

CS 6041

Theory of Computation

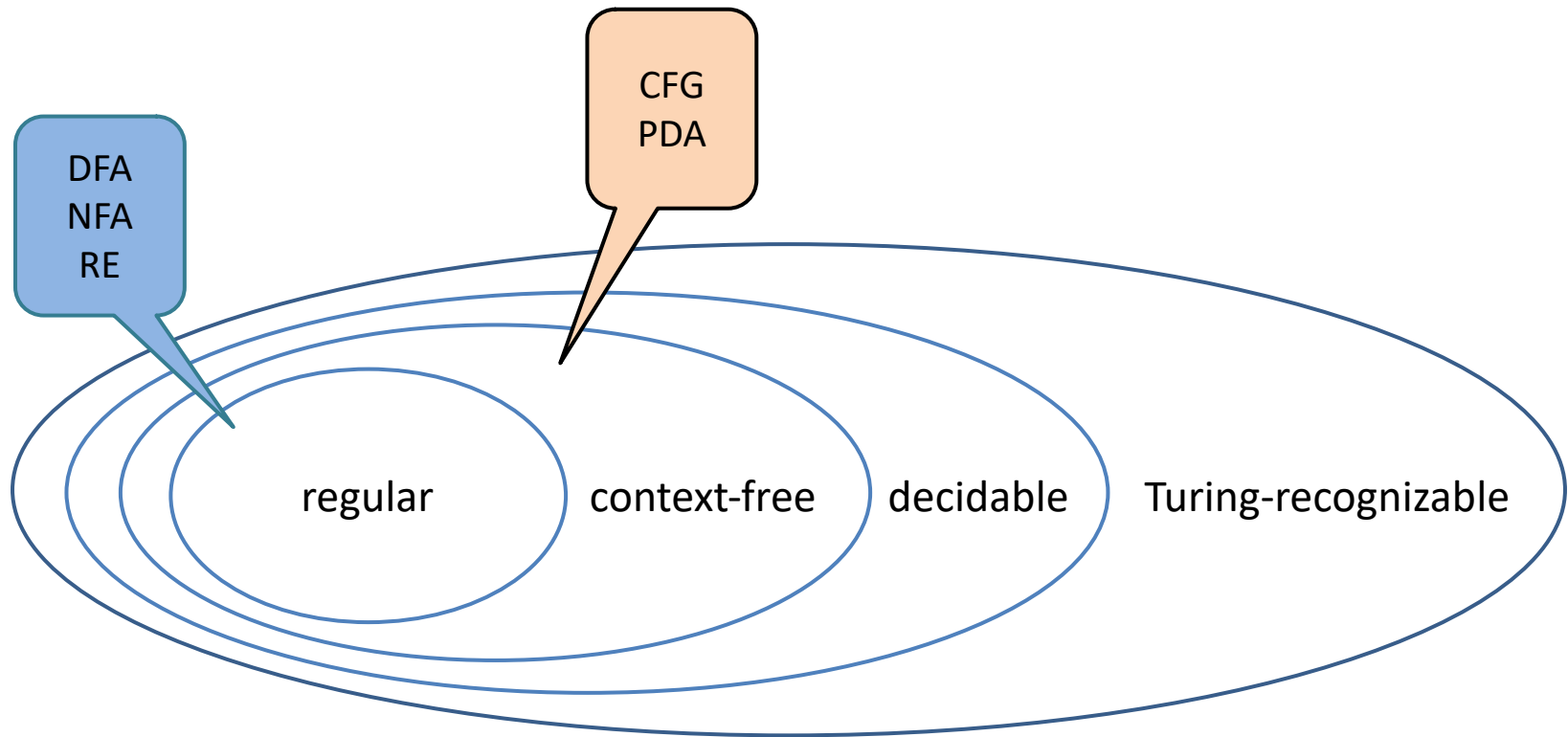
Pushdown Automata

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 \implies some PDA recognizes it
- A language is CFL \iff some PDA recognizes it



A language is CFL \Rightarrow 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: $A \Rightarrow 0A1 \Rightarrow$
 $00A11 \Rightarrow 000A111 \Rightarrow$
 $000B111 \Rightarrow 000\#111$

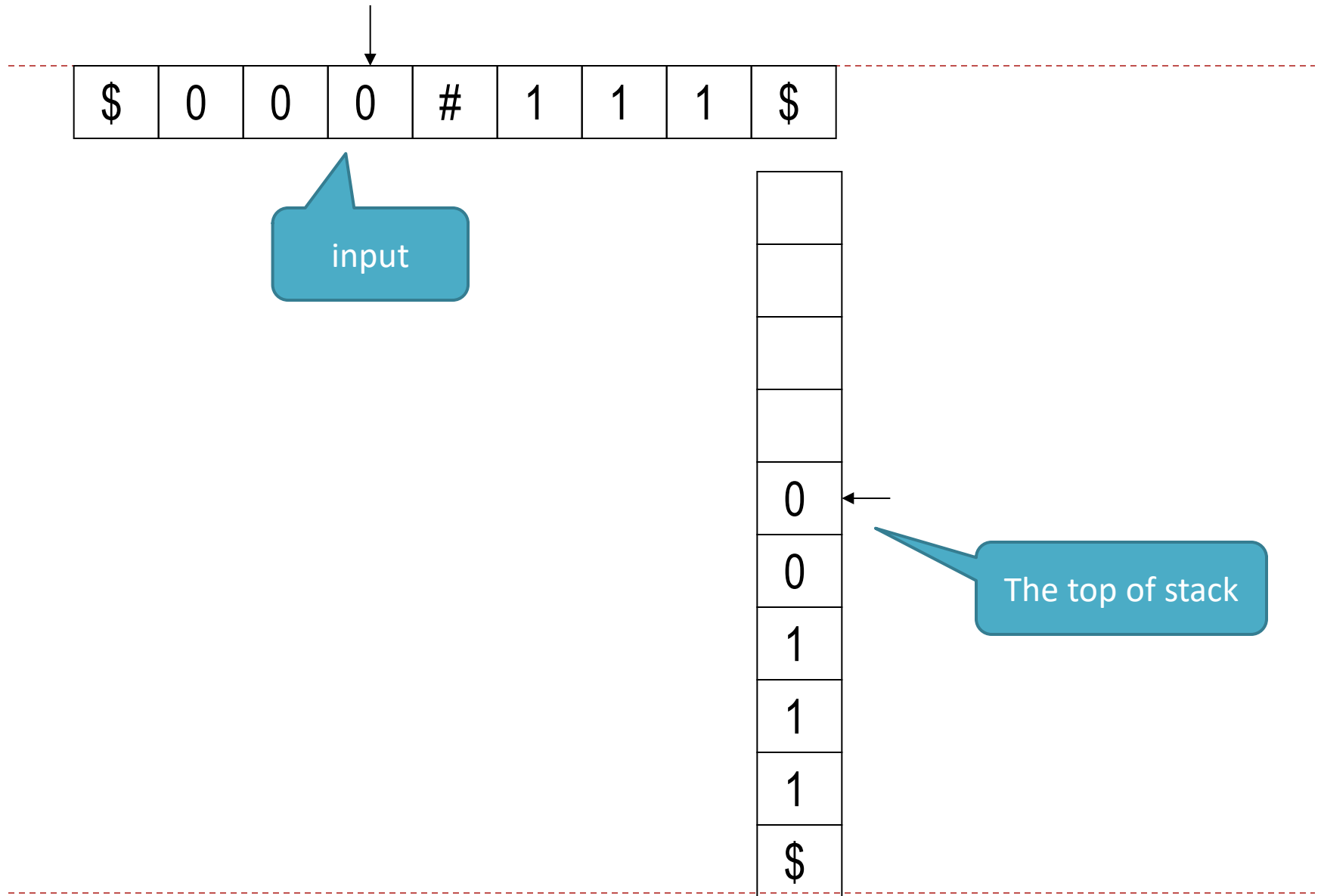
How to create a PDA to recognize $000\#111$

A language is CFL \Rightarrow 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

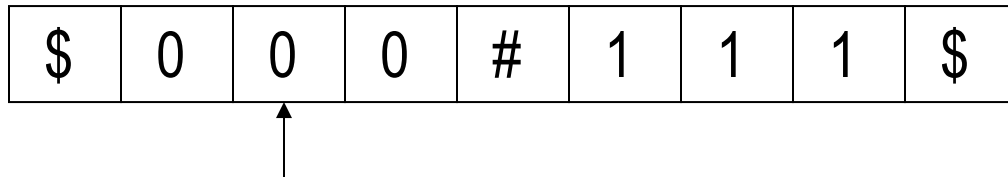


A language is CFL \Rightarrow some PDA recognizes it

- 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



- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

- Derivation : $A \Rightarrow 0A1$

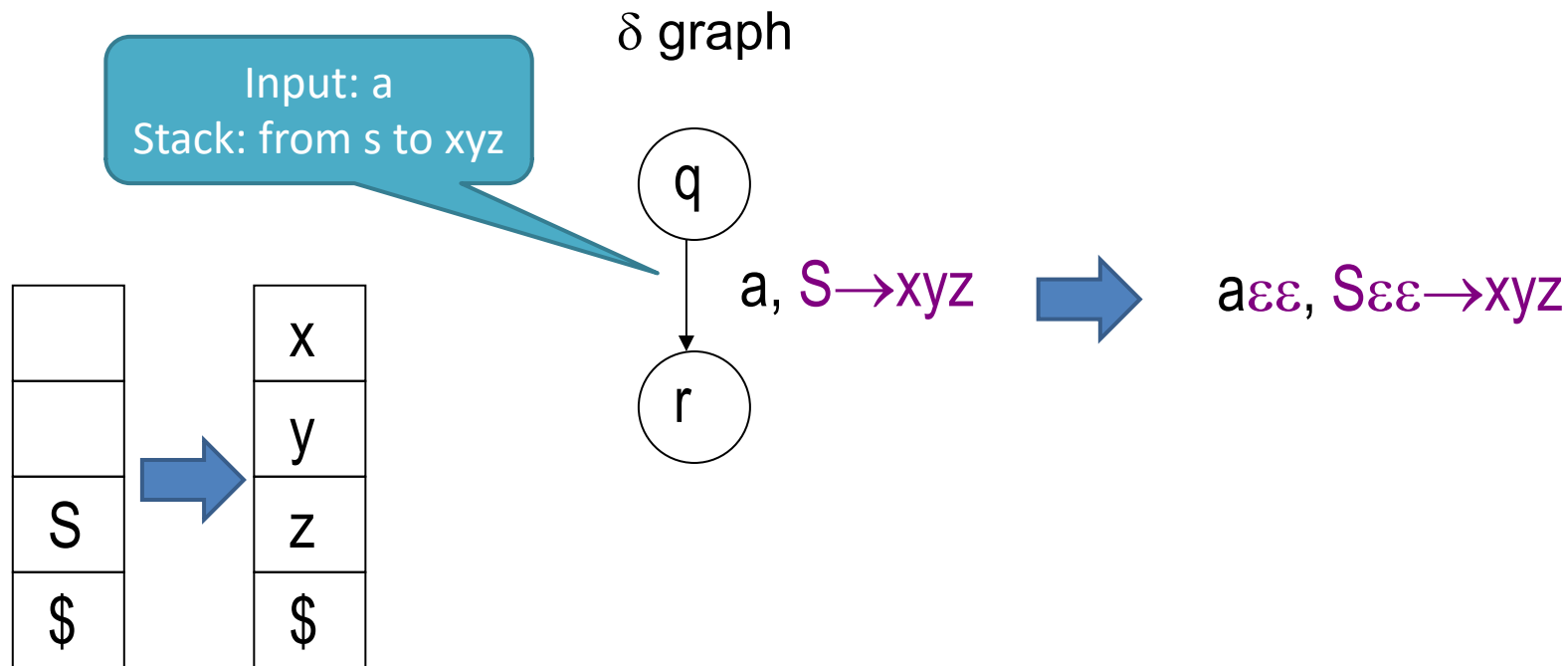


The top of stack is variable



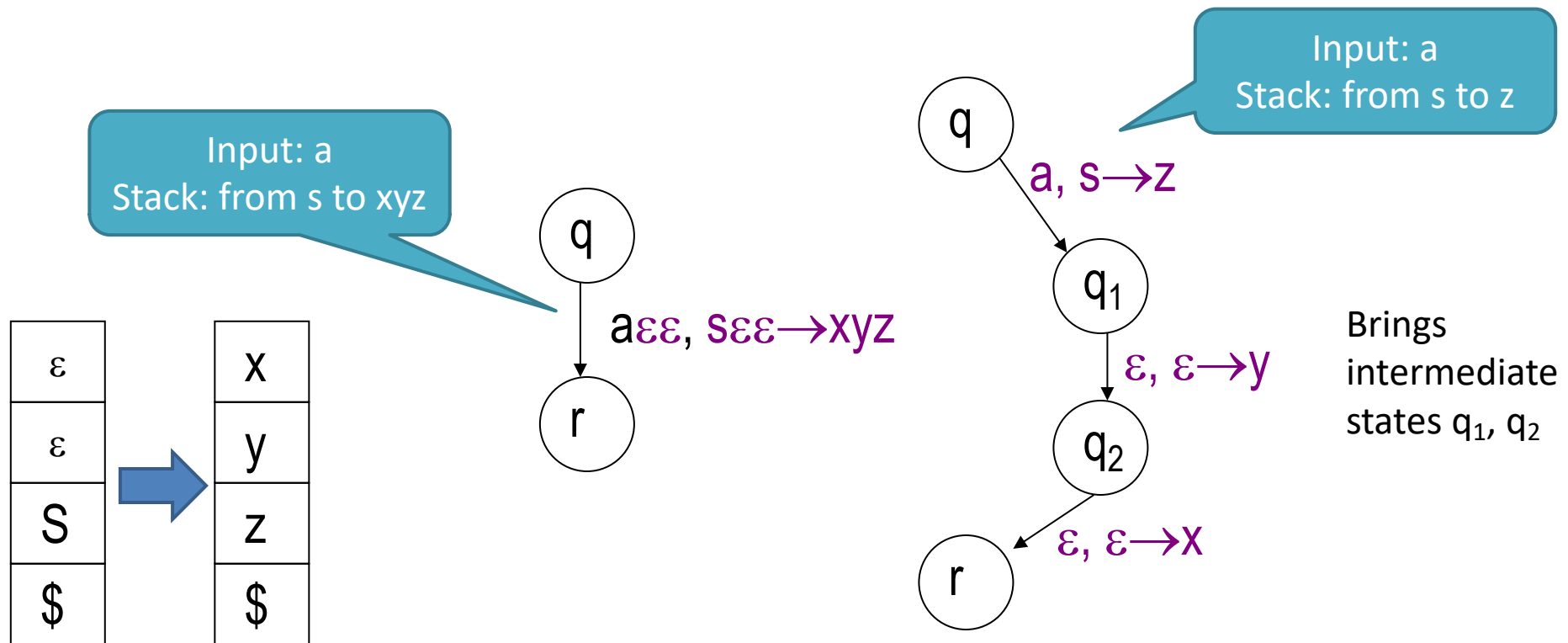
Step 2: If the top of stack is variable, then simulate the derivation

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



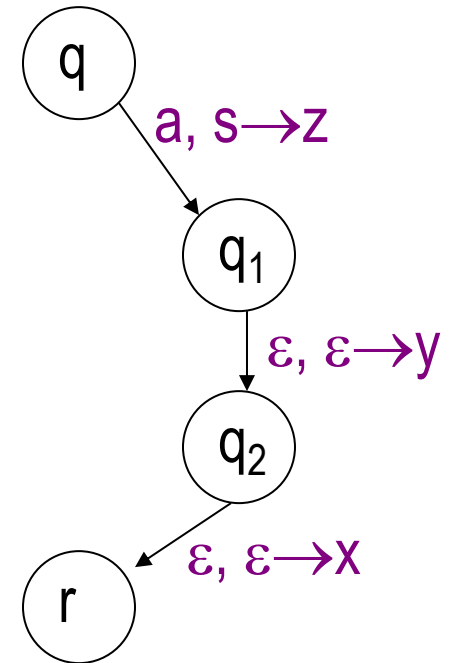
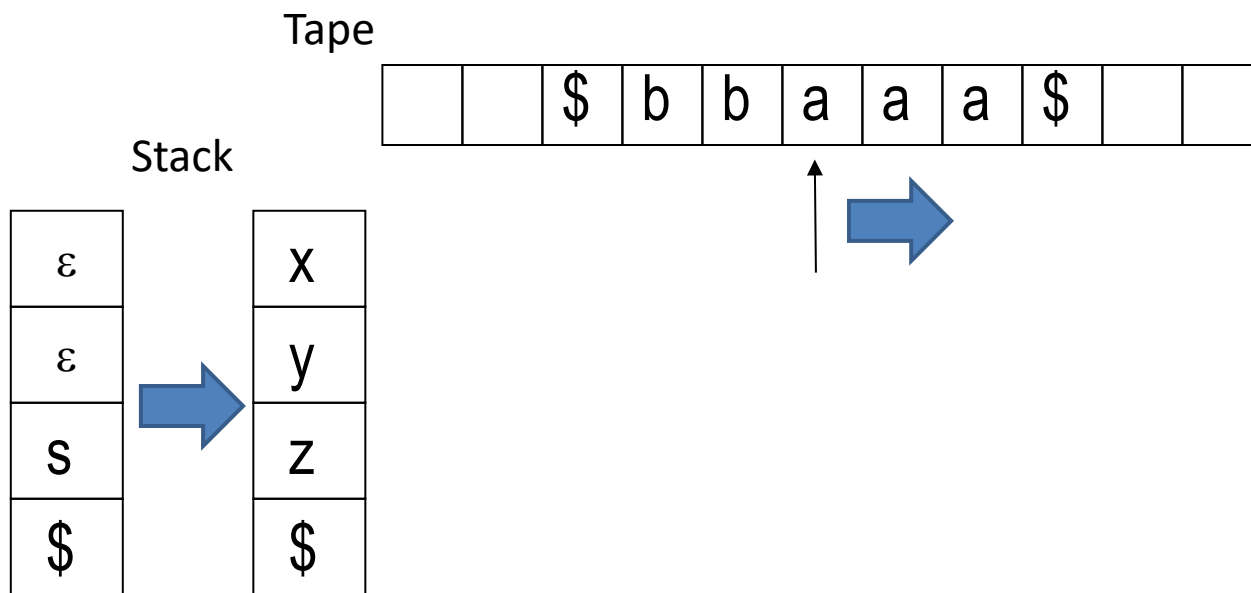
Step 2: If the top of stack is variable, then simulate the derivation

- 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'

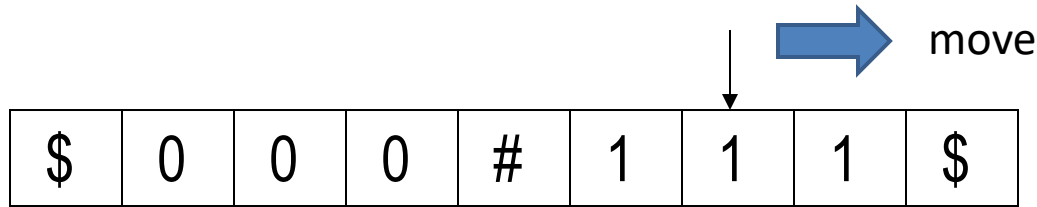


A language is CFL \Rightarrow some PDA recognizes it



- 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**

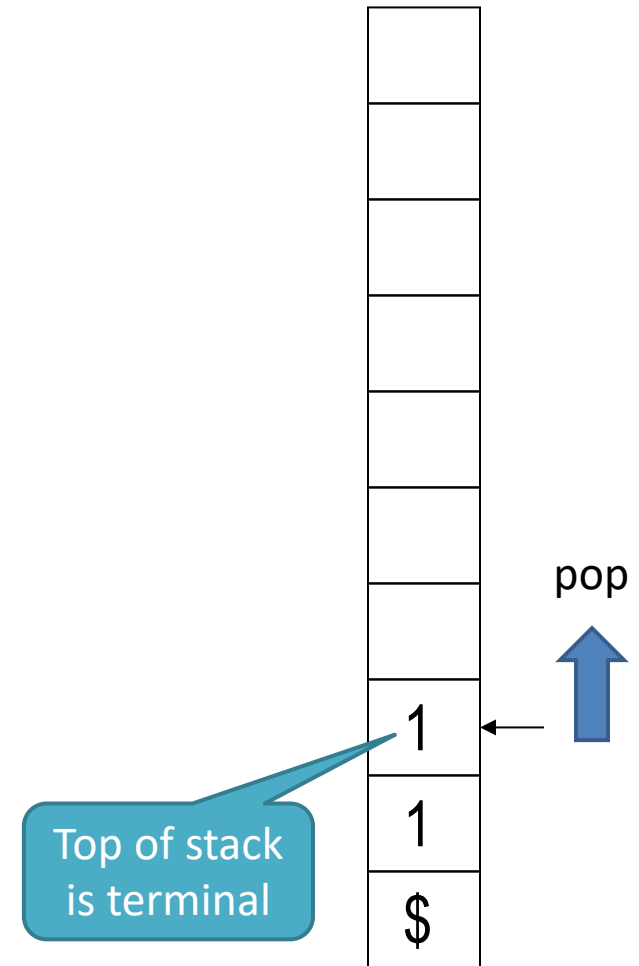


Step 3: If the top of stack is terminal, then do the match



- Compare the current input and top element on stack, if match succeeds:

