Theory of Computation
CSC 339 – Spring 2021

Chapter-2: part2
Context-free Languages

King Saud University

Department of Computer Science

Dr. Azzam Alsudais

Outline

- Introduction
- Pushdown automata (PDA)

Introduction

- **Finite automata are limited in terms of memory and how much information they can retain at any given time.**

- **What if we could augment FA with additional memory?**
  - **Pushdown automata (PDA)**
  - **Just like NFA, but with extra memory (stack)**

Pushdown Automata (PDA)

- **PDA are equivalent to CFG**
- **PDA *recognize* context-free languages, and CFGs can *generate* those languages.**
- **Using a stack gives the automata more power to remember things in a certain order (LIFO).**
- **“A PDA can write symbols on the stack and read them back later”.**

Pushdown Automata (PDA): Stack

- **The stack follows a last-in-first-out order.**
  - **Last element “pushed” onto the stack is the first element “popped” from the stack.**

Pushdown Automata (PDA): Stack

- **The stack follows a last-in-first-out order.**
  - **Last element “pushed” onto the stack is the first element “popped” from the stack.**
- **Writing a symbol on the stack “pushes down” all other symbol previously added to the stack.**

Pushdown Automata (PDA): Stack

- **The stack follows a last-in-first-out order.**
  - **Last element “pushed” onto the stack is the first element “popped” from the stack.**
- **Writing a symbol on the stack “pushes down” all other symbol previously added to the stack.**

Pushdown Automata (PDA): Stack

- $L = \{0^n1^n \mid n \geq 0\}$
  - A PDA can recognize this language.
    - We can use the stack to store how many 0s the PDA has seen.
    - Then, for each 1 in the input, we pop a 0 from the stack.
    - If all input is consumed, and no 0s remain in the stack, we accept the string.
- The alphabet used for the stack can be different than the one used for the language (input).

Pushdown Automata (PDA): Transition Function

- **The transition function of a PDA is determined by the following:**
  - **The current state**
  - **The input symbol**
  - **The symbol popped from the stack**

Pushdown Automata (PDA): Transition Function

- **The transition function of a PDA is determined by the following:**
  - **The current state**
  - **The input symbol**
  - **The symbol popped from the stack**

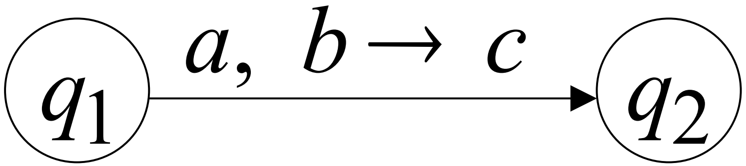
Since we're dealing with nondeterministic PDA, these can be the empty string

Pushdown Automata (PDA): Transition Function

- **The transition function of a PDA is determined by the following:**
  - **The current state**
  - **The input symbol**
  - **The symbol popped from the stack**
- **Transitioning from state to state may result in pushing some symbol onto the stack.**

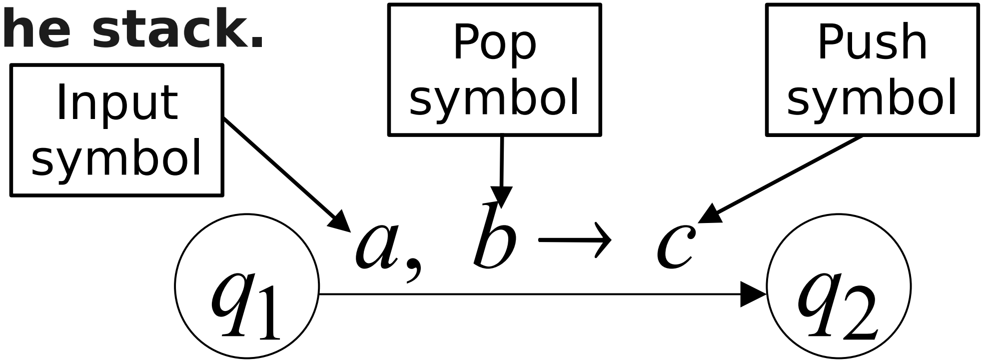
Pushdown Automata (PDA): Transition Function

- **The transition function of a PDA is determined by the following:**
  - **The current state**
  - **The input symbol**
  - **The symbol popped from the stack**
- **Transitioning from state to state may result in pushing some symbol onto the stack.**



Pushdown Automata (PDA): Transition Function

- **The transition function of a PDA is determined by the following:**
  - **The current state**
  - **The input symbol**
  - **The symbol popped from the stack**
- **Transitioning from state to state may result in pushing some symbol onto the stack.**



Pushdown Automata (PDA): Formal Definition

- **A PDA is a 6-tuple (Q, Σ, Γ, δ, q_0, F), where Q, Σ, Γ, and F are all finite sets**

Pushdown Automata (PDA): Formal Definition

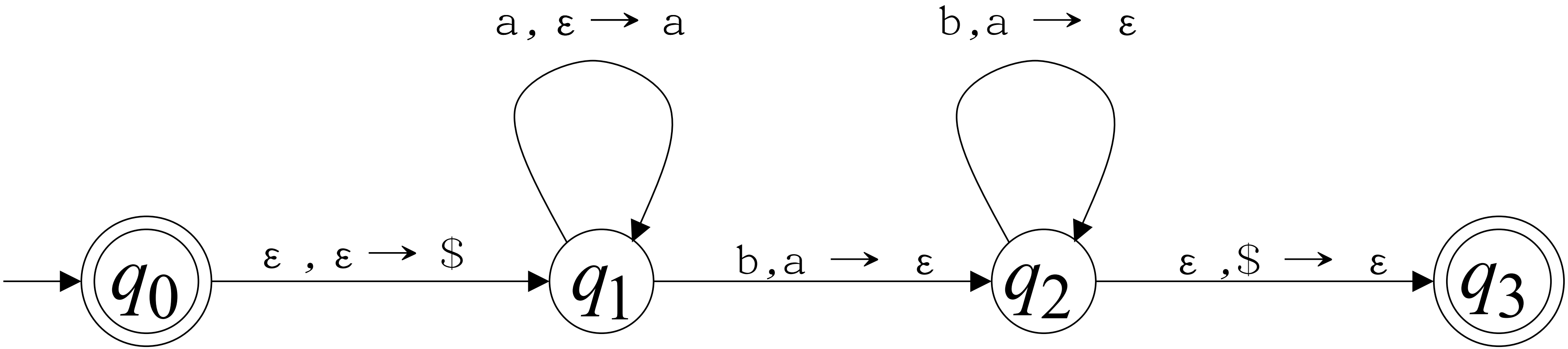
- **A PDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q, Σ, Γ, and F are all finite sets**
  - **Q is the set of states,**
  - **Σ is the input alphabet,**
  - **Γ is the stack alphabet,**
  - **$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$ is the transition function,**
  - **$q_0 \in Q$ is the start state, and**
  - **$F \subseteq Q$ is the set of accept states.**

