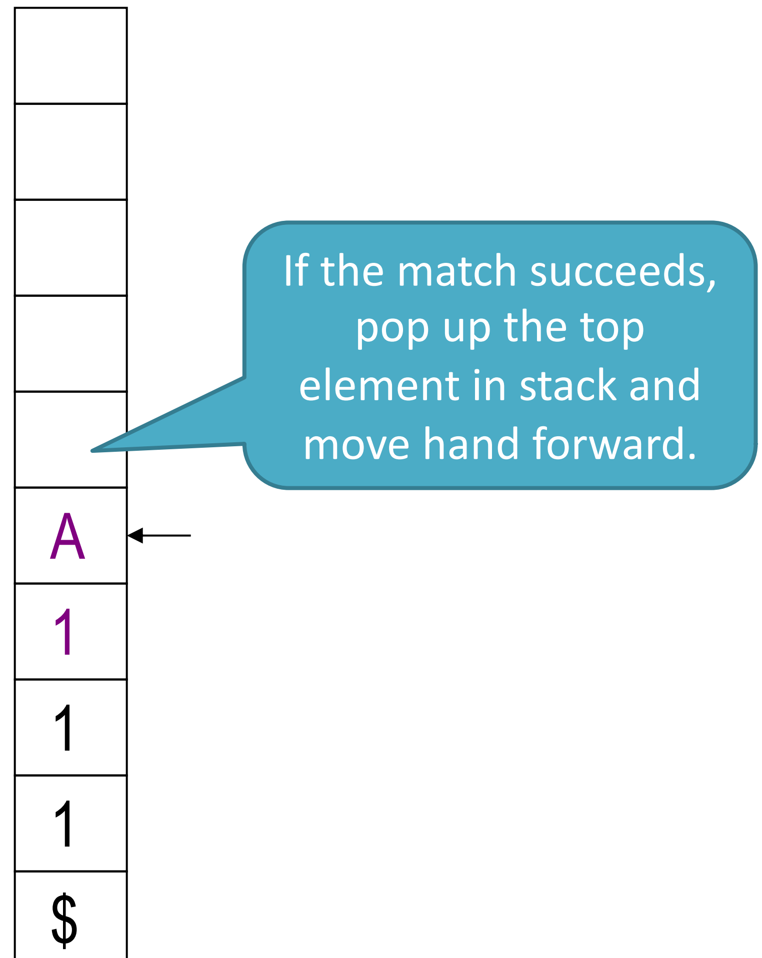


- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$



- Derivation : $A \Rightarrow 0A1$

$\Rightarrow 00A11 \Rightarrow 000A111$

Example: how PDA recognizes the input 000#111

$	0	0	0	#	1	1	1	$

- Grammar G_1:

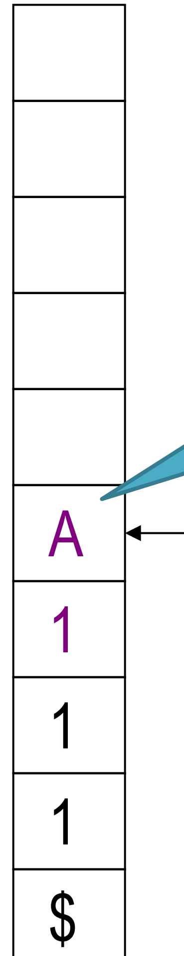
$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

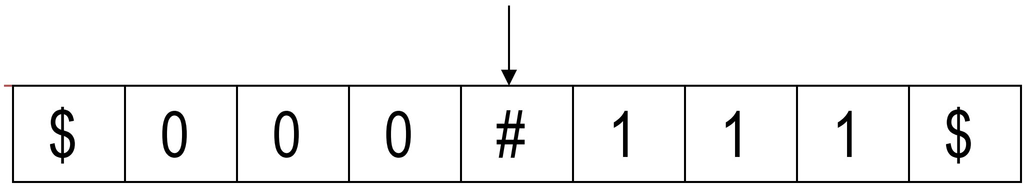
- Derivation : $A \Rightarrow 0A1$

$$\Rightarrow 00A11 \Rightarrow 000A111$$



The input is # now, we need other derivation to generate #.

Example: how PDA recognizes the input 000#111



- Grammar G_1:

$$A \rightarrow 0A1$$

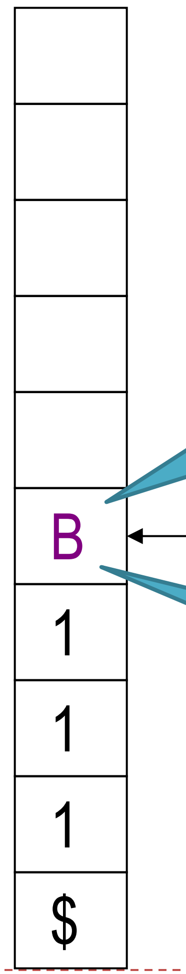
$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1 \Rightarrow$

$00A11 \Rightarrow 000A111 \Rightarrow$

$000B111$



This time, we perform the derivation using grammar $A \rightarrow B$

After that, as the top element is still variable, keep the derivation (step 2)

Example: how PDA recognizes the input 000#111

$	0	0	0	#	1	1	1	$

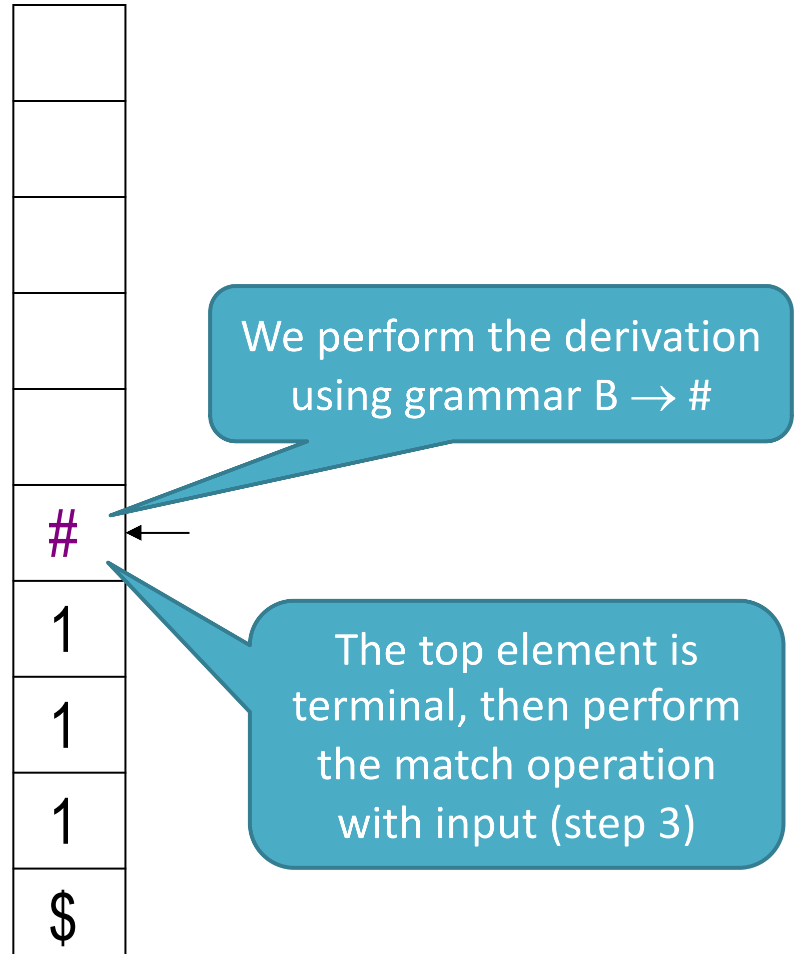
- Grammar G_1:

$$A \rightarrow 0A1$$

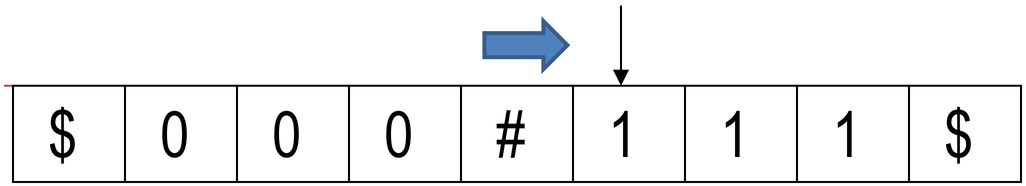
$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$



Example: how PDA recognizes the input 000#111



- Grammar G_1:

$$A \rightarrow 0A1$$

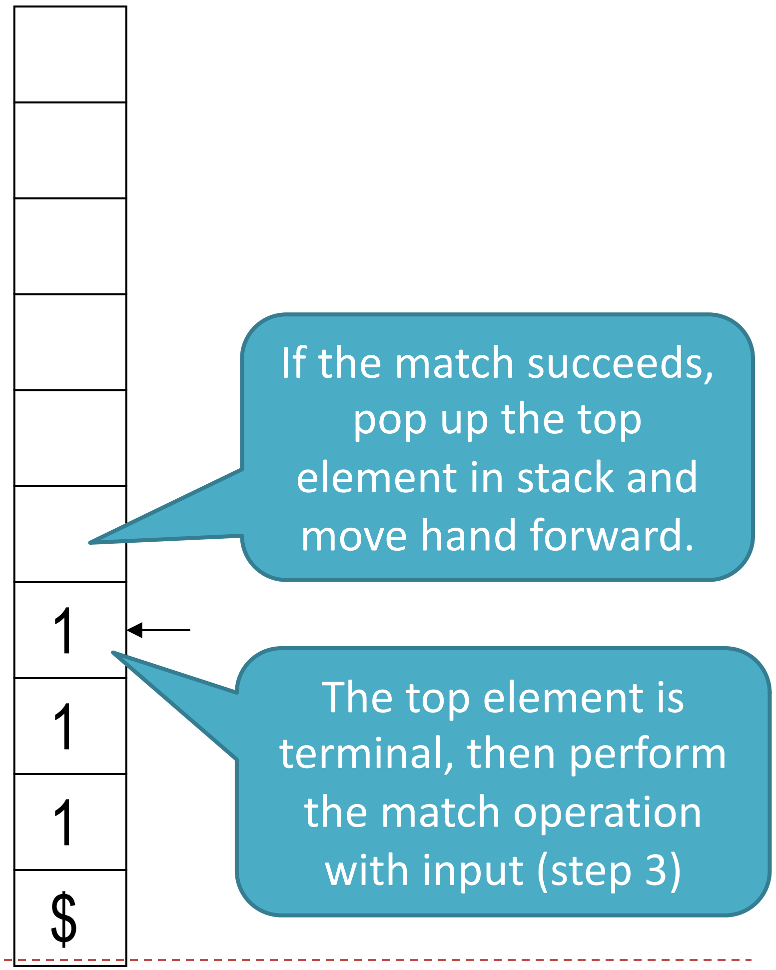
$$A \rightarrow B$$

$$B \rightarrow \#$$

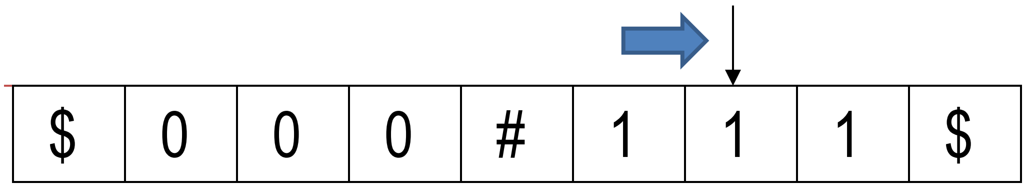
- Derivation : $A \Rightarrow 0A1 \Rightarrow$

$00A11 \Rightarrow 000A111 \Rightarrow$

$000B111 \Rightarrow 000\#111$



Example: how PDA recognizes the input 000#111



- Grammar G_1:

$$A \rightarrow 0A1$$

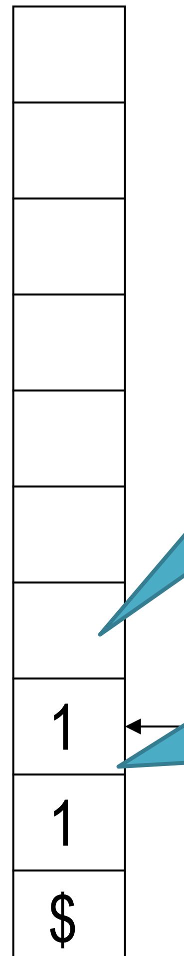
$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1 \Rightarrow$

$00A11 \Rightarrow 000A111 \Rightarrow$

$000B111 \Rightarrow 000\#111$



If the match succeeds, pop up the top element in stack and move hand forward.

The top element is terminal, then perform the match operation with input (step 3)

Example: how PDA recognizes the input 000#111

$	0	0	0	#	1	1	1	$

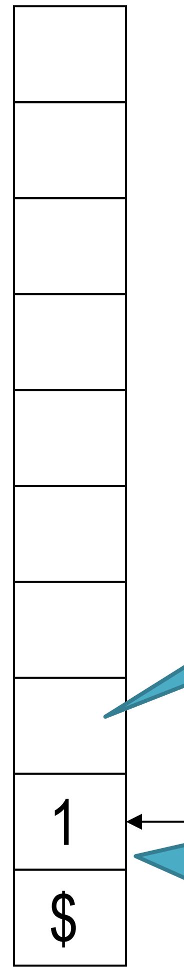
- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

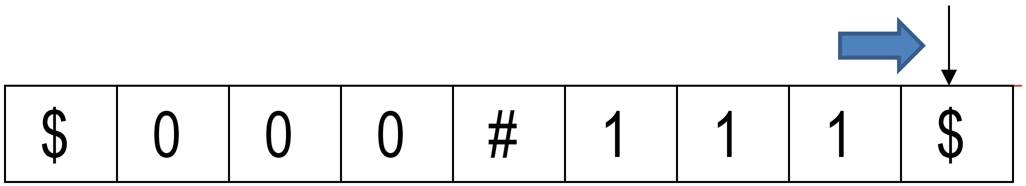
- Derivation : $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$



If the match succeeds, pop up the top element in stack and move hand forward.

The top element is terminal, then perform the match operation with input (step 3)

Example: how PDA recognizes the input 000#111



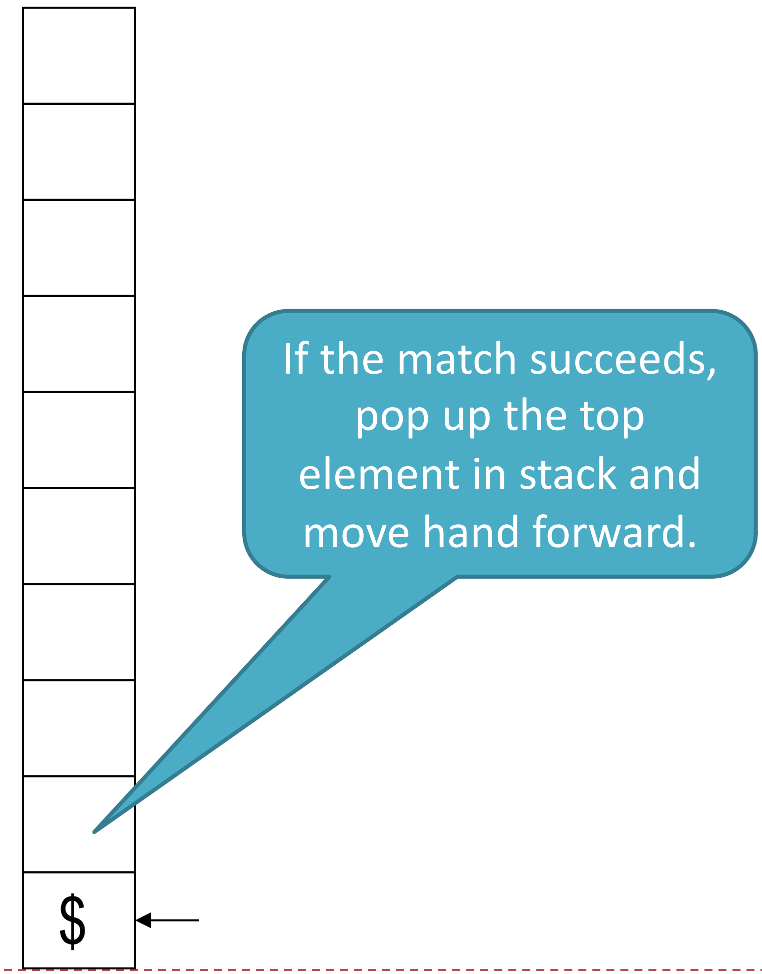
- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$



Example: how PDA recognizes the input 000#111

$	0	0	0	#	1	1	1	$

- Grammar G_1:

$$A \rightarrow 0A1$$

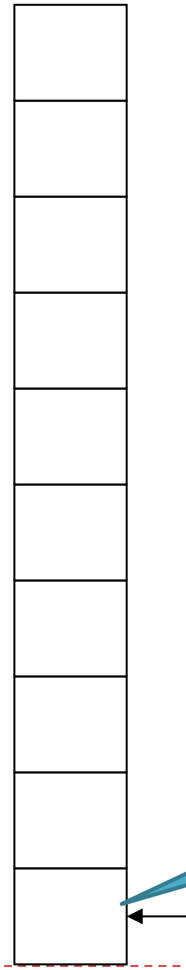
$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1 \Rightarrow$

$$00A11 \Rightarrow 000A111 \Rightarrow$$

$$000B111 \Rightarrow 000\#111$$



If the input is finished and stack is empty, accept; otherwise reject.