- The header move forward 
- The stack pop one element 

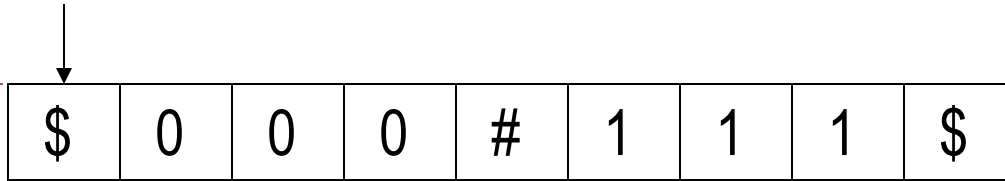


A language is CFL \Rightarrow some PDA recognizes it

- 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



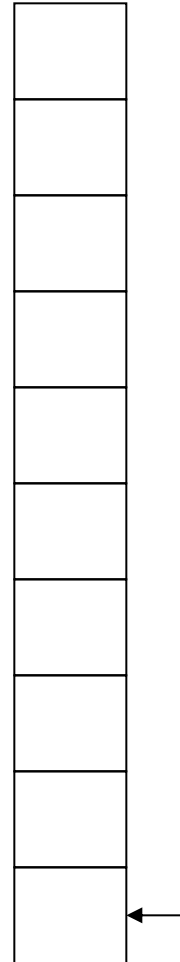
- Grammar G_1 :

$A \rightarrow 0A1$

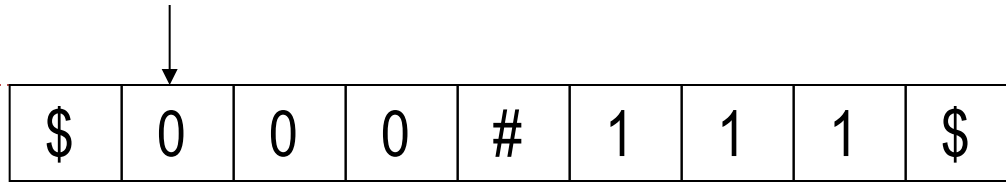
$A \rightarrow B$

$B \rightarrow \#$

- Derivation :



Example: how PDA recognizes the input 000#111



- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

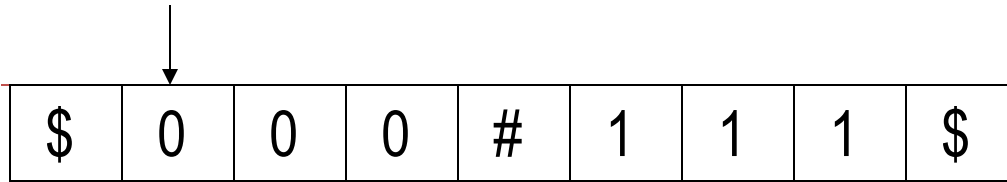
- Derivation :



The bottom of stack \$
Start reading from input



Example: how PDA recognizes the input 000#111

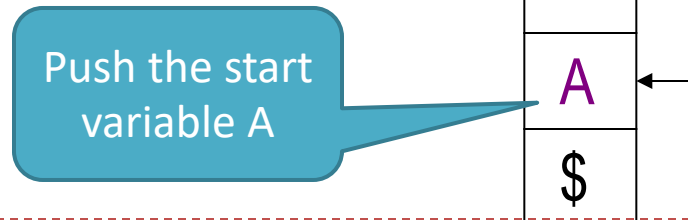


- Grammar G_1 :

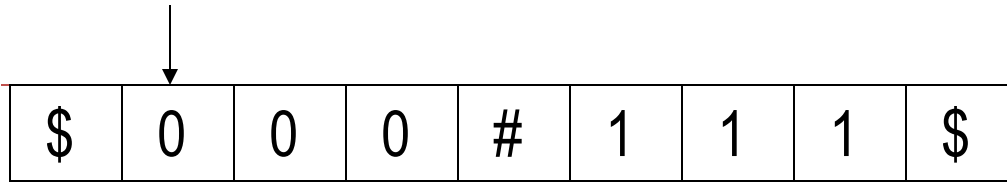
$$A \rightarrow 0A1$$
$$A \rightarrow B$$

B \rightarrow **#**

- Derivation : A



Example: how PDA recognizes the input 000#111



- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

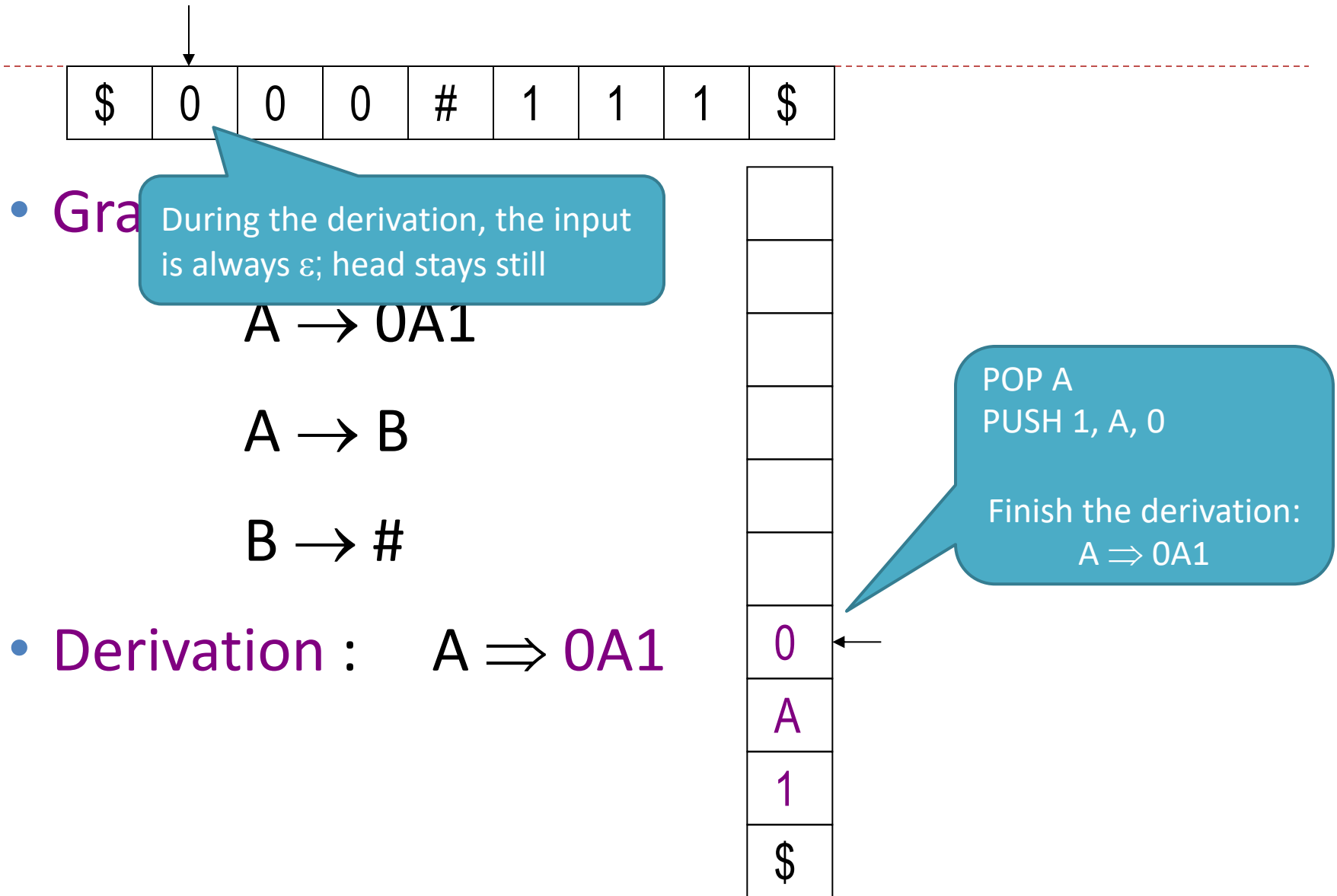
- Derivation : $A \Rightarrow 0A1$



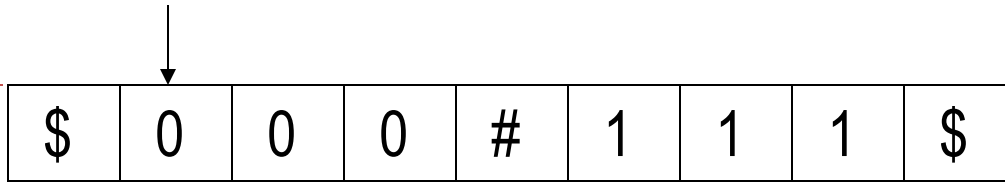
Step 2: The top of stack is variable, so simulate the derivation



Example: how PDA recognizes the input 000#111



Example: how PDA recognizes the input 000#111



- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

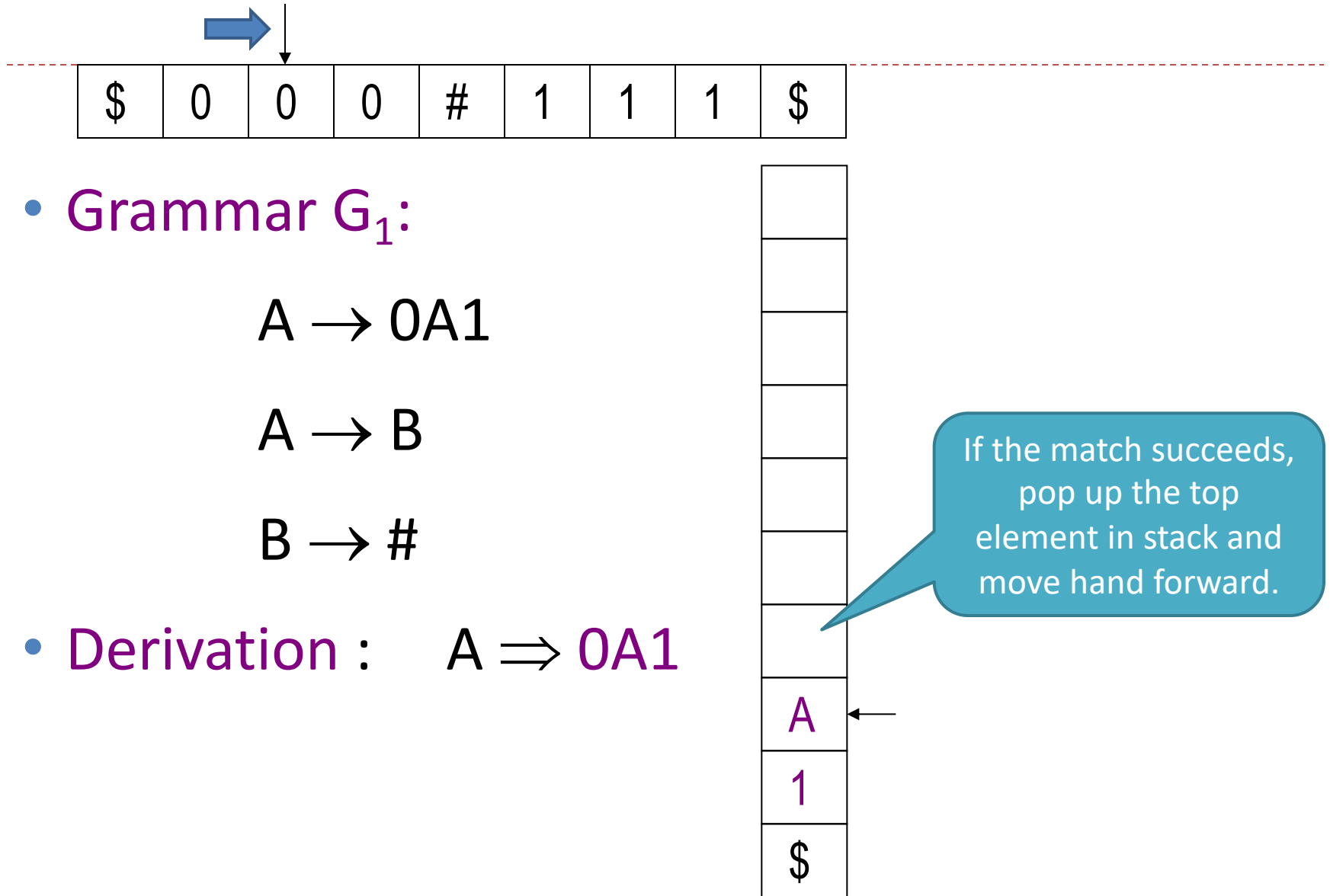
- Derivation : $A \Rightarrow 0A1$



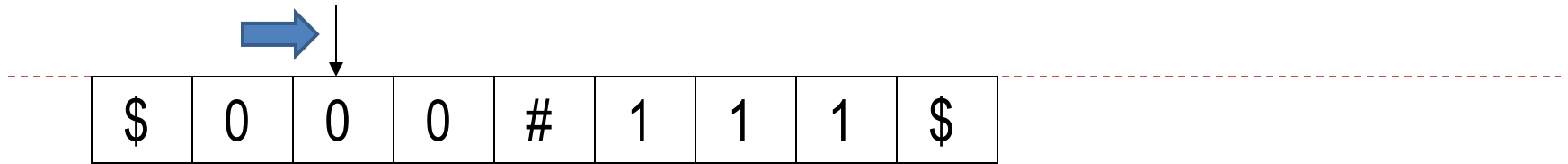
Step 3: If it is terminal, do the match operation.



Example: how PDA recognizes the input 000#111



Example: how PDA recognizes the input 000#111



- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

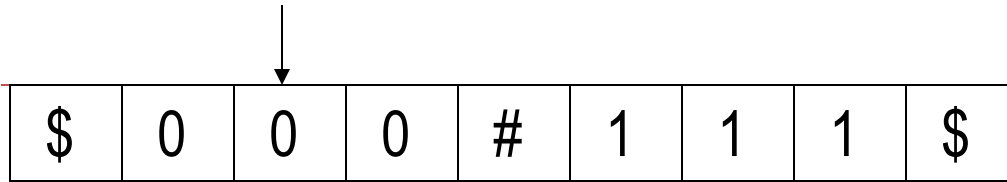
- Derivation : $A \Rightarrow 0A1$



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



Example: how PDA recognizes the input 00#111



- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

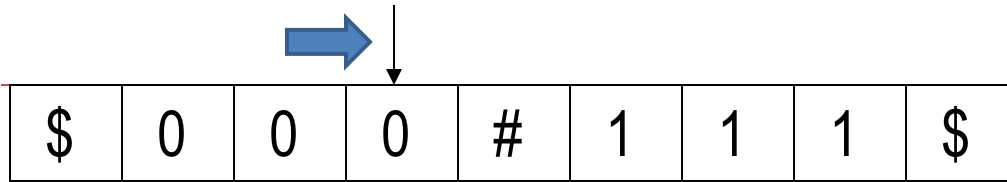
- Derivation : $A \Rightarrow 0A1$
 $\Rightarrow 00A11$



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 : $A \Rightarrow 0A1$
 $\Rightarrow 00A11$

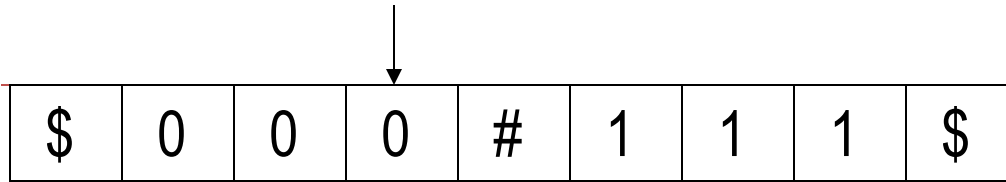


If the match succeeds, pop up the top element in stack and move hand forward.

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 :

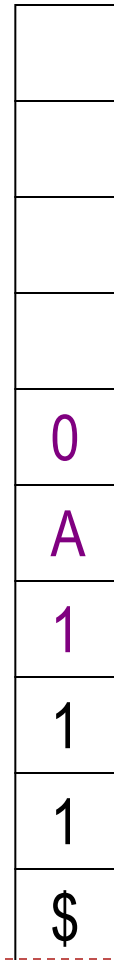
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

- Derivation : $A \Rightarrow 0A1$

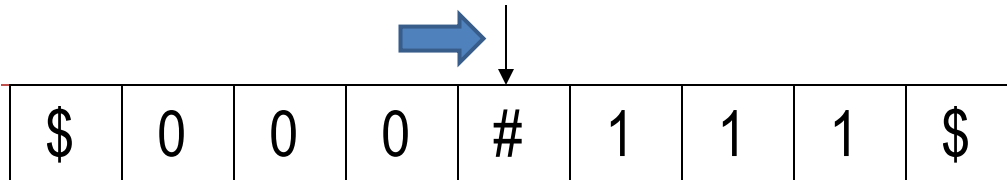
$\Rightarrow 00A11 \Rightarrow 000A111$



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 : $A \Rightarrow 0A1$

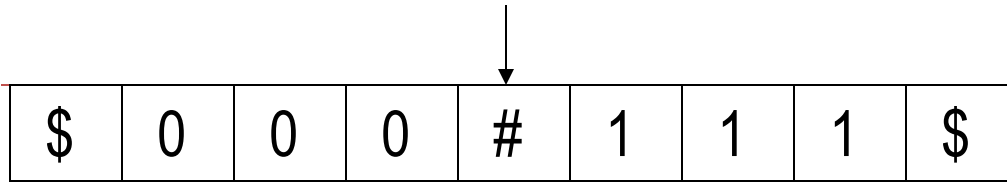
$\Rightarrow 00A11 \Rightarrow 000A111$



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 : $A \Rightarrow 0A1$

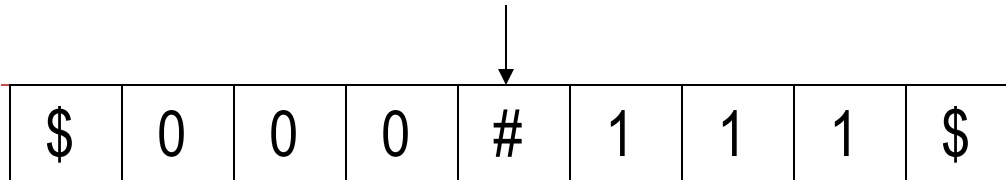
$\Rightarrow 00A11 \Rightarrow 000A111$



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 : $A \Rightarrow 0A1 \Rightarrow$

$00A11 \Rightarrow 000A111 \Rightarrow$

$000B111$

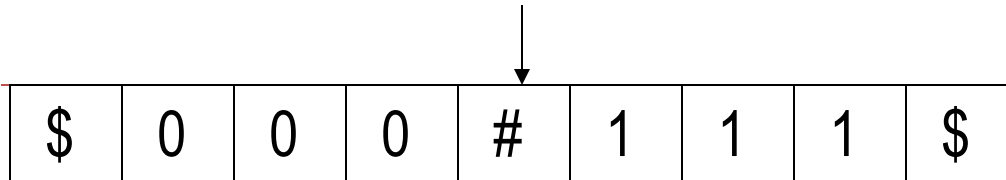


This time, we perform the derivation using grammar $A \rightarrow B$

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



Example: how PDA recognizes the input 000#111



- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

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

$00A11 \Rightarrow 000A111 \Rightarrow$

$000B111 \Rightarrow 000\#111$

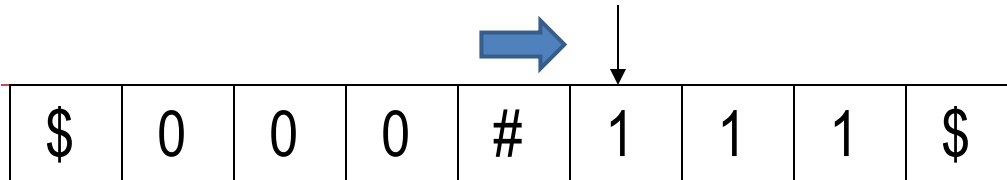


We perform the derivation using grammar $B \rightarrow \#$

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 : $A \Rightarrow 0A1 \Rightarrow$

