CS420

Introduction to the Theory of Computation

Lecture 18: PDA $\Longleftrightarrow$ CFG

Tiago Cogumbreiro

Today we will learn...

- Exercises on designing a PDA
- Convert a PDA into a CFG
- Convert a CFG into a PDA

> Section 2.2
> Supplementary material: Professor David Chiang's lecture notes [1] [2]; Professor Siu On Chan slides

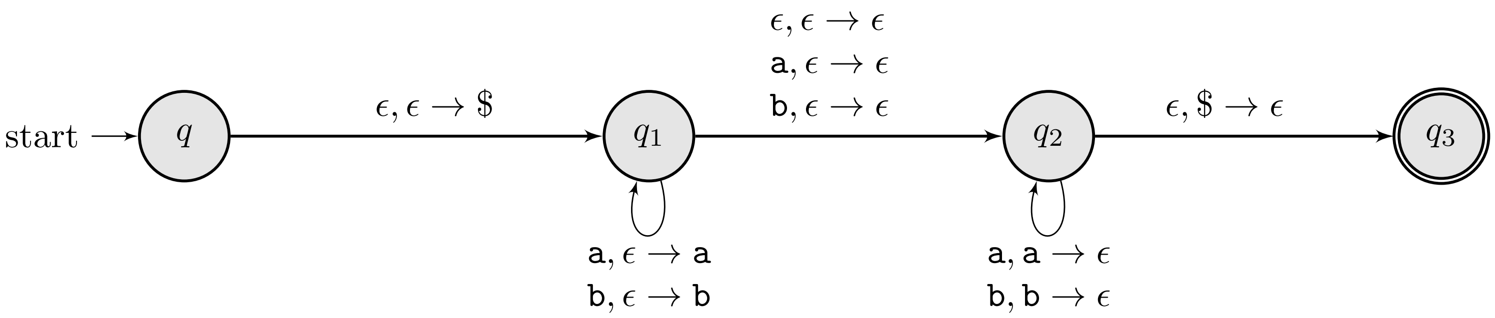
Exercise 1

Exercise 1

1. aa is a palindrome
2. aba is a palindrome
3. bbb is a palindrome
4. ϵ is a palindrome
5. a is a palindrome

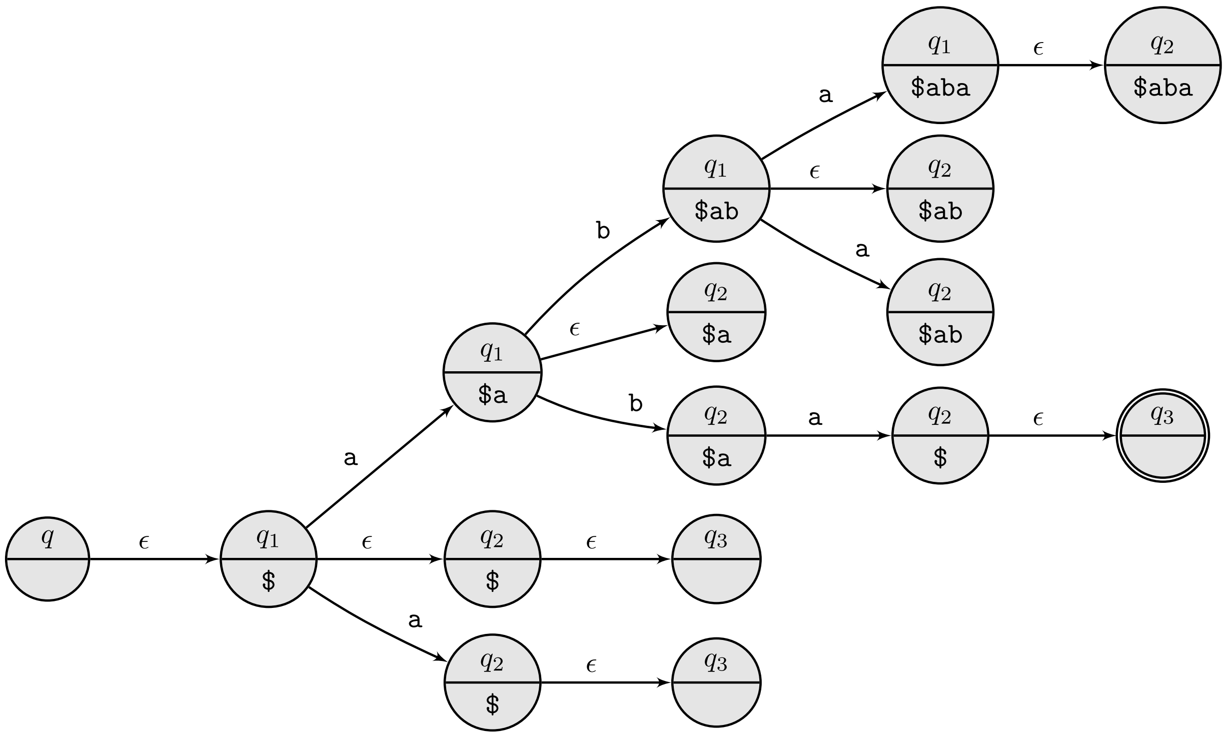
Give a PDA that recognizes palindromes and show it accepts aba and rejects abb