A language is CFL $\Longrightarrow$ some PDA recognizes it

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

Question

\$	a	b	a	b	\$

- Grammar G_3:

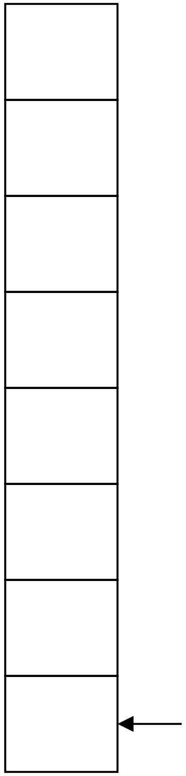
$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question

$	a	b	a	b	$

- Grammar G_3:

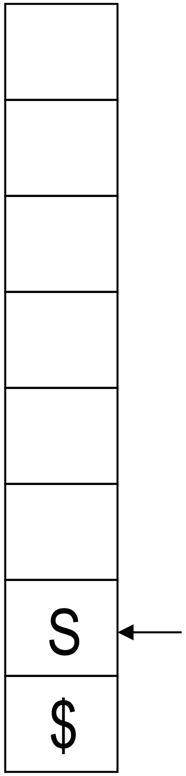
$$S \rightarrow aSb$$

$$S \rightarrow SS$$

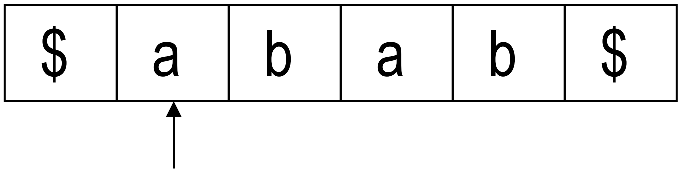
$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question

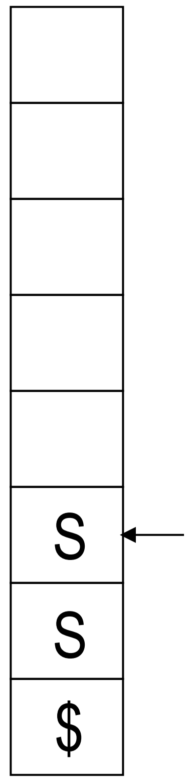


- Grammar G_3:

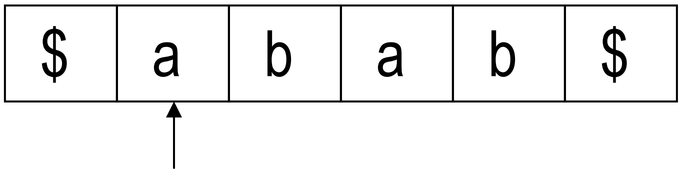
$$S \rightarrow aSb$$
$$S \rightarrow SS$$
$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question



- Grammar G_3:

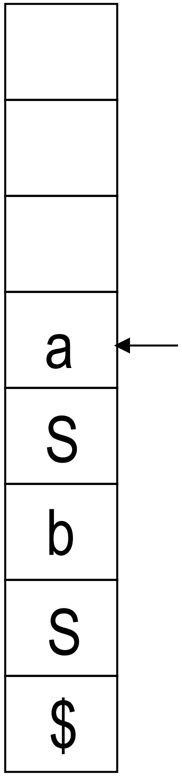
$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question

\$	a	b	a	b	\$

- Grammar G_3:

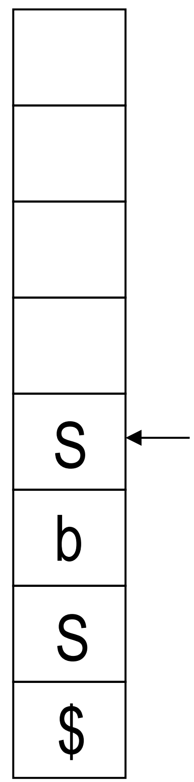
$$S \to aSb$$

$$S \to SS$$

$$S \to \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question

$	a	b	a	b	$

- Grammar G_3:

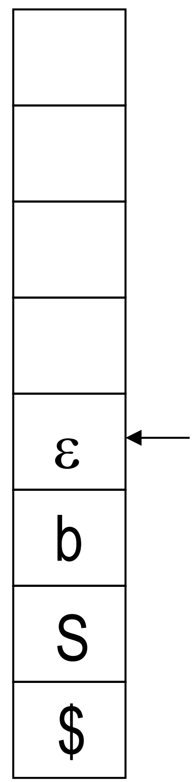
$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question

$	a	b	a	b	$

- Grammar G_3:

$$S \to aSb$$

$$S \to SS$$

$$S \to \varepsilon$$

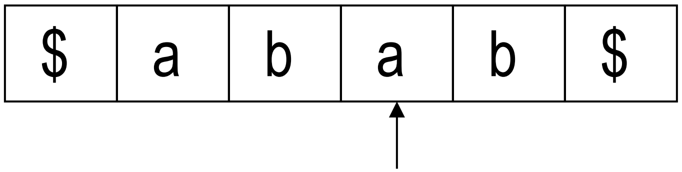
- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

b
S
$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

Question



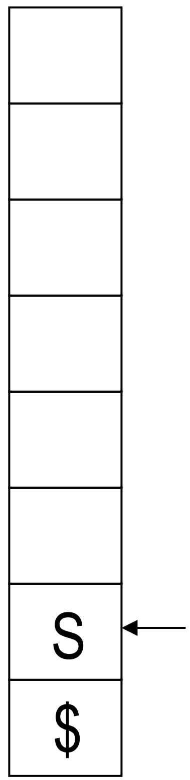
- Grammar G_3:

$$S \to aSb$$

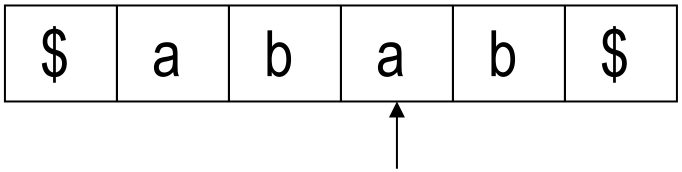
$$S \to SS$$

$$S \to \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$



Question



- Grammar G_3:

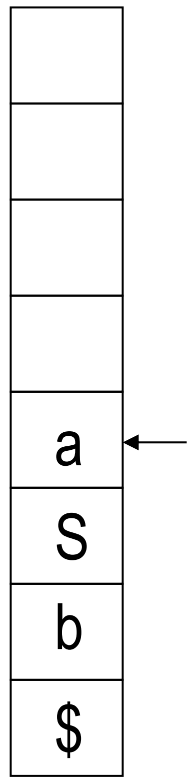
$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

\$	a	b	a	b	\$

- Grammar G_3:

$$S \to aSb$$

$$S \to SS$$

$$S \to \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

S
b
\$

Question

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

\$	a	b	a	b	\$

ε
b
\$

- Grammar G_3:

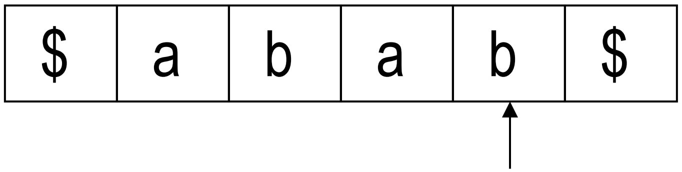
$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

Question

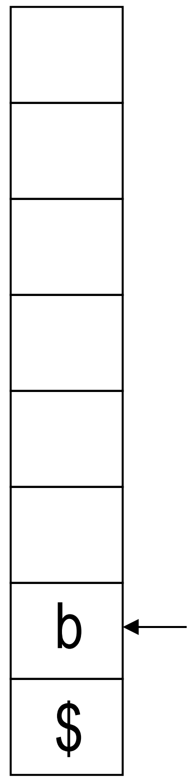


- Grammar G_3:

$$S \rightarrow aSb$$
$$S \rightarrow SS$$
$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question

$	a	b	a	b	$

- Grammar G_3:

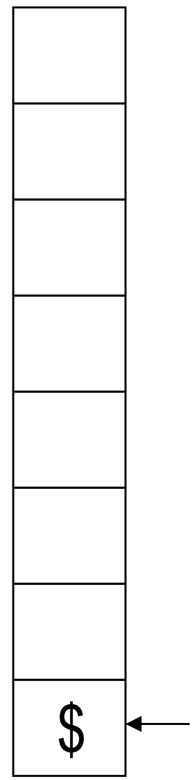
$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

- Derivation : $S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



Question 2

\$	a	+	a	x	a	\$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

- Grammar G_5:

$$E \rightarrow E+E \mid E\times E \mid (E) \mid a$$

- Derivation : $E \Rightarrow E+E$

$$\Rightarrow a+E \Rightarrow a+E\times E$$

$$\Rightarrow a+a\times E \Rightarrow a+a\times a$$

Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

\$	a	+	a	x	a	\$

- Grammar G_5:

$$E \rightarrow E+E \mid E\times E \mid (E) \mid a$$

- Derivation : $E \Rightarrow E+E$

$$\Rightarrow a+E \Rightarrow a+E\times E$$

$$\Rightarrow a+a\times E \Rightarrow a+a\times a$$

E
\$

Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

\$	a	+	a	x	a	\$

- Grammar G_5:

$E \rightarrow E+E \mid E\times E \mid (E) \mid a$

- Derivation : $E \Rightarrow E+E$

$\Rightarrow a+E \Rightarrow a+E\times E$

$\Rightarrow a+a\times E \Rightarrow a+a\times a$

E
+
E
\$

Question 2

$	a	+	a	x	a	$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

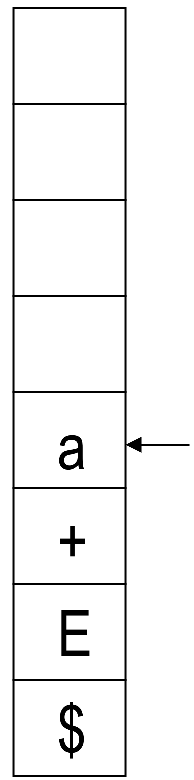
- Grammar G_5:

$$E \rightarrow E+E \mid E\times E \mid (E) \mid a$$

- Derivation : $E \Rightarrow E+E$

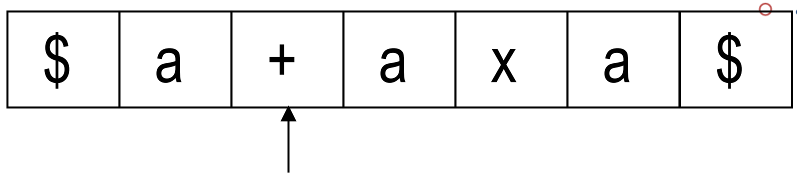
$$\Rightarrow a+E \Rightarrow a+E\times E$$

$$\Rightarrow a+a\times E \Rightarrow a+a\times a$$



Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



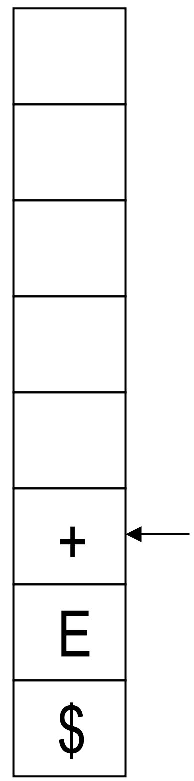
- Grammar G_5:

$E \rightarrow E+E \mid E\times E \mid (E) \mid a$

- Derivation : $E \Rightarrow E+E$

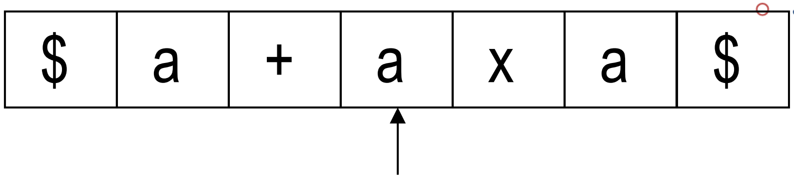
$\Rightarrow a+E \Rightarrow a+E\times E$

$\Rightarrow a+a\times E \Rightarrow a+a\times a$



Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



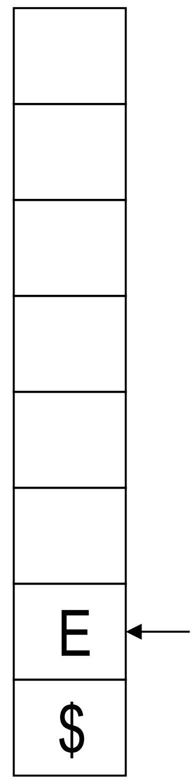
- Grammar G_5:

$E \rightarrow E+E \mid E\times E \mid (E) \mid a$

- Derivation : $E \Rightarrow E+E$

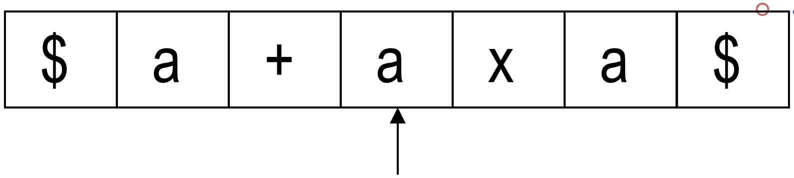
$\Rightarrow a+E \Rightarrow a+E\times E$

$\Rightarrow a+a\times E \Rightarrow a+a\times a$



Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match



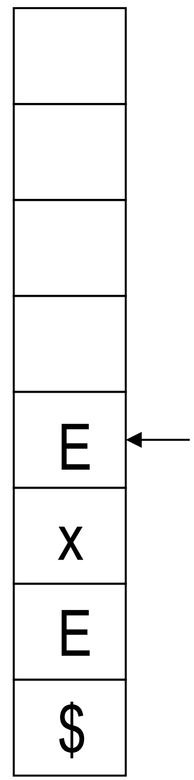
- Grammar G_5:

$$E \rightarrow E+E \mid E\times E \mid (E) \mid a$$

- Derivation : $E \Rightarrow E+E$

$$\Rightarrow a+E \Rightarrow a+E\times E$$

$$\Rightarrow a+a\times E \Rightarrow a+a\times a$$



Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

\$	a	+	a	x	a	\$

- Grammar G_5:

$$E \rightarrow E+E \mid E\times E \mid (E) \mid a$$

- Derivation : $E \Rightarrow E+E$

$$\Rightarrow a+E \Rightarrow a+E\times E$$

$$\Rightarrow a+a\times E \Rightarrow a+a\times a$$

a
x
E
\$

Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

$	a	+	a	x	a	$

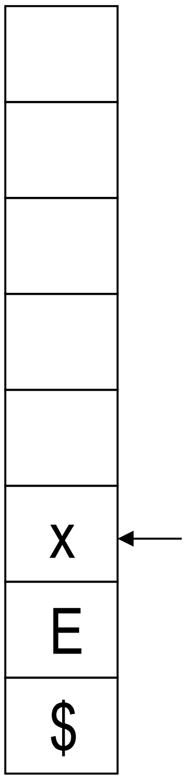
- Grammar G_5:

$$E \rightarrow E+E \mid E\times E \mid (E) \mid a$$

- Derivation : $E \Rightarrow E+E$

$$\Rightarrow a+E \Rightarrow a+E\times E$$

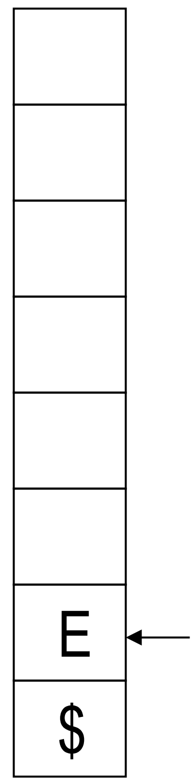
$$\Rightarrow a+a\times E \Rightarrow a+a\times a$$



Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

\$	a	+	a	x	a	\$



- Grammar G_5:

$$E \rightarrow E+E \mid E\times E \mid (E) \mid a$$

- Derivation : $E \Rightarrow E+E$

$$\Rightarrow a+E \Rightarrow a+E\times E$$

$$\Rightarrow a+a\times E \Rightarrow a+a\times a$$

Question 2

$	a	+	a	x	a	$

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

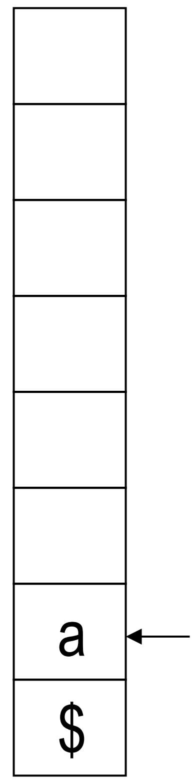
- Grammar G_5:

$E \rightarrow E+E \mid E\times E \mid (E) \mid a$

- Derivation : $E \Rightarrow E+E$

$\Rightarrow a+E \Rightarrow a+E\times E$

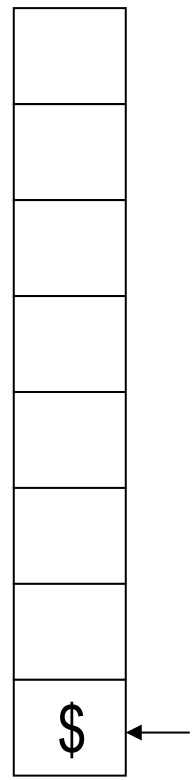
$\Rightarrow a+a\times E \Rightarrow a+a\times a$



Question 2

- Details: how PDA recognizes a string of a CFL
  - Step 1: Compare the input with the top of stack
  - Step 2: If the top of stack is variable, then simulate the derivation
  - Step 3: If the top of stack is terminal, then do the match

\$	a	+	a	x	a	\$



- Grammar G_5:

$E \rightarrow E+E \mid E\times E \mid (E) \mid a$

- Derivation : $E \Rightarrow E+E$

$\Rightarrow a+E \Rightarrow a+E\times E$

$\Rightarrow a+a\times E \Rightarrow a+a\times a$

A language is CFL $\Longrightarrow$ some PDA recognizes it

- Proof:

Suppose A is CFL, based on definition

we have the grammar of A, **CFG $G=(V,\Sigma,R,S)$.**

Build the following PDA to recognize A

$P = (\{q_{start}, q_{loop}, q_{accept}\} \cup E,$

$\Sigma,$

$V \cup \Sigma,$

$\delta,$

$q_{start},$

$\{q_{accept}\}).$

Add states q_{start}, q_{loop}, q_{accept}
E is the intermediate state

Input is terminals in CFG

Stack could place variables and terminals in CFG

A language is CFL $\Longrightarrow$ some PDA recognizes it

Define E:

Suppose we read input **a**, state changes from **q** to **r**, and the top of stack changes from **s** to $\mathbf{u_1u_2…u_k}$

introduce new states $q_1,q_2,…,q_{k-1}$,

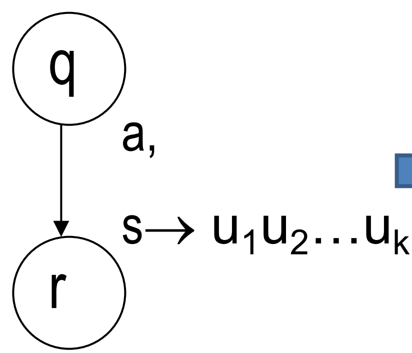
let $\delta(q,a,s)$ contains (q_1,u_k),

$\delta(q_1,\varepsilon,\varepsilon)=\{(q_2,u_{k-1})\}$,

$\delta(q_2,\varepsilon,\varepsilon)=\{(q_3,u_{k-2})\}$,

…,

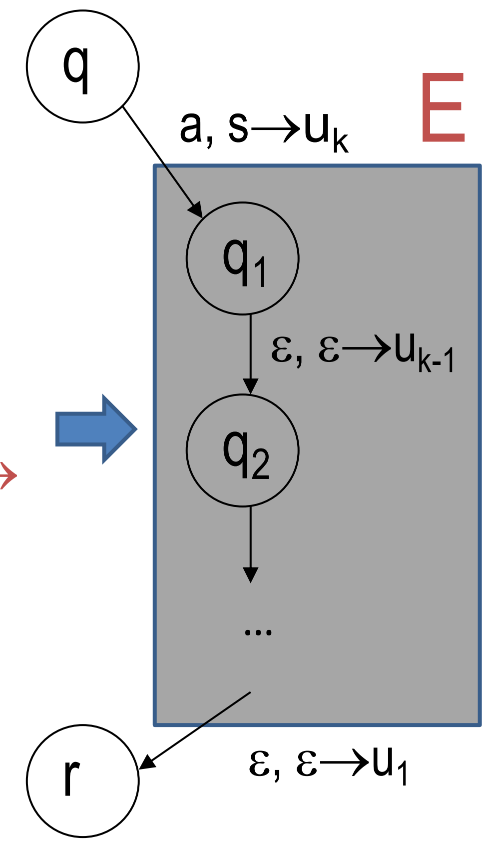
$\delta(q_{k-1},\varepsilon,\varepsilon)=\{(r,u_1)\}$.



$a\varepsilon\varepsilon$ ….. ε,

$\varepsilon\varepsilon$ ….. $\varepsilon s \rightarrow$

$u_1u_2…u_k$



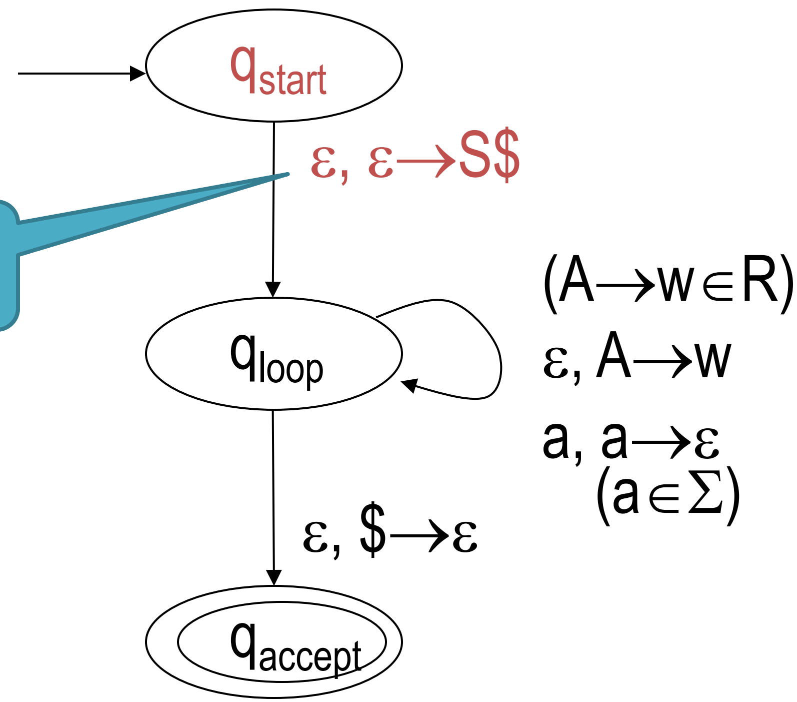
Conclude all the above as $(r,u)\in\delta(q,a,s)$ and mark all the new states as E

A language is CFL $\Longrightarrow$ some PDA recognizes it

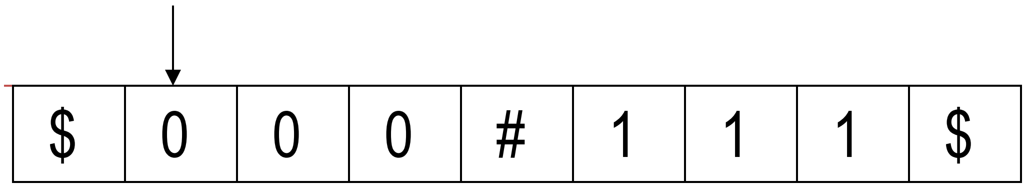
Define δ :

$\delta(q_{start},\varepsilon,\varepsilon)=\{(q_{loop},S\$)\}$,

Put $ and start variable into stack



Example: Put $ and start variable into stack



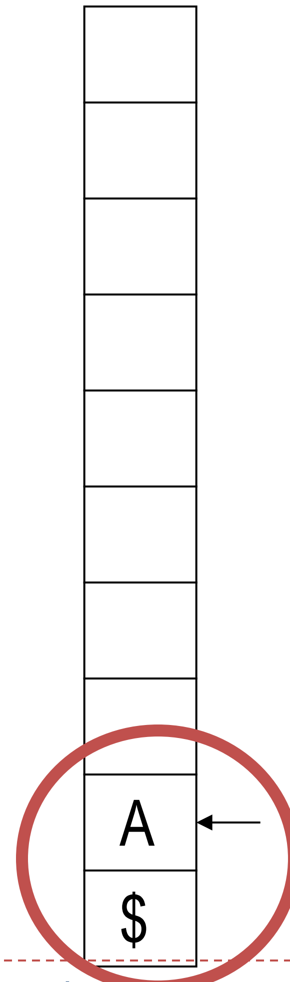
- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : A



ge is CFL $\Longrightarrow$ some PDA recognizes it

top of stack is variable A

replace A by w

pick up one rule $A \to w$

Define δ

$\delta(q_{loop},\varepsilon,A)=\{ (q_{loop},w) \mid A\to w \text{ is rule of } R\}$,

$\delta(q_{loop},a,a)=\{(q_{loop},\varepsilon)\}$,

top of stack is terminal a

Match succeeds and pop

// repeat the following steps

If the top of stack is variable A,

then pick up one rule A→w, and replace A by w, simulate

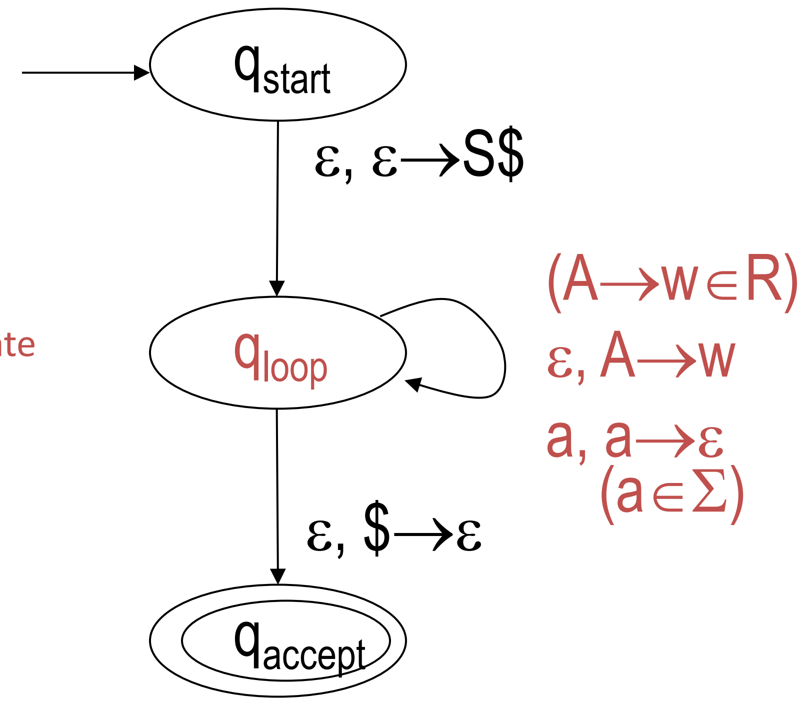
the **derivation (Step 2)**.

If the top of stack is terminal a,

then compare a with the symbol of current input

if **match(Step 3)**, repeat;

if not match, reject this branch.



A language is CFL $\Longrightarrow$ some PDA recognizes it

top of stack is \$

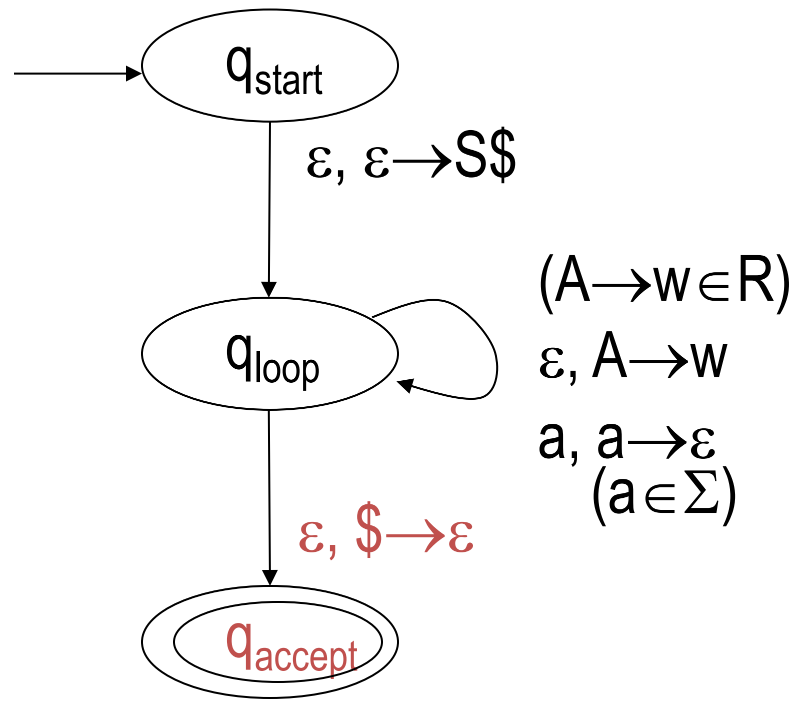
Define δ :

the input is over

$\delta(q_{loop},\varepsilon,\$)=\{(q_{accept},\varepsilon)\}$.

// If the top of stack is \$,

if the input is over, accept.



Example: when stack and input are empty

$	0	0	0	#	1	1	1	$

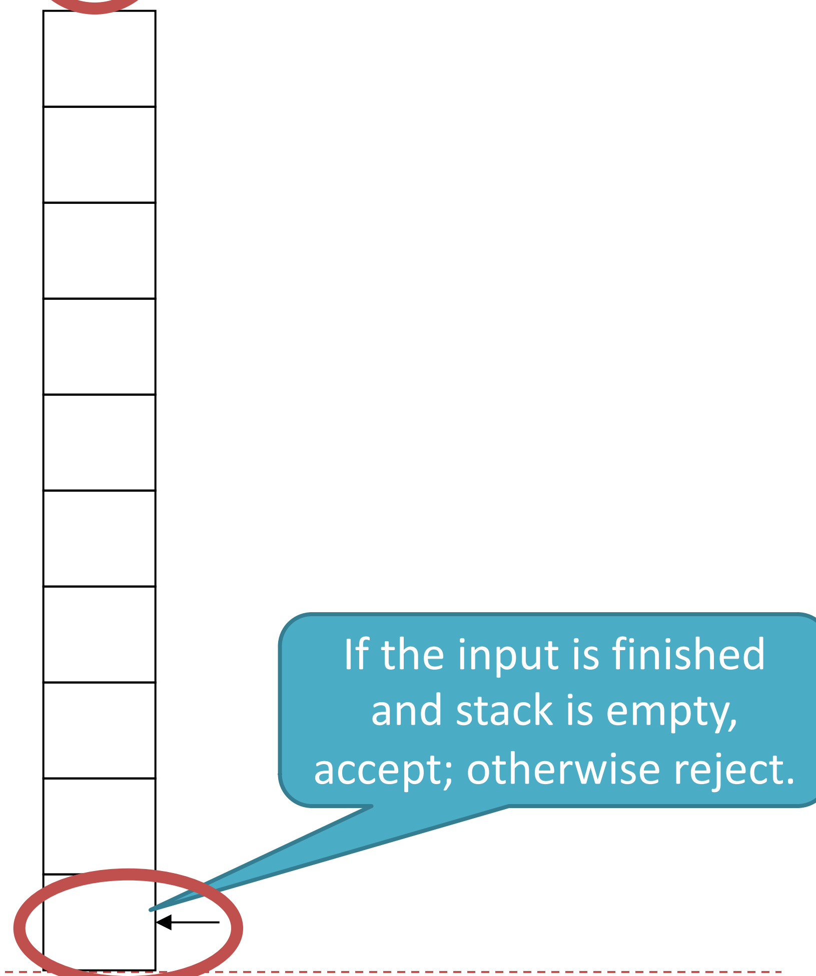
- Grammar G_1:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Derivation : $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$



A language is CFL $\Longrightarrow$ some PDA recognizes it

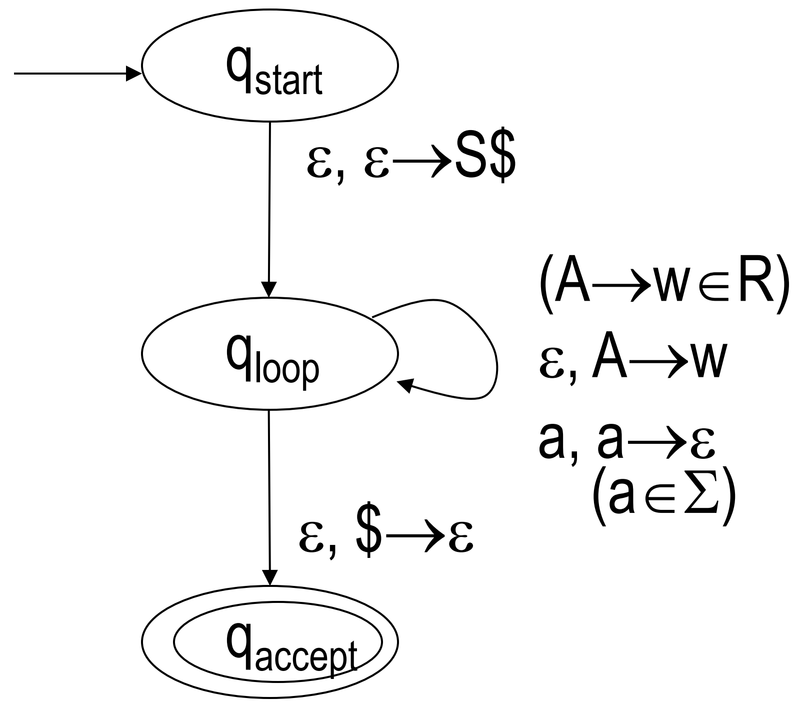
Define δ :

$\delta(q_{start},\varepsilon,\varepsilon)=\{(q_{loop},S\$)\}$, //step 1

$\delta(q_{loop},\varepsilon,A)=\{ (q_{loop},w) \mid A\rightarrow w \text{ is rule of } R\}$,

$\delta(q_{loop},a,a)=\{(q_{loop},\varepsilon)\}$, //step 2 or 3

$\delta(q_{loop},\varepsilon,\$)=\{(q_{accept},\varepsilon)\}$.