$00A11 \Rightarrow 000A111 \Rightarrow$

$000B111 \Rightarrow 000\#111$



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 : $A \Rightarrow 0A1 \Rightarrow$

$00A11 \Rightarrow 000A111 \Rightarrow$

$000B111 \Rightarrow 000\#111$

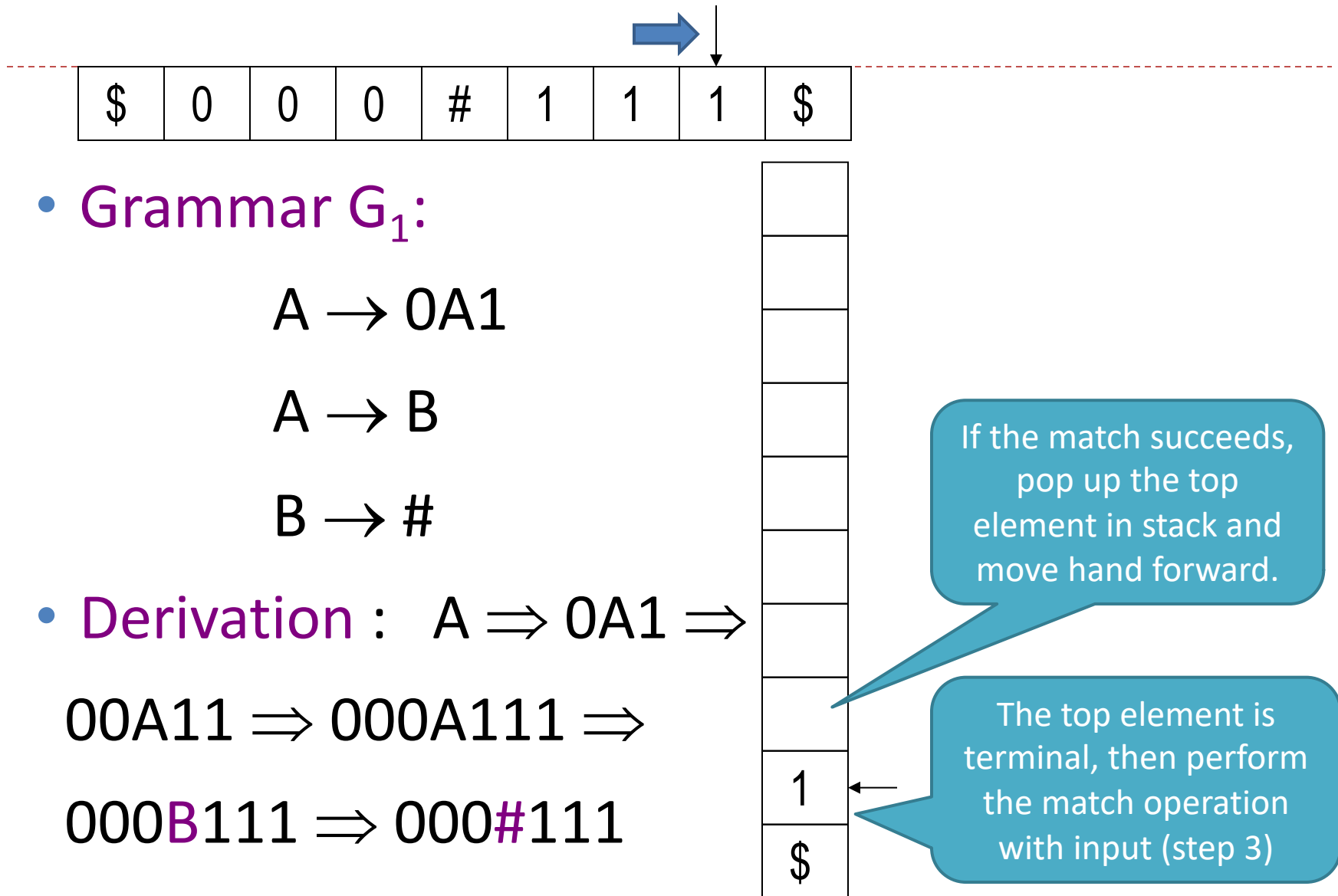
1
1
\$

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

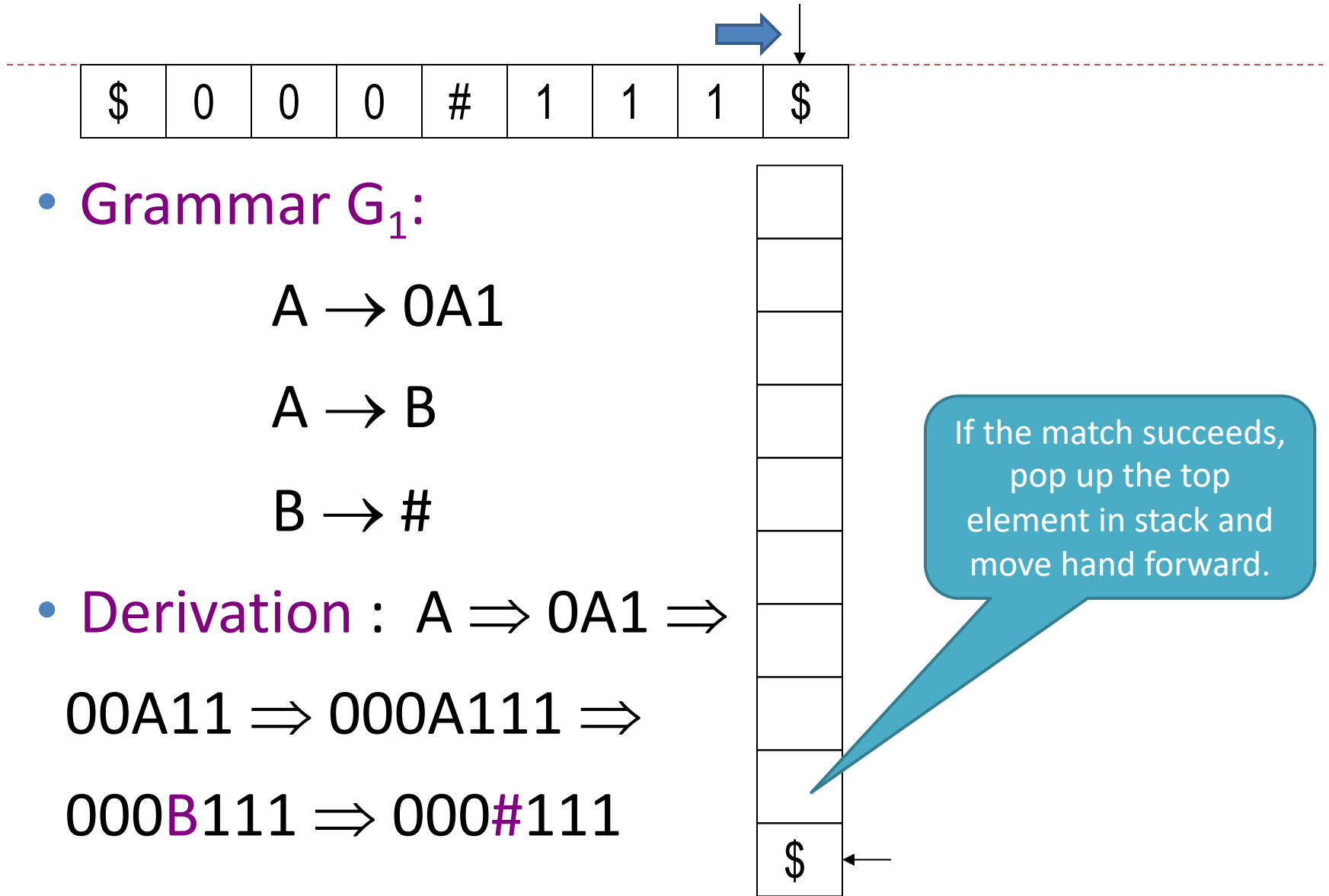
The top element is terminal, then perform the match operation with input (step 3)



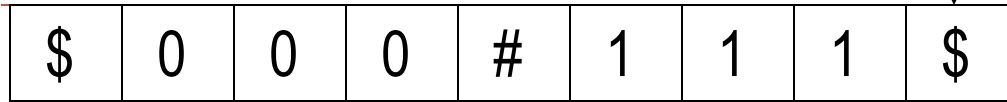
Example: how PDA recognizes the input 000#111



Example: how PDA recognizes the input 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 input is finished and stack is empty, accept; otherwise reject.

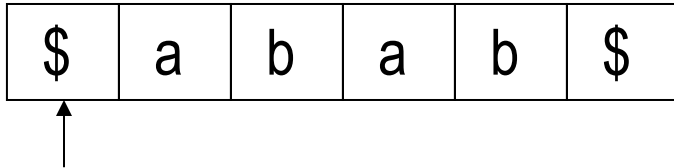


A language is CFL \Rightarrow some PDA recognizes it

- 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



- Grammar G_3 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

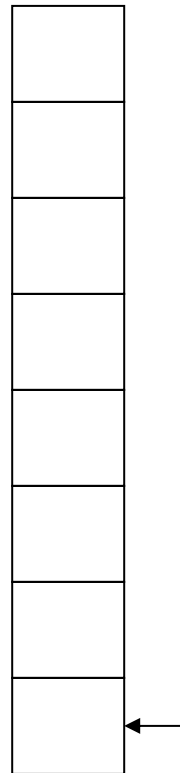
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

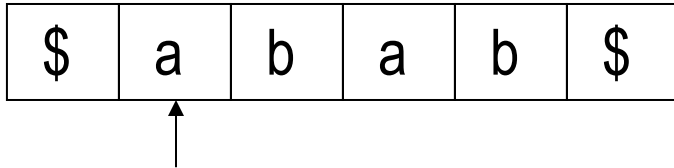
$$abaSb \Rightarrow abab$$

- 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 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

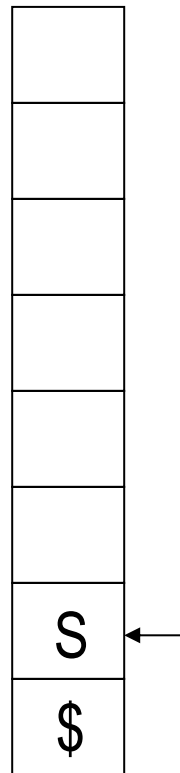
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

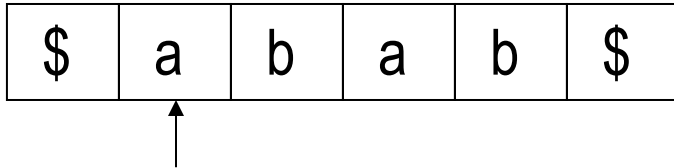
$$abaSb \Rightarrow abab$$

- 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 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

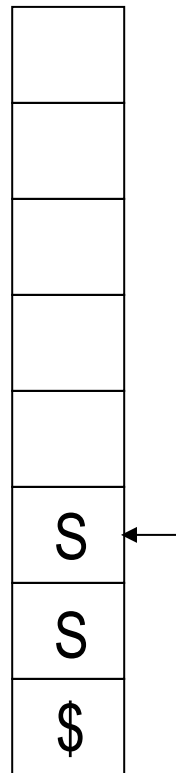
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

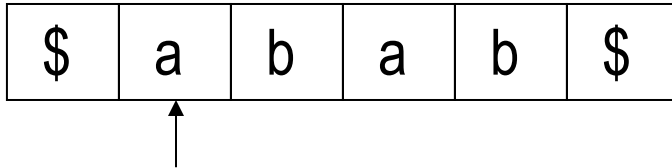
$$abaSb \Rightarrow abab$$

- 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



- Grammar G_3 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

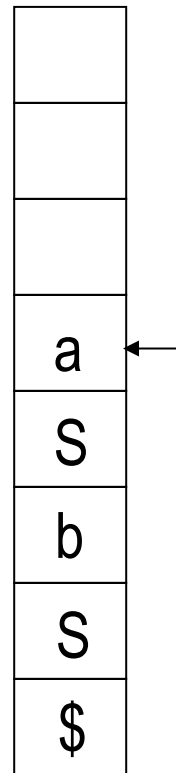
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

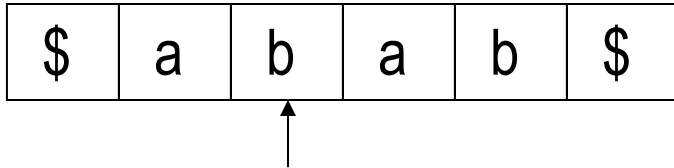
$$abaSb \Rightarrow abab$$

- 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 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

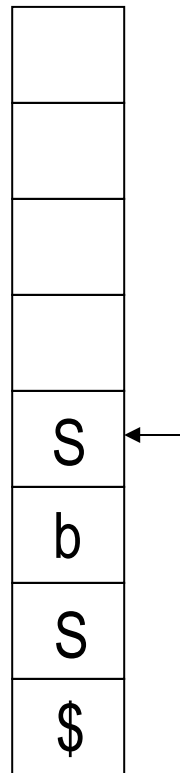
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

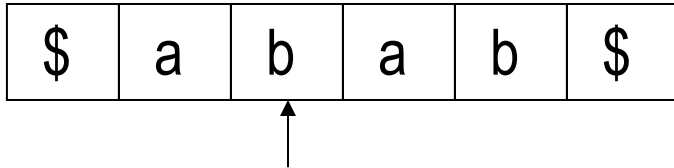
$$abaSb \Rightarrow abab$$

- 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 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

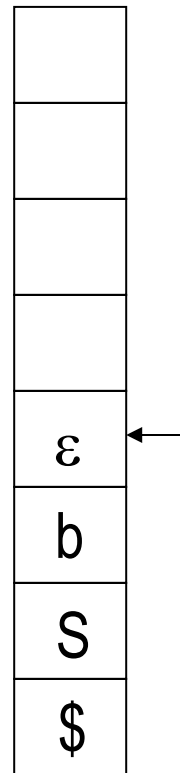
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

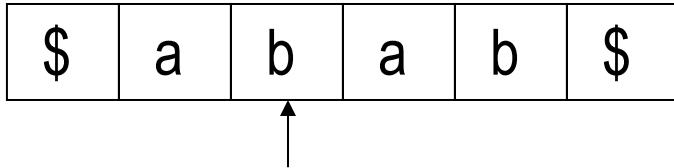
$$abaSb \Rightarrow abab$$

- 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



- Grammar G_3 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

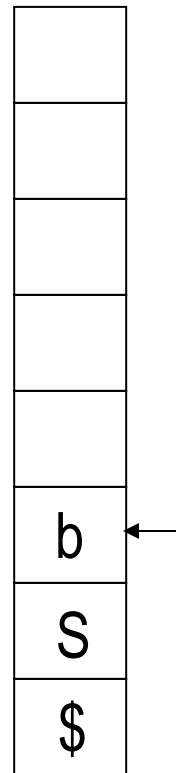
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

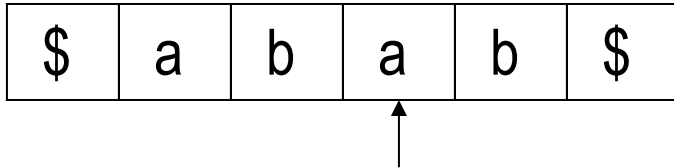
$$abaSb \Rightarrow abab$$

- 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 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

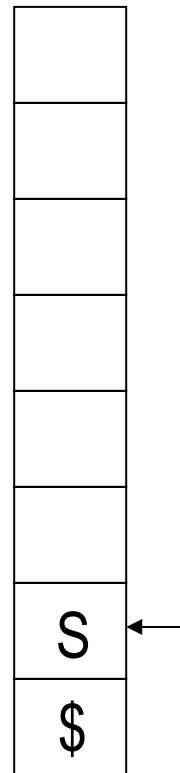
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

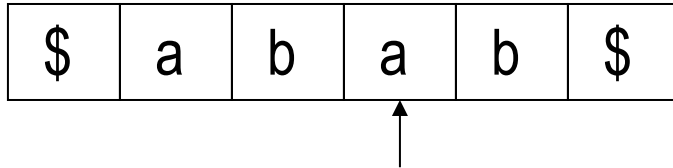
$$abaSb \Rightarrow abab$$

- 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 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

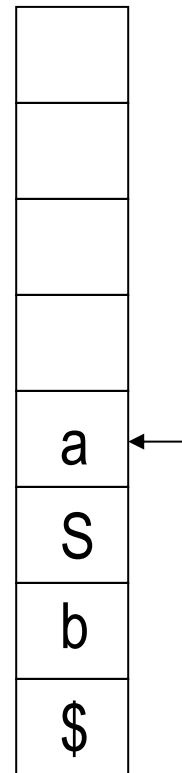
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

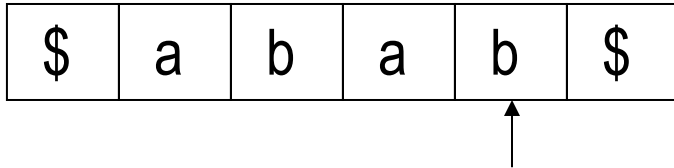
$$abaSb \Rightarrow abab$$

- 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



- Grammar G_3 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

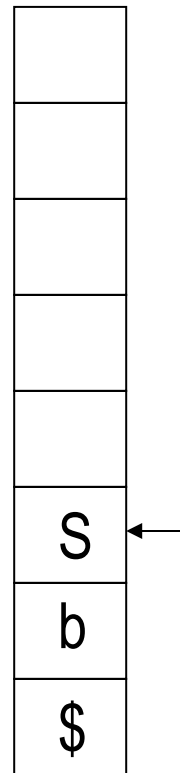
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

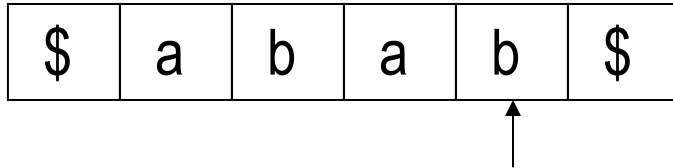
$$abaSb \Rightarrow abab$$

- 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 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

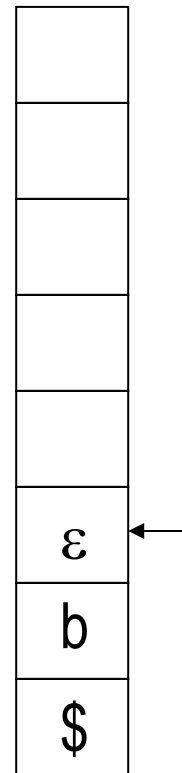
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

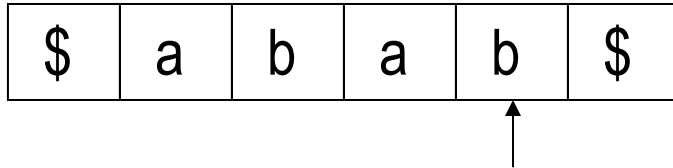
$$abaSb \Rightarrow abab$$

- 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 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

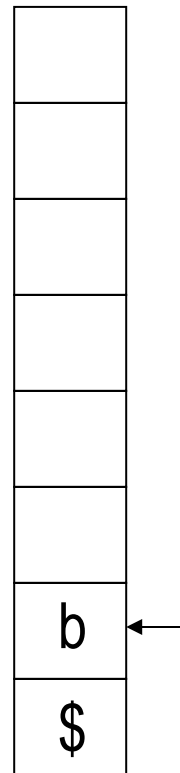
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

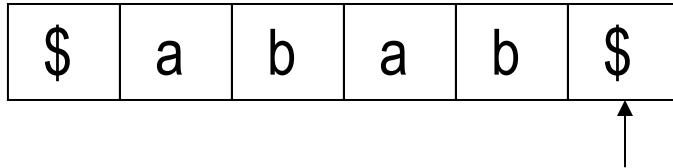
$$abaSb \Rightarrow abab$$

- 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



- Grammar G_3 :

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \varepsilon$$

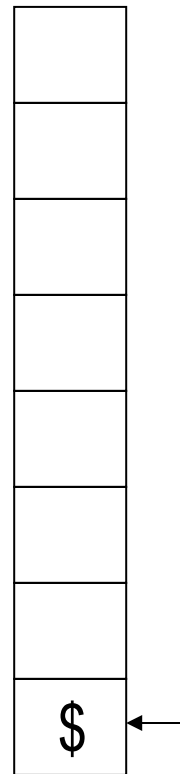
- Derivation : $S \Rightarrow SS$

$$\Rightarrow aSbS \Rightarrow abS \Rightarrow$$

$$abaSb \Rightarrow abab$$

- Details: how PDA recognizes a string of a CFL

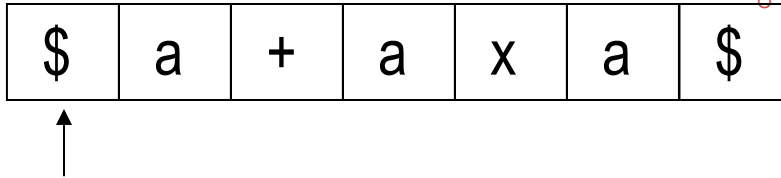
- Step 1: Compare the input with the top of stack
- Step 2: If the top of stack is variable, then simulate the derivation
- Step 3: If the top of stack is terminal, then do the match