Pushdown Automata (PDA): Example

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

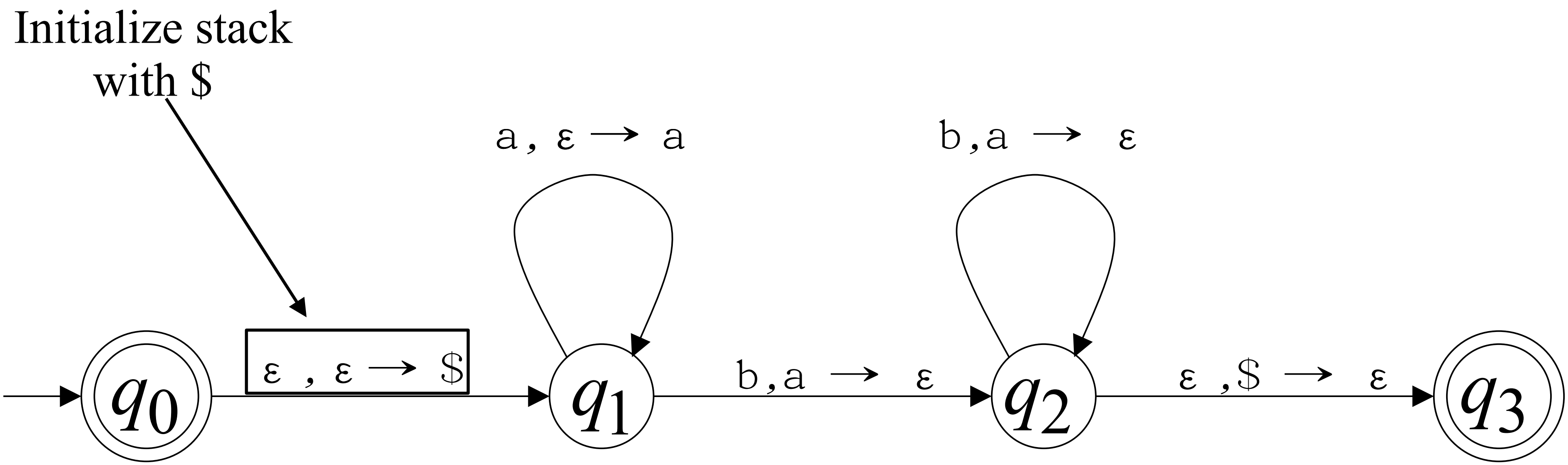
Pushdown Automata (PDA): Example

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



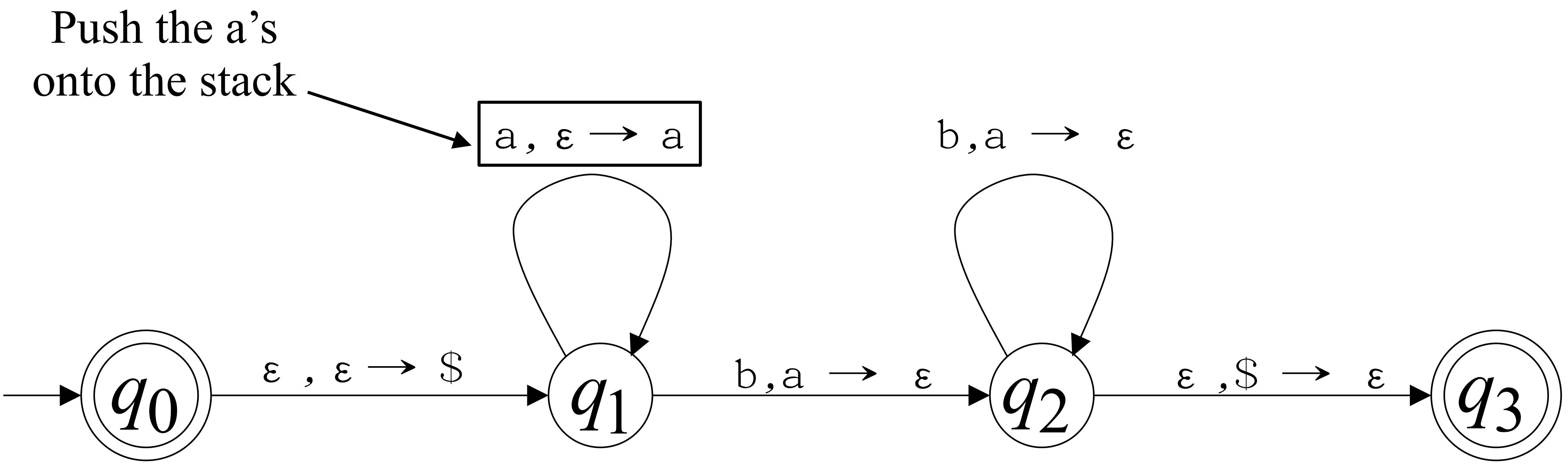
Pushdown Automata (PDA): Example

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



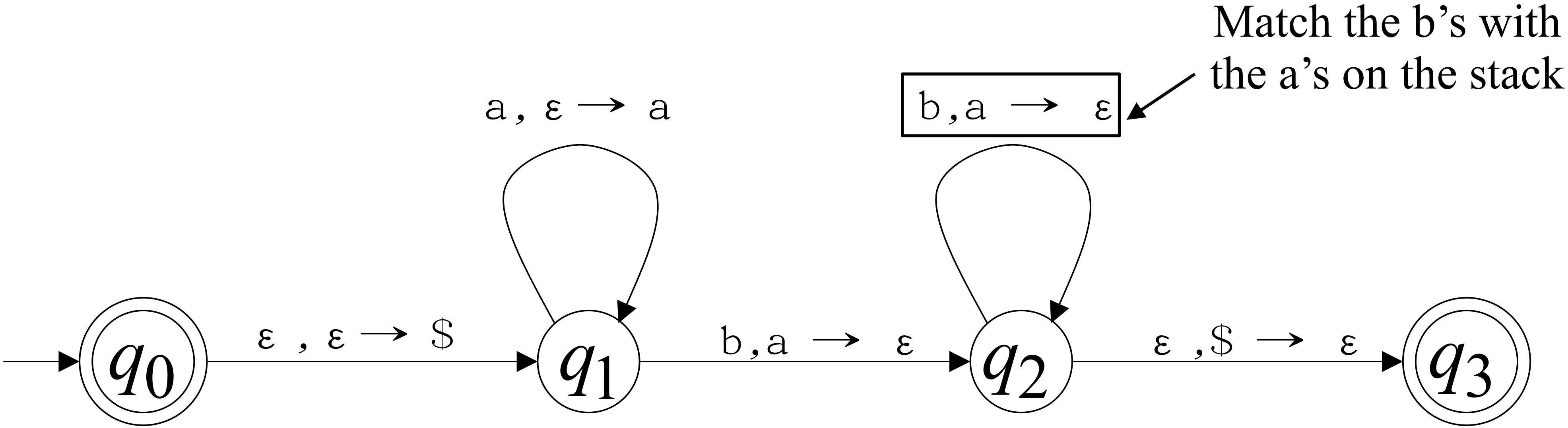
Pushdown Automata (PDA): Example

L(M) = {$a^n b^n$ | n ≥ 0}



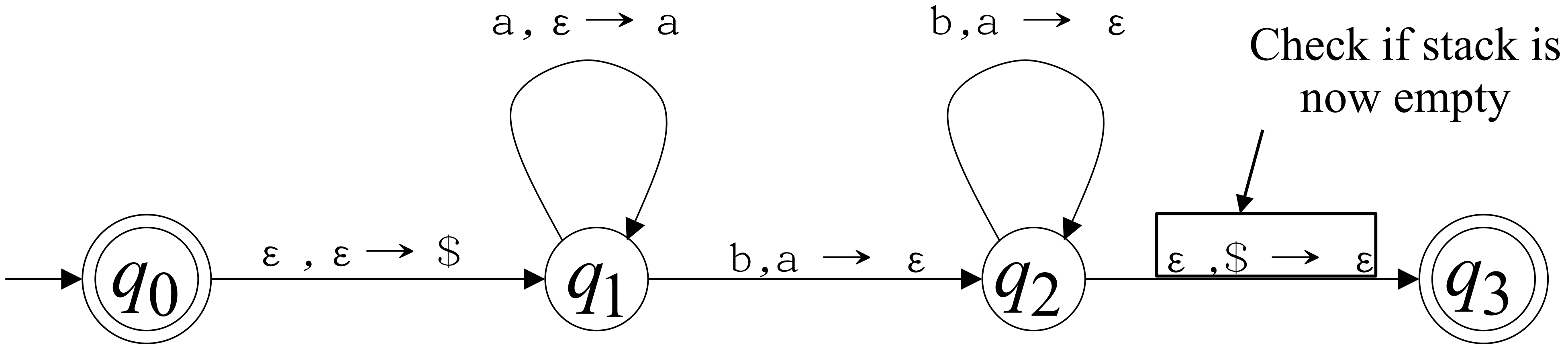
Pushdown Automata (PDA): Example

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



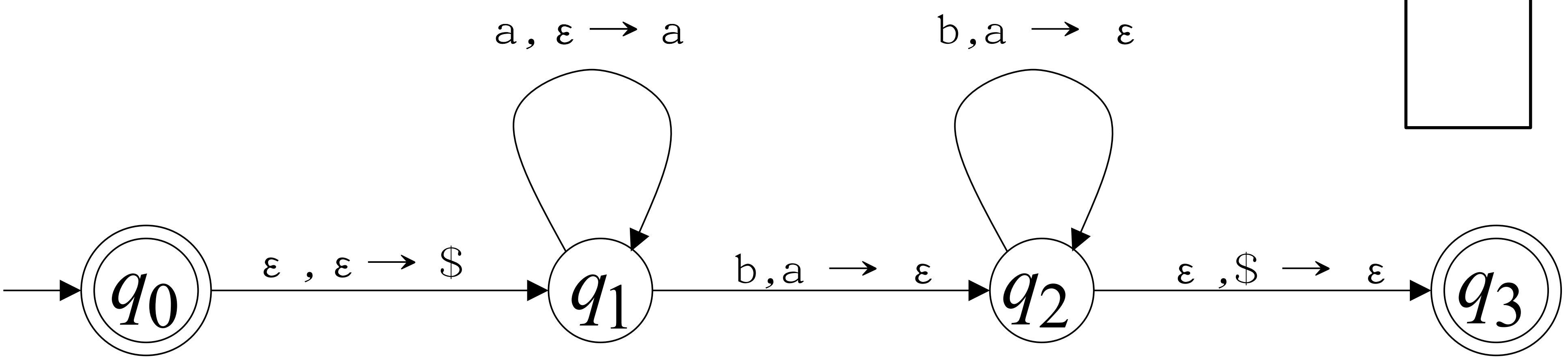