Conclusion for A language is CFL $\Longrightarrow$ some PDA recognizes it

Put \$ and start variable into stack

Repeat the following steps

- If the top of stack is variable A,

  then pick up one rule $A \rightarrow w$, and replace A by w.

- If the top of stack is terminal a,

  then compare a with the symbol of current input

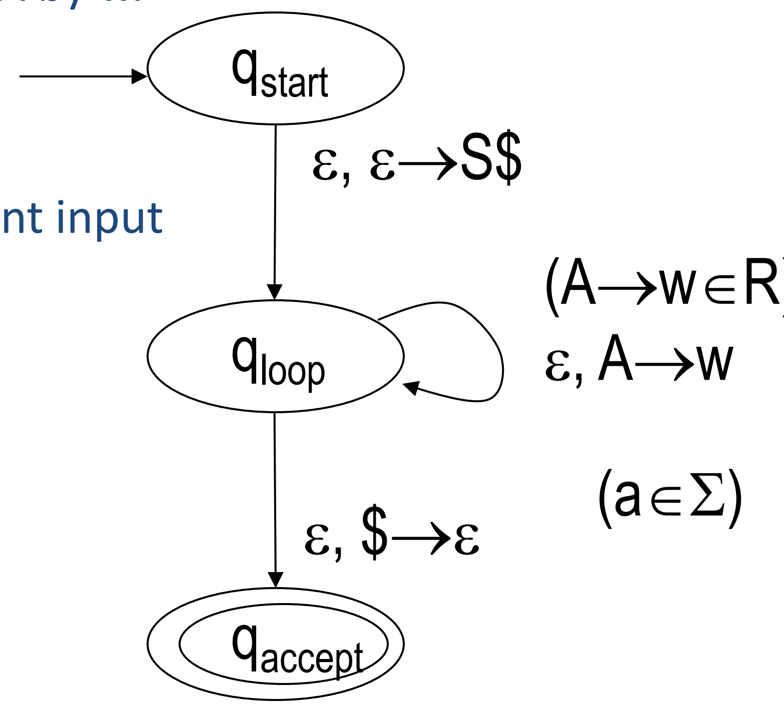
  if match, repeat;

  if not match, reject this branch.

If the top of stack is \$,

if the input is over, accept.

Create a PDA like this:



Example of CFL $\Longrightarrow$ PDA

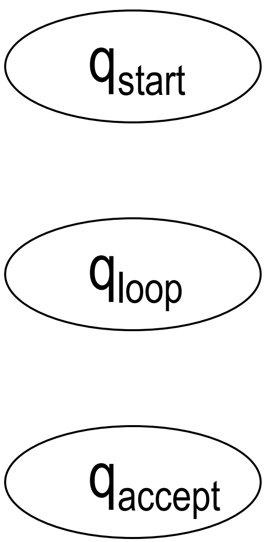
- CFG G: $S \rightarrow aTb$, $T \rightarrow Ta$, $S \rightarrow b$, $T \rightarrow \varepsilon$

  construct an equivalent PDA P_1.

Example of CFL $\Longrightarrow$ PDA

- CFG G: $S \rightarrow aTb$, $T \rightarrow Ta$, $S \rightarrow b$, $T \rightarrow \varepsilon$

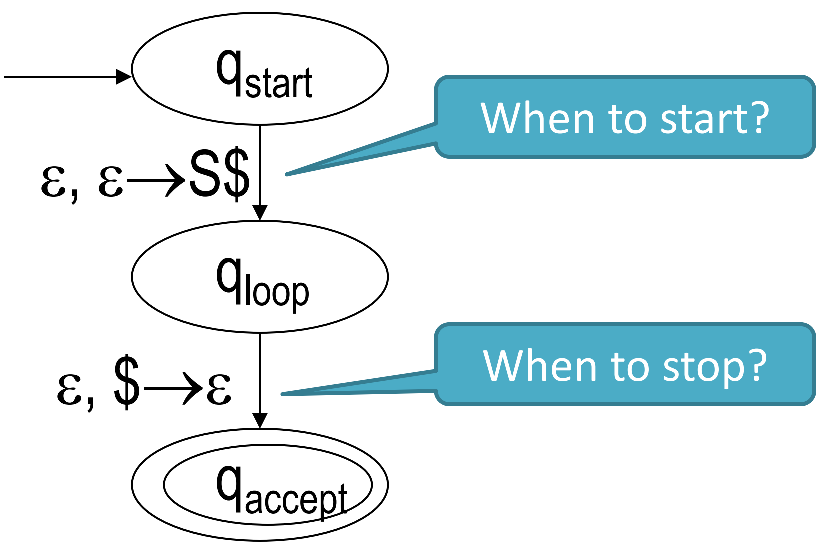
  construct an equivalent PDA P_1.



Example of CFL $\Longrightarrow$ PDA

- CFG G: $S \to aTb$, $T \to Ta$, $S \to b$, $T \to \varepsilon$

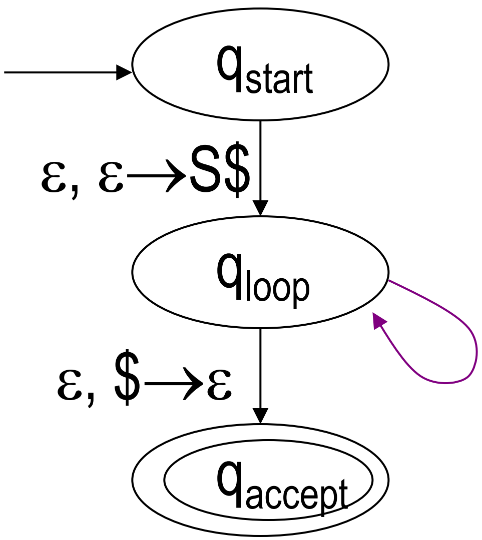
  construct an equivalent PDA P_1.



Example of CFL $\Longrightarrow$ PDA

- CFG G: $S \rightarrow aTb$, $T \rightarrow Ta$, $S \rightarrow b$, $T \rightarrow \varepsilon$

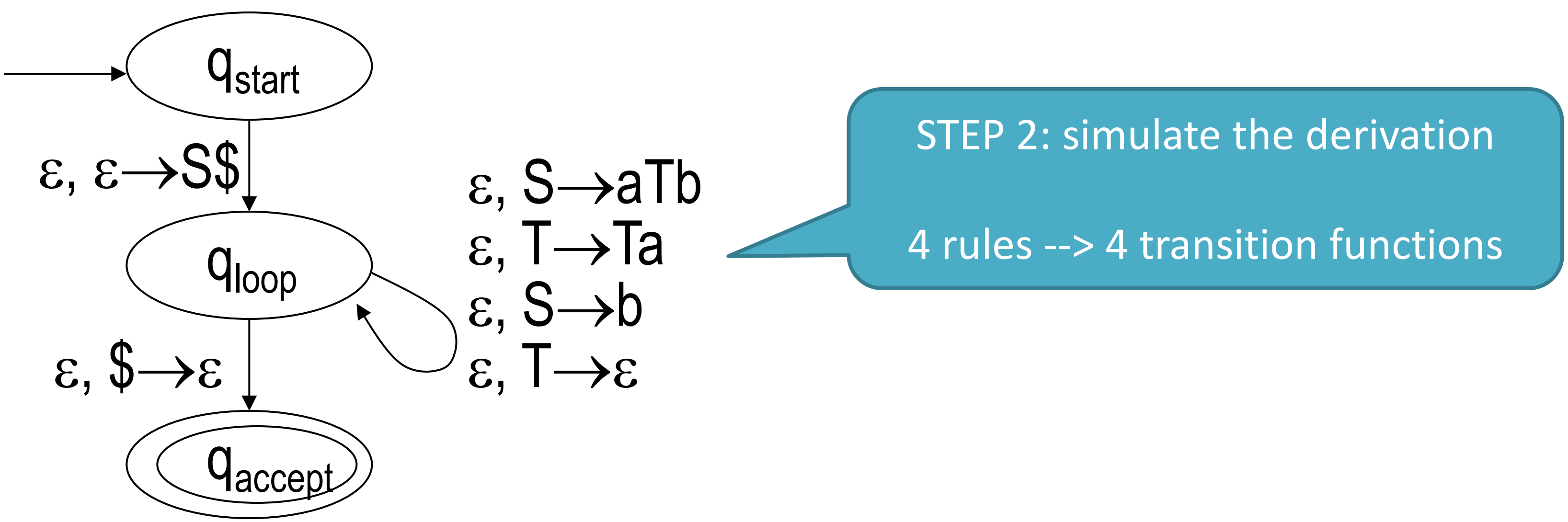
  construct an equivalent PDA P_1.



Example of CFL $\Longrightarrow$ PDA

- CFG G: $S \rightarrow aTb$, $T \rightarrow Ta$, $S \rightarrow b$, $T \rightarrow \varepsilon$

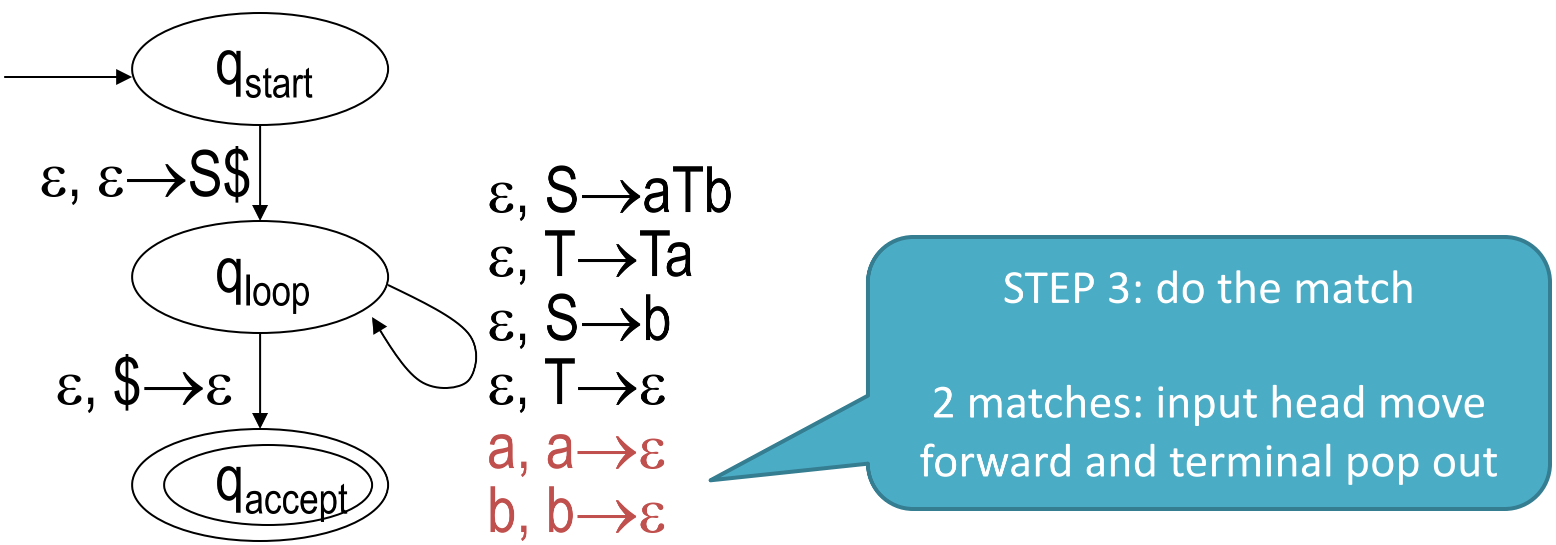
  construct an equivalent PDA P_1.



Example of CFL ⟹ PDA

- CFG G: $S \rightarrow aTb$, $T \rightarrow Ta$, $S \rightarrow b$, $T \rightarrow \varepsilon$

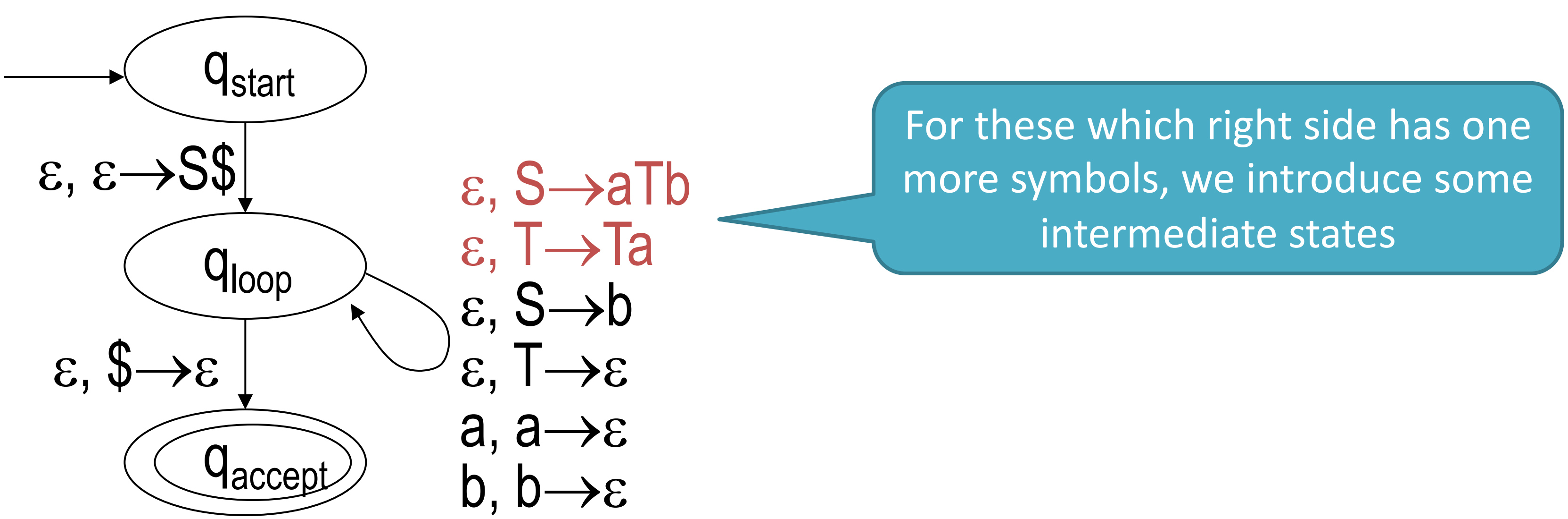
  construct an equivalent PDA P_1.



Example of CFL $\Longrightarrow$ PDA

- CFG G: $S \rightarrow aTb$, $T \rightarrow Ta$, $S \rightarrow b$, $T \rightarrow \varepsilon$

  construct an equivalent PDA P_1.