Question 2

- 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 : $E \Rightarrow E+E$

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

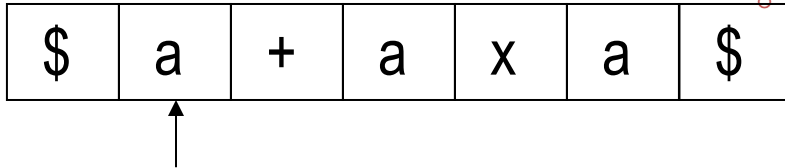
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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



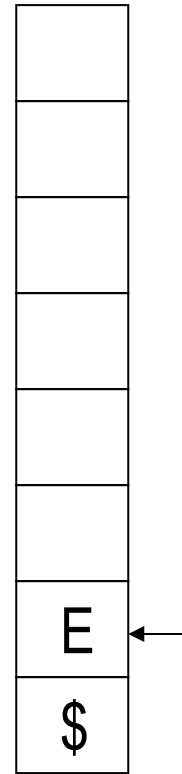
- Grammar G_5 :

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

- Derivation : $E \Rightarrow E+E$

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

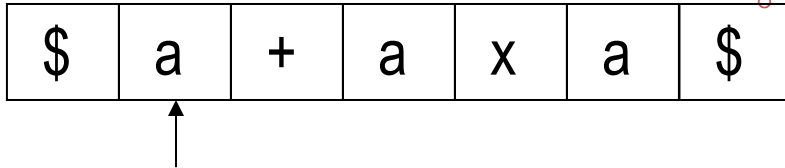
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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



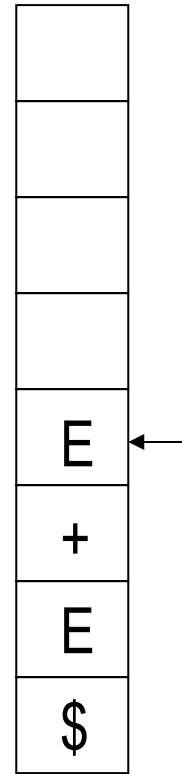
- Grammar G_5 :

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

- Derivation : $E \Rightarrow E+E$

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

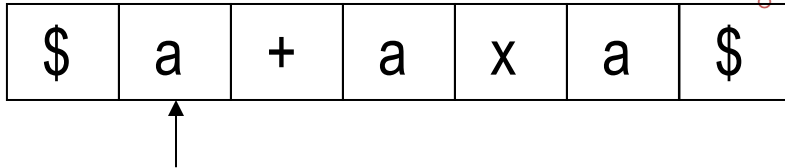
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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



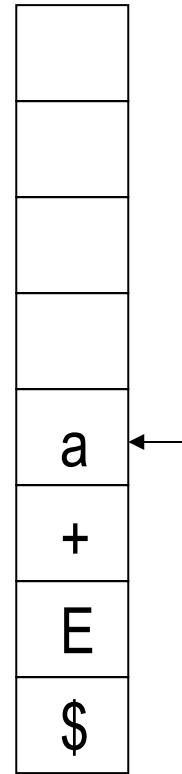
- Grammar G_5 :

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

- Derivation : $E \Rightarrow E+E$

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

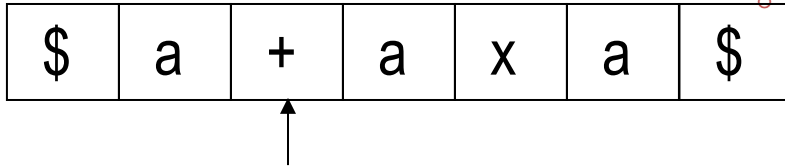
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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



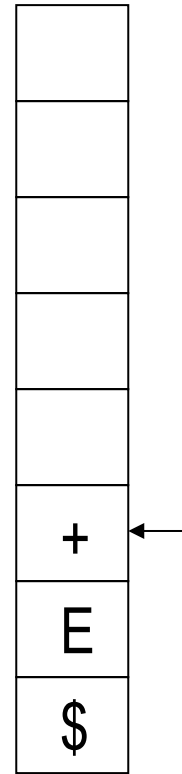
- Grammar G_5 :

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

- Derivation : $E \Rightarrow E+E$

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

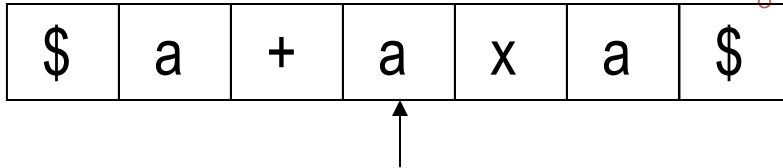
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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



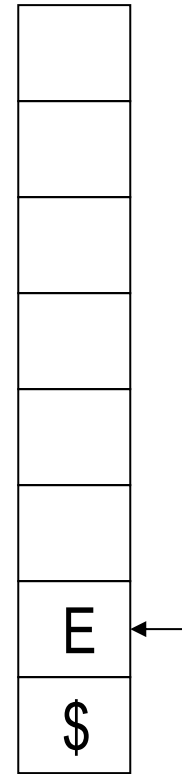
- Grammar G_5 :

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

- Derivation : $E \Rightarrow E+E$

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

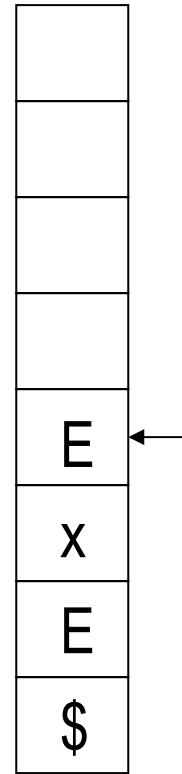
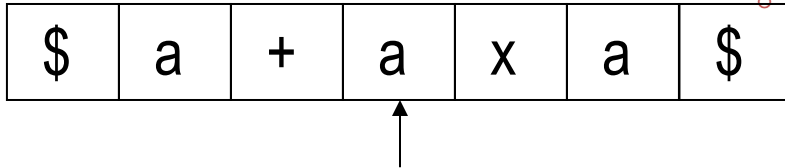
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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 : $E \Rightarrow E+E$

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

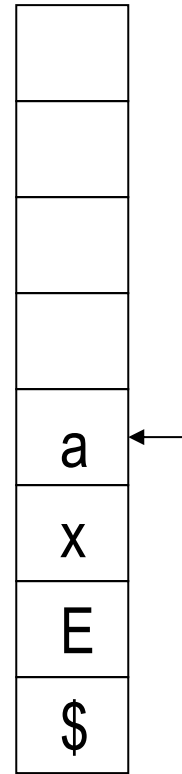
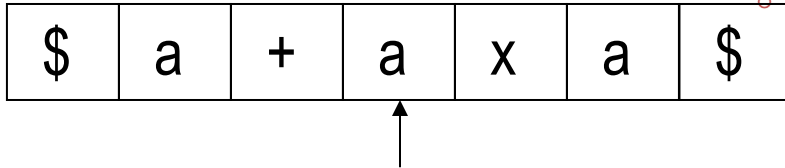
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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 : $E \Rightarrow E+E$

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

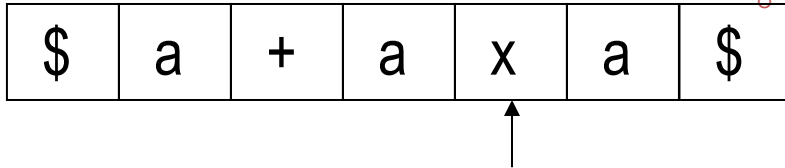
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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



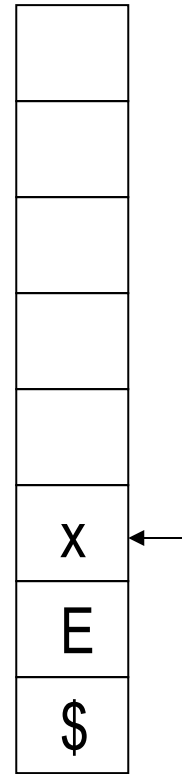
- Grammar G_5 :

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

- Derivation : $E \Rightarrow E+E$

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

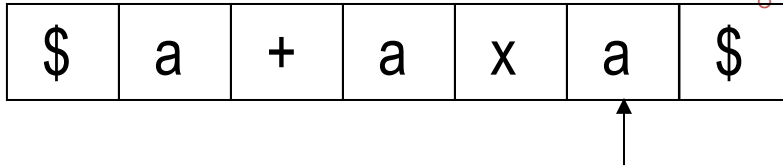
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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



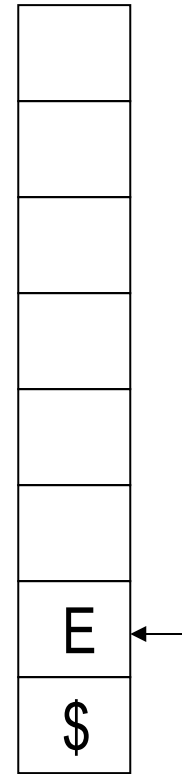
- Grammar G_5 :

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

- Derivation : $E \Rightarrow E+E$

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

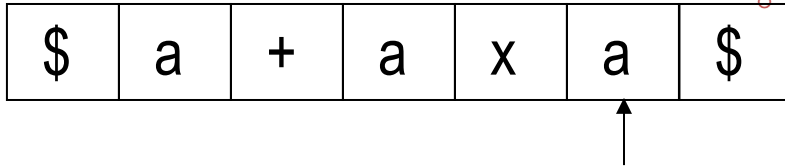
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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



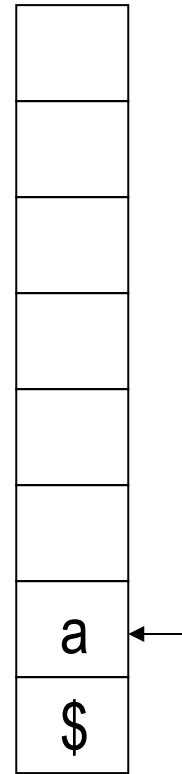
- Grammar G_5 :

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

- Derivation : $E \Rightarrow E+E$

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

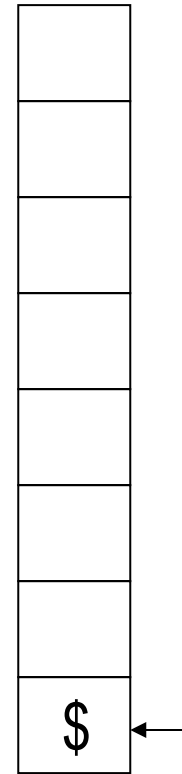
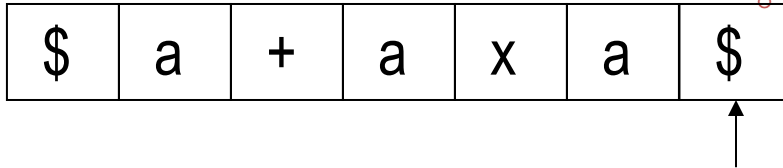
$$\Rightarrow a+a \times E \Rightarrow a+a \times a$$



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 : $E \Rightarrow E+E$

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

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



A language is CFL \Rightarrow 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 \Rightarrow 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 **$u_1u_2...u_k$**

introduce new states q_1, q_2, \dots, q_{k-1} ,

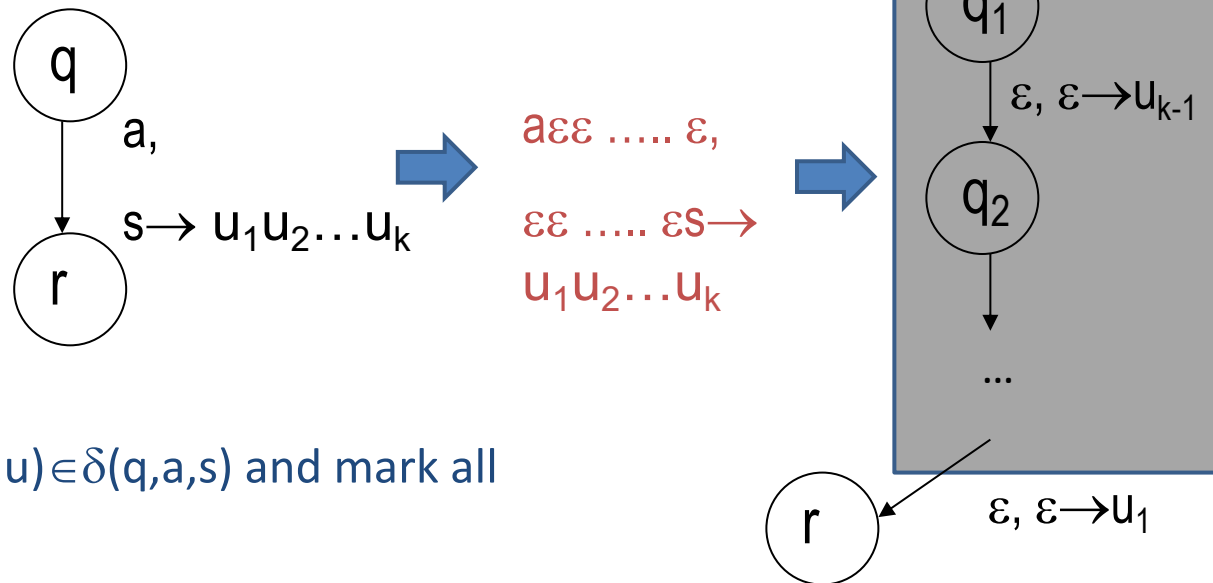
let $\delta(q, a, s)$ contains (q_1, u_k) ,

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

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

...

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



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

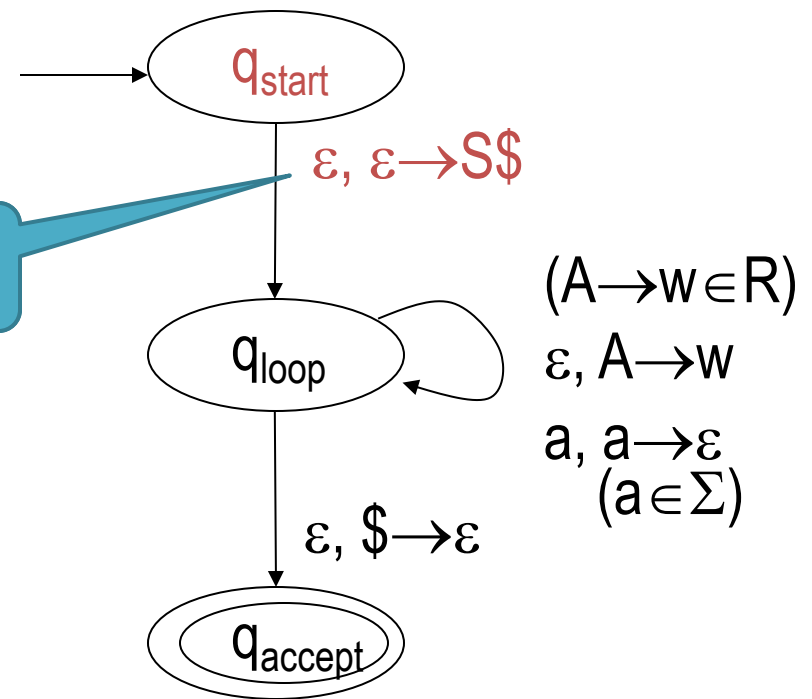


A language is CFL \Rightarrow some PDA recognizes it

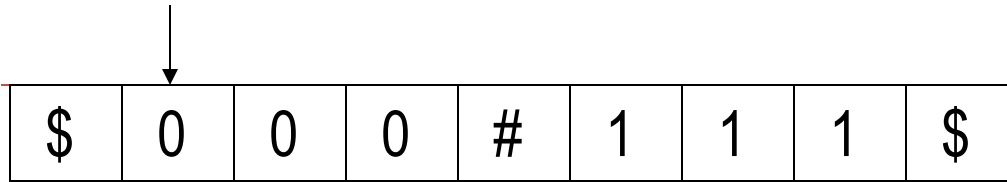
Define δ :

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

Put \$ and start variable into stack



Example: Put \$ and start variable into stack



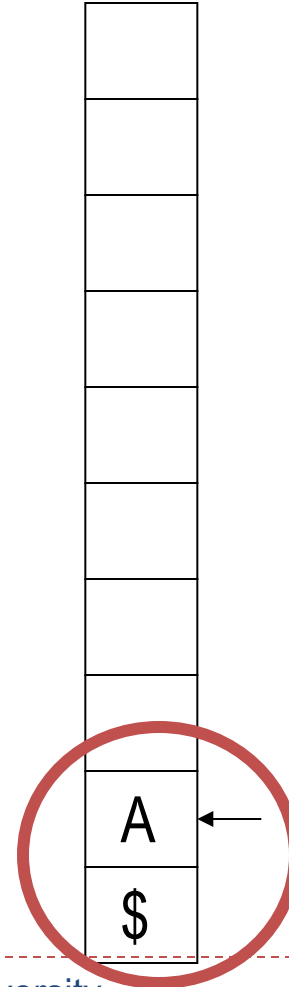
- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

- Derivation : A



top of stack is
variable A

replace A
by w

pick up one
rule $A \rightarrow w$

language is CFL \Rightarrow some PDA recognizes it

Define δ

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

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

Match succeeds
and pop

top of stack is terminal a

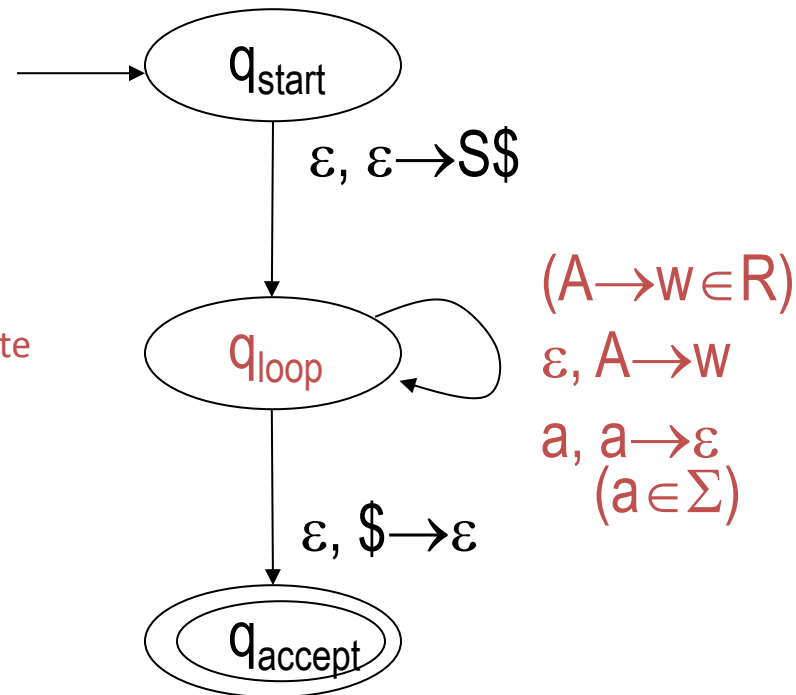
// repeat the following steps

If the top of stack is variable A,
then pick up one rule $A \rightarrow w$, and replace A by w, 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 \Rightarrow some PDA recognizes it

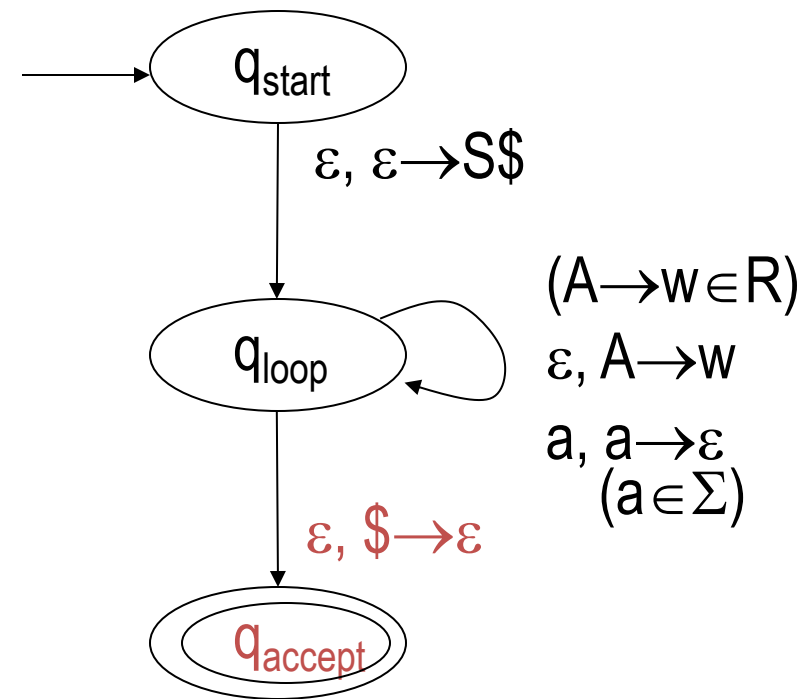
top of stack
is \$

Defin δ :

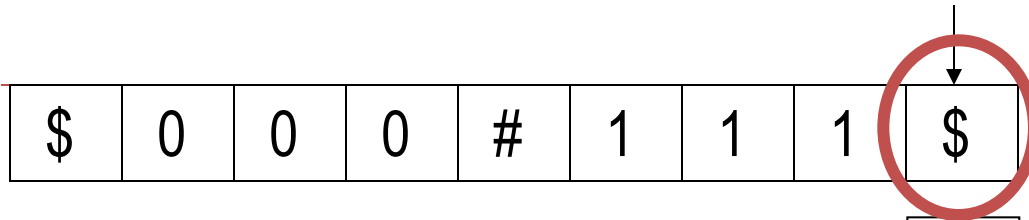
the input
is over

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

// If the top of stack is \$,
if the input is over, accept.