Pushdown Automata (PDA): Example

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



Pushdown Automata (PDA): Example

Input:
a a a b b b

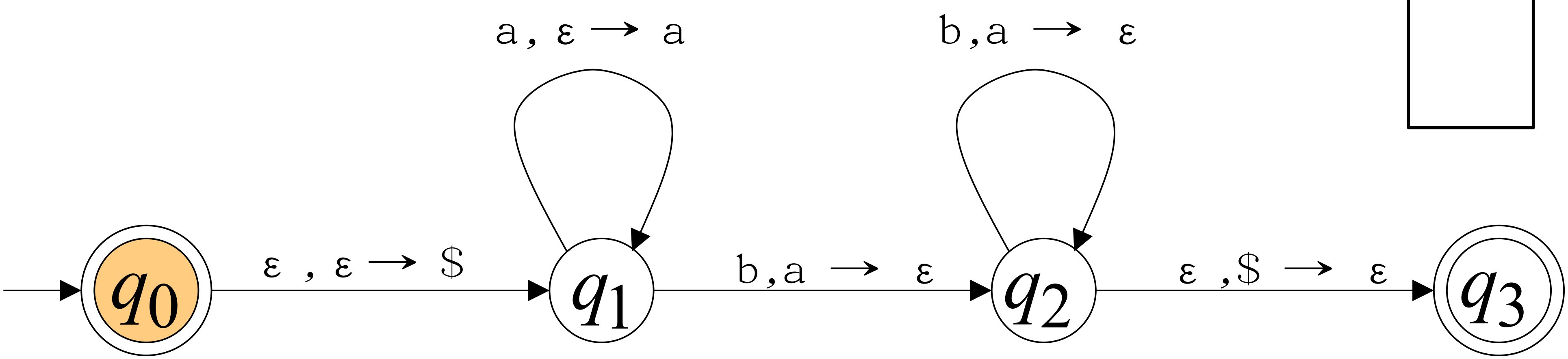


Stack

Pushdown Automata (PDA): Example

Stack

Input:
a a a b b b

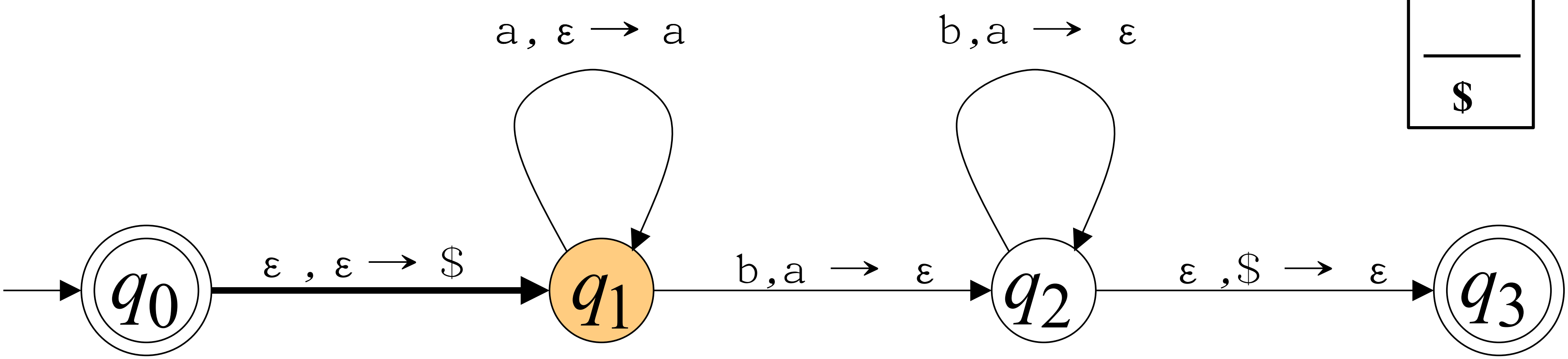


Pushdown Automata (PDA): Example

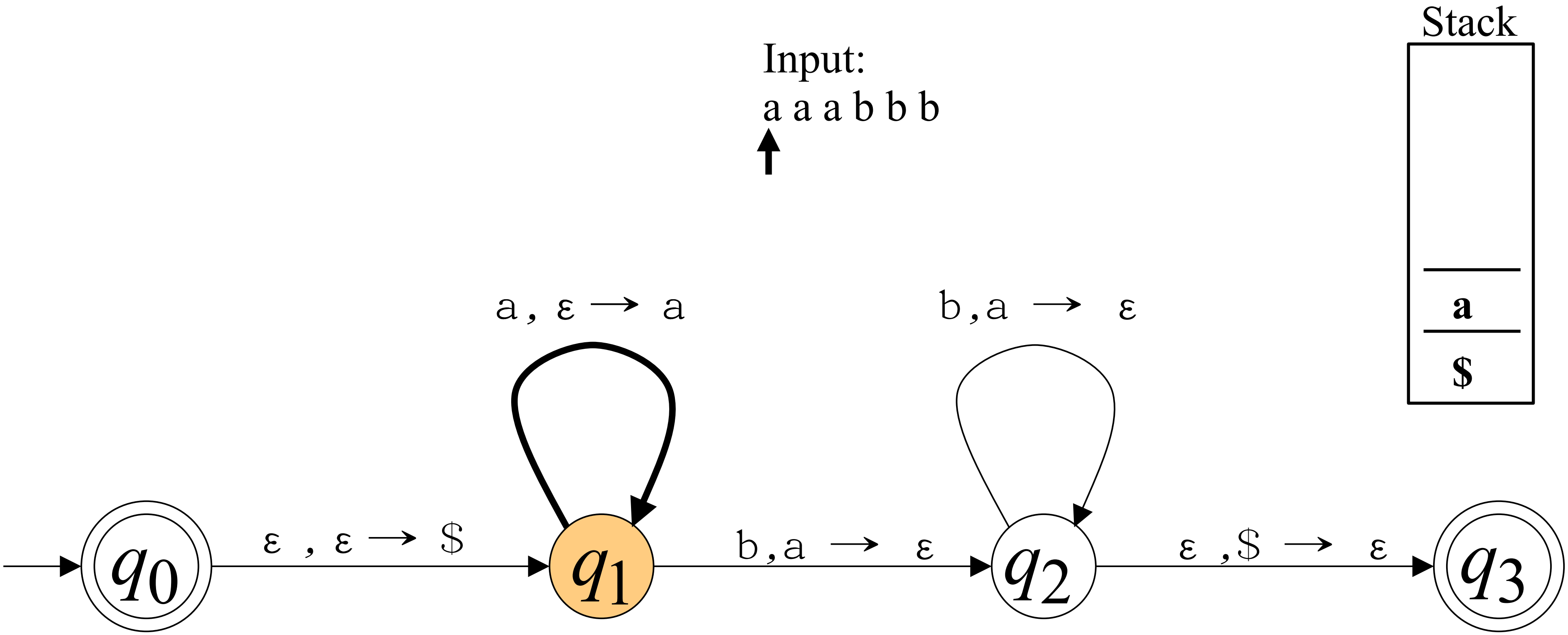
Input:
a a a b b b

Stack

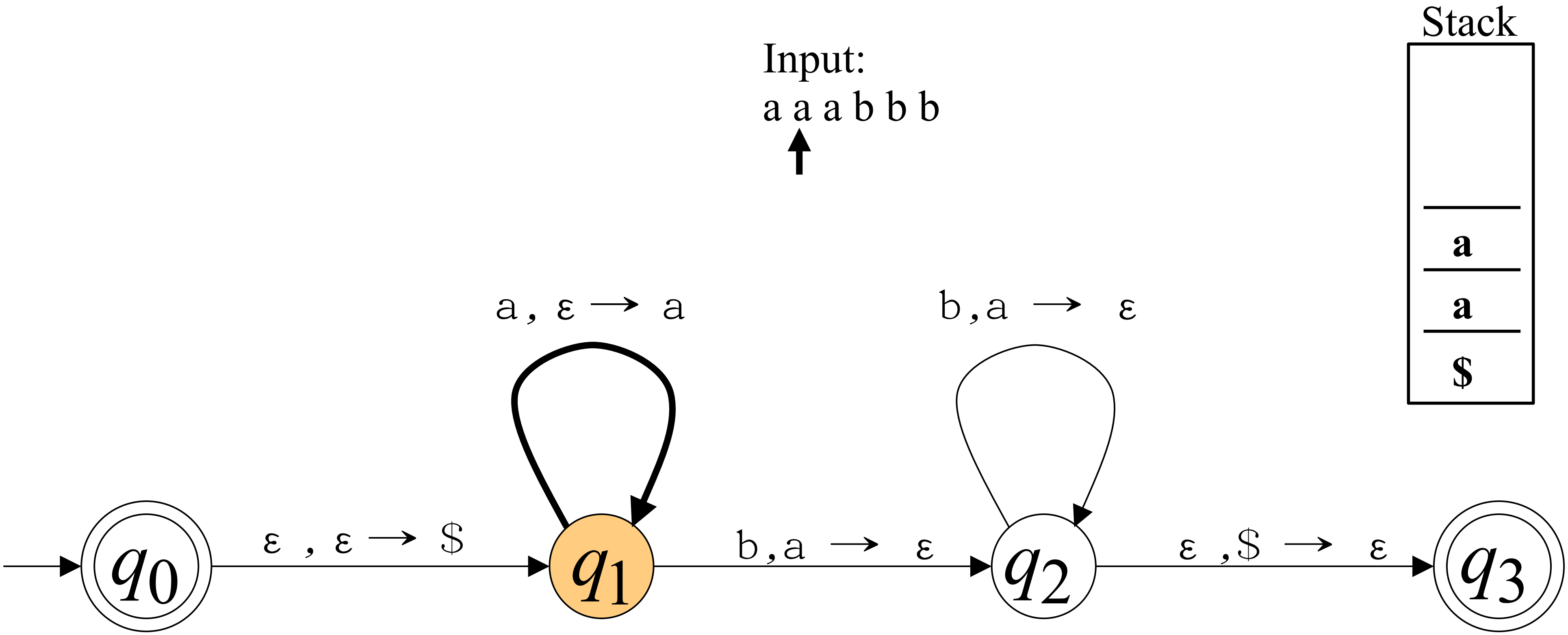
$



Pushdown Automata (PDA): Example



Pushdown Automata (PDA): Example

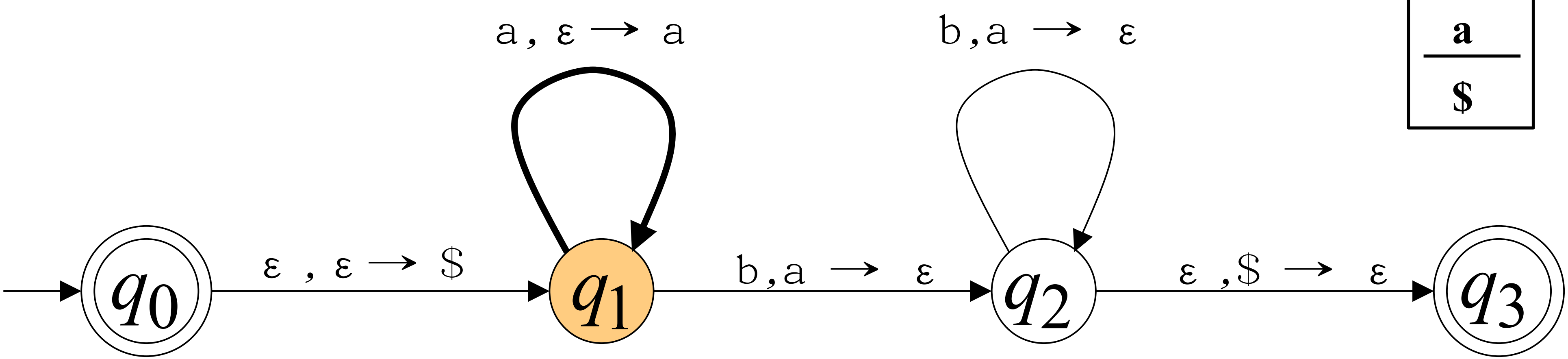


Pushdown Automata (PDA): Example

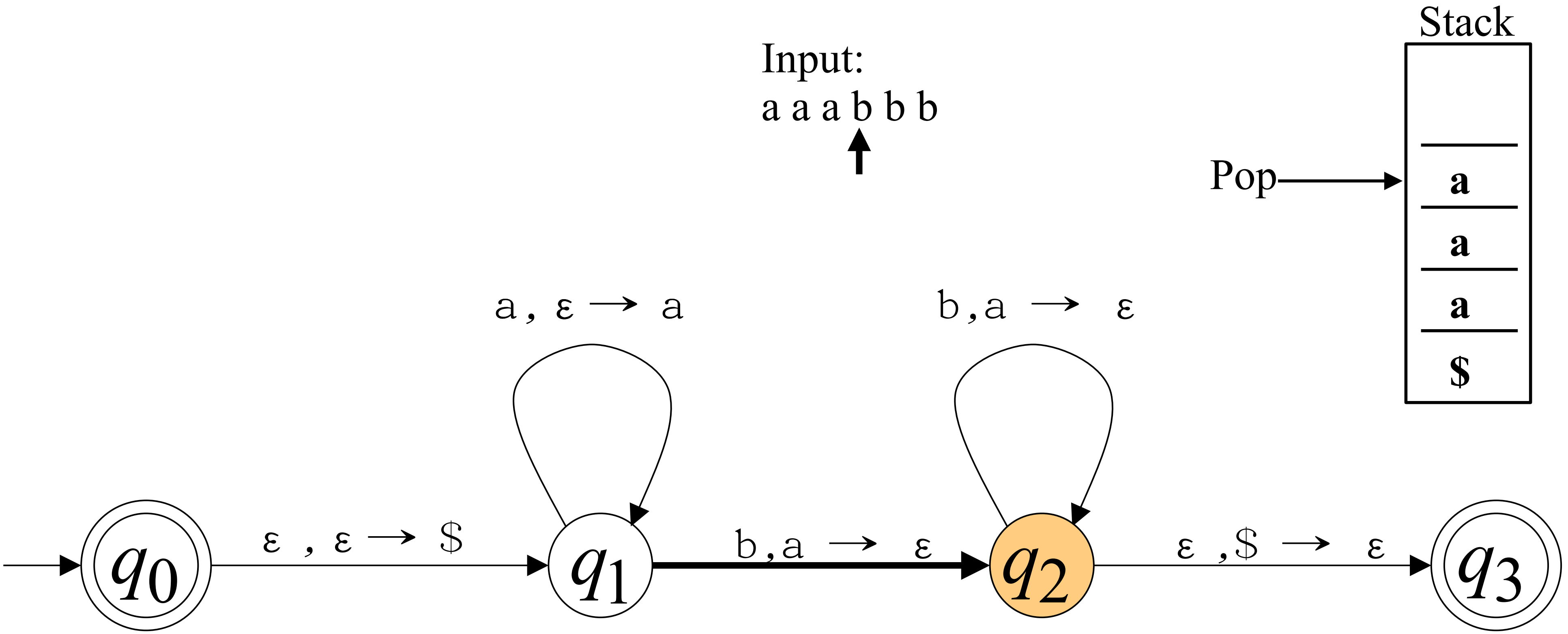