Exercise palindrome



Accepts aba

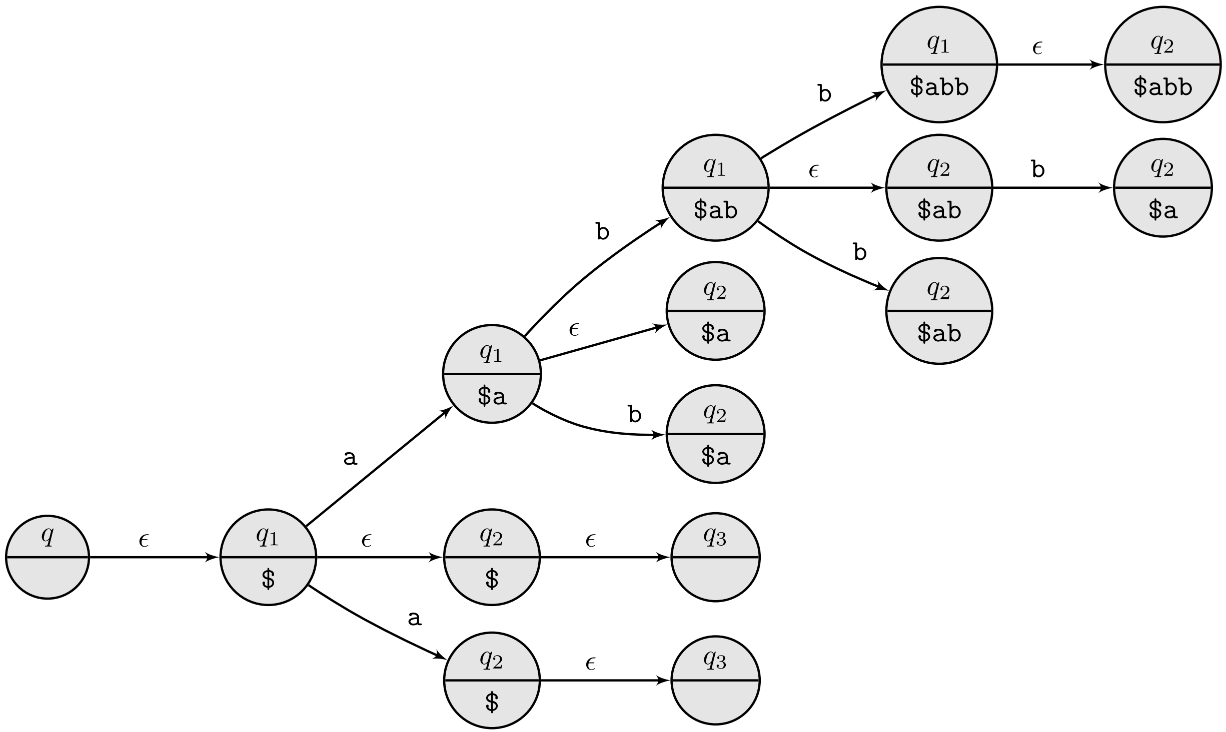
Accepts aba



q
ϵ
q_1
$
ϵ
q_2
$
ϵ
q_3
a
q_2
$
ϵ
q_3
a
q_1
$a
ϵ
q_2
$a
b
q_2
$a
a
q_2
$
ϵ
q_3
b
q_1
$ab
a
q_1
$aba
ϵ
q_2
$aba
ϵ
q_2
$ab
a
q_2
$ab

Rejects abb

Rejects abb



q
ϵ
q1
$
a
ϵ
a
q1
$a
ϵ
b
q2
$
ϵ
q3
q2
$
ϵ
q3
q1
$ab
q2
$a
q2
$a
b
ϵ
b
q1
$abb
q2
$ab
q2
$ab
ϵ
b
q2
$abb
q2
$a