Example: when stack and input are empty



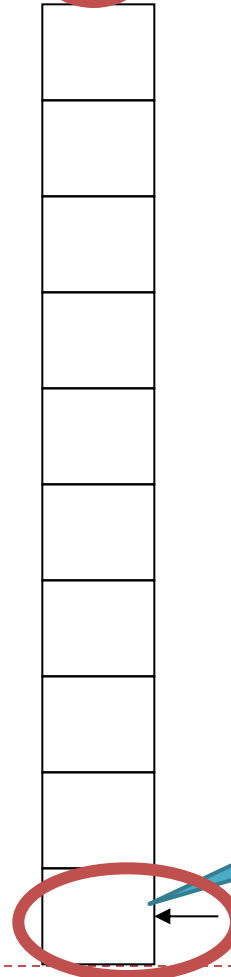
- Grammar G_1 :

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

- 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 \Rightarrow some PDA recognizes it

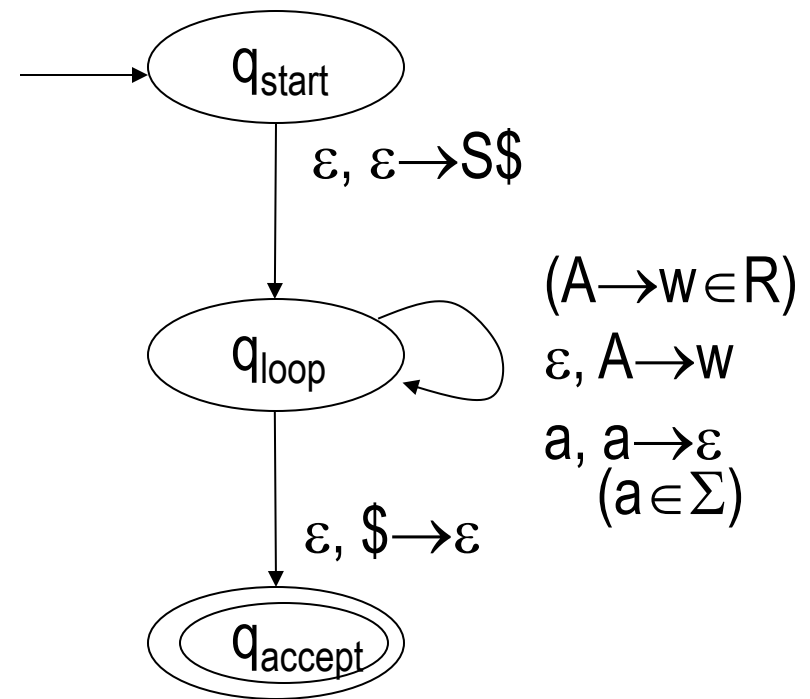
Define δ :

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

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

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

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



Conclusion for A language is CFL \Rightarrow some PDA recognizes it

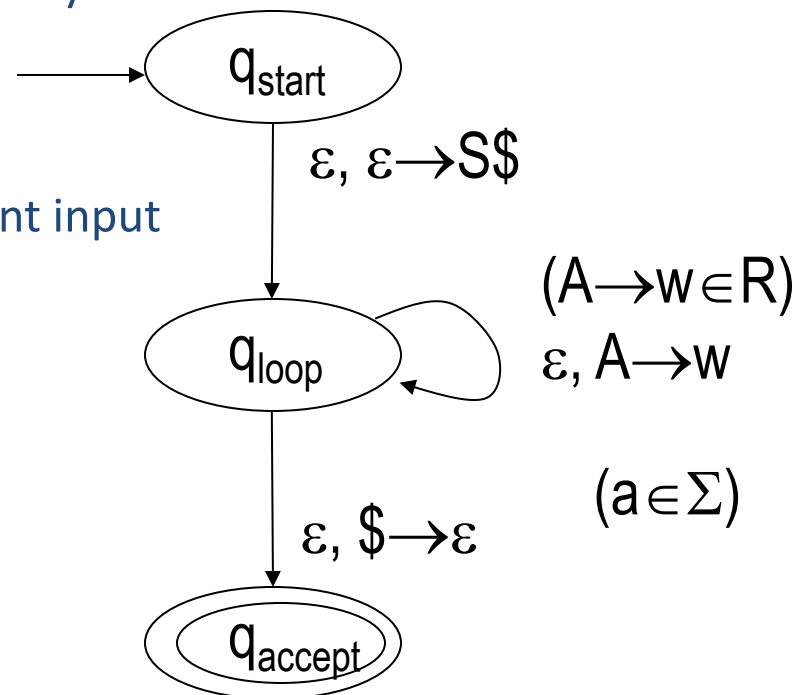
Put \$ and start variable into stack

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 \Rightarrow PDA

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

construct an equivalent PDA P_1 .



Example of CFL \Rightarrow PDA

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

construct an equivalent PDA P_1 .

q_{start}

q_{loop}

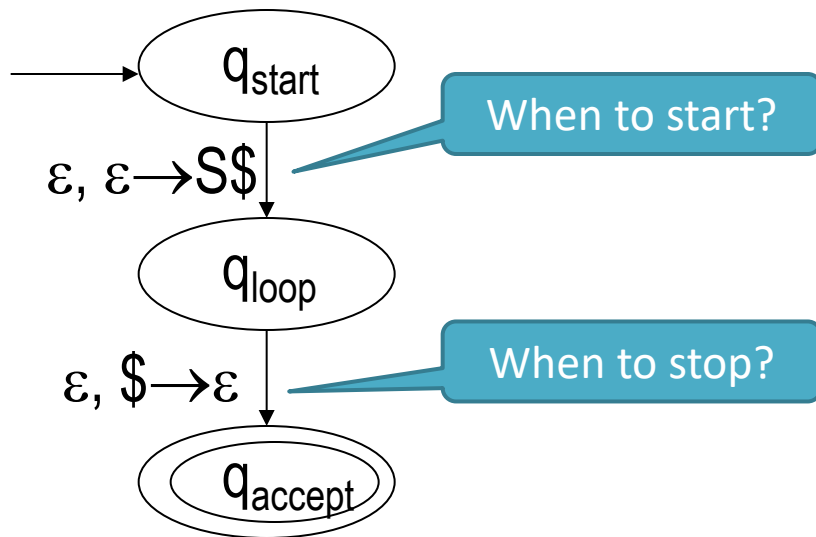
q_{accept}



Example of CFL \Rightarrow PDA

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

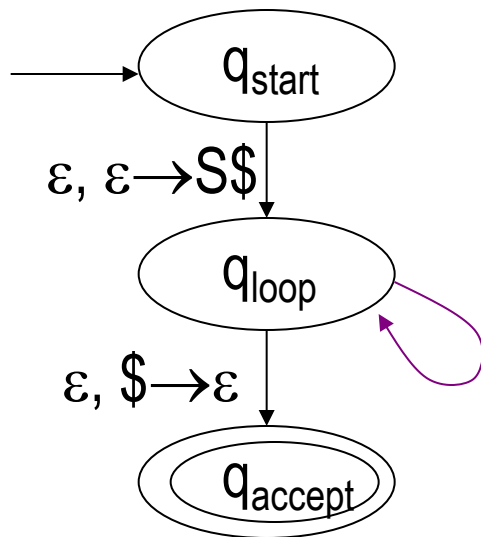
construct an equivalent PDA P_1 .



Example of CFL \Rightarrow PDA

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

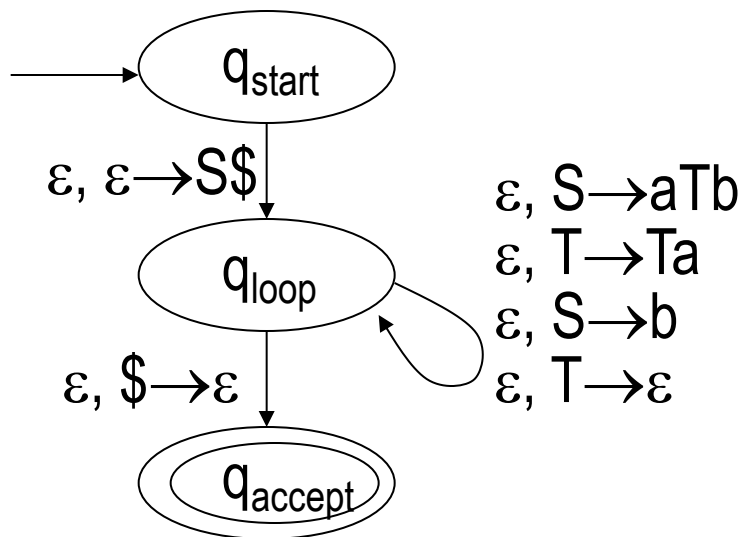
construct an equivalent PDA P_1 .



Example of CFL \Rightarrow PDA

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

construct an equivalent PDA P_1 .

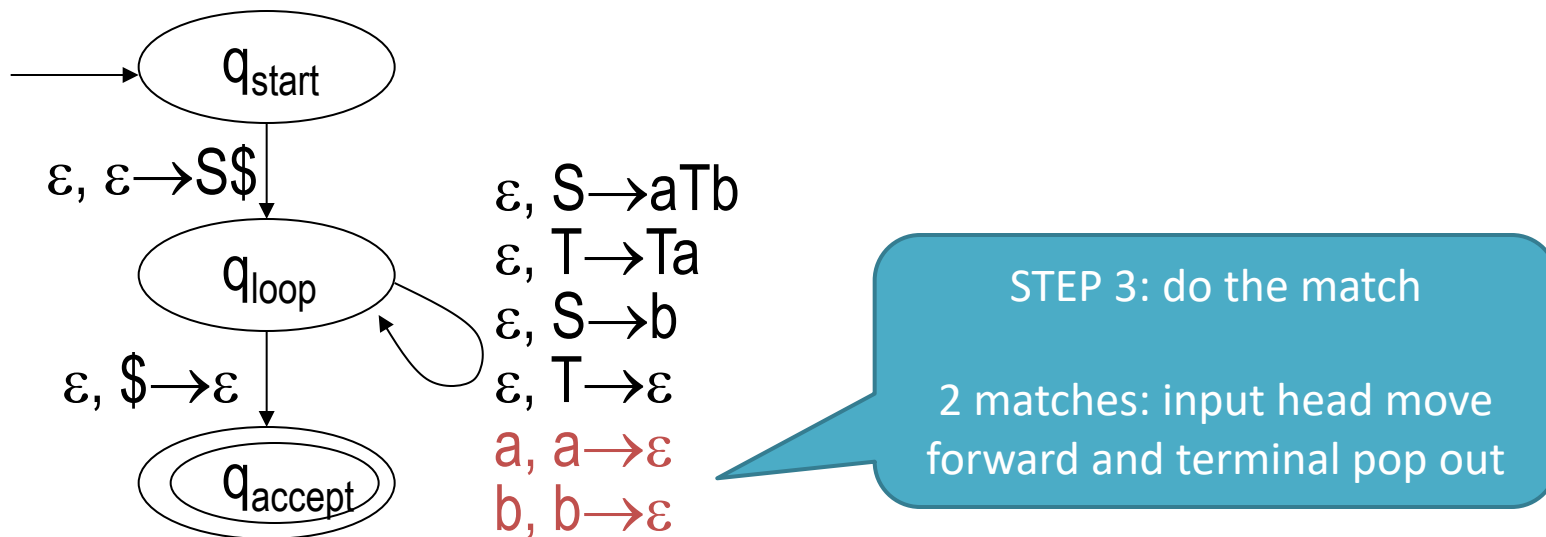


STEP 2: simulate the derivation
4 rules \rightarrow 4 transition functions

Example of CFL \Rightarrow PDA

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

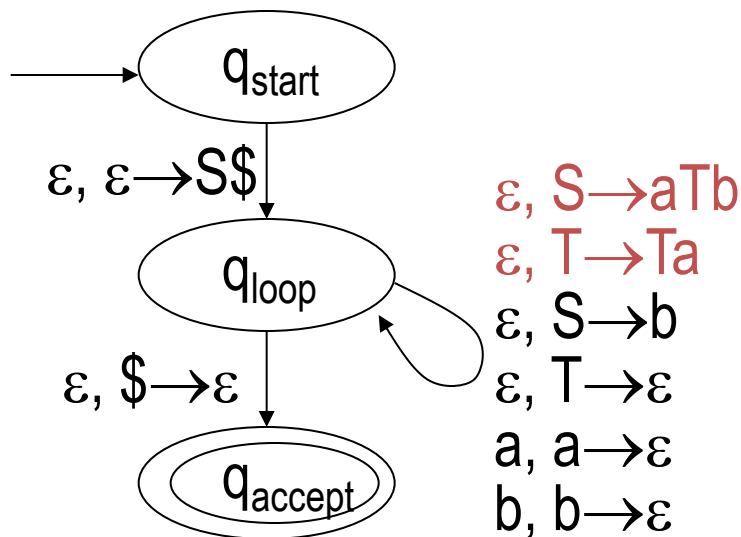
construct an equivalent PDA P_1 .



Example of CFL \Rightarrow PDA

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

construct an equivalent PDA P_1 .

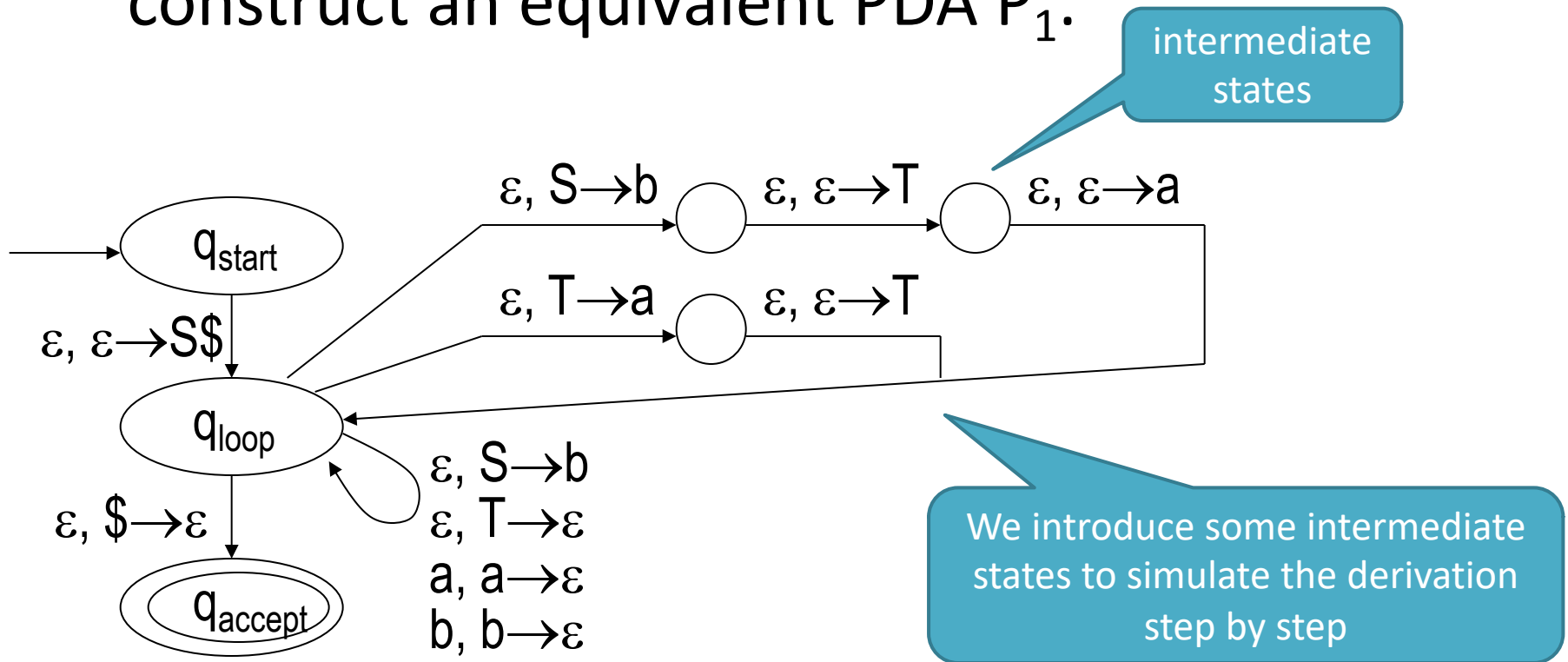


For these which right side has one more symbols, we introduce some intermediate states

Example of CFL \Rightarrow PDA

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

construct an equivalent PDA P_1 .