Input:
a a a b b b

Stack

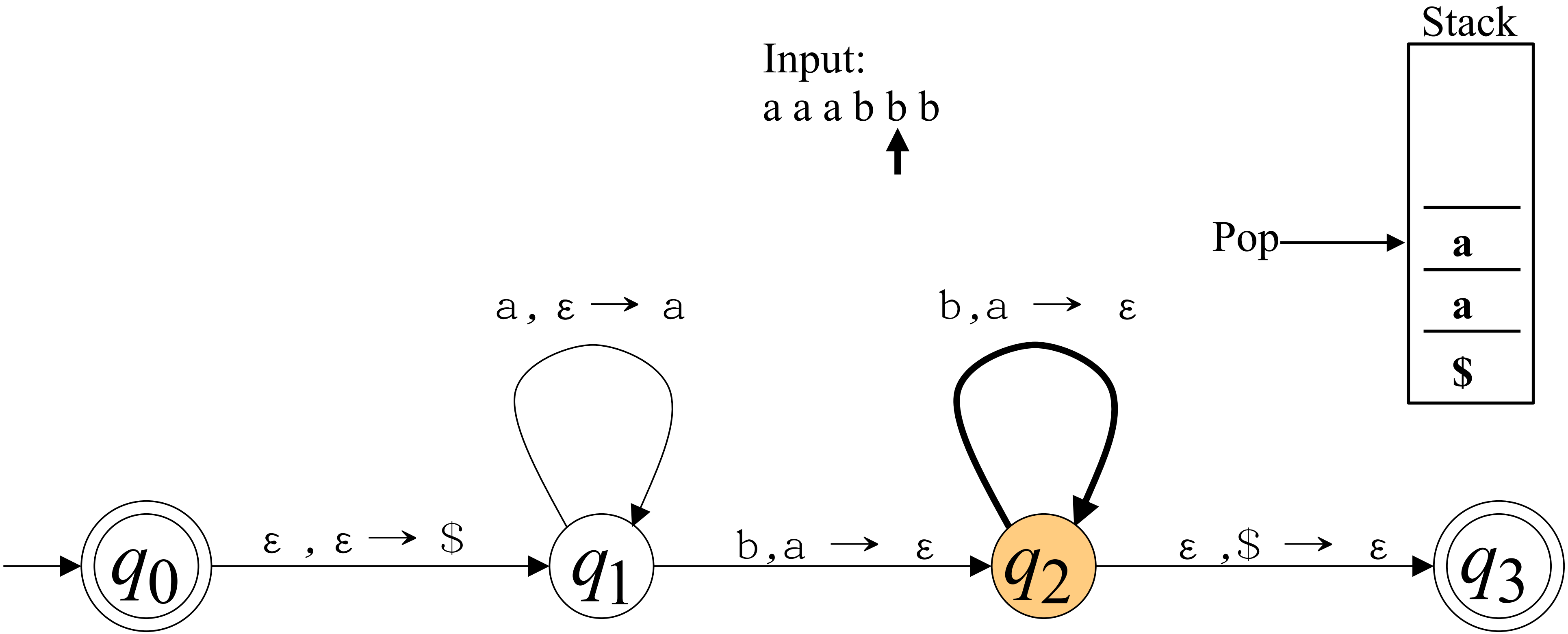
a
a
a
$



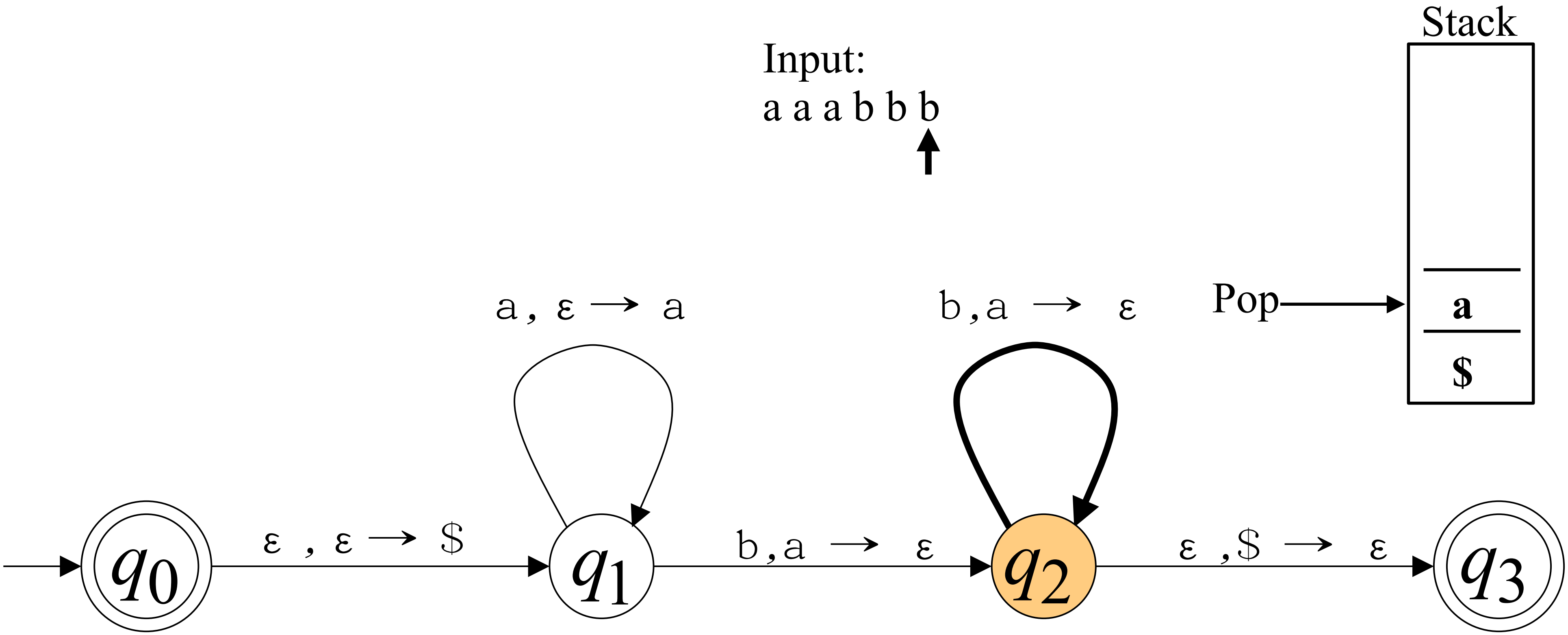
Pushdown Automata (PDA): Example



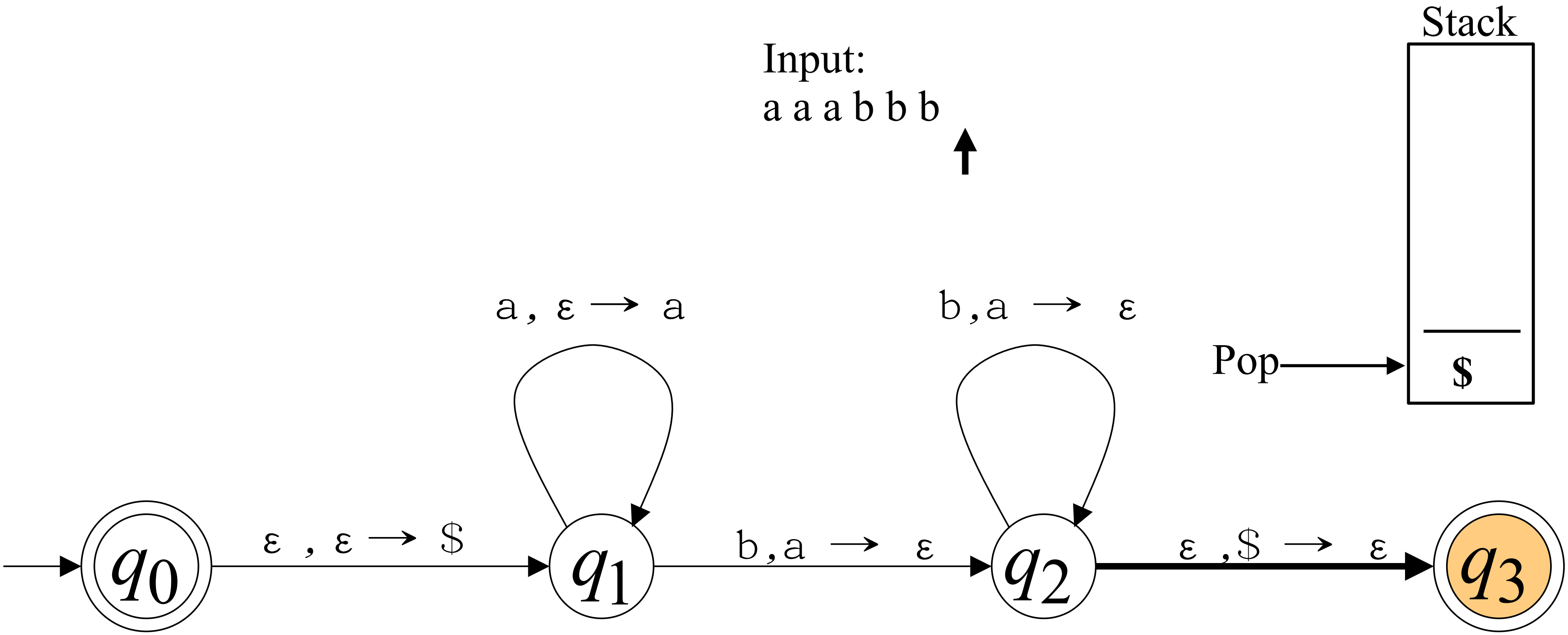
Pushdown Automata (PDA): Example



Pushdown Automata (PDA): Example



Pushdown Automata (PDA): Example

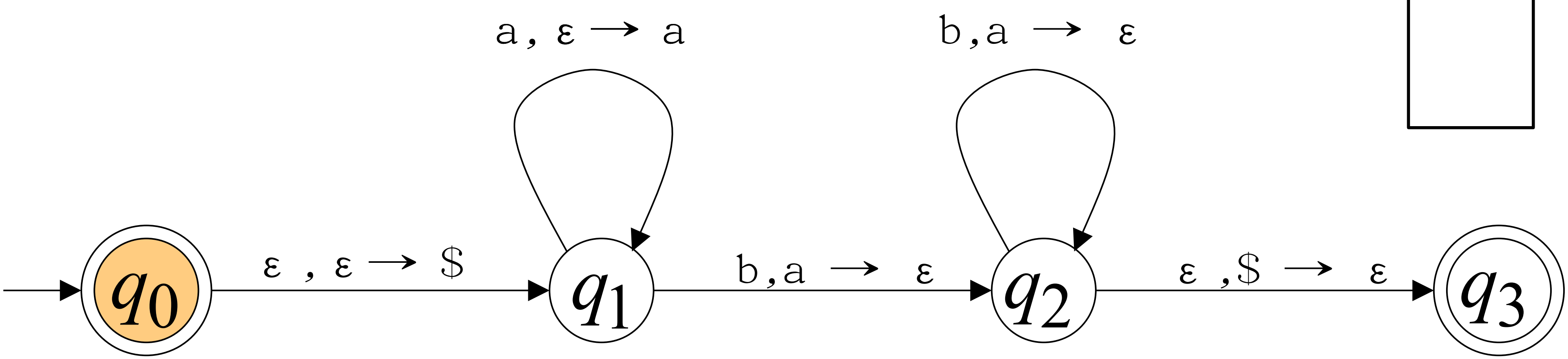


Pushdown Automata (PDA): Example

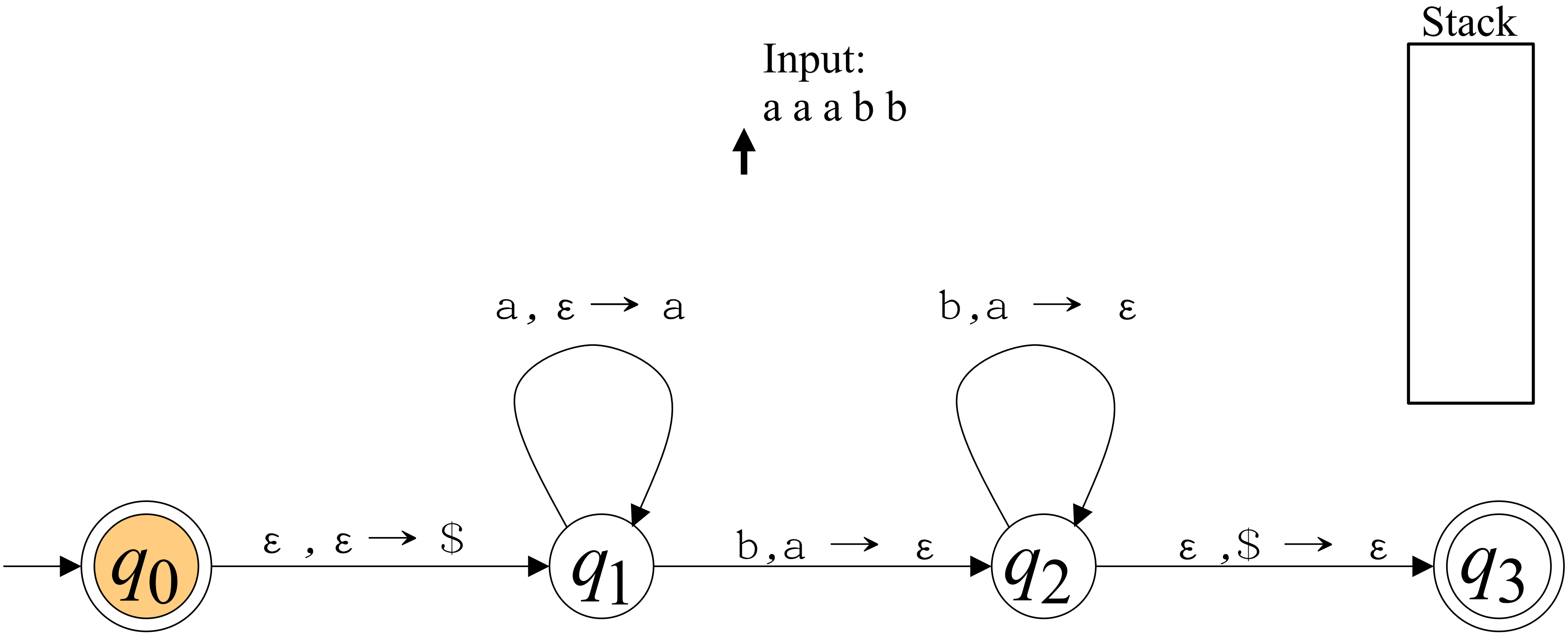
Another example

Input:
a a a b b

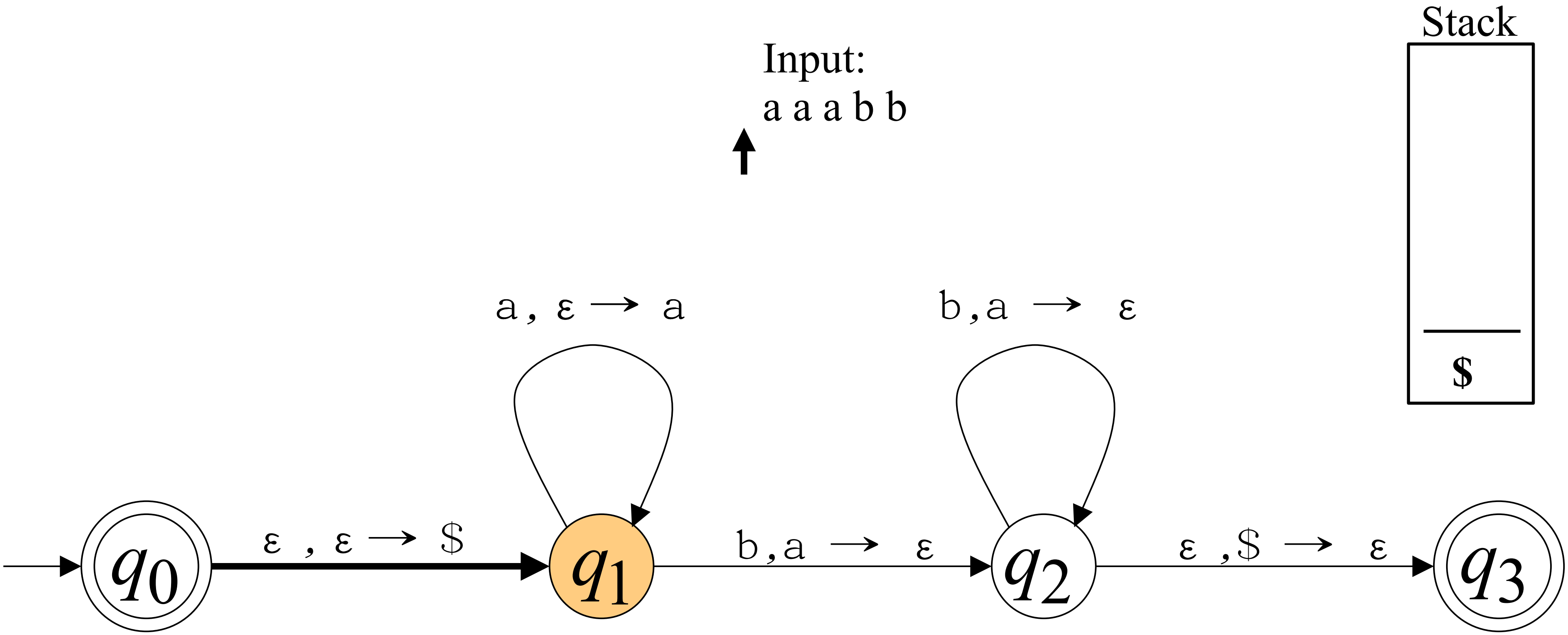
Stack



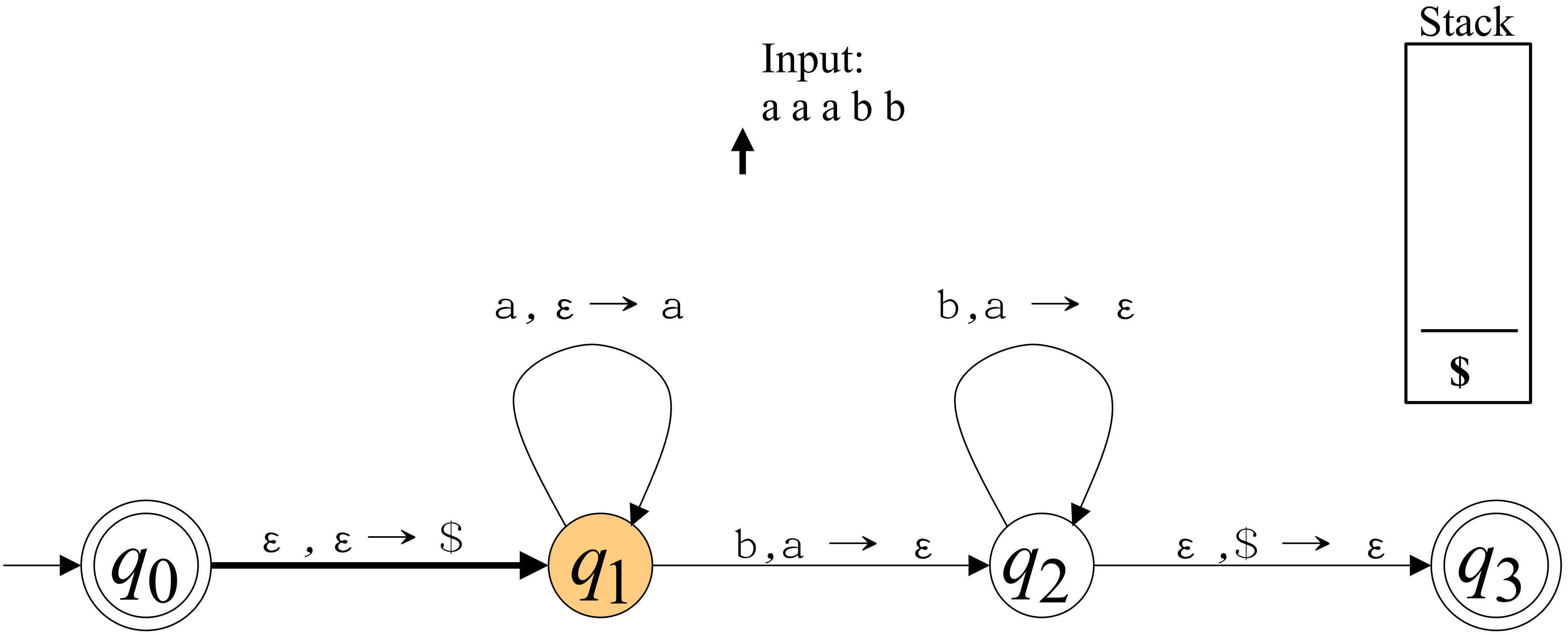
Pushdown Automata (PDA): Example



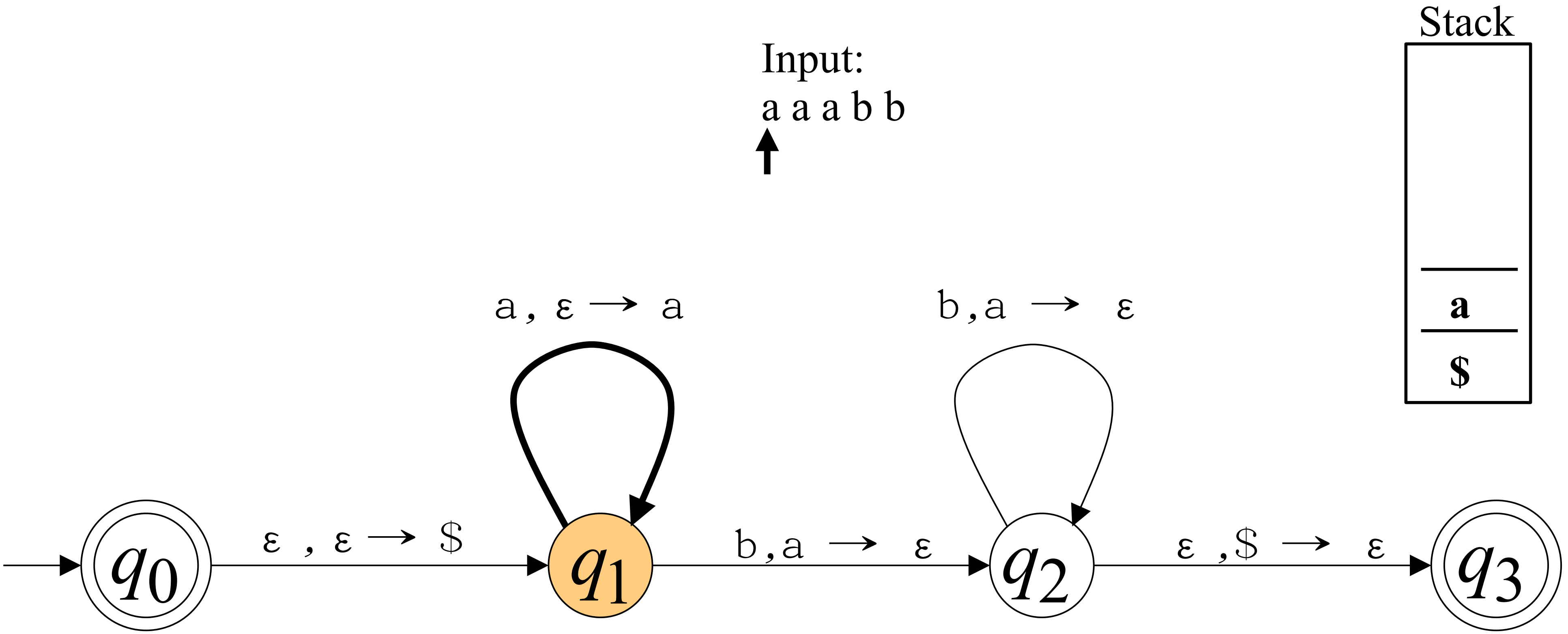