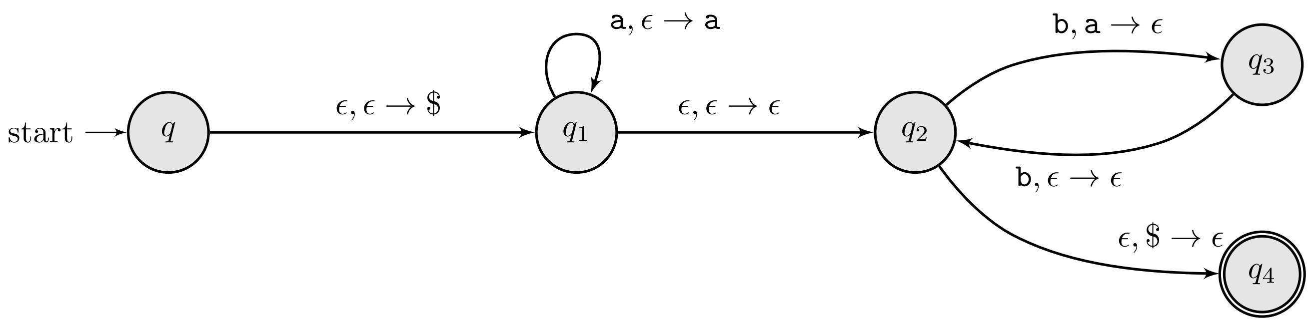
Exercise 2

Exercise 2

$L_2 = \{a^n b^{2n} \mid n \geq 0\}$

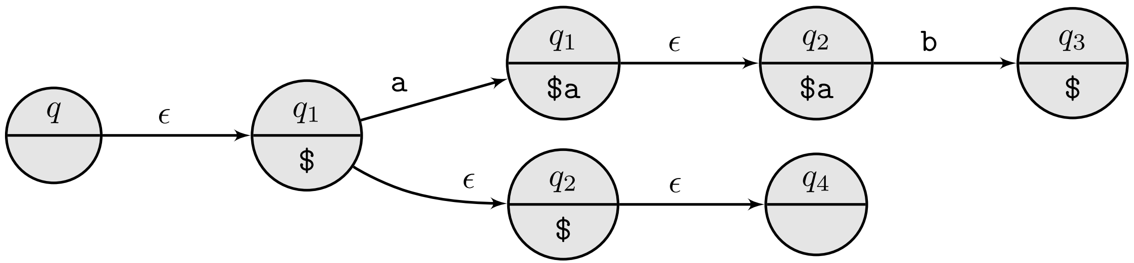
Give a PDA that recognizes $\boldsymbol{L_2}$ and show it rejects aba and accepts abb

Exercise 2 solution



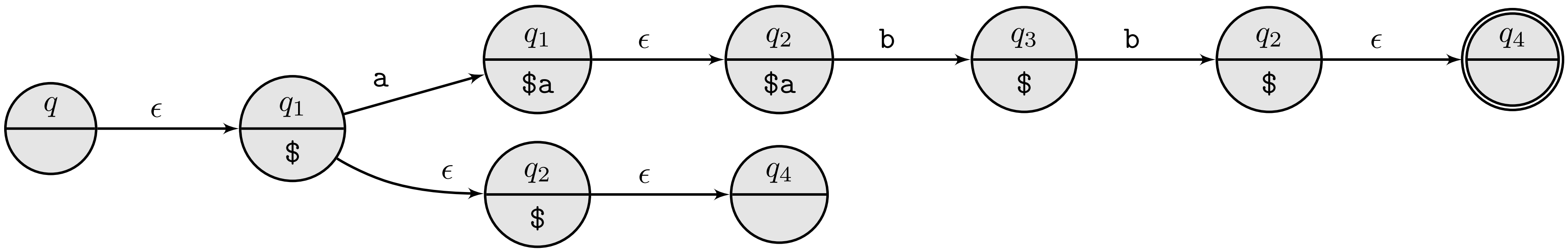
L_2 does not contain aba

L_2 does not contain aba



L_2 contains abb

L_2 contains abb



Context Free Languages

Main result

Context free languages

Theorem: Language L has a context free grammar if, and only if, L is recognized by some pushdown automaton.