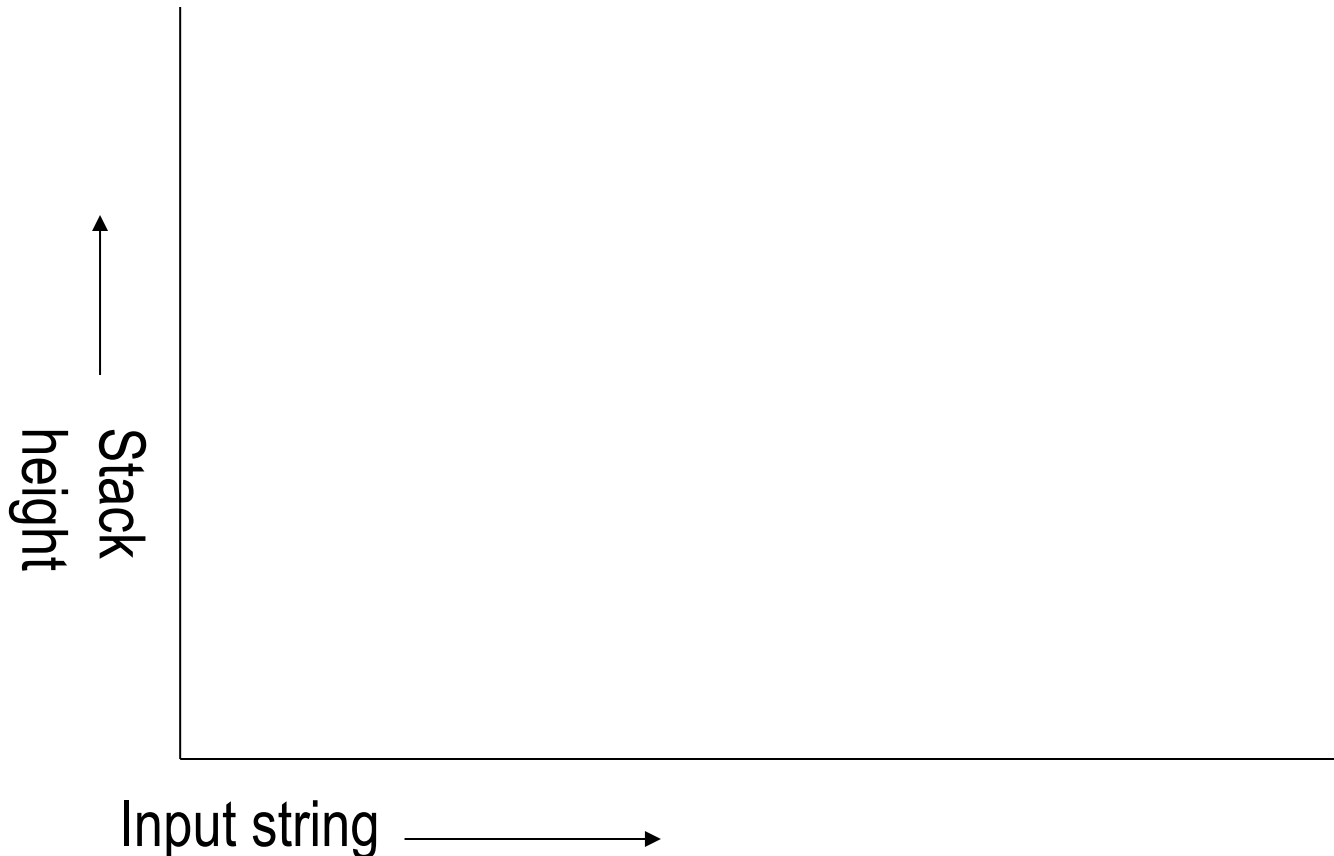
Outline

- Equivalence of PDA and CFG:
- A language is CFL \implies some PDA recognizes it
- **A language is CFL \iff some PDA recognizes it**



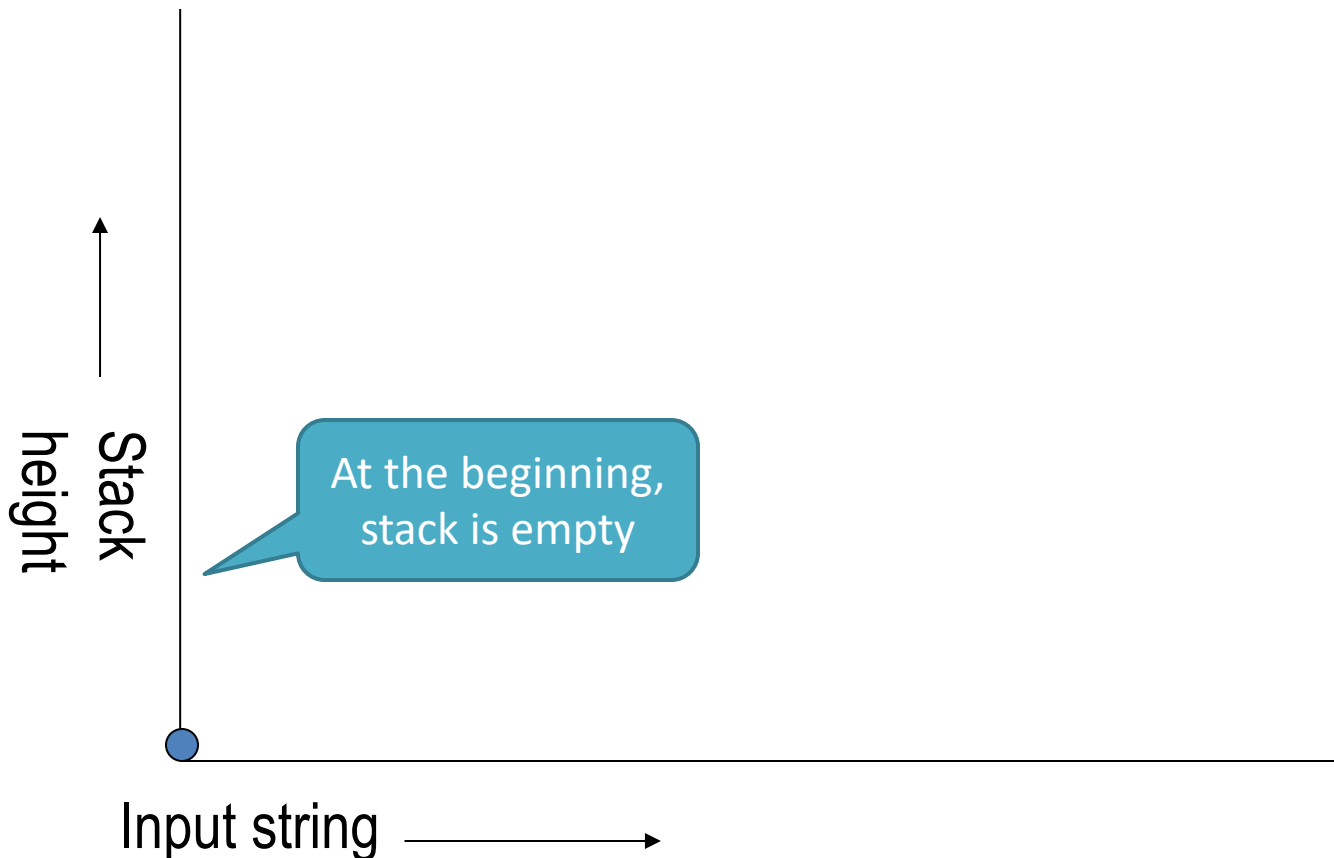
A language is CFL \Leftrightarrow some PDA recognizes it

- The process of PDA recognizes some inputs



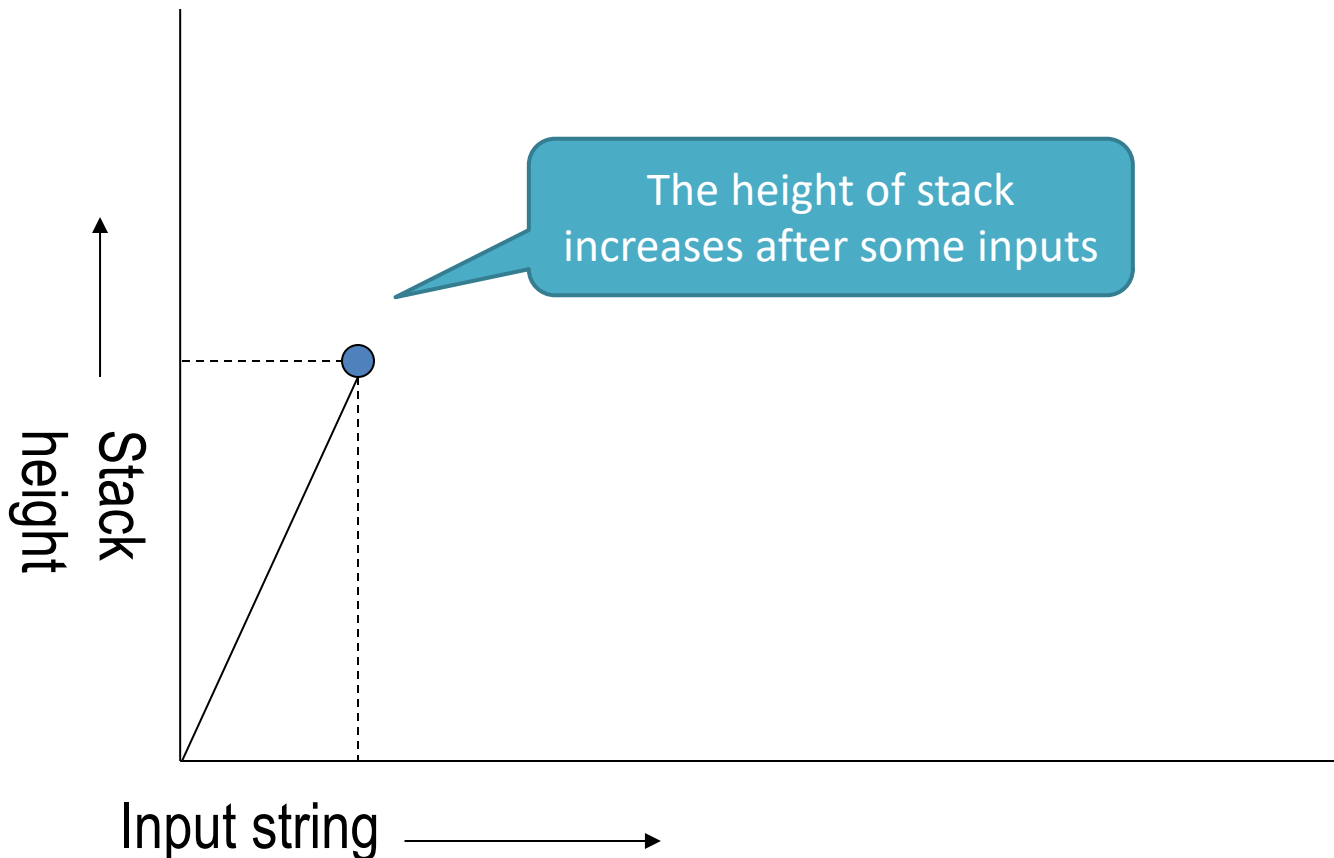
A language is CFL \Leftrightarrow some PDA recognizes it

- The process of PDA recognizes some inputs



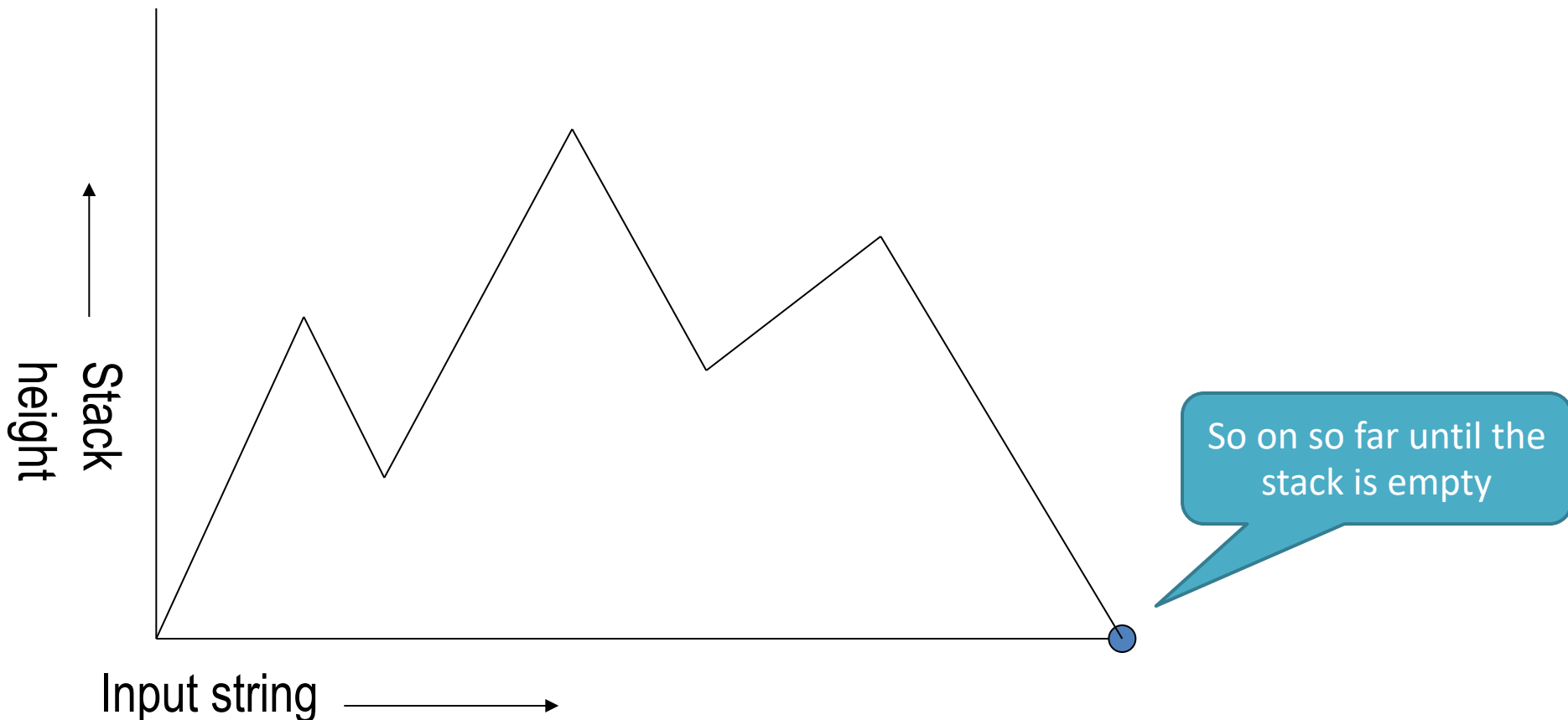
A language is CFL \Leftrightarrow some PDA recognizes it

- The process of PDA recognizes some inputs



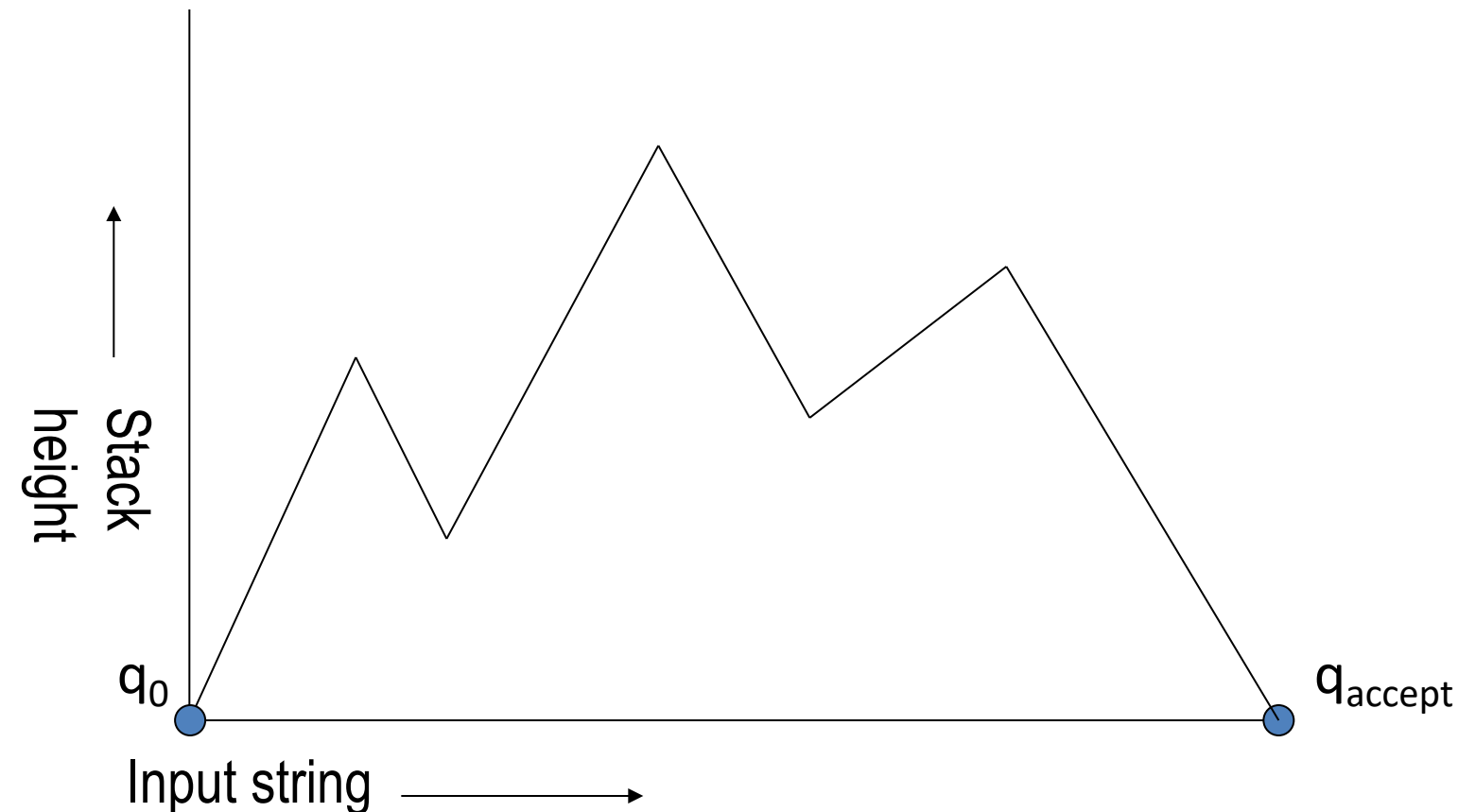
A language is CFL \Leftarrow some PDA recognizes it

- The process of PDA recognizes some inputs



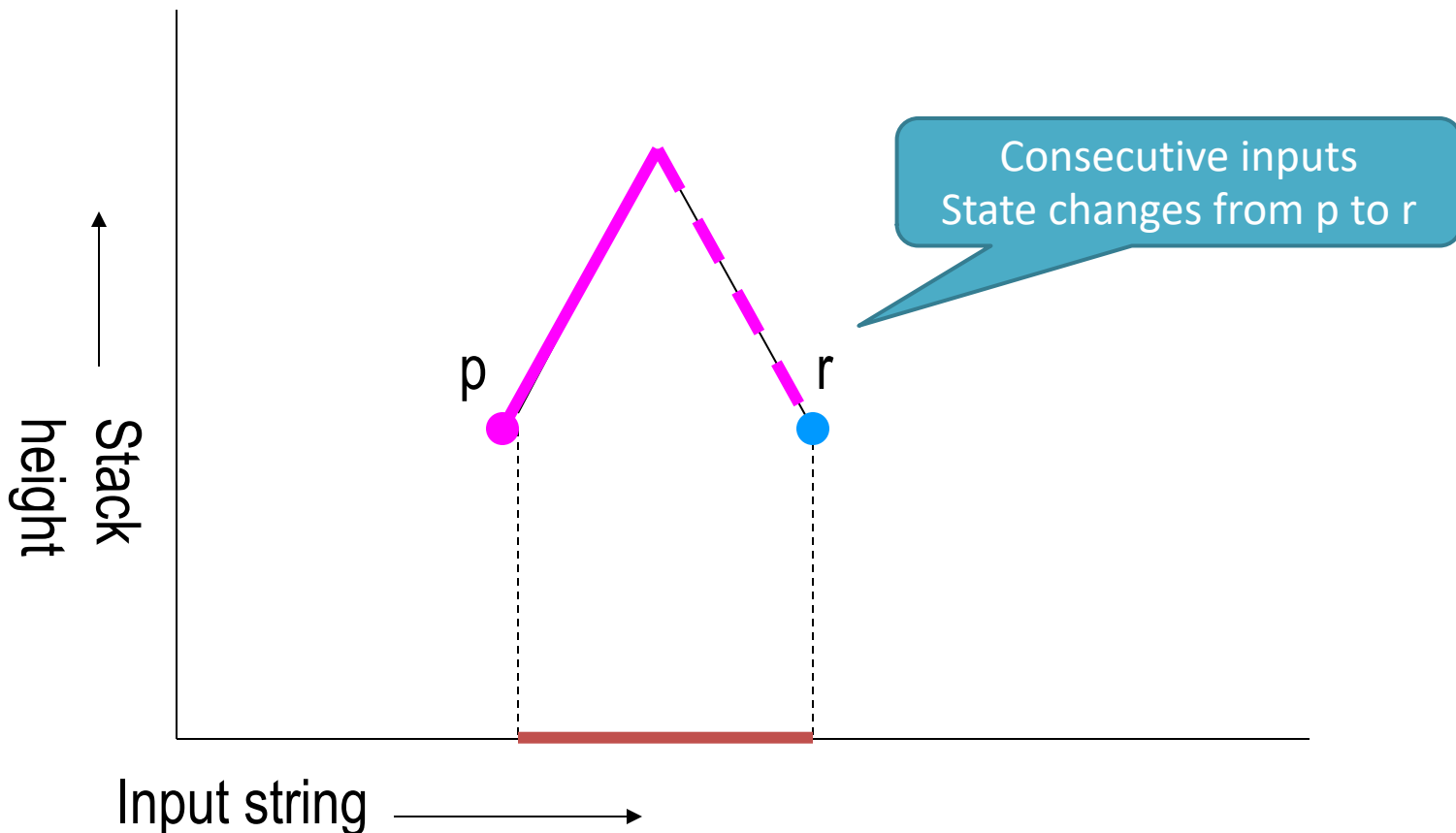
A language is CFL \Leftrightarrow some PDA recognizes it