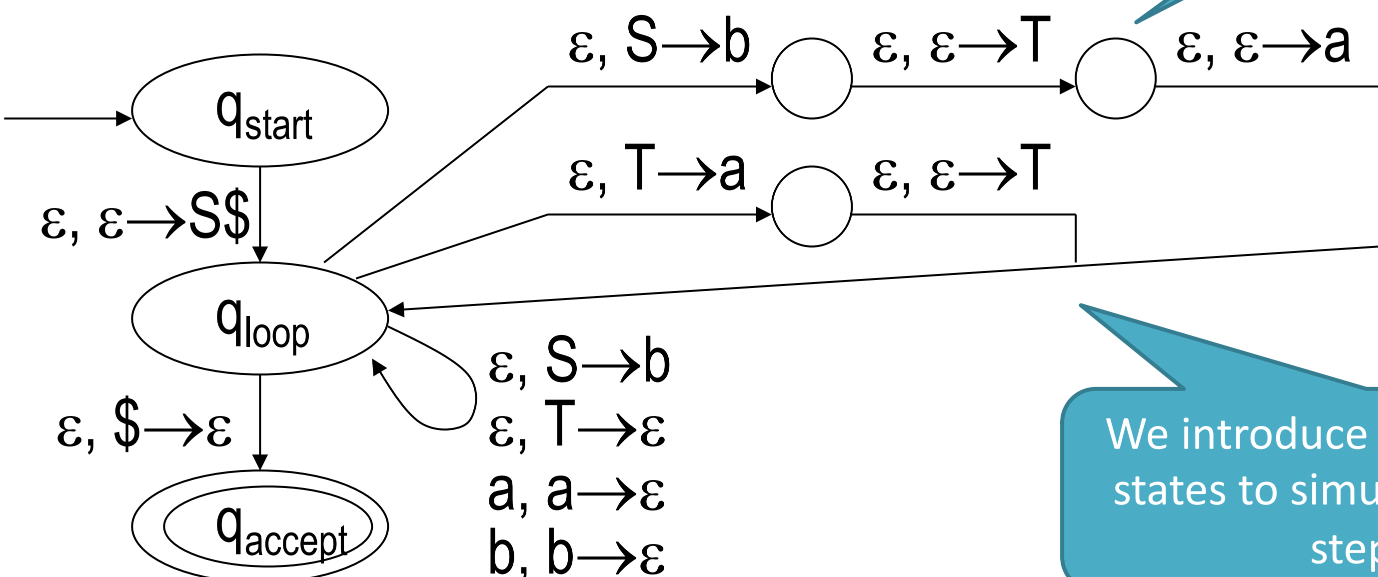


Example of CFL ⟹ PDA

- CFG G: $S \rightarrow aTb$, $T \rightarrow Ta$, $S \rightarrow b$, $T \rightarrow \varepsilon$

  construct an equivalent PDA P_1.

intermediate states

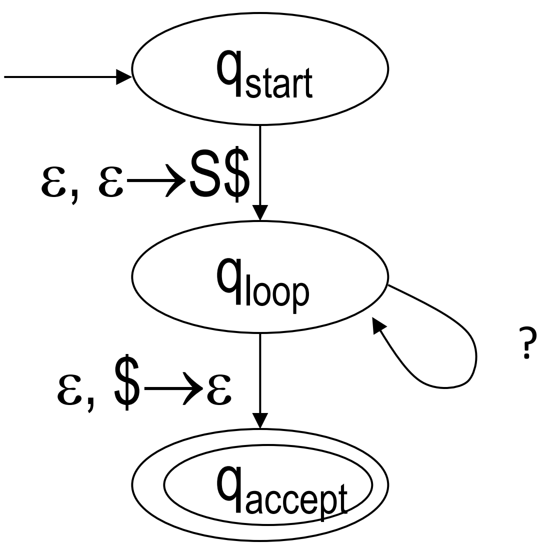


We introduce some intermediate states to simulate the derivation step by step

Question

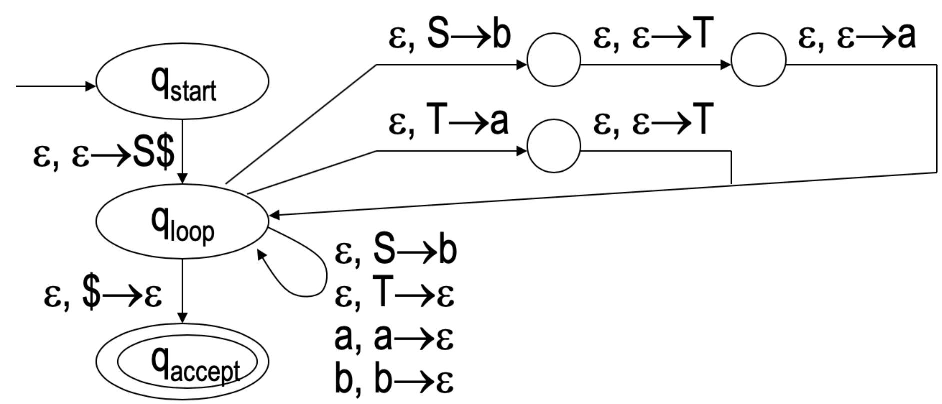
- CFG G: $E \rightarrow E+E \mid E\times E \mid (E) \mid a$

  construct an equivalent PDA P_2.



- CFG G: $S\rightarrow aTb$, $T\rightarrow Ta$, $S\rightarrow b$, $T\rightarrow \varepsilon$

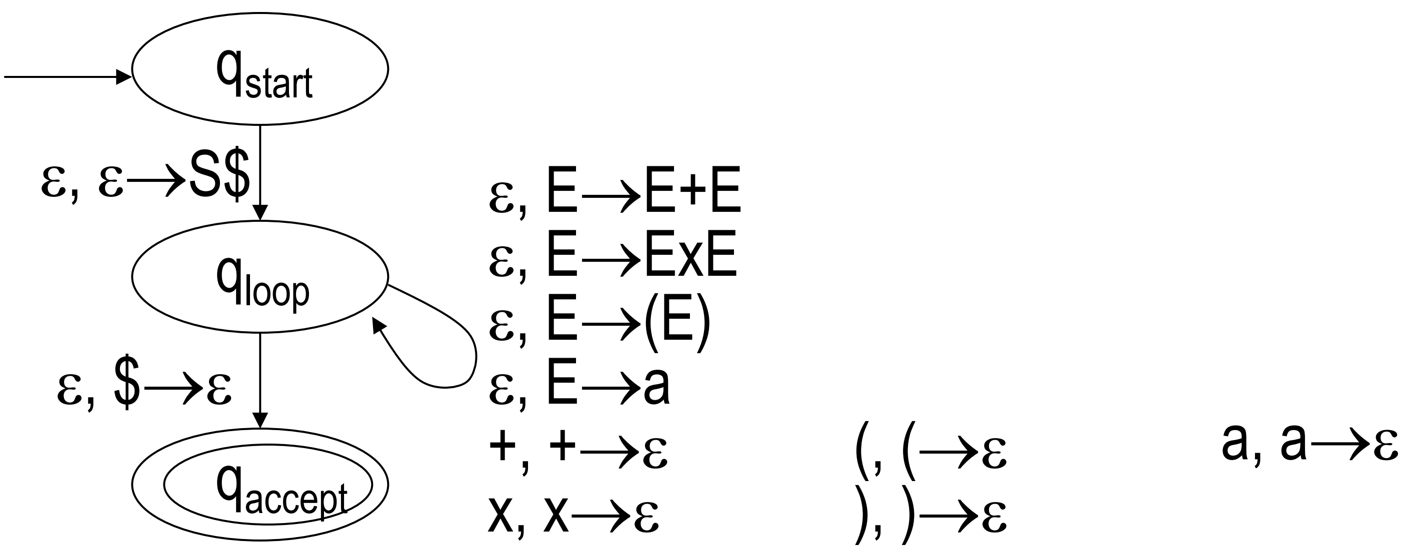
  construct an equivalent PDA P_1.



Question

- CFG G: $E \rightarrow E+E \mid E\times E \mid (E) \mid a$

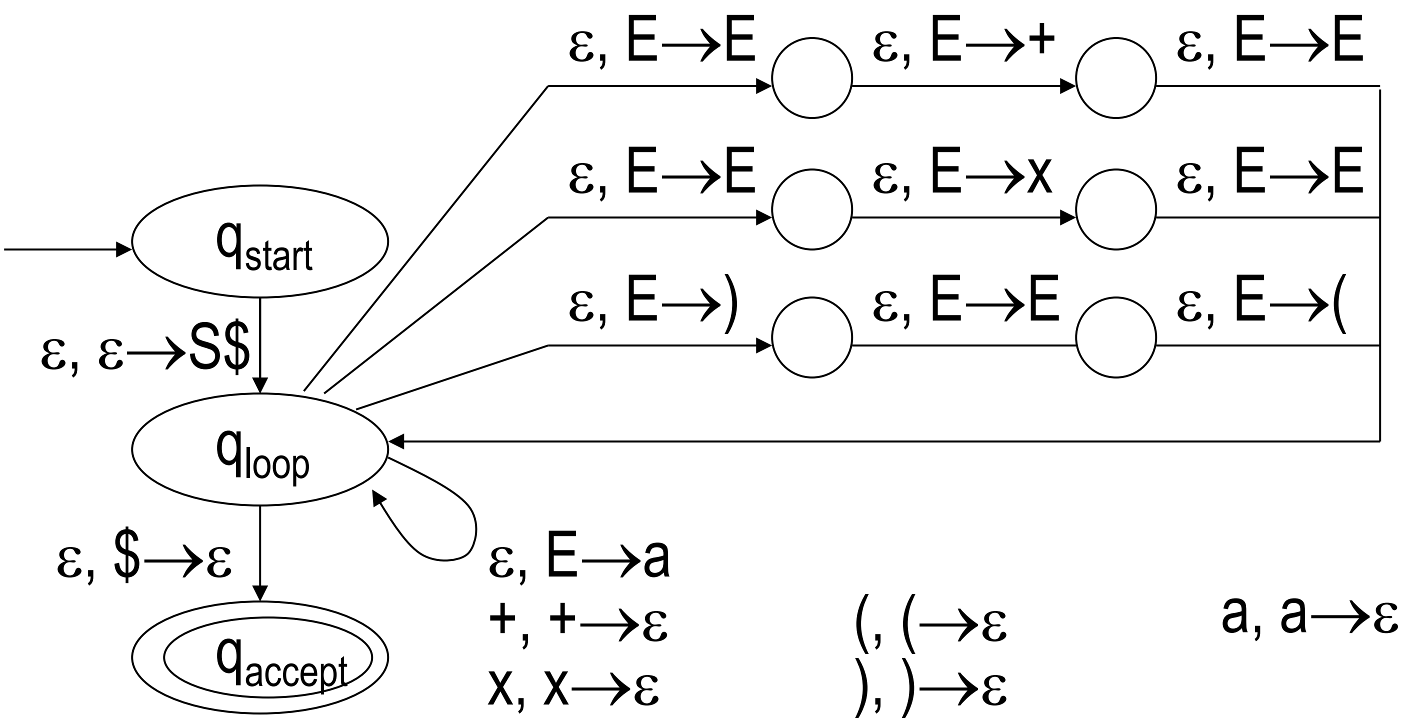
  construct an equivalent PDA P_2.



Question

- CFG G: $E \rightarrow E{+}E \mid E{\times}E \mid (E) \mid a$

construct an equivalent PDA P_2.



Outline

- Equivalence of PDA and CFG:

- A language is CFL $\Longrightarrow$ some PDA recognizes it

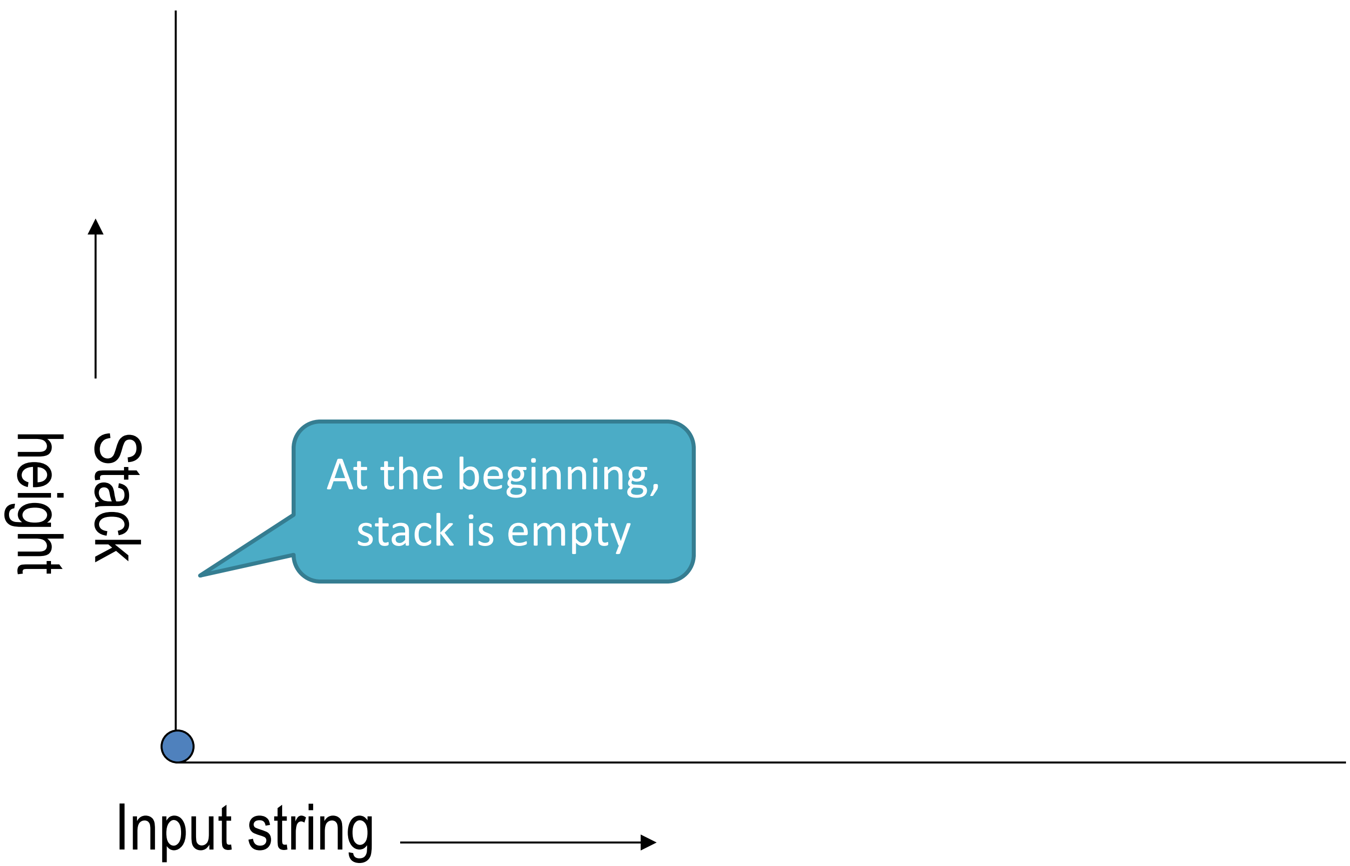
- **A language is CFL $\Longleftarrow$ some PDA recognizes it**

A language is CFL ⇐ some PDA recognizes it

- The process of PDA recognizes some inputs

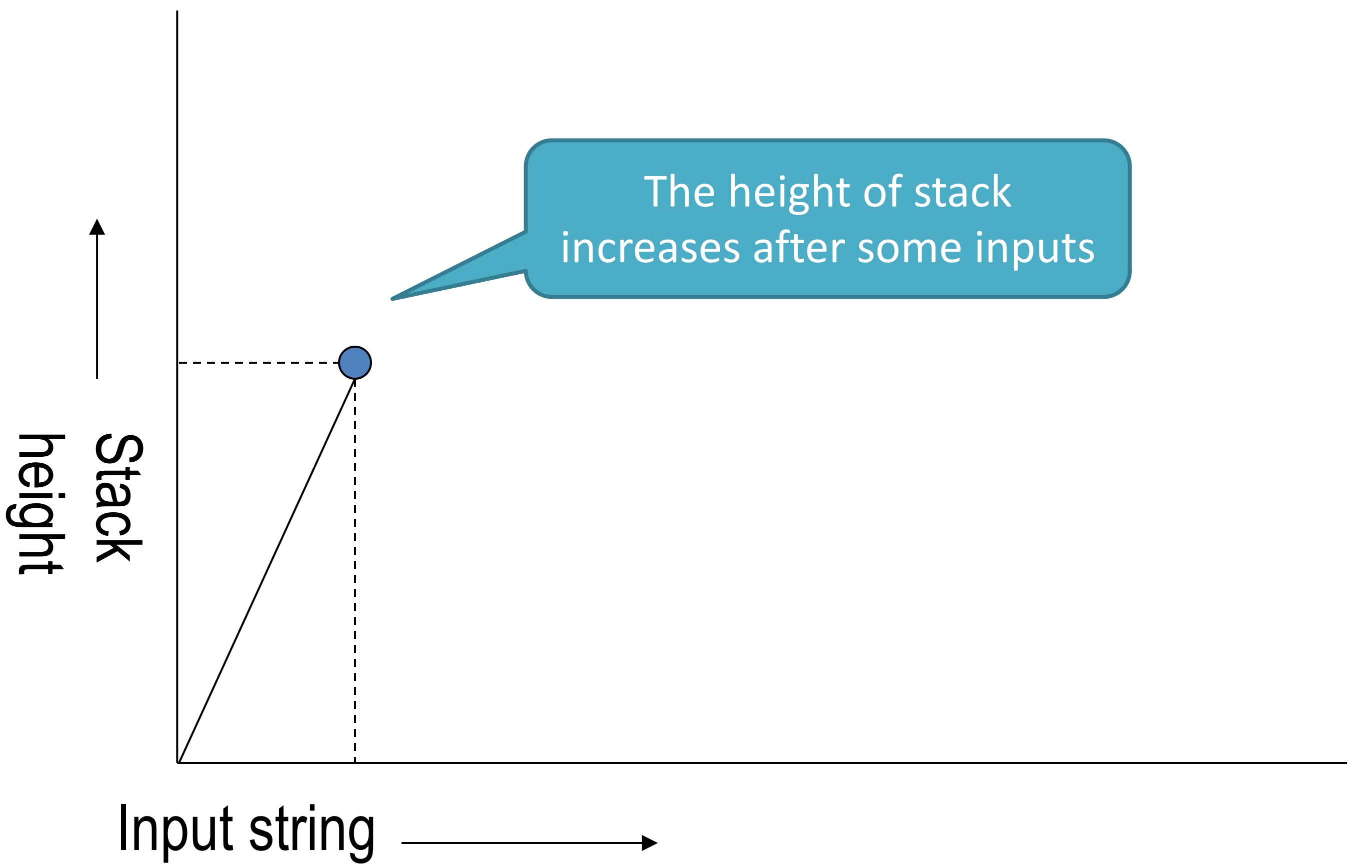
A language is CFL $\Leftarrow$ some PDA recognizes it