Next

1. We show that from a CFG we can build an equivalent† PDA
2. We show that from a PDA we can build an equivalent† CFG

---

† Equivalence with respect to recognized languages. Let P be a PDA and C a CFG we say that P is equivalent to C (and vice versa) if, and only if, $L(P) = L(C)$

Converting a CFG into a PDA

Converting a CFG into a PDA

- (0) Push the sentinel \$ to the stack
- (1) Push the initial variable S to the stack
- In a loop:
  - (2) Every rule $S \to w$ corresponds to poping S and pushing w (in reverse)
  - (3) Pop terminals from stack
  - (4) Empty stack means recognized

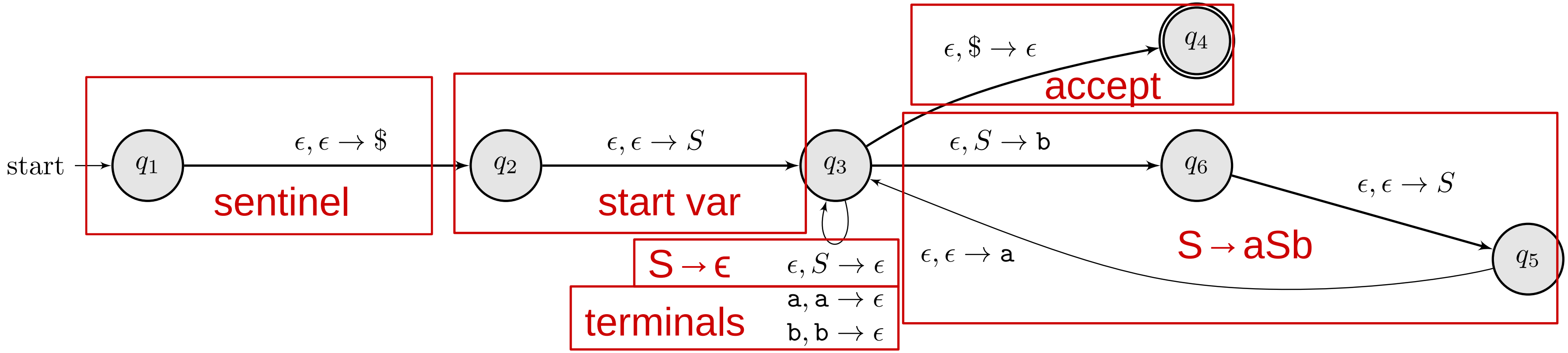
Example $L_3 = \{a^n b^n \mid n \geq 0\}$

$$S \to aSb \mid \epsilon$$

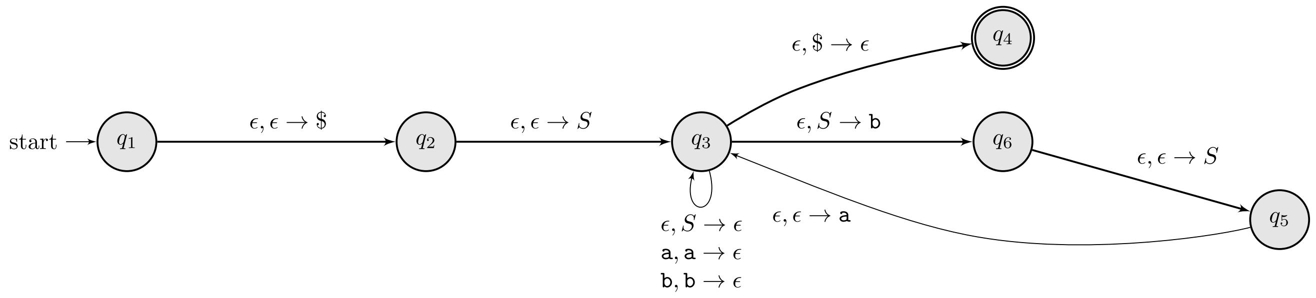
PDA operation	Output	Accept?
(0) $\epsilon, \epsilon \to \$$	\$	
(1) $\epsilon, \epsilon \to S$	S\$	
(2) $\epsilon, S \to aSb$	aSb\$	
(3) $\epsilon, a \to \epsilon$	Sb\$	a
(2) $\epsilon, S \to aSb$	aSbb\$	a
(3) $\epsilon, a \to \epsilon$	Sbb\$	aa
(2) $\epsilon, S \to \epsilon$	bb\$	aa
(3) $\epsilon, b \to \epsilon$	b\$	aab
(3) $\epsilon, b \to \epsilon$	\$	aabb