- Create CFGs for such movements



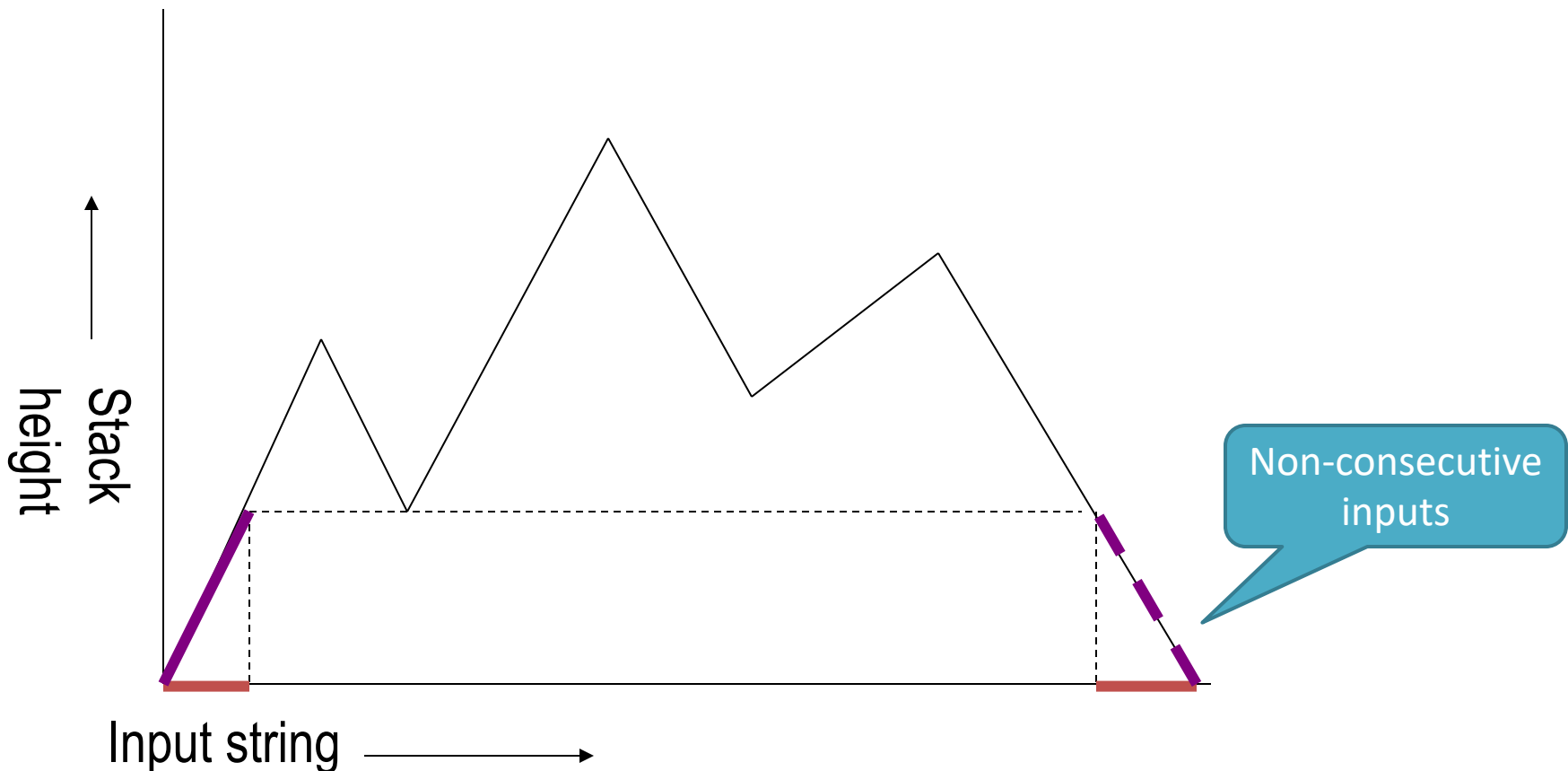
A language is CFL \Leftarrow some PDA recognizes it

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

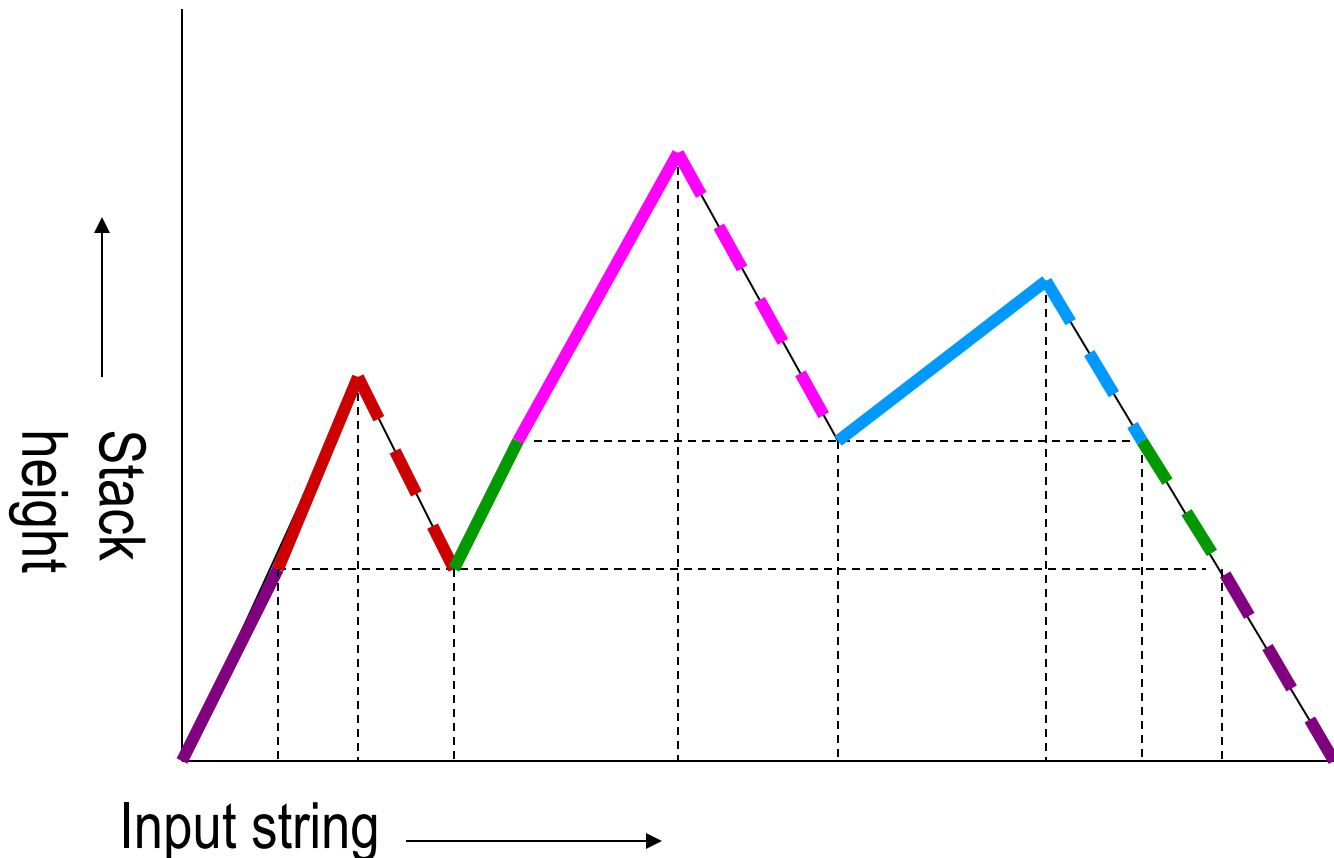


A language is CFL \Leftarrow some PDA recognizes it

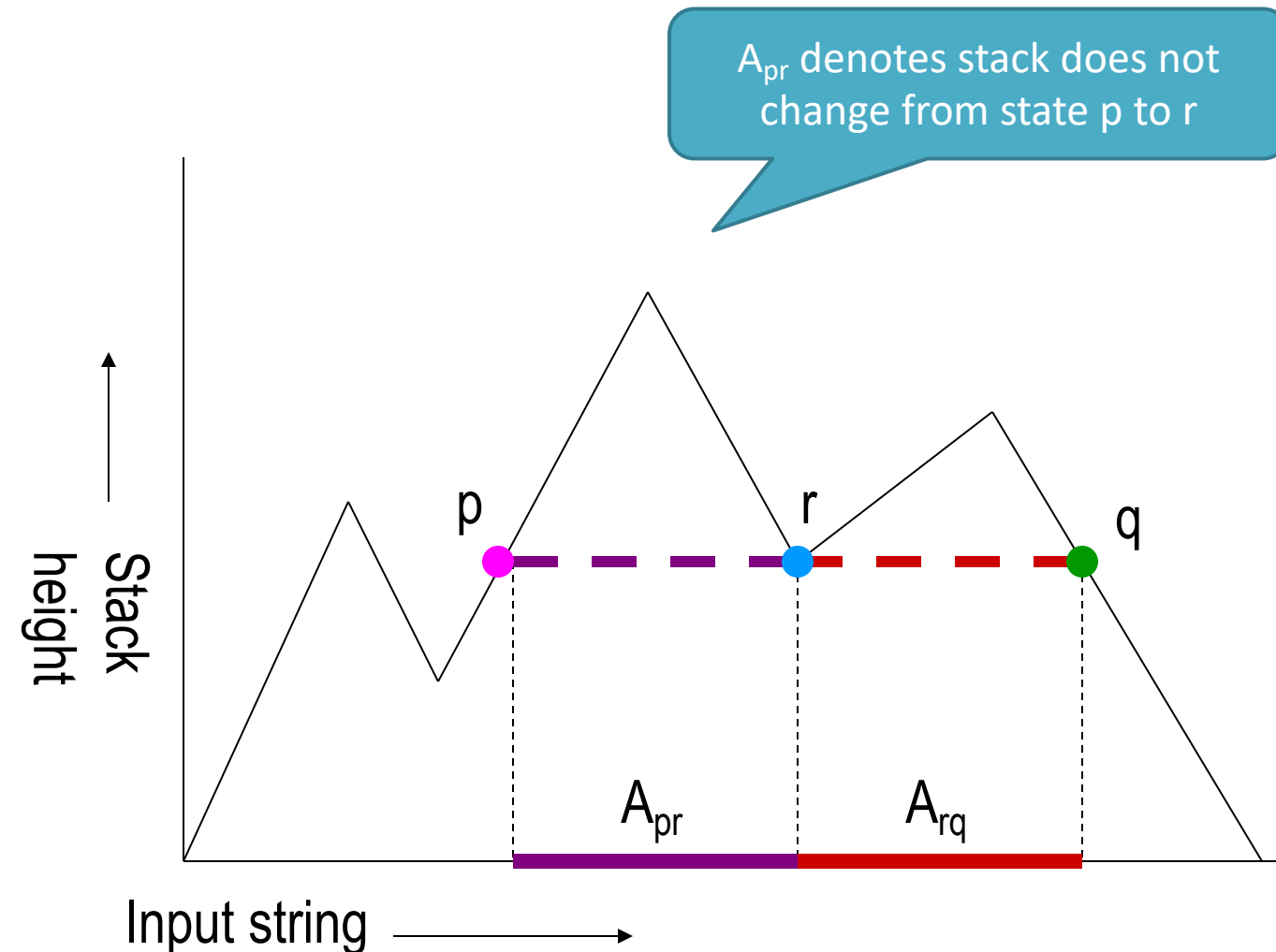
- Case 2: the stack does not change (increase and then decrease) after some **non-consecutive** inputs



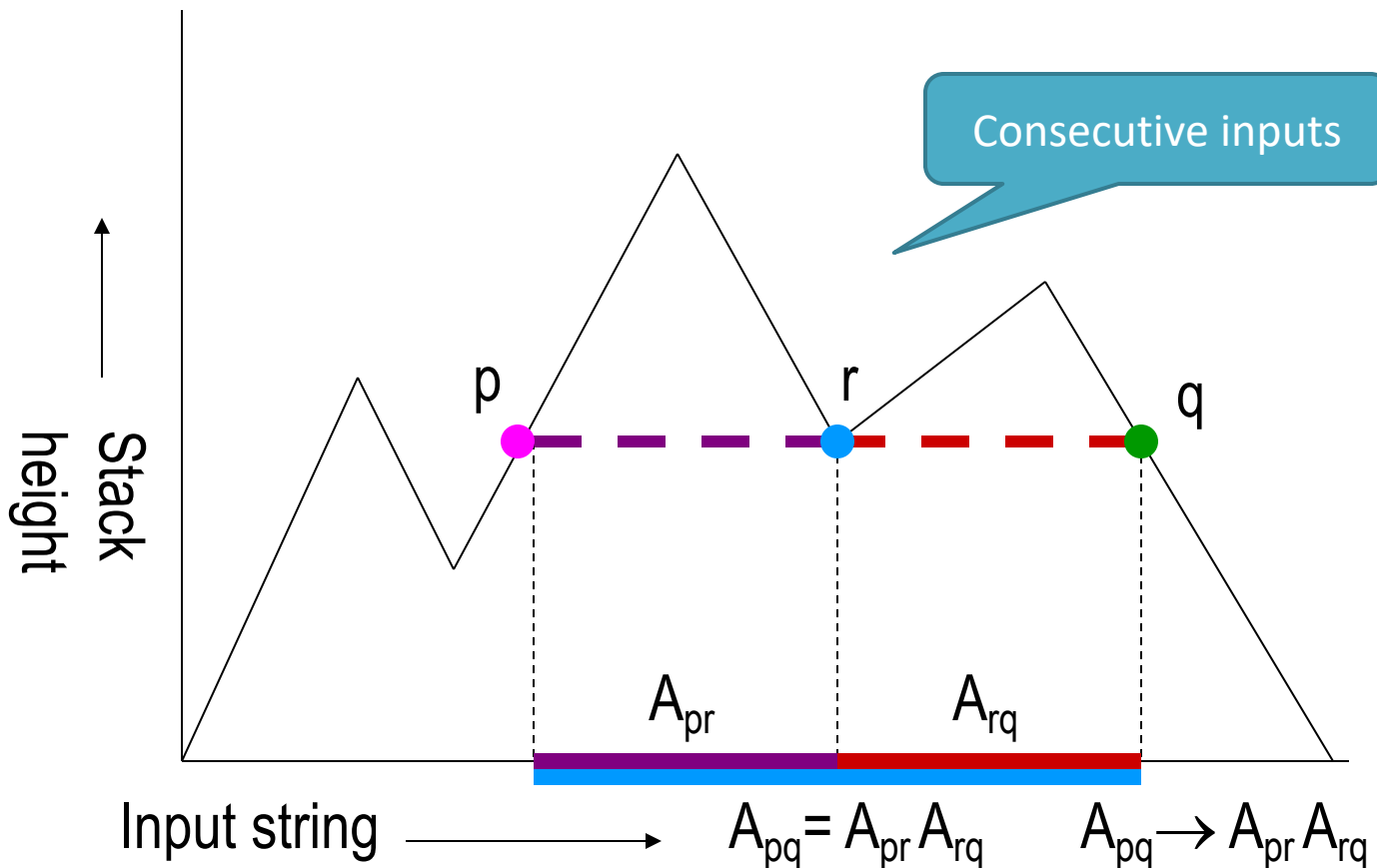
A language is CFL \Leftrightarrow some PDA recognizes it



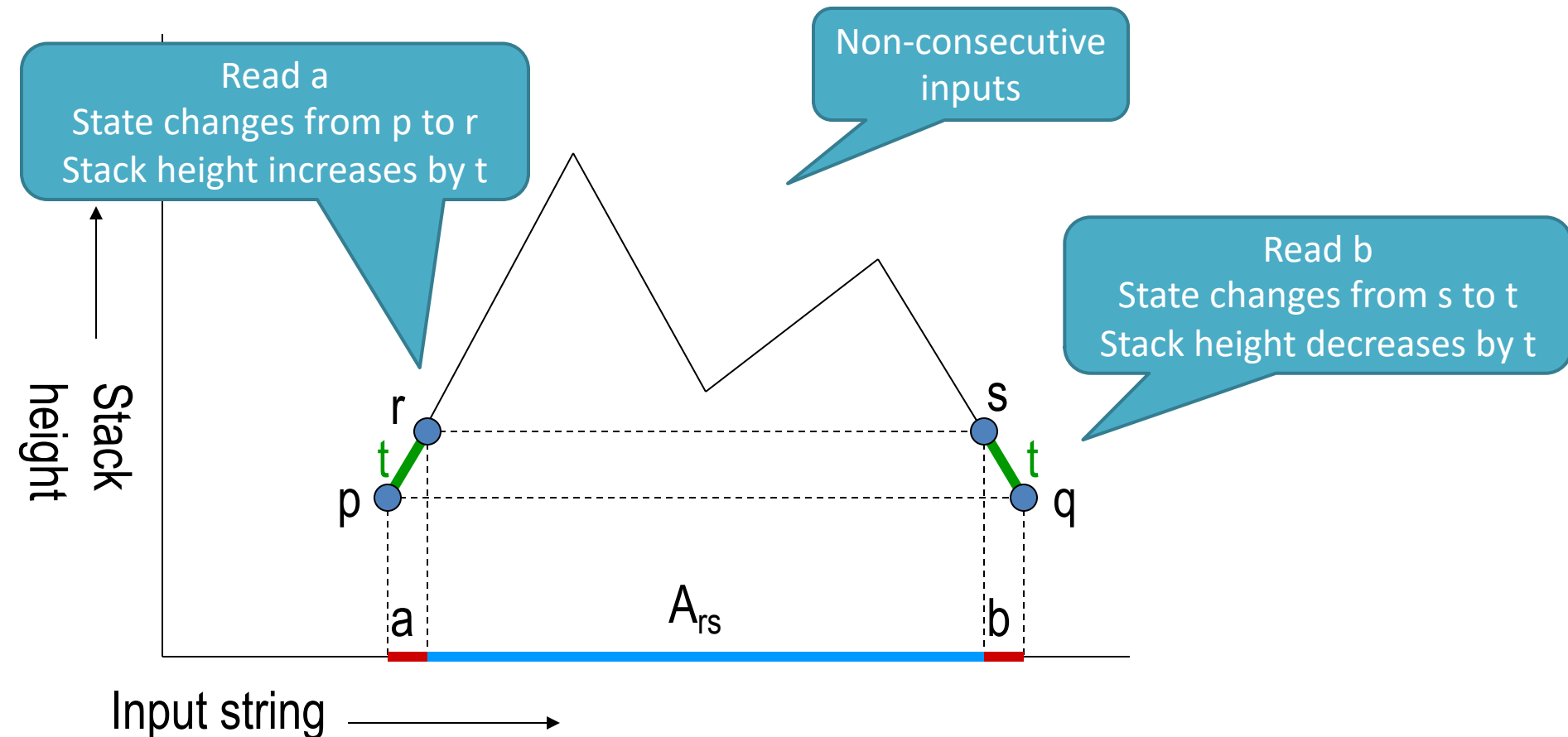
A language is CFL \Leftrightarrow some PDA recognizes it



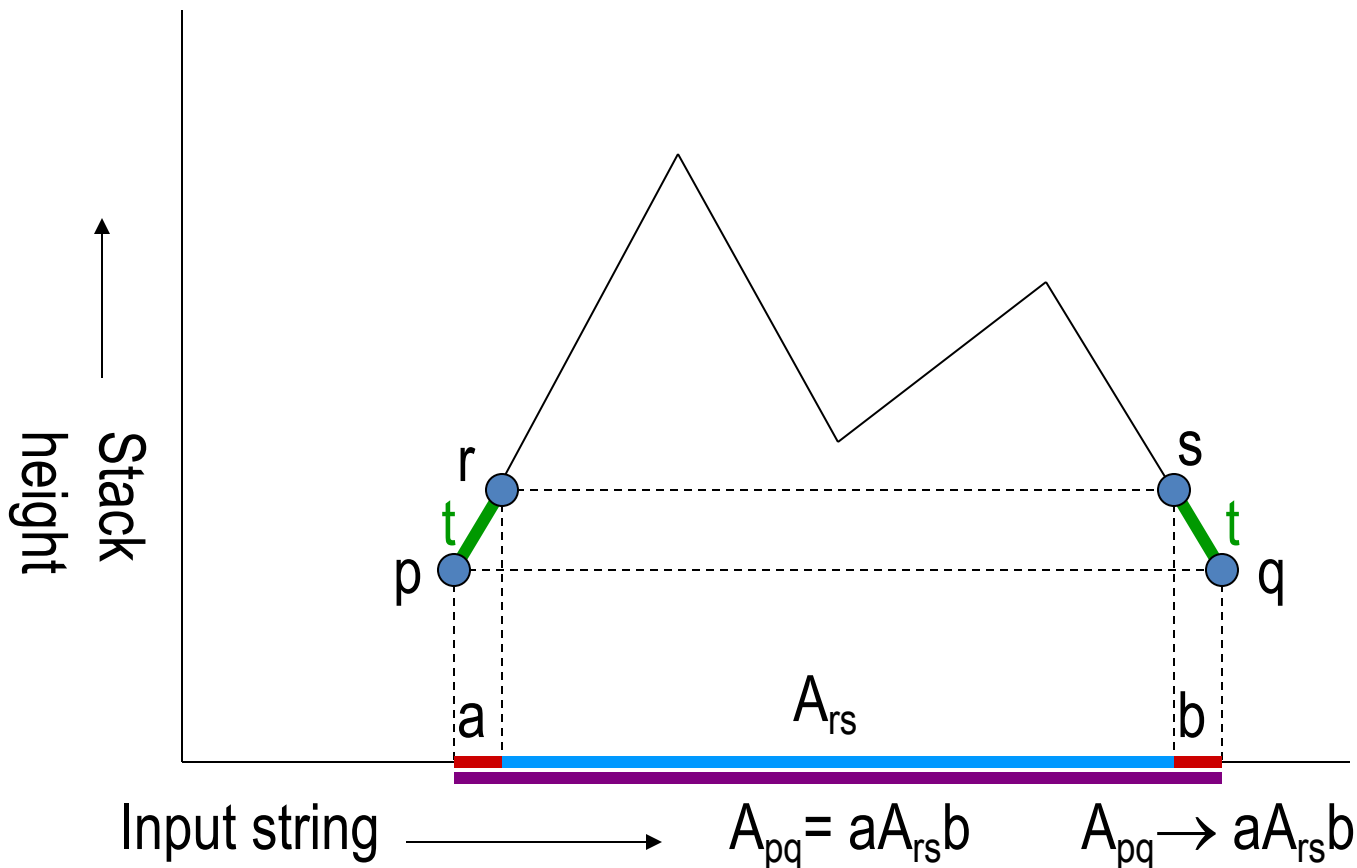
A language is CFL \Leftarrow some PDA recognizes it