- The process of PDA recognizes some inputs



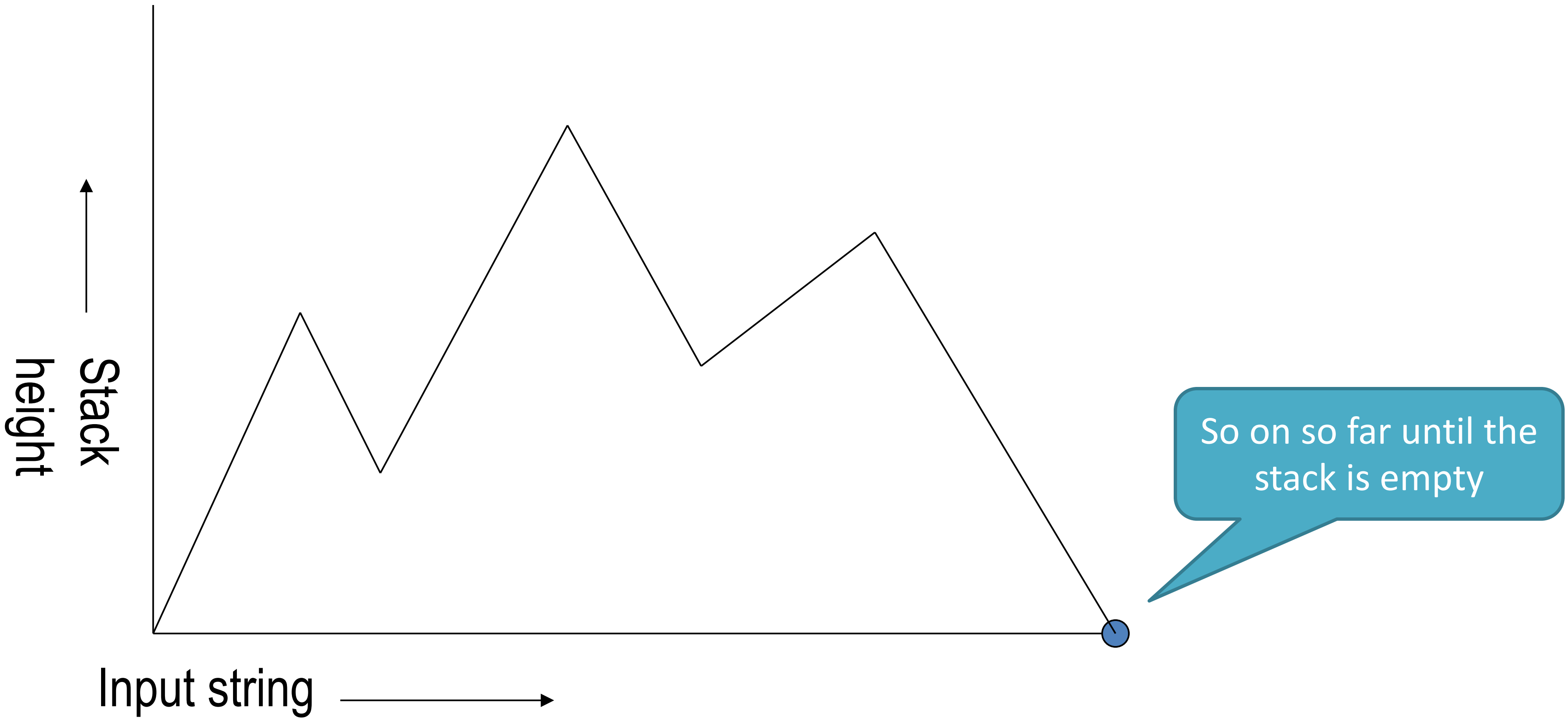
A language is CFL ⇐ some PDA recognizes it

- The process of PDA recognizes some inputs



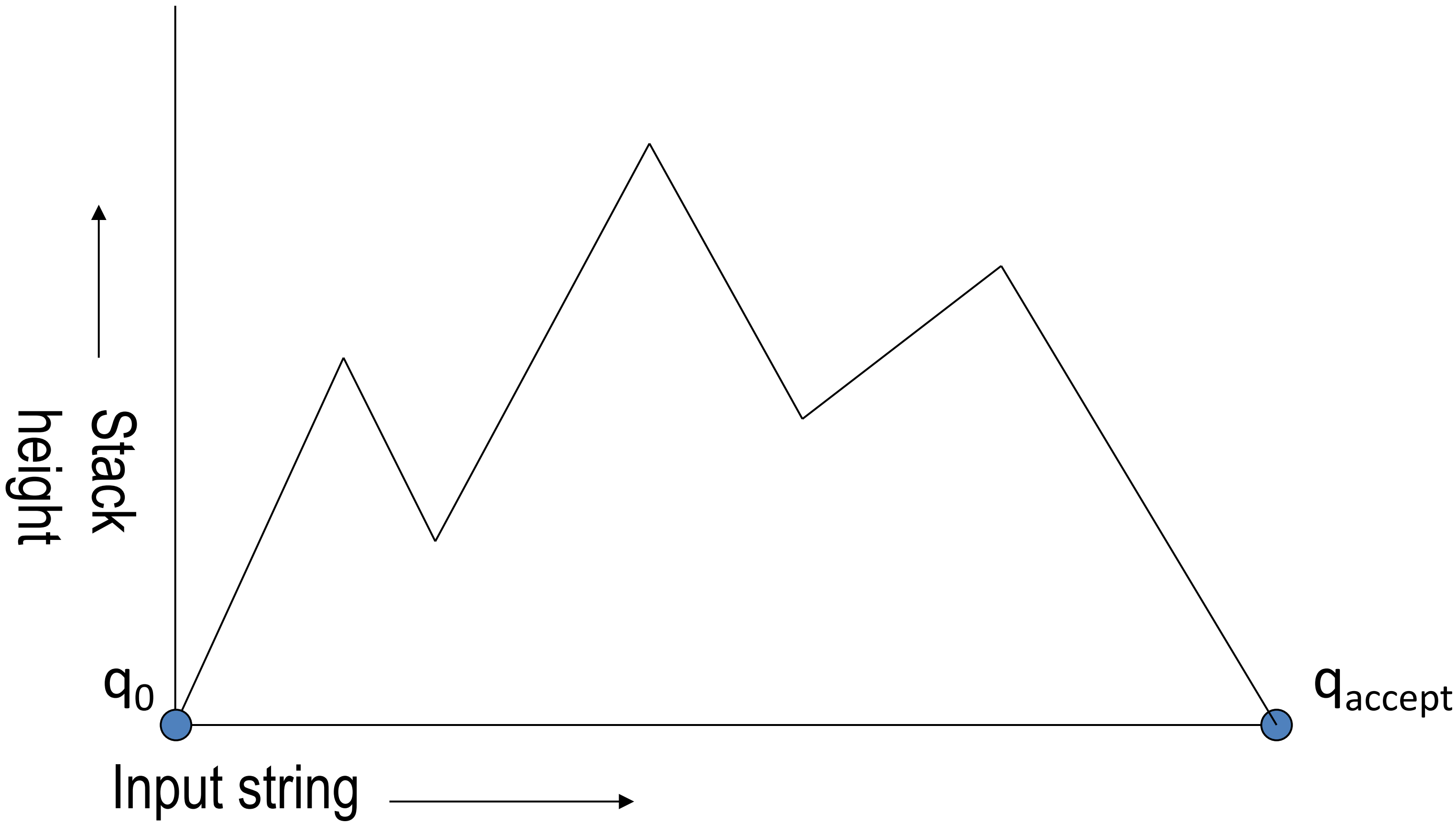
A language is CFL $\Leftarrow$ some PDA recognizes it

- The process of PDA recognizes some inputs



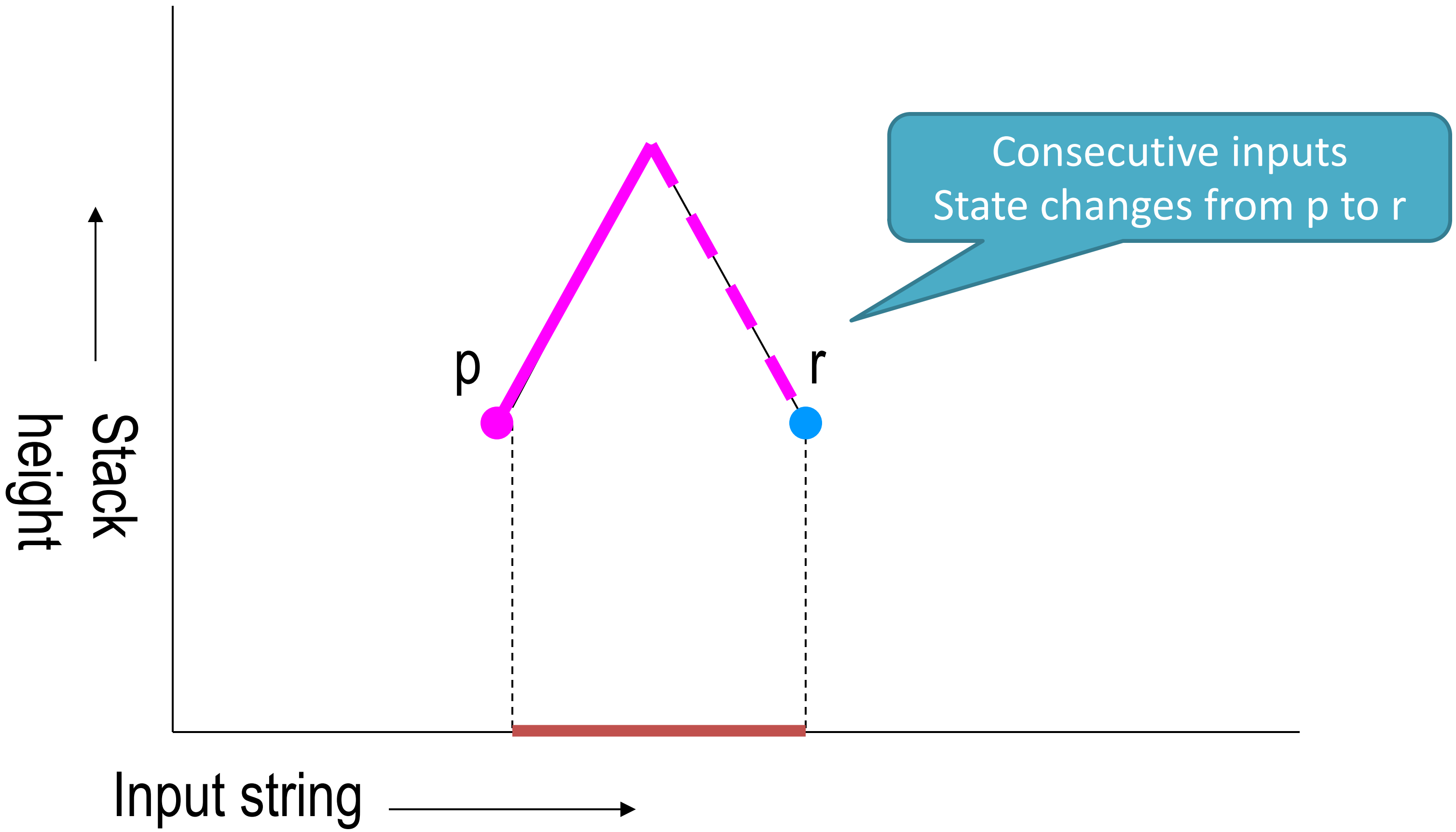
A language is CFL $\Leftarrow$ some PDA recognizes it

- Create CFGs for such movements



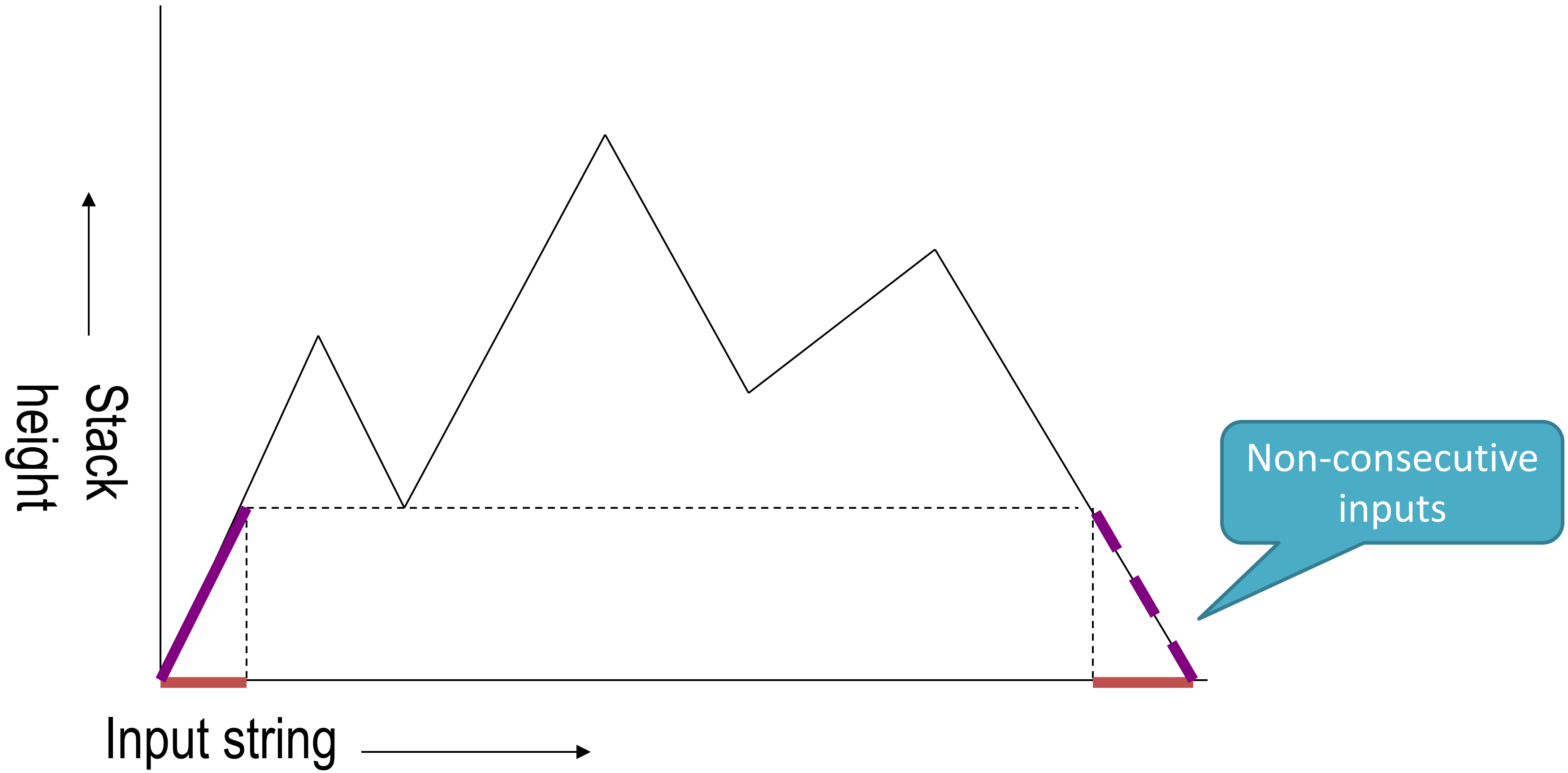
A language is CFL ⇐ some PDA recognizes it

- Case 1: the stack does not change (increase and then decrease) after some consecutive inputs