Pushdown Automata (PDA): Example



Pushdown Automata (PDA): Example



Pushdown Automata (PDA): Example

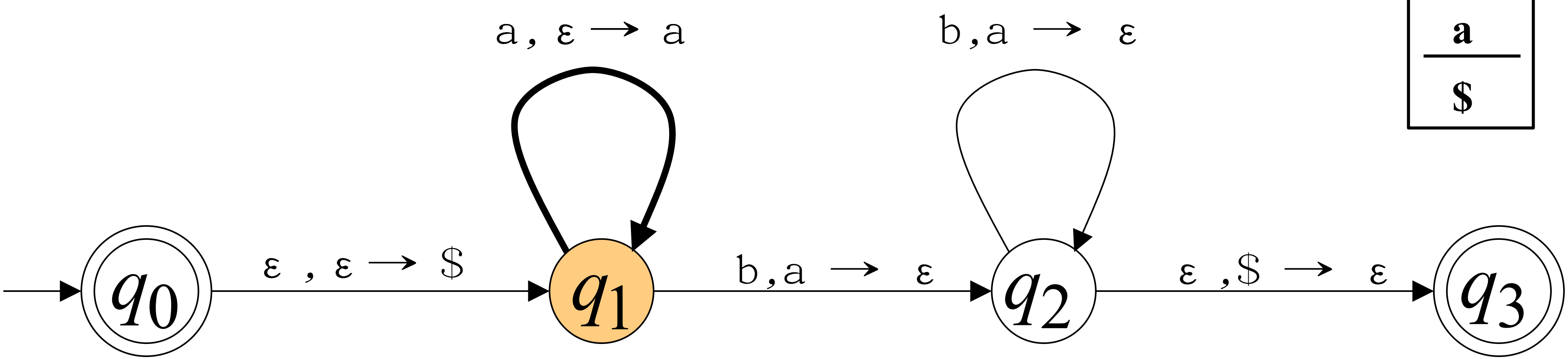


Pushdown Automata (PDA): Example

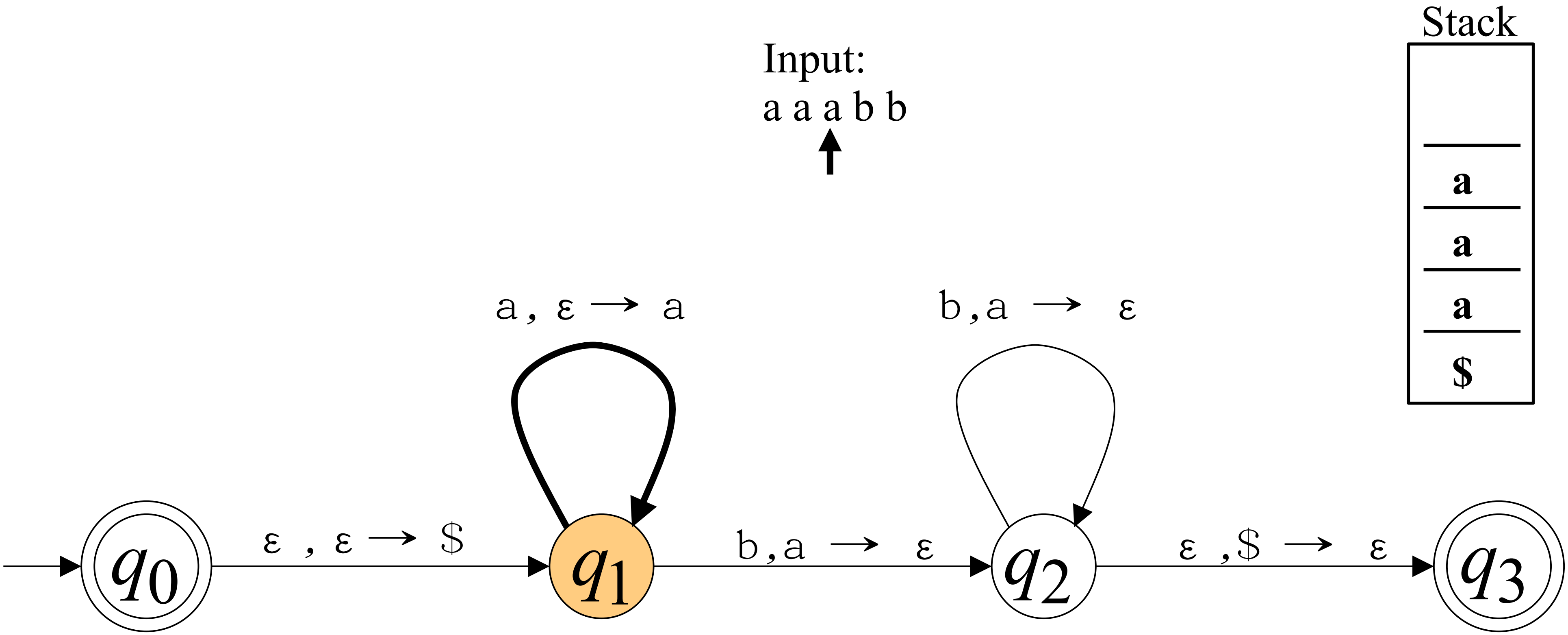
Input:
a a a b b

Stack

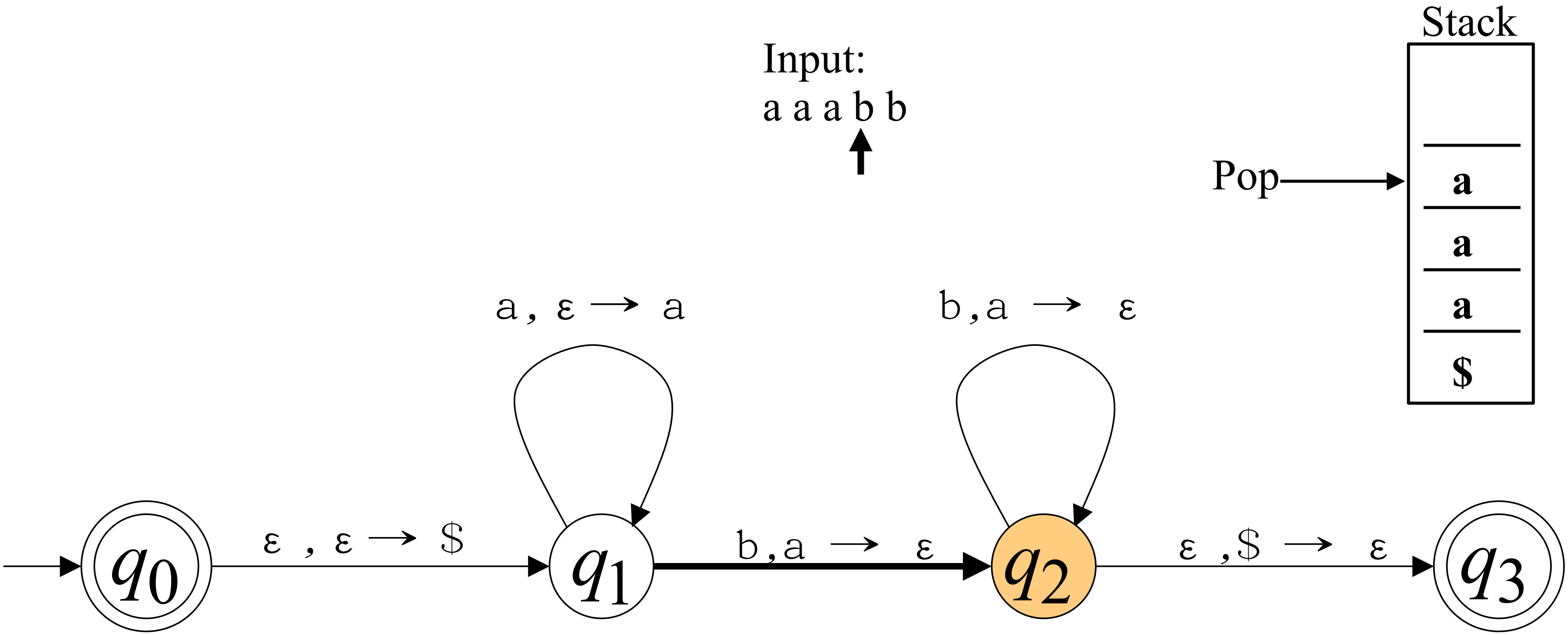
a
a
$



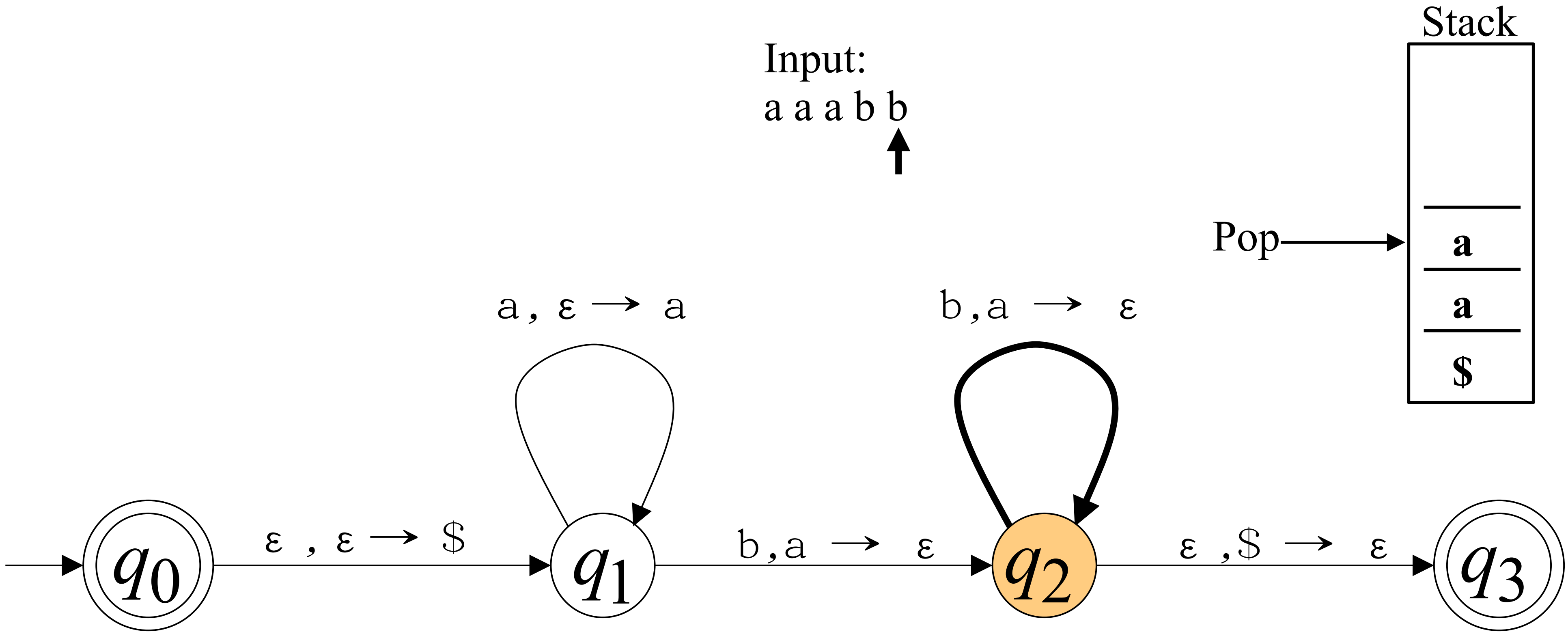
Pushdown Automata (PDA): Example



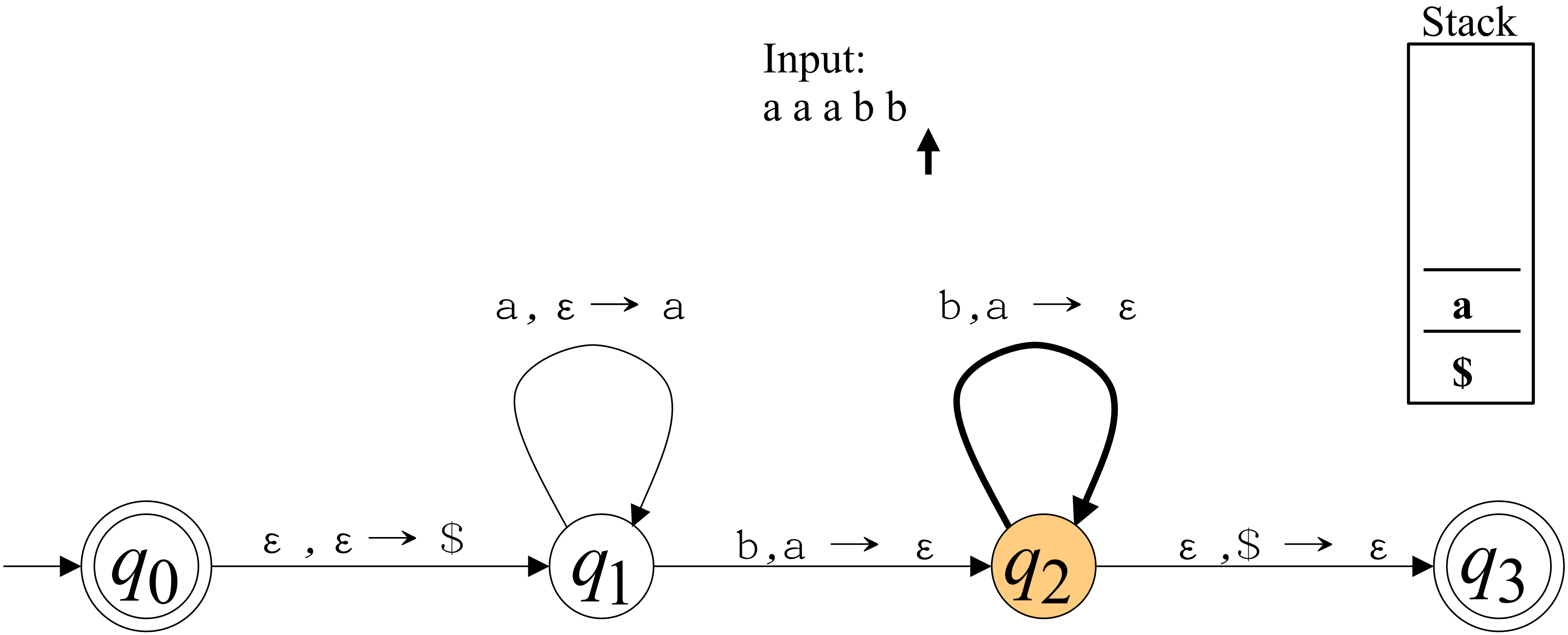
Pushdown Automata (PDA): Example



Pushdown Automata (PDA): Example



Pushdown Automata (PDA): Example



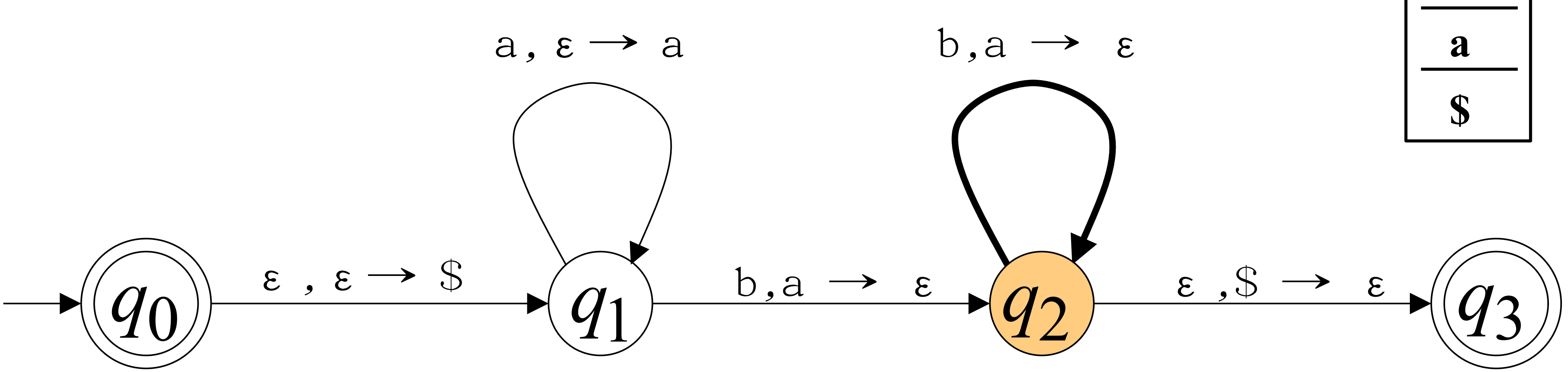
Pushdown Automata (PDA): Example

The PDA can't move to state q3

Input:

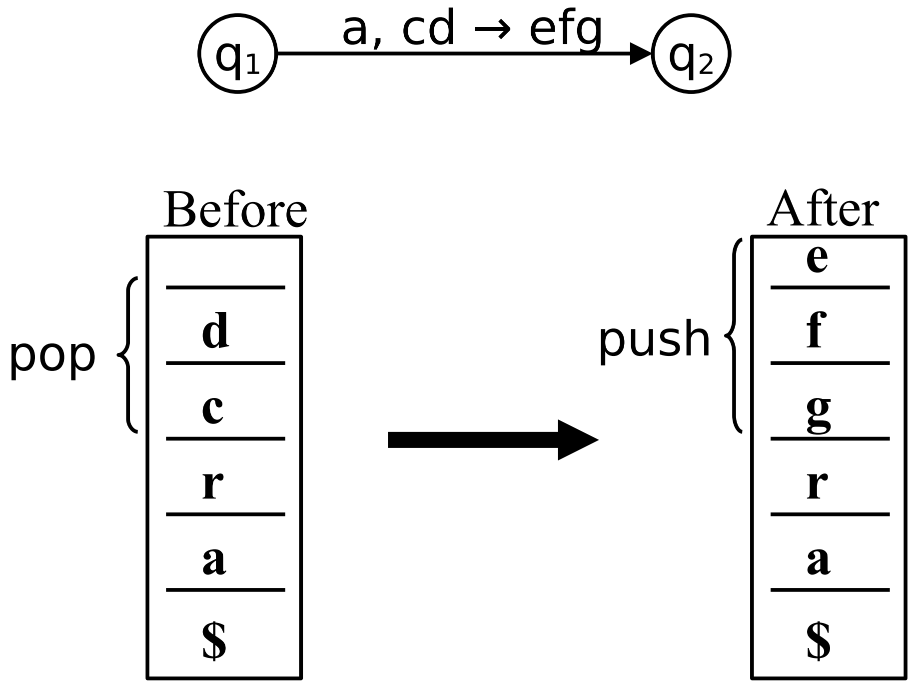
a a a b b

Stack



Pushdown Automata (PDA)

- **Transition labels can take the following form**



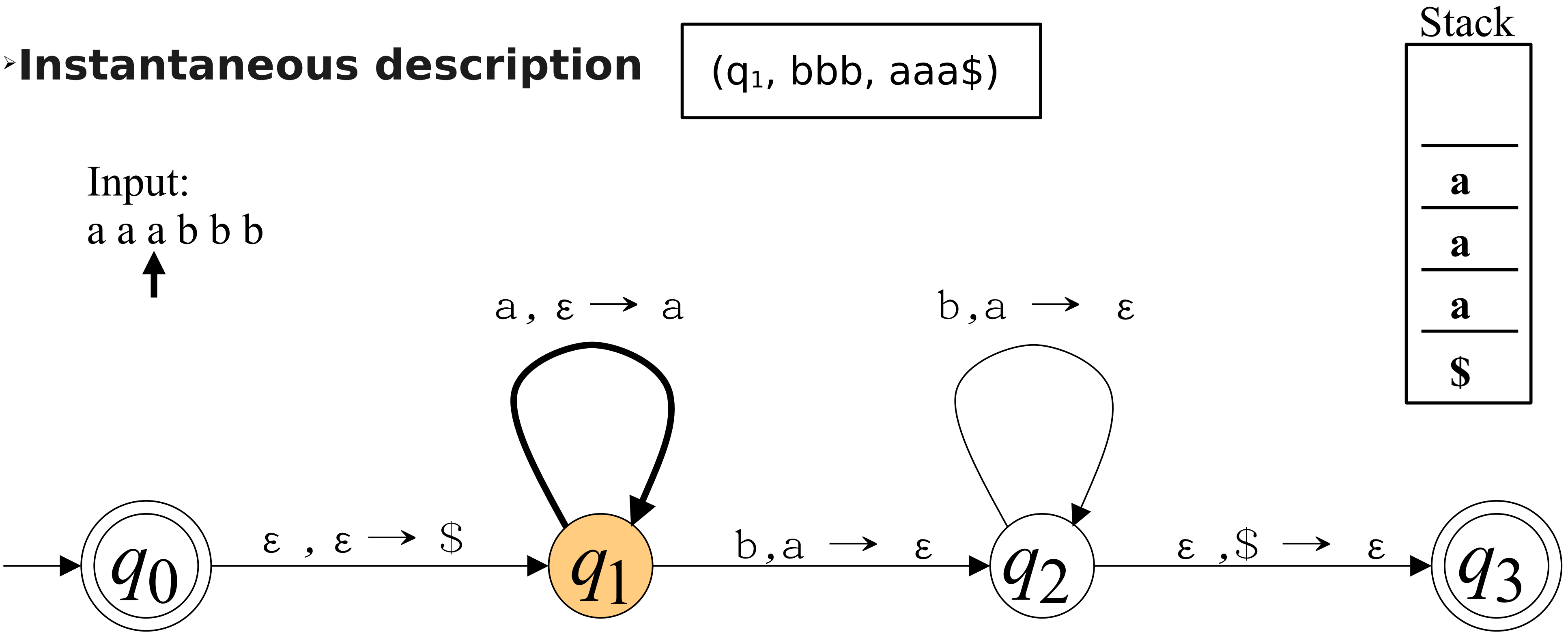
Pushdown Automata (PDA)

- **Instantaneous description**

$$\boxed{(q_1, bbb, aaa\$)}$$

Pushdown Automata (PDA)

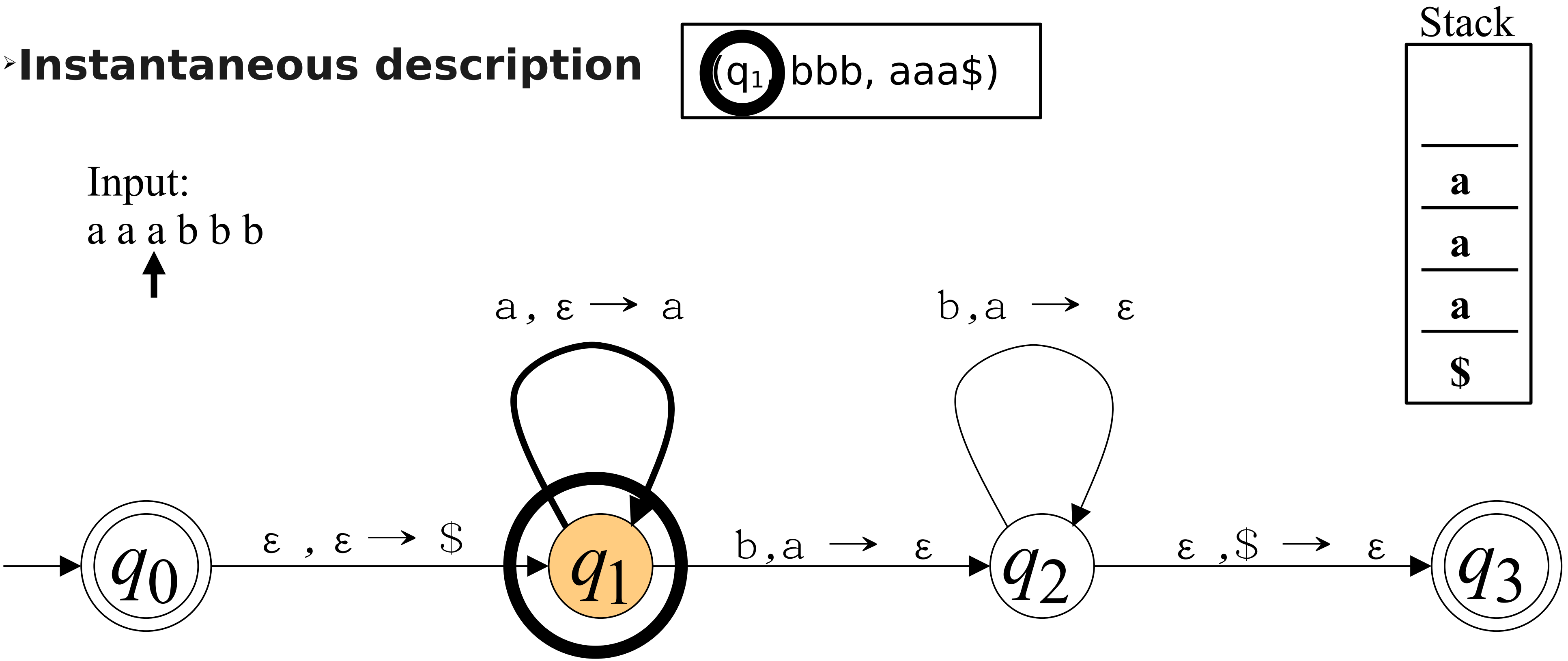
- **Instantaneous description** $(q_1, bbb, aaa\$)$



Pushdown Automata (PDA)

- **Instantaneous description**

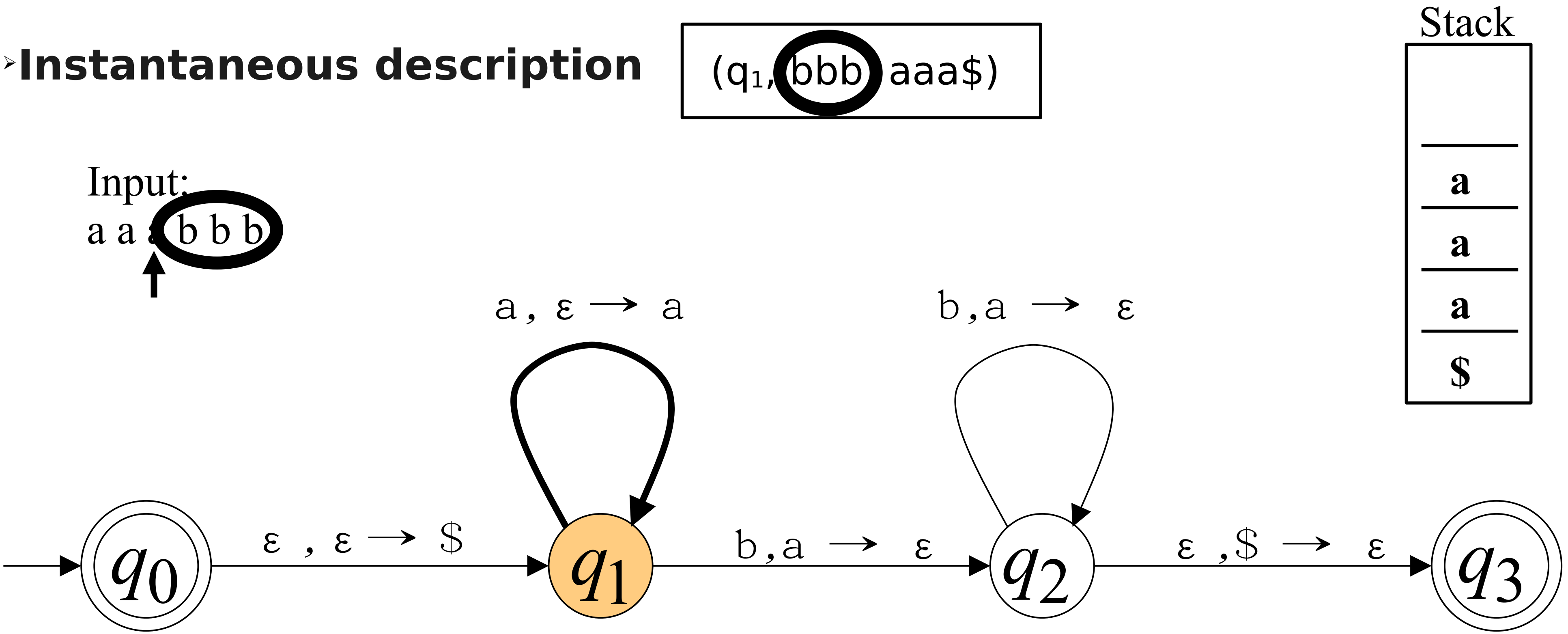
(q₁, bbb, aaa$)



Pushdown Automata (PDA)

- **Instantaneous description**

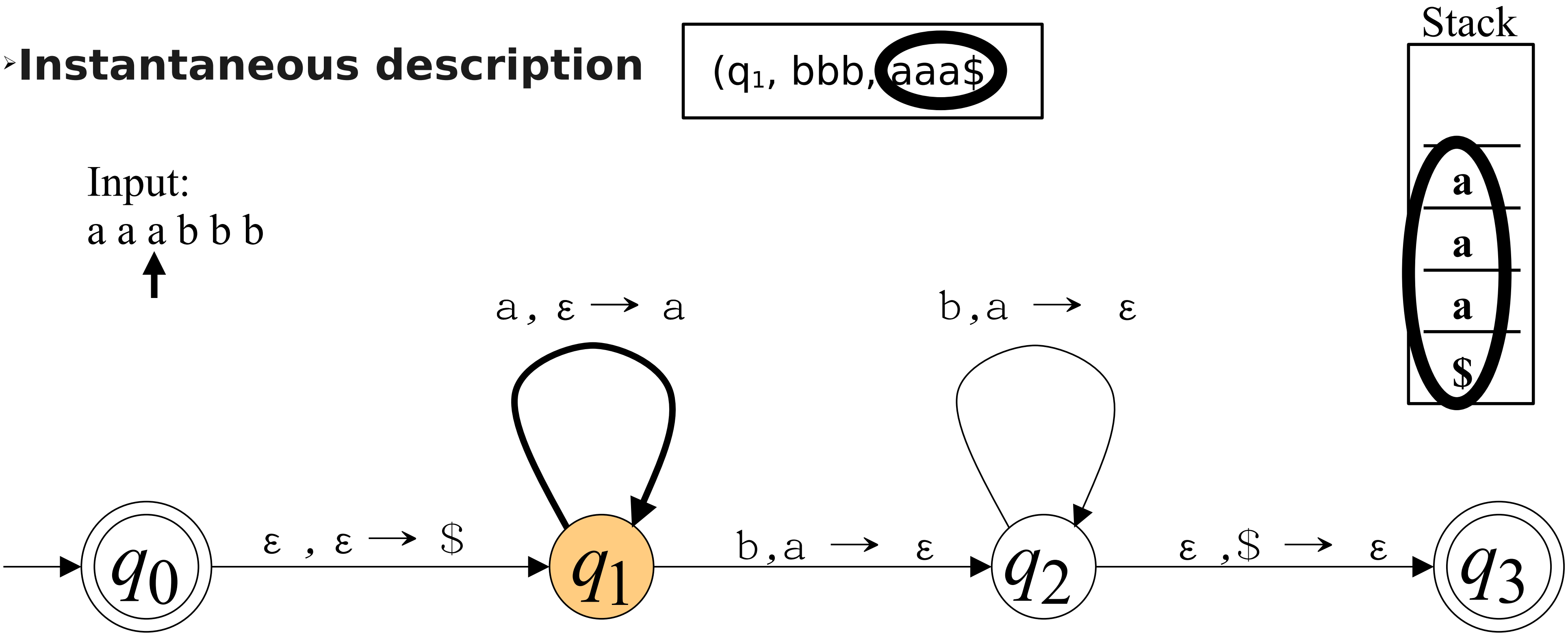
$(q_1, bbb, aaa\$)$



Pushdown Automata (PDA)

- **Instantaneous description**

$(q_1, bbb, aaa\$)$



Pushdown Automata (PDA): Theorem 2.20

"A language is context free <u>if and only if</u> some pushdown automaton recognizes it."

Pushdown Automata (PDA): Theorem 2.20

"A language is context free <u>if and only if</u> some pushdown automaton recognizes it."

- **Two directions to prove**

  1)If a language is context-free, then there is some PDA that recognizes it.

Pushdown Automata (PDA): Theorem 2.20

"A language is context free <u>if and only if</u> some pushdown automaton recognizes it."