A language is CFL \Leftrightarrow some PDA recognizes it



A language is CFL \Leftarrow some PDA recognizes it



CFL \Leftarrow PDA

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

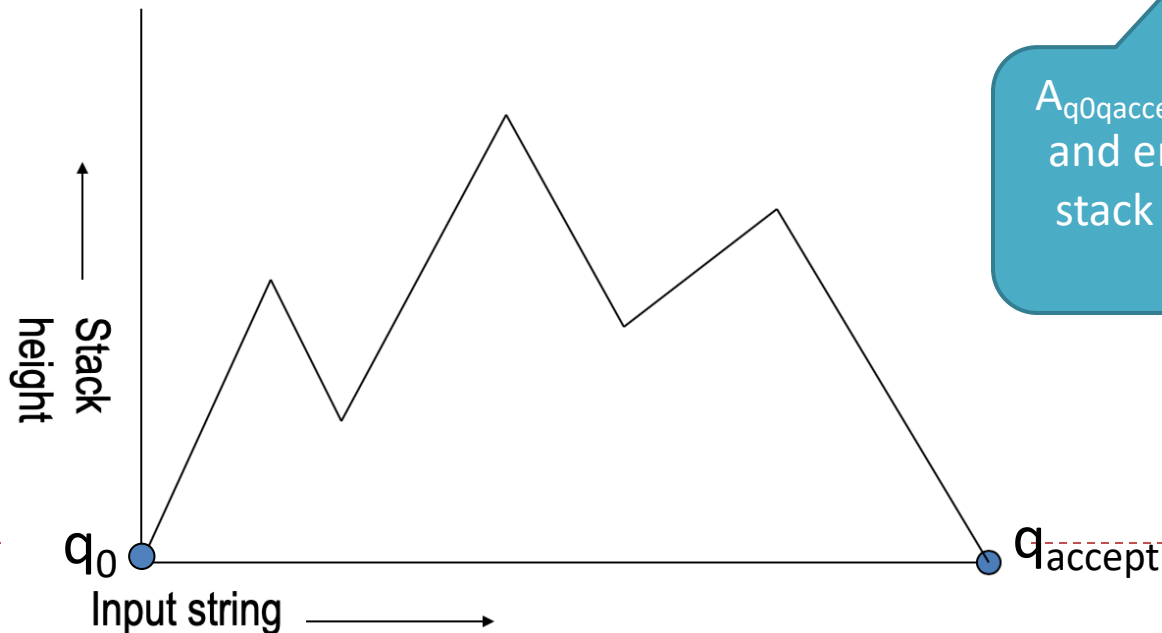
- For a PDA

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

then create equivalent CFG

$G = (\{A_{pq} \mid p, q \in Q\}, \Sigma, R, A_{q_0q_{\text{accept}}})$,

$A_{q_0q_{\text{accept}}}$ requires starting from q_0 and ending in q_{accept} and making stack is empty. It defines where the PDA starts.

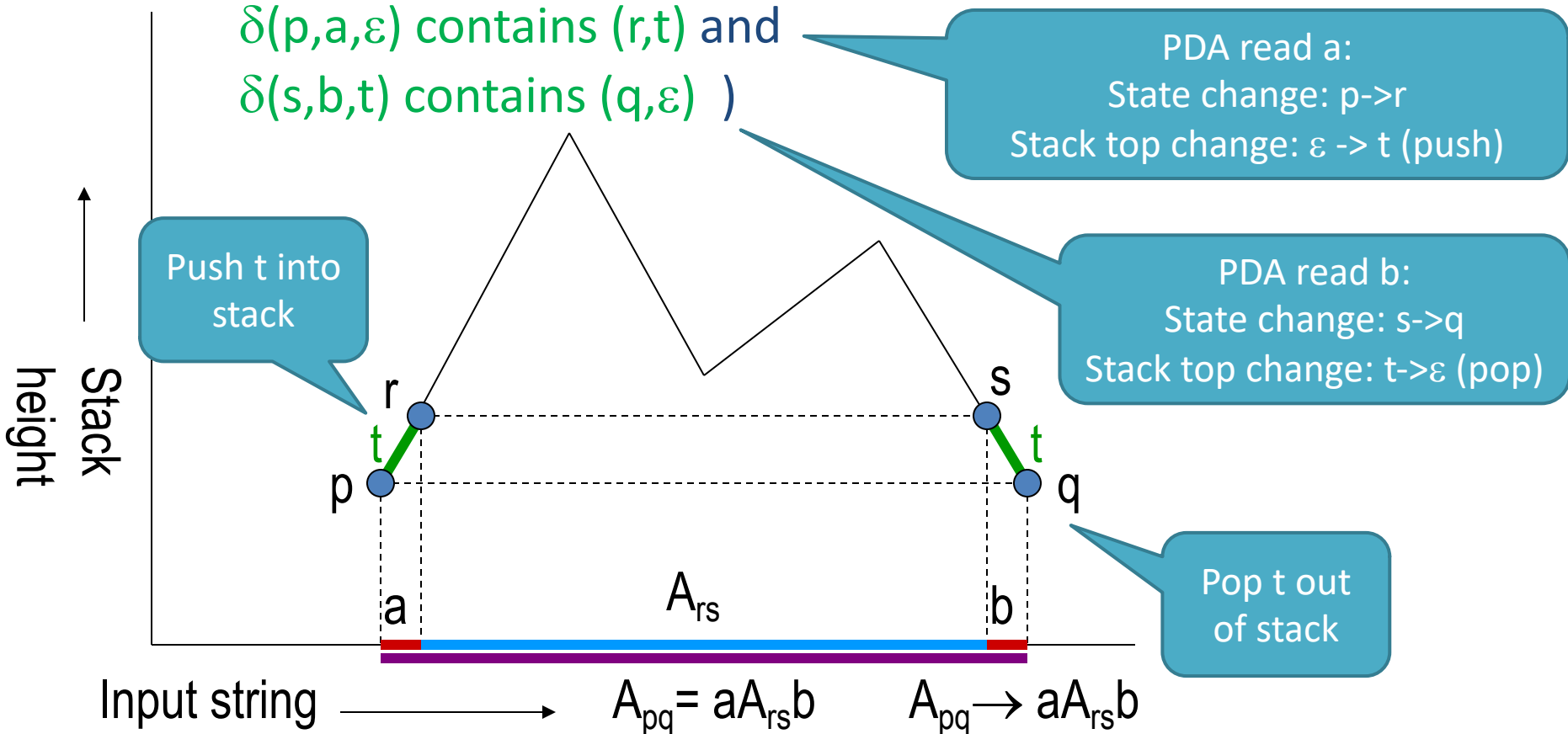


Non-Consecutive input rule

R has the following rules:

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

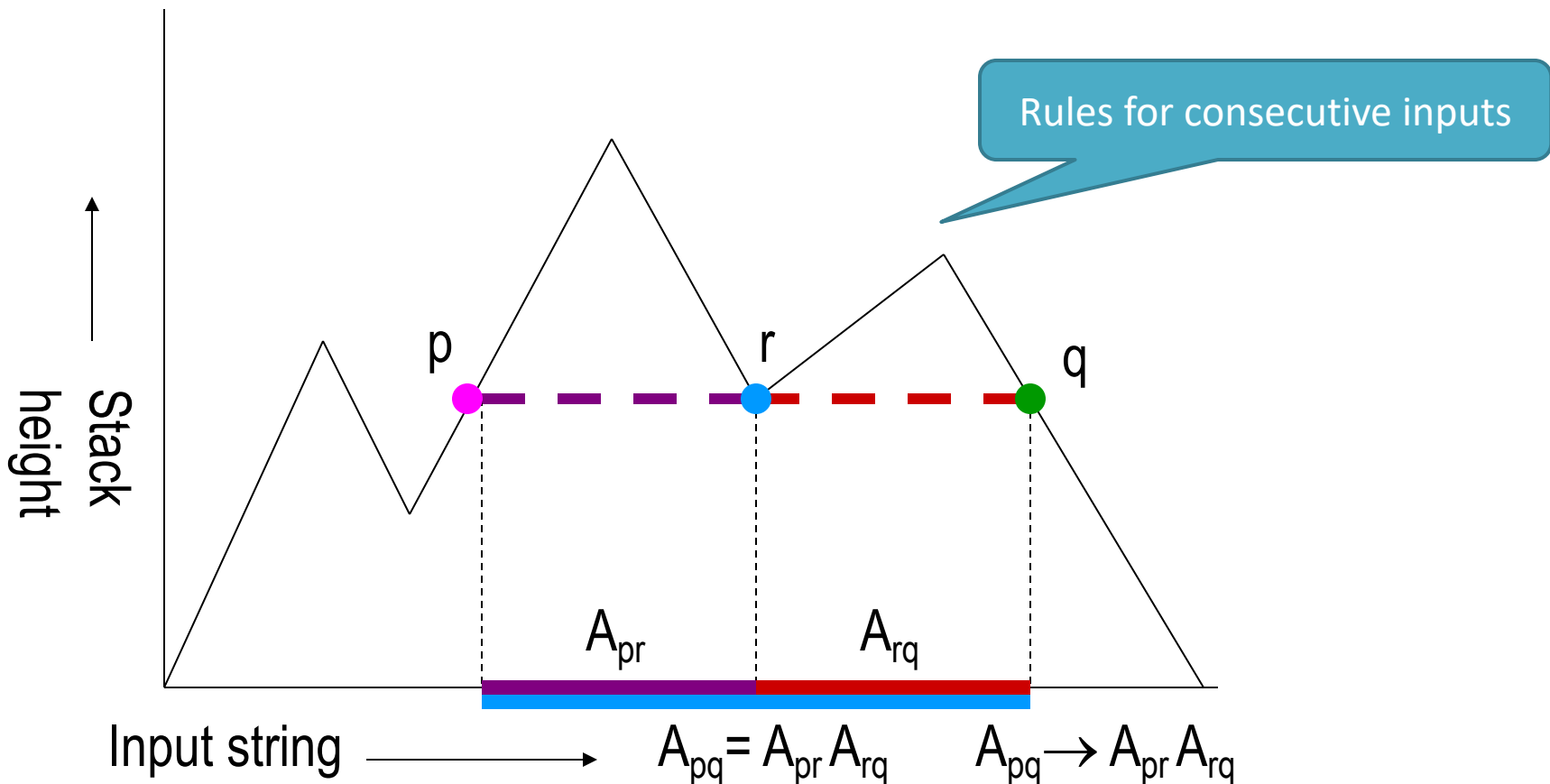
$\delta(p, a, \epsilon)$ contains (r, t) and
 $\delta(s, b, t)$ contains (q, ϵ))



Consecutive input rule

R has the following rules:

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

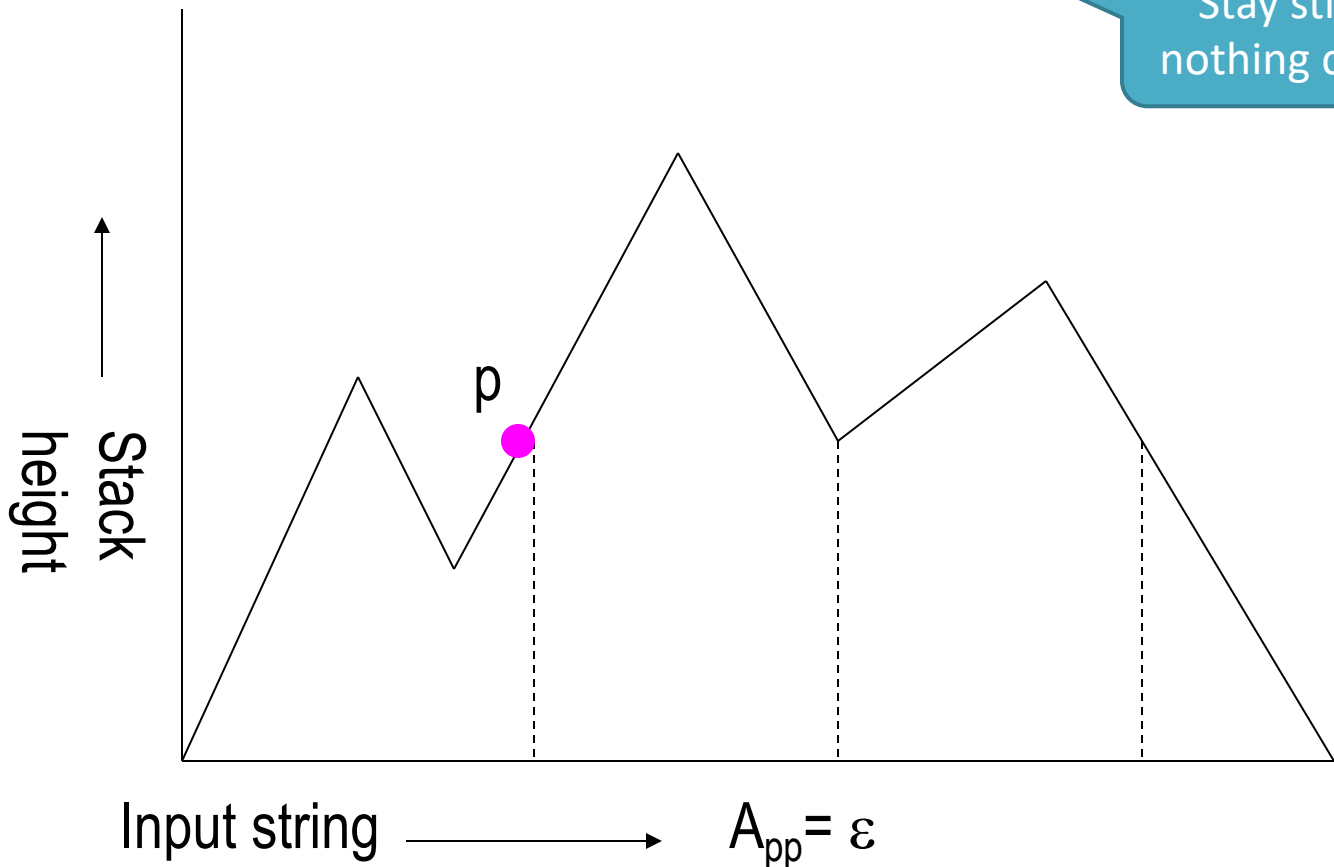


ε rule

R has the following rules:

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

Stay still and
nothing changes



A language is CFL \Leftarrow some PDA recognizes it

- For a PDA

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

then create equivalent CFG

$$G=(\{A_{pq} \mid p,q \in Q\}, \Sigma, \quad R, A_{q_0q_{\text{accept}}}),$$

Key idea:

We can create CFGs to simulate the movement of PDA



Conclusion

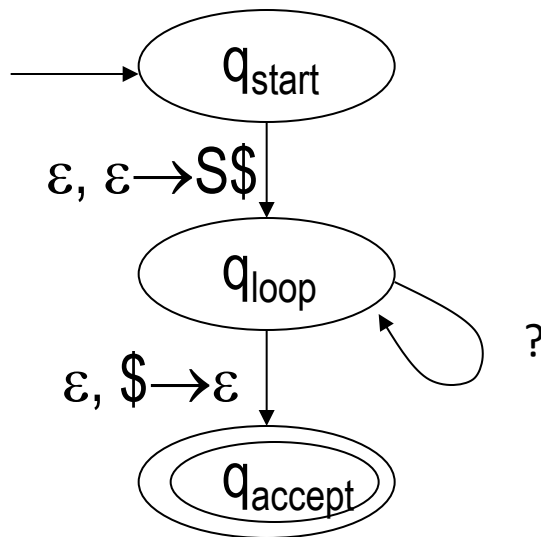
- Equivalence of PDA and CFG:
- A language is CFL \Rightarrow some PDA recognizes it
- A language is CFL \Leftarrow some PDA recognizes it



Question

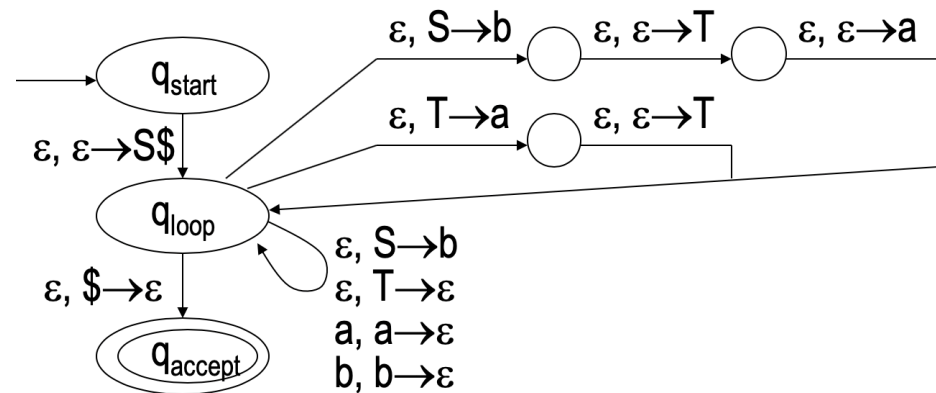
- CFG G: $E \rightarrow E + E \mid E \times E \mid (E) \mid a$

construct an equivalent PDA P_2 .



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

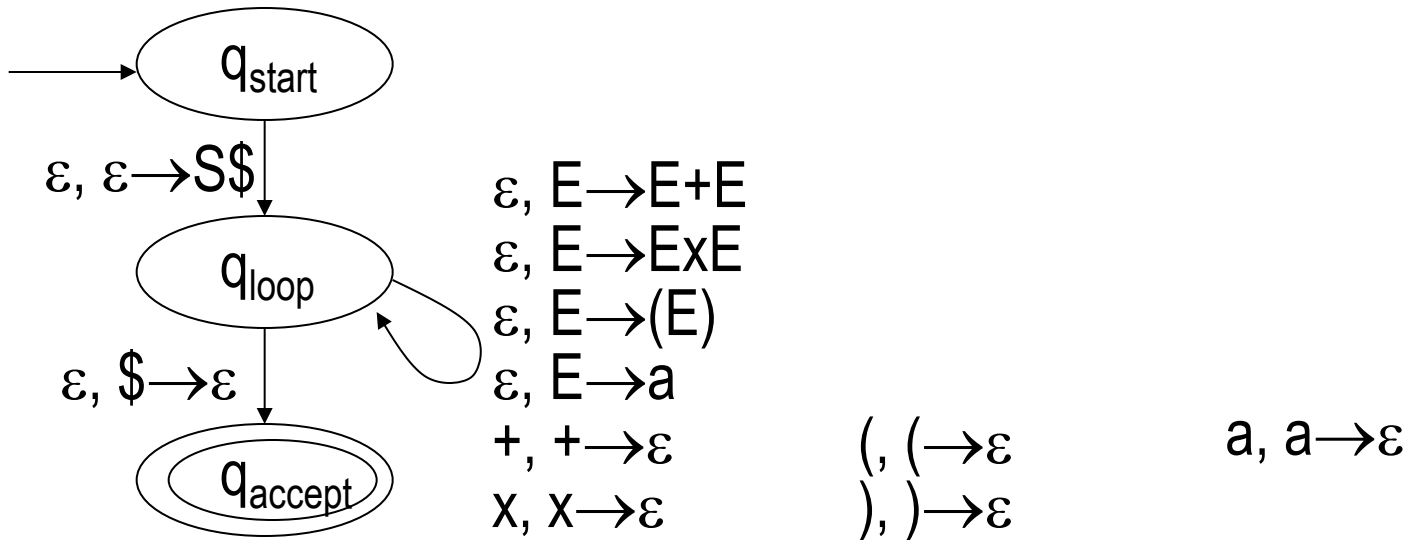
construct an equivalent PDA P_1 .



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 .