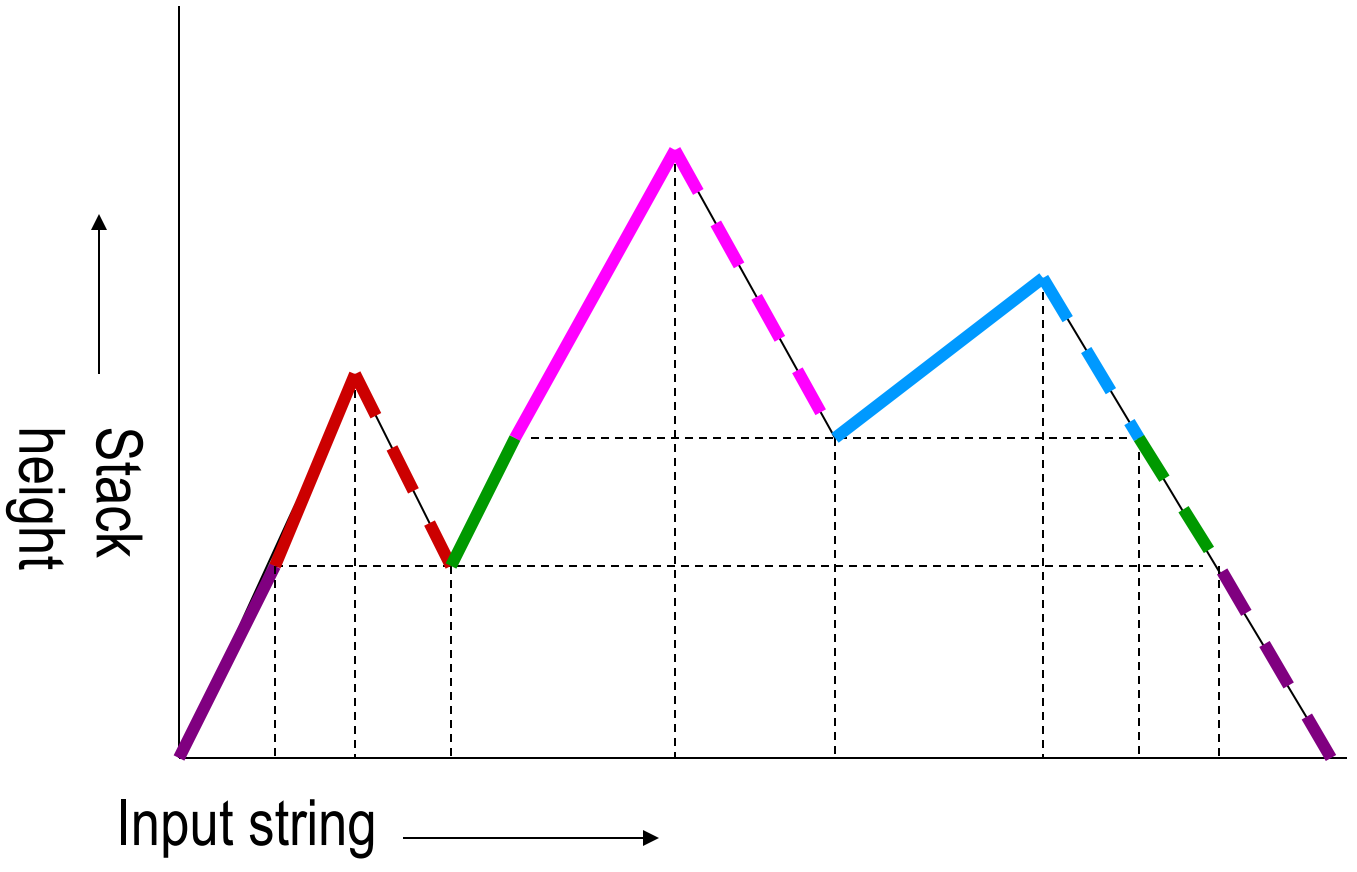


A language is CFL ⇐ some PDA recognizes it

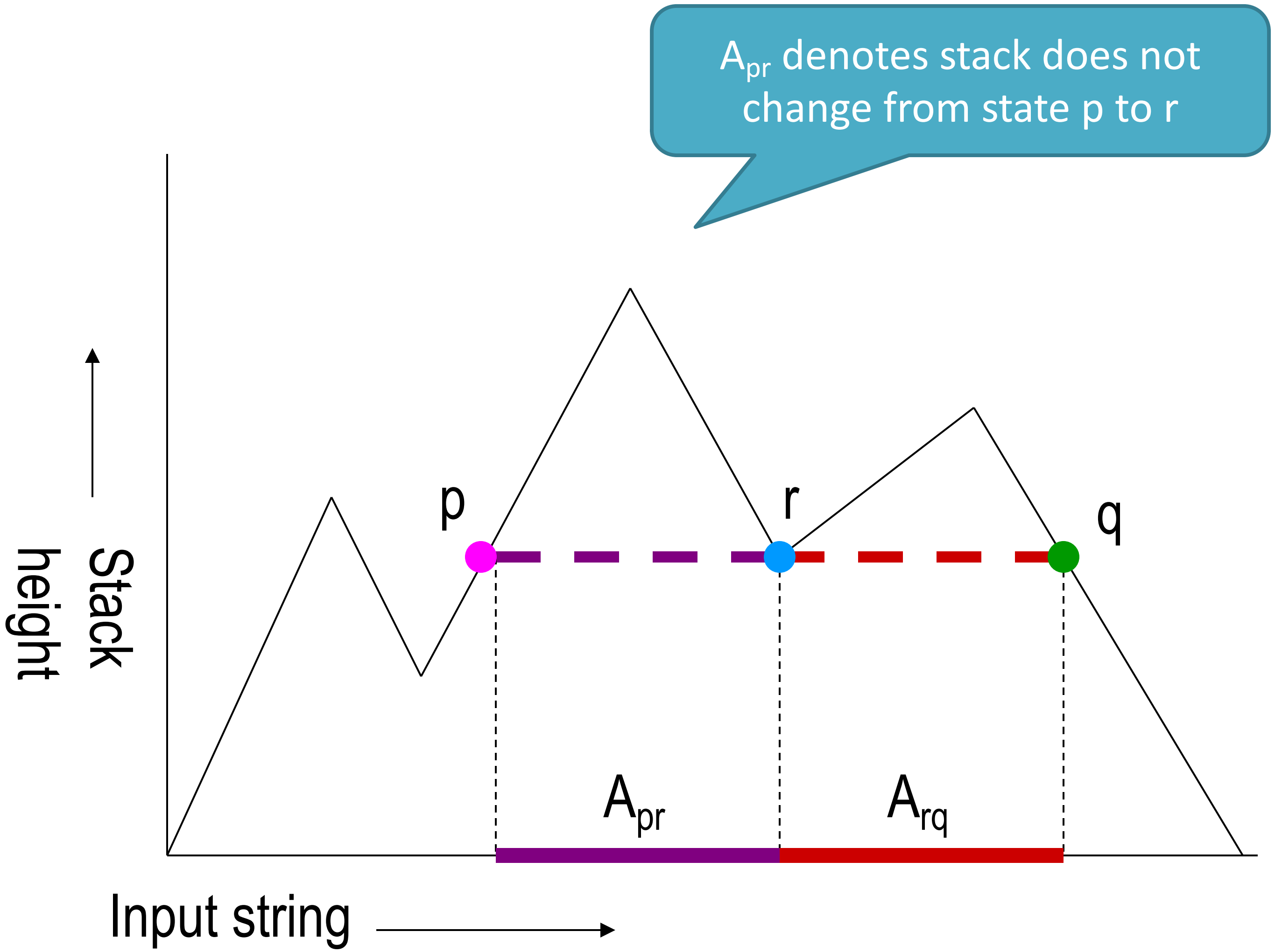
- Case 2: the stack does not change (increase and then decrease) after some non-consecutive inputs



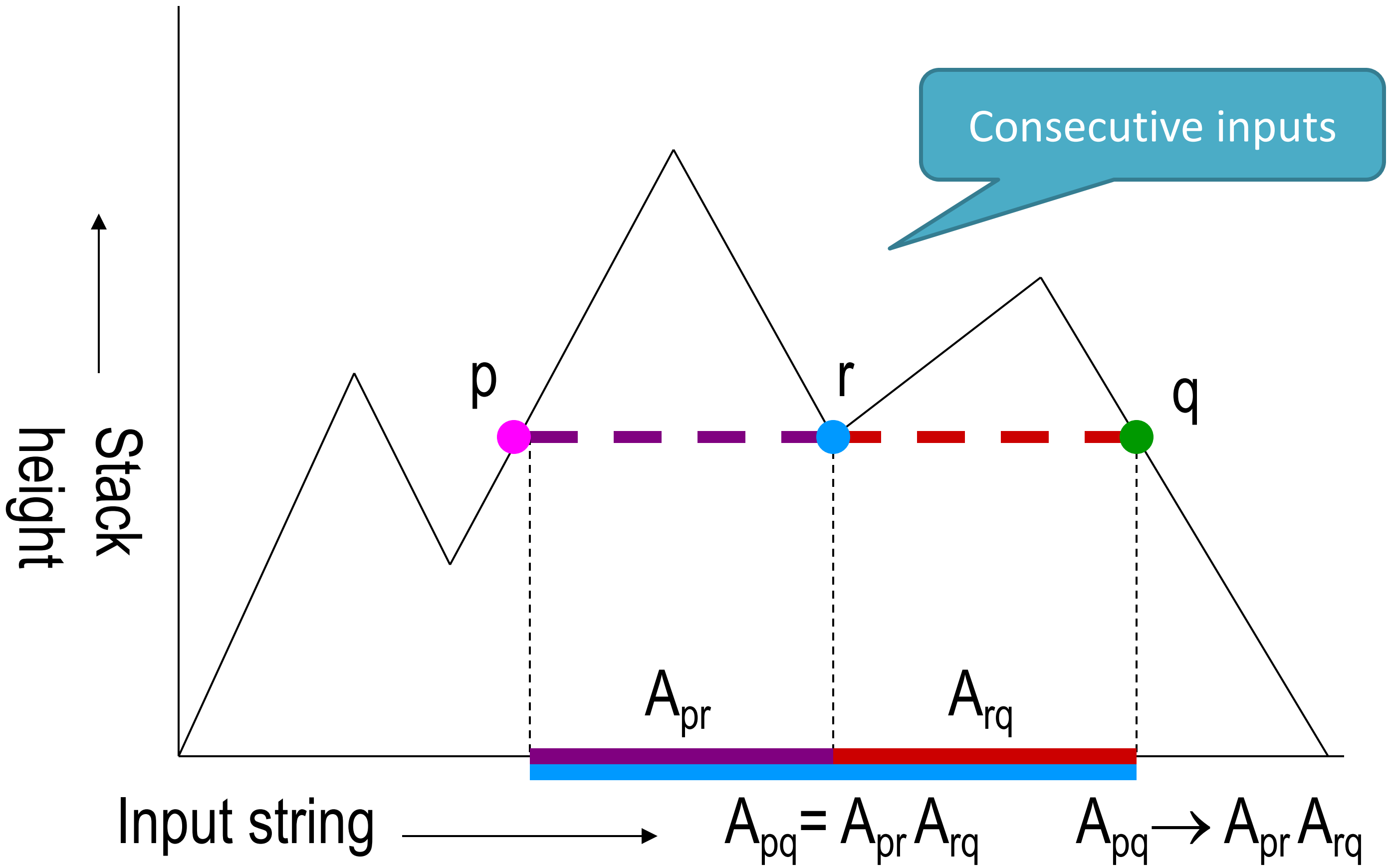
A language is CFL ⇐ some PDA recognizes it



A language is CFL ⇐ some PDA recognizes it

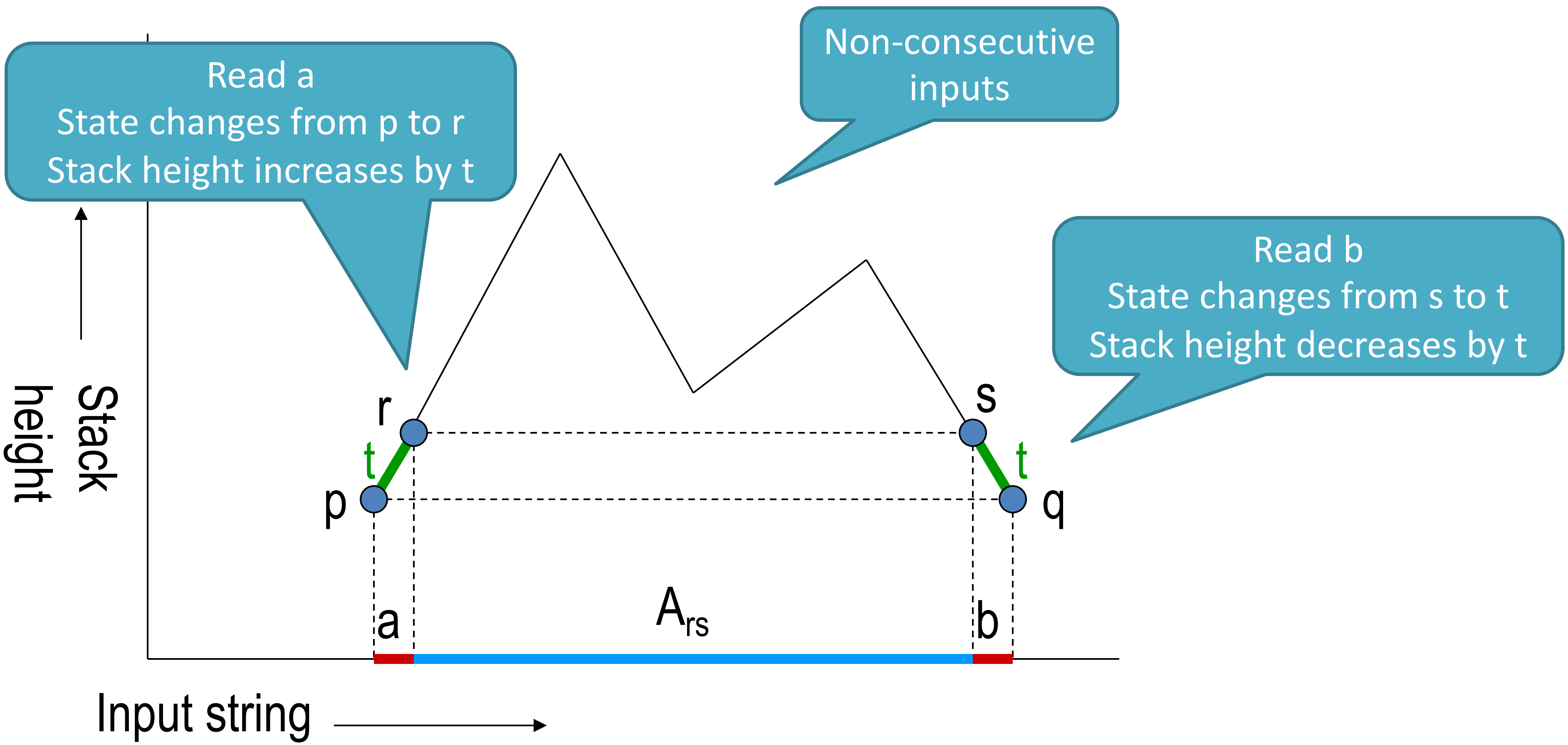


A language is CFL ⇐ some PDA recognizes it

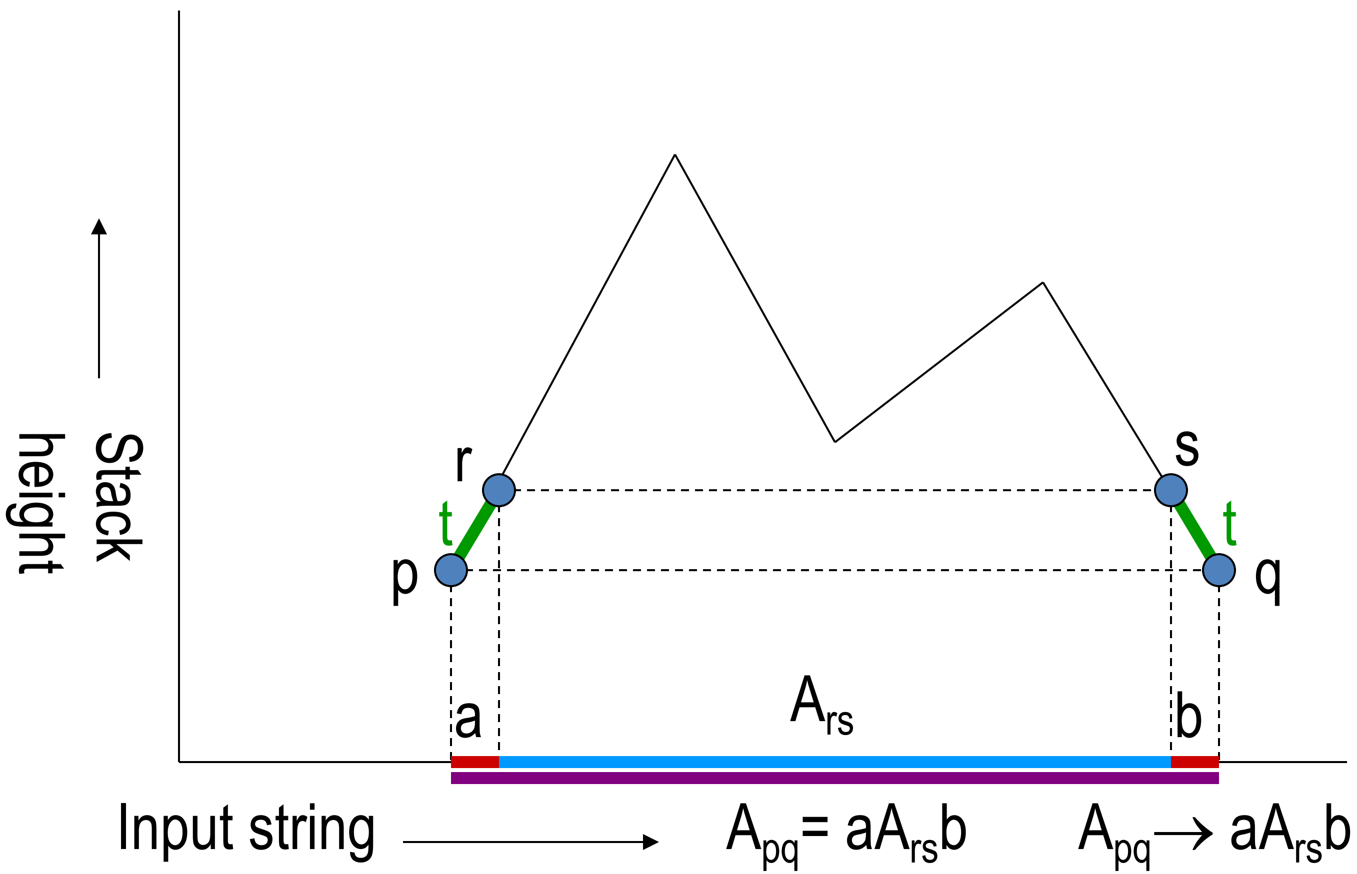


$A_{pq} = A_{pr} A_{rq}$ $\quad$ $A_{pq} \rightarrow A_{pr} A_{rq}$

A language is CFL ⇐ some PDA recognizes it



A language is CFL ⇐ some PDA recognizes it



CFL $\Leftarrow$ PDA

A_{pq} denotes stack movement from p to q (p, q are any two states)