- **Two directions to prove**

  1) **If a language is context-free, then there is some PDA that recognizes it.**
  2) **If a PDA recognizes some language, then it is context-free.**

Pushdown Automata (PDA): Theorem 2.20

- **If a language is context-free, then there is a PDA that recognizes it.**
- **Now, we'll look at a technique to convert CFG into PDA.**

Pushdown Automata (PDA): CFG → PDA

- **Goal is to construct a PDA that recognizes the same language that is generated using the CFG at hand.**

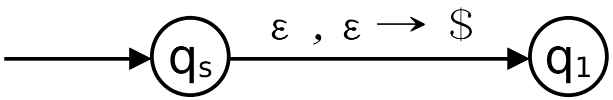
G_1

$$\boxed{\begin{array}{l} S \rightarrow xBz \mid xy \\ B \rightarrow SB \mid \varepsilon \end{array}}$$

Pushdown Automata (PDA): CFG → PDA

G_1

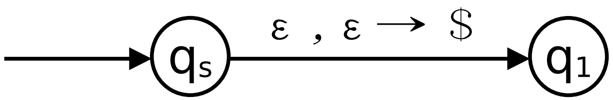
$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$

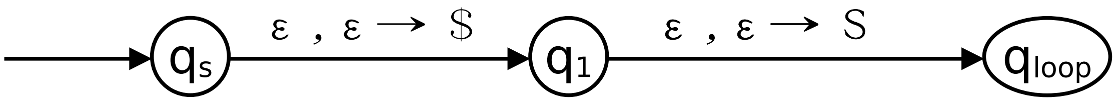


Always initialize the stack

Pushdown Automata (PDA): CFG → PDA

G_1

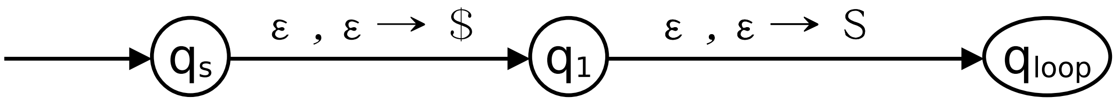
$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$

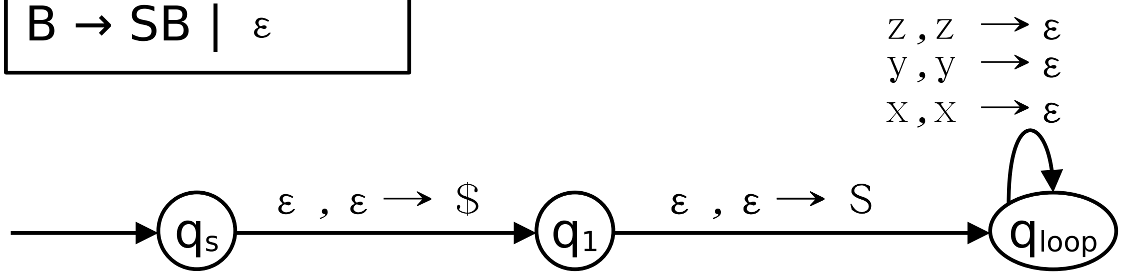


We start by pushing the start variable onto the stack

Pushdown Automata (PDA): CFG → PDA

G_1

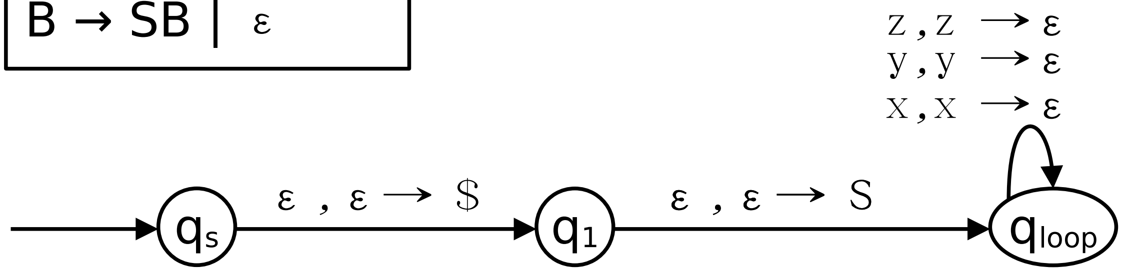
$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

$S \to xBz \mid xy$
$B \to SB \mid \varepsilon$

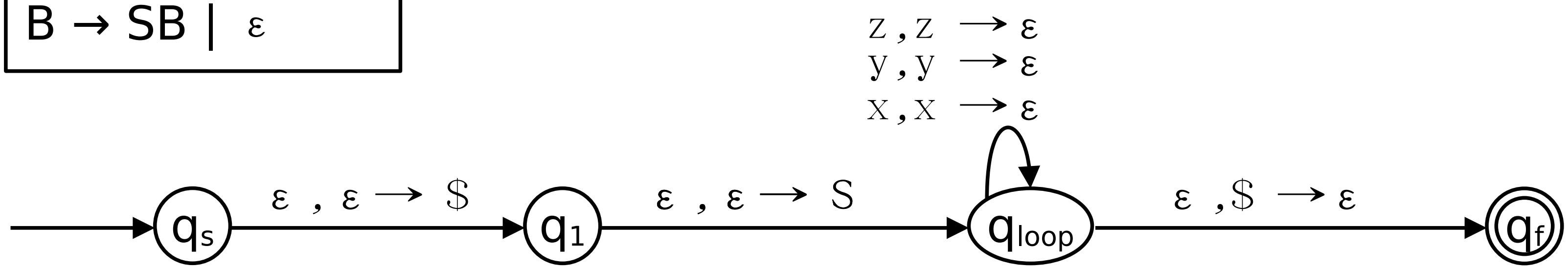


For every terminal we read, we should be able to pop it from the top of the stack

Pushdown Automata (PDA): CFG → PDA

G_1

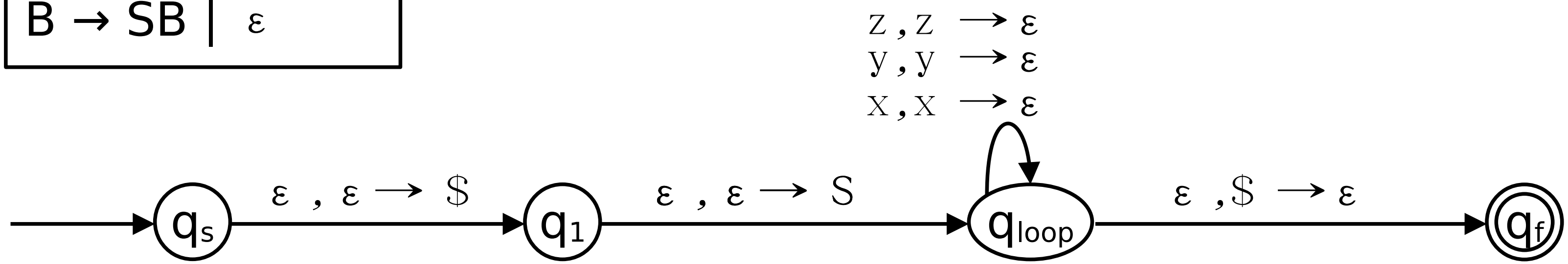
$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$

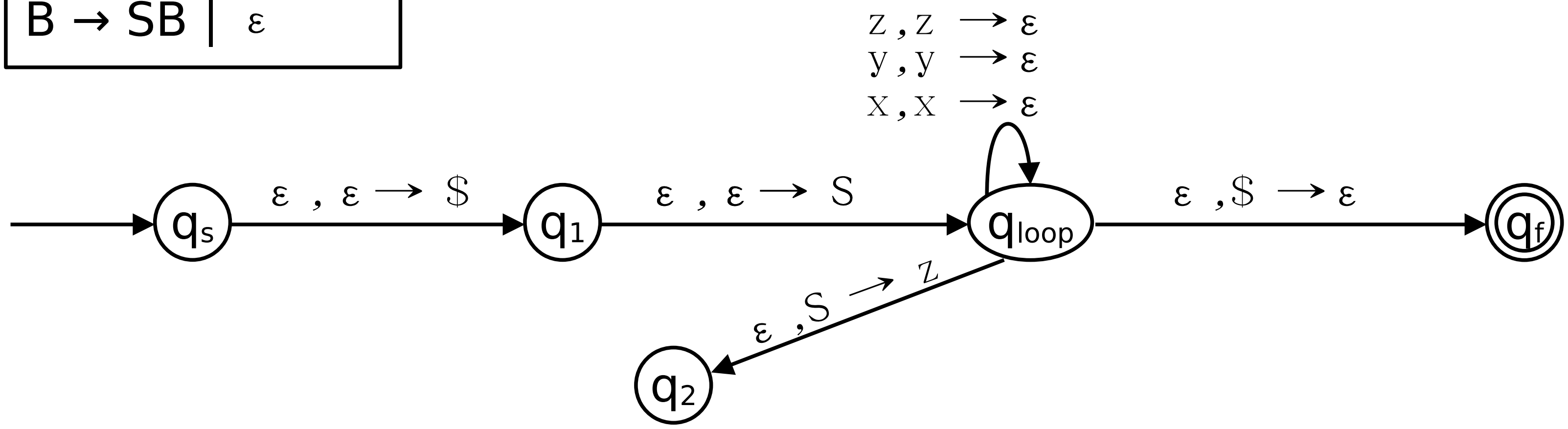


Now, we simulate the rules (productions) via transitioning through some intermediate states

Pushdown Automata (PDA): CFG → PDA

G_1

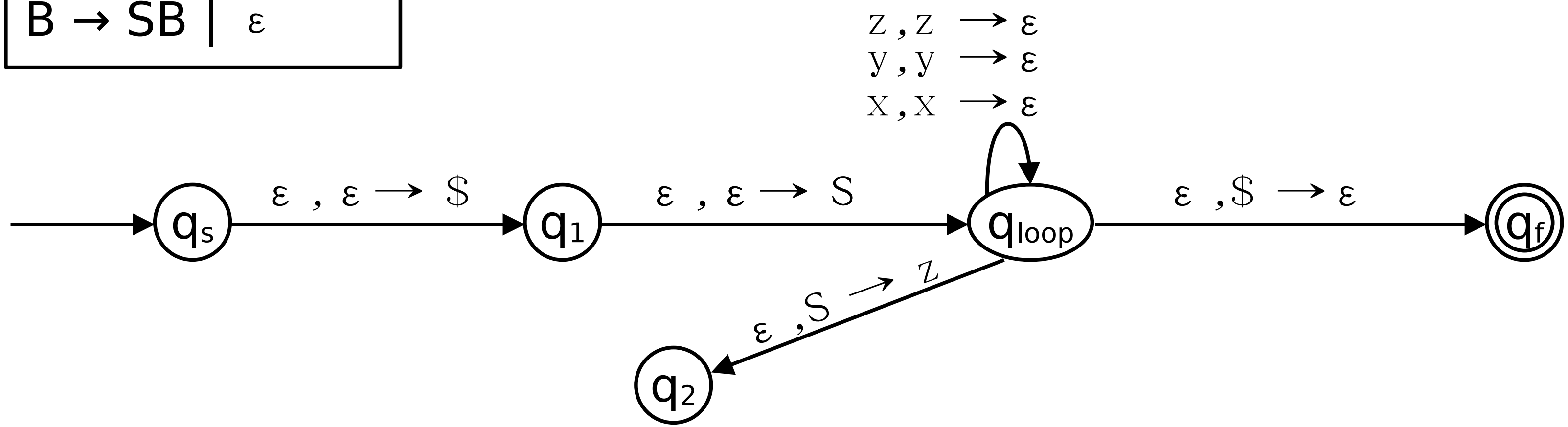
$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$