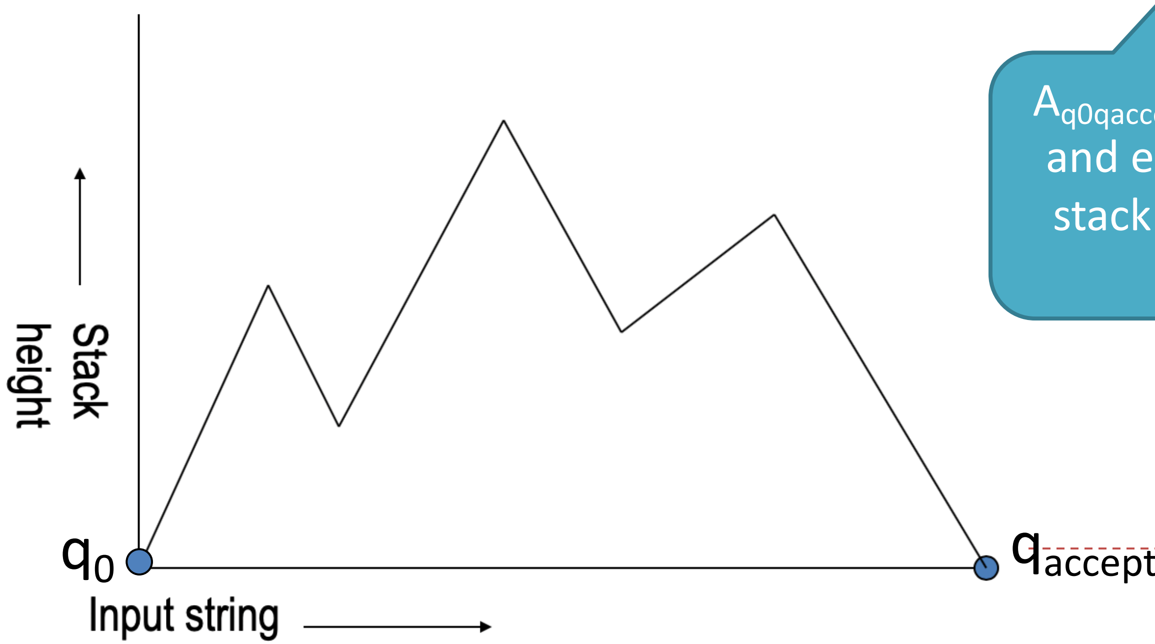
- For a PDA

  $P=(Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$,

  then create equivalent CFG

  $G=(\{A_{pq} \mid p,q \in Q\}, \Sigma, R, A_{q0qaccept})$,

$A_{q0qaccept}$ requires starting from q_0 and ending in q_{accept} and making stack is empty. It defines where the PDA starts.



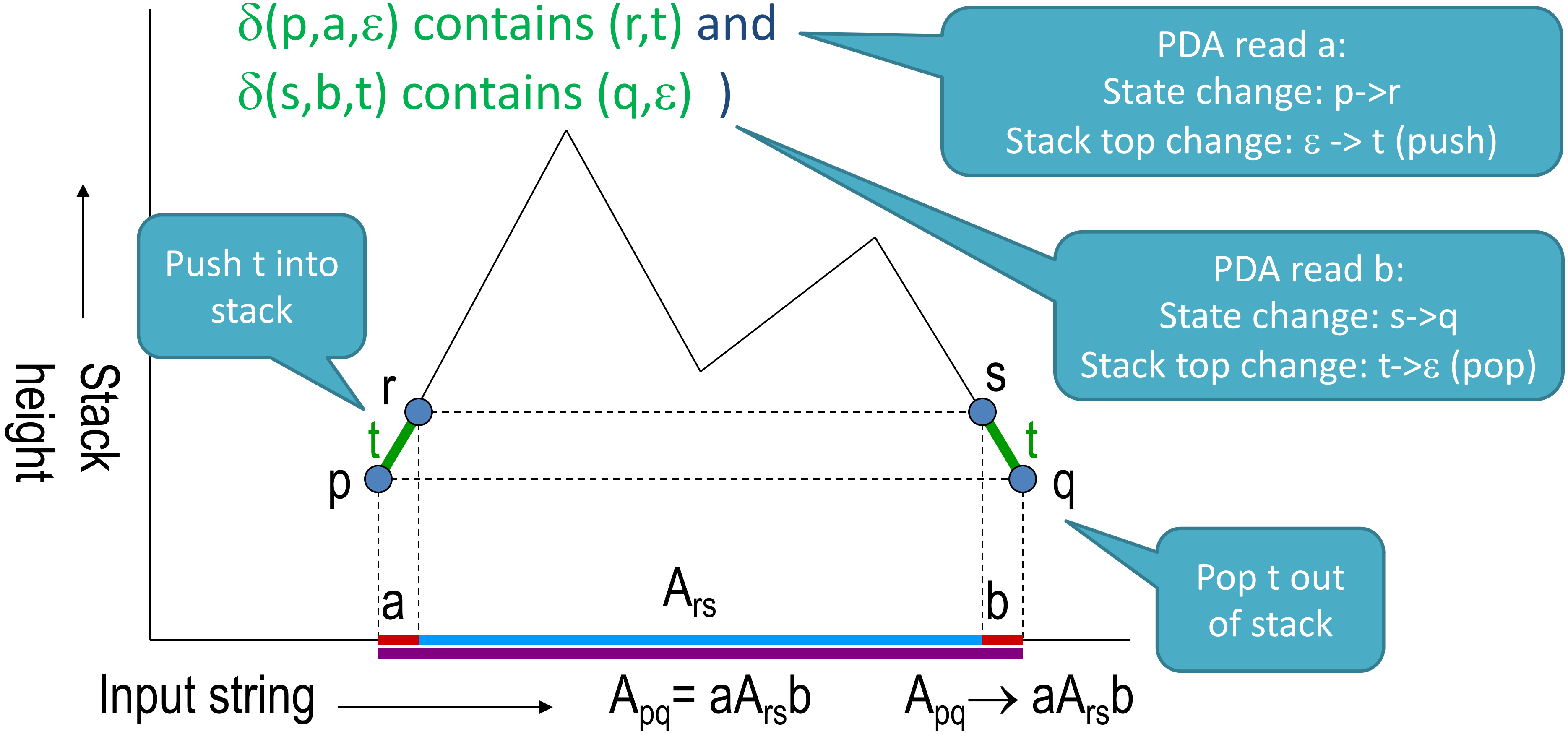
Non-Consecutive input rule

R has the following rules:

(1) $A_{pq} \rightarrow aA_{rs}b$, (for each $p,q,r,s \in Q$, $a,b \in \Sigma_\varepsilon$, $t \in \Gamma$,

$\delta(p,a,\varepsilon)$ contains (r,t) and

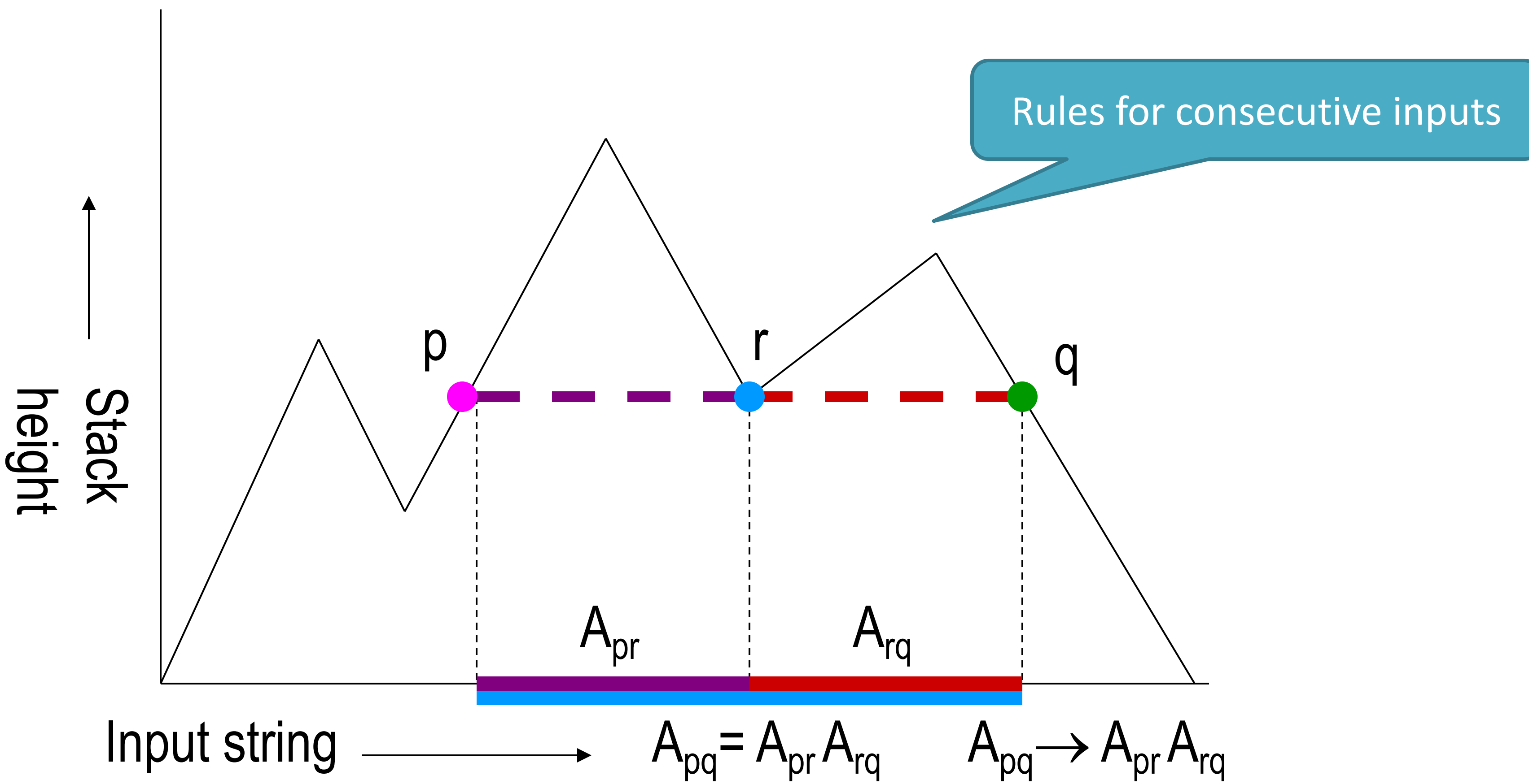
$\delta(s,b,t)$ contains (q,ε))



Consecutive input rule

R has the following rules:

(2) $A_{pq} \rightarrow A_{pr}A_{rq}$, (for each $p,q,r \in Q$)

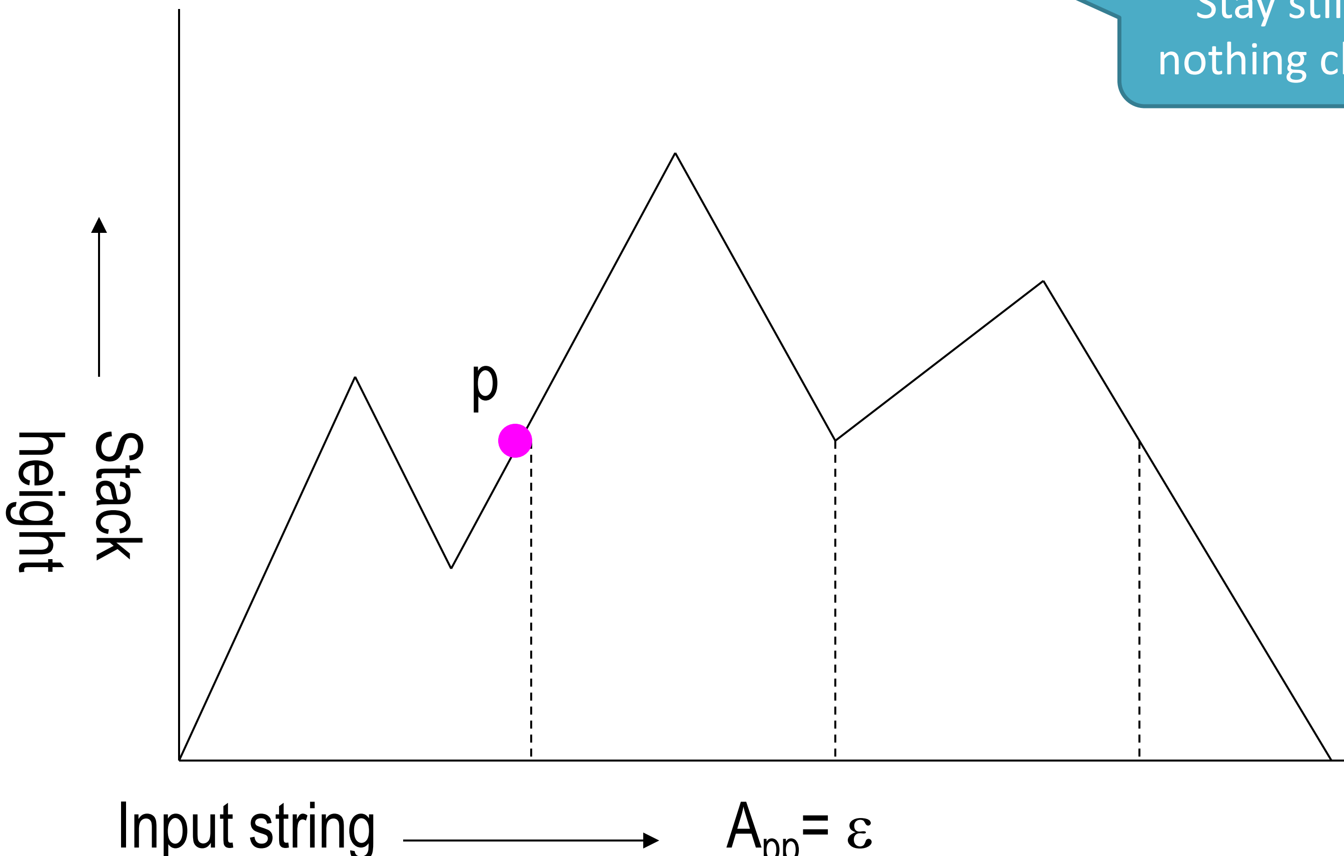


ε rule

R has the following rules:

(3) $A_{pp} \rightarrow \varepsilon$, (for each $p \in Q$)

Stay still and nothing changes



$A_{pp} = \varepsilon$

A language is CFL $\Leftarrow$ some PDA recognizes it

- For a PDA

  $P=(Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$,

  then create equivalent CFG

  $G=(\{A_{pq} \mid p,q \in Q\}, \Sigma, R, A_{q0qaccept})$,

Key idea:

We can create CFGs to simulate the movement of PDA

Conclusion

- Equivalence of PDA and CFG:

- A language is CFL $\Longrightarrow$ some PDA recognizes it

- A language is CFL $\Longleftarrow$ some PDA recognizes it