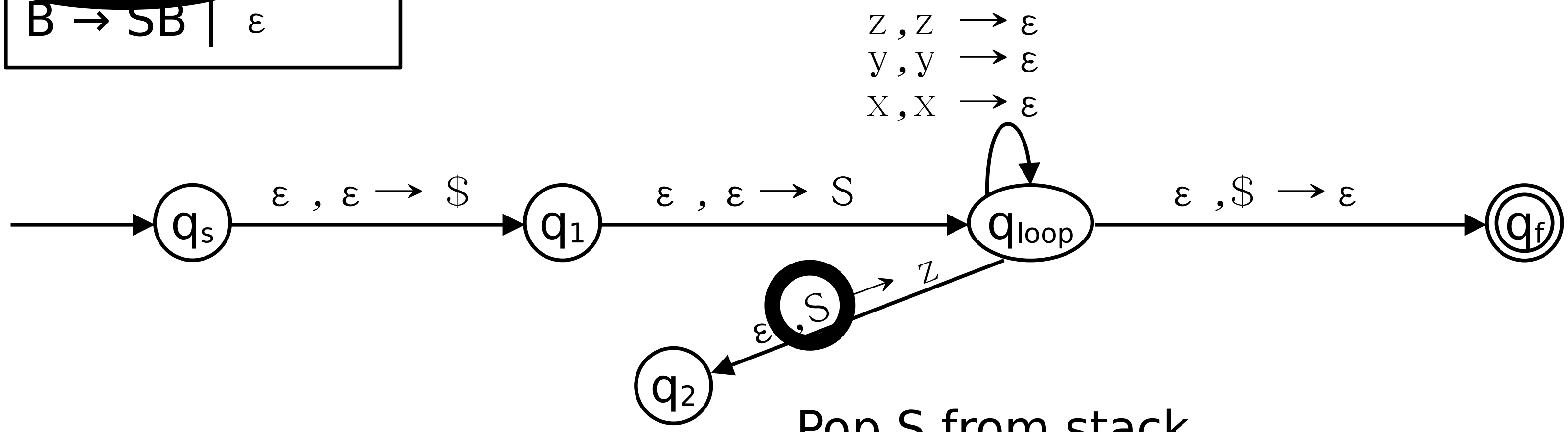


Since we're dealing with a stack we push the last element in the rule first

Pushdown Automata (PDA): CFG → PDA

G_1

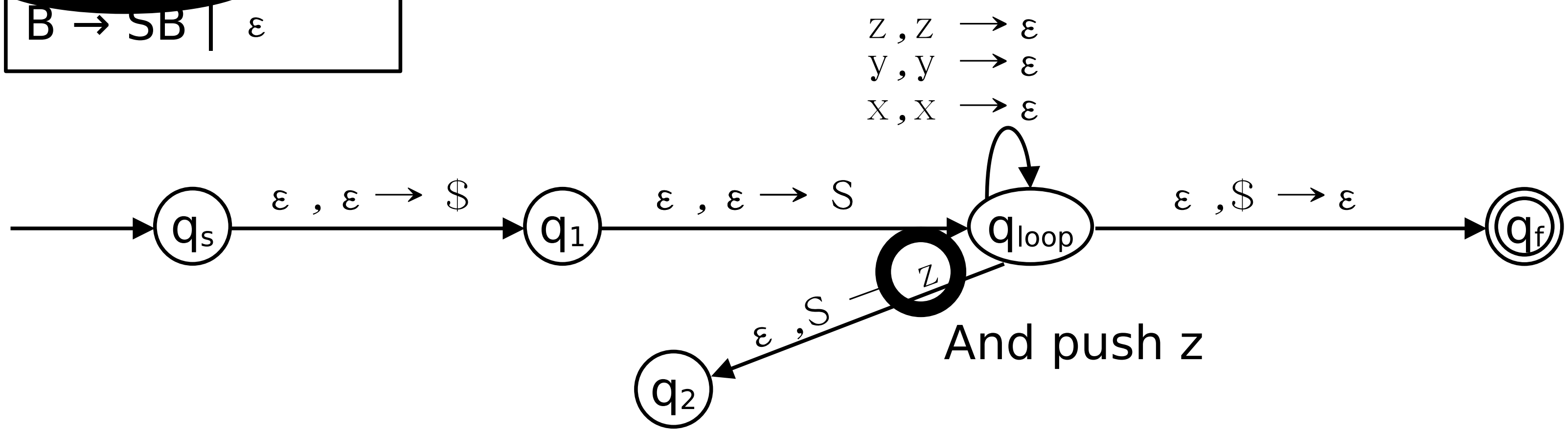
S → xBz	|	xy
B → SB	|	ε



Pop S from stack

G_1

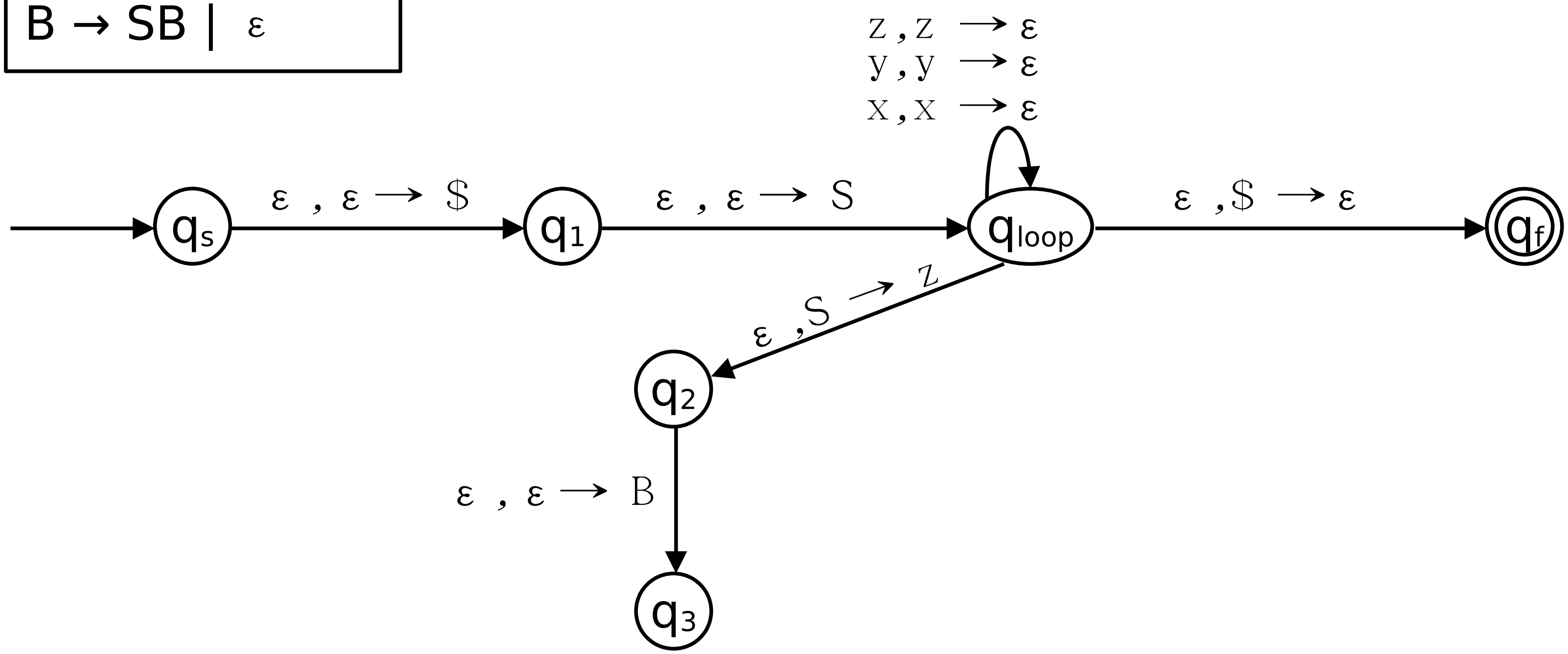
$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

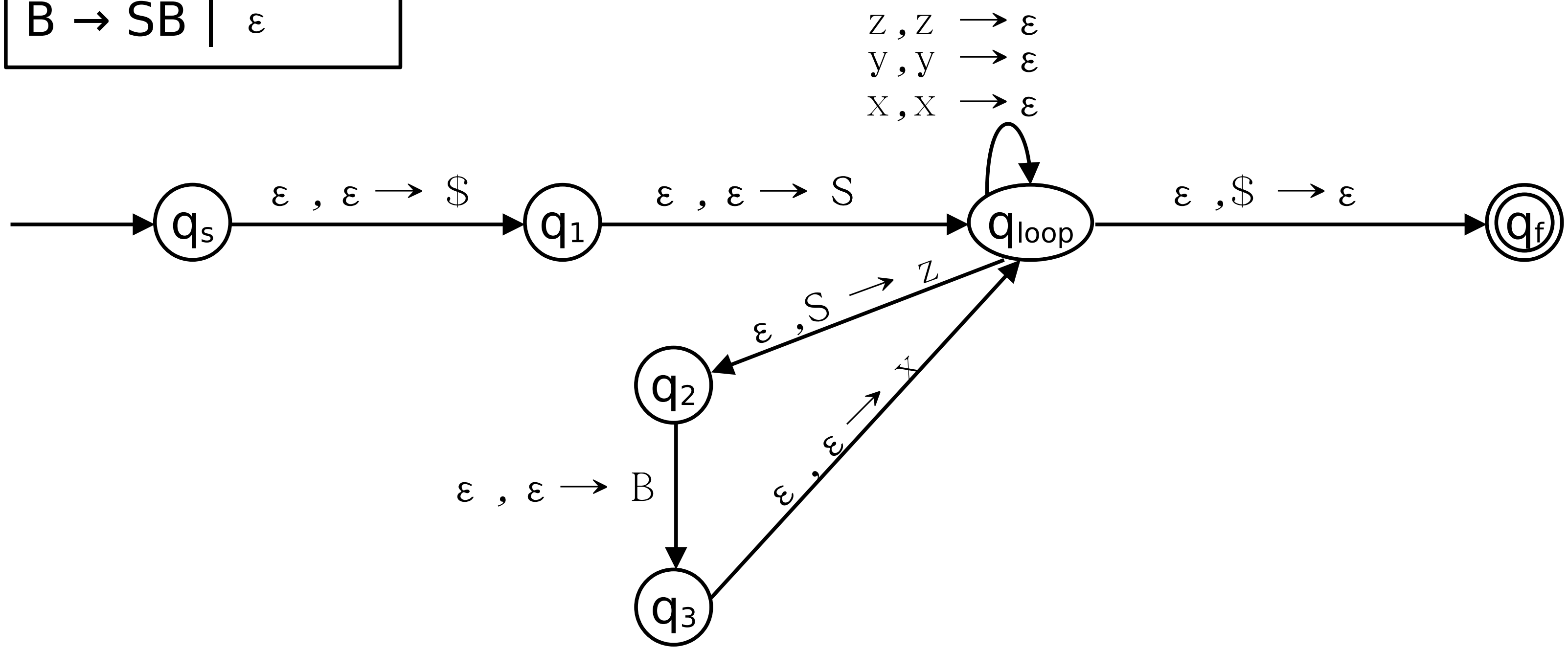
$S \to xBz \mid xy$
$B \to SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

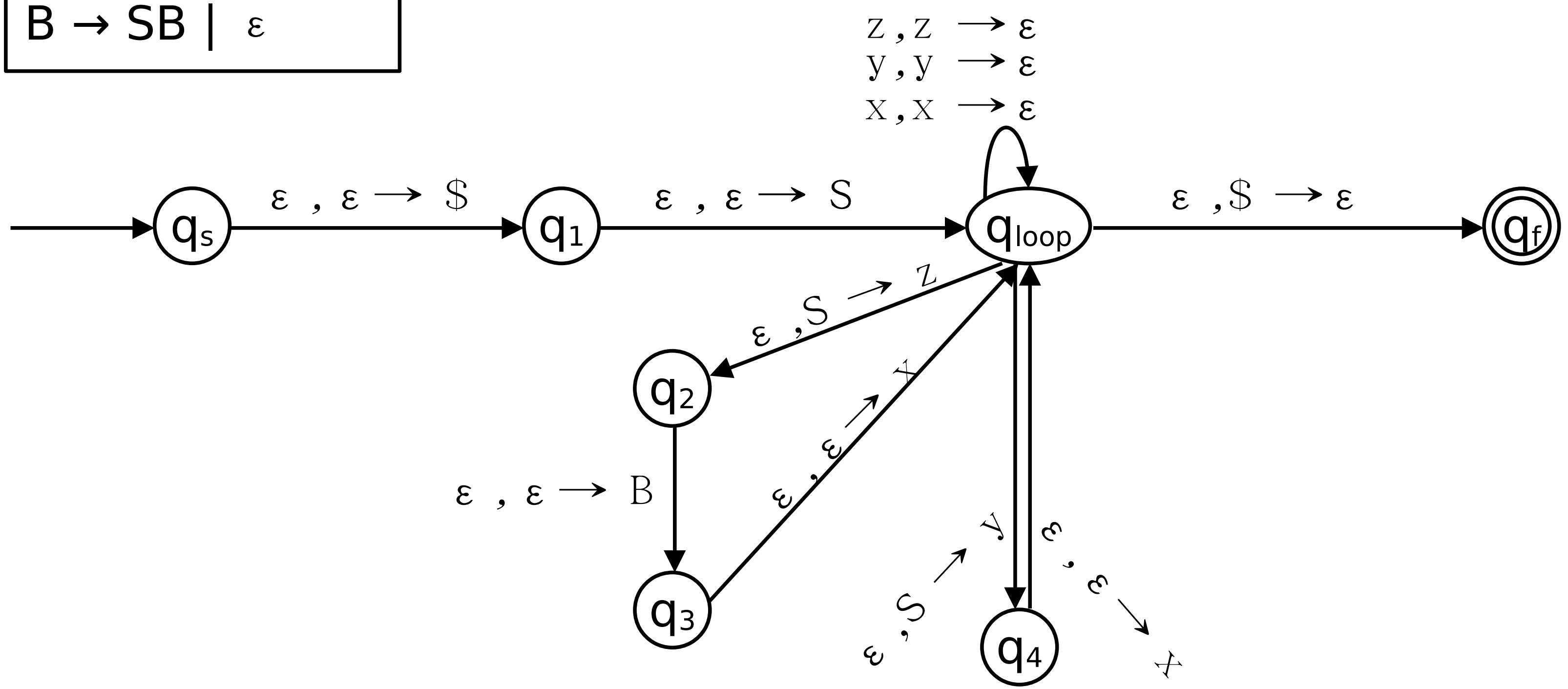
$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

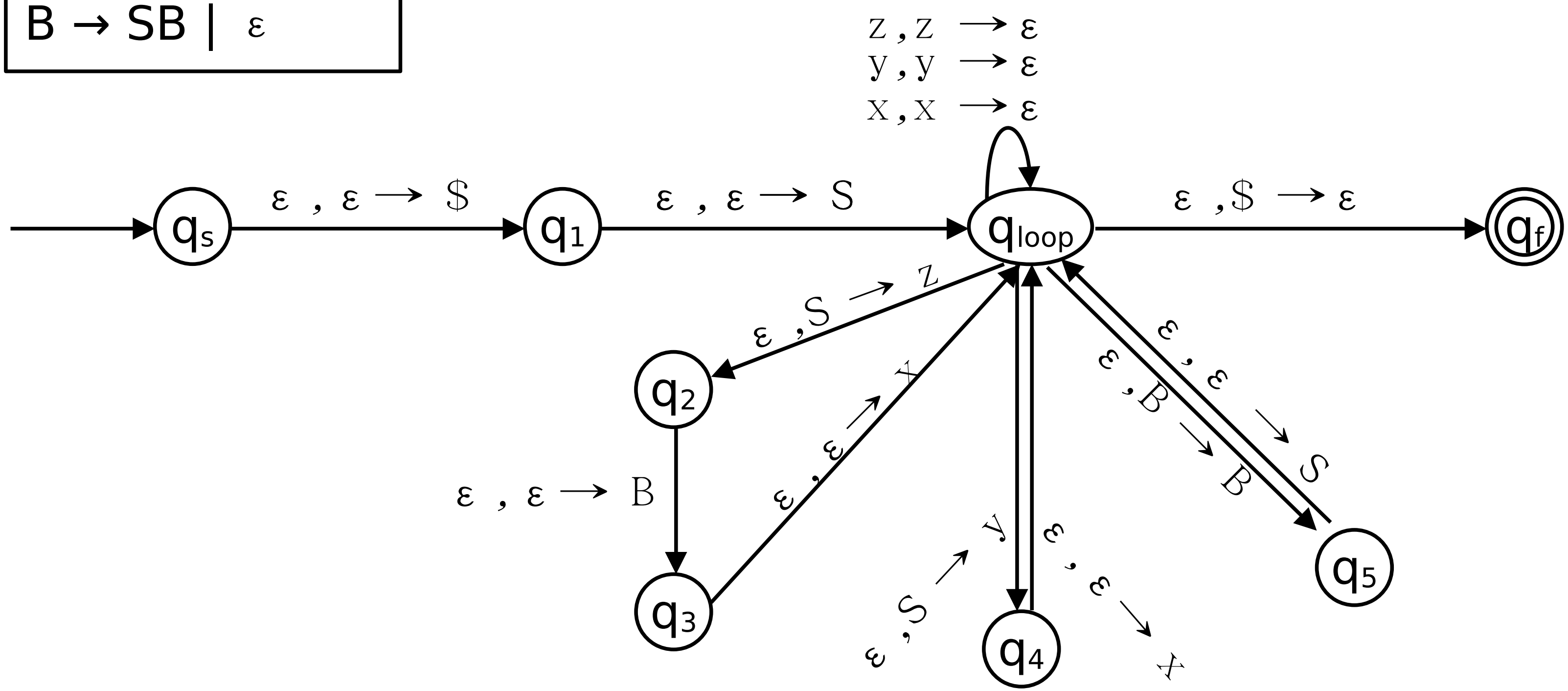
$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$



Pushdown Automata (PDA): CFG → PDA

G_1

$S \rightarrow xBz \mid xy$
$B \rightarrow SB \mid \varepsilon$