Overview

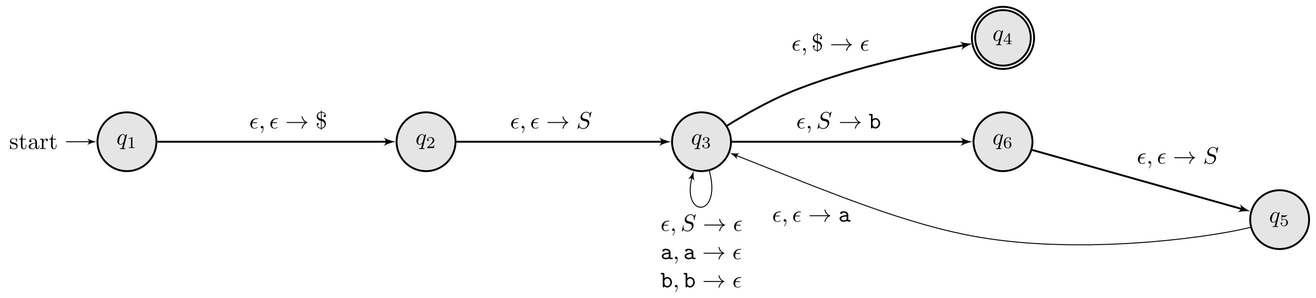
1. **Initial variable:** From the initial state q_1 push the initial variable onto the stack via ϵ and move to the loop state (q_2)
2. **Productions:** For each rule ($S \to aSb$), perform a multi-push edge via ϵ from q_2 back to q_2, by popping popping the variable of the rule S and performing a multi-push of the body aSb.
3. **Alphabet:** For each letter a of the grammar draw a self loop to q_2 that reads a and pops a from the stack
4. **Final transition:** Once the stack is empty transition to the final state q_3 via ϵ

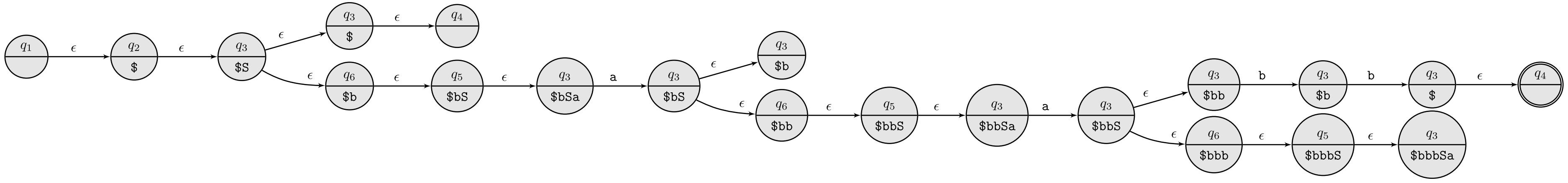


aabb is in $L_3 = \{a^n b^n \mid n \geq 0\}$, show acceptance



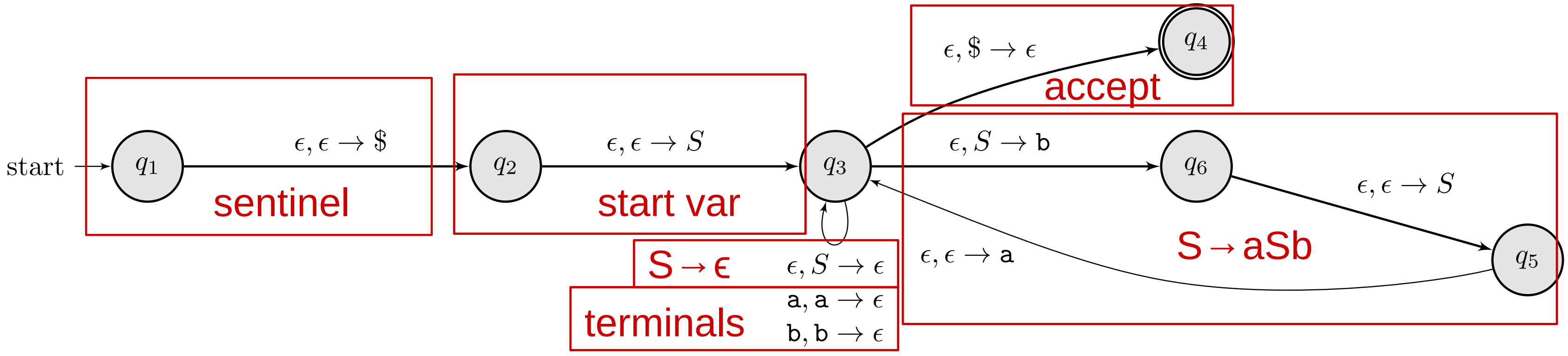
aabb is in $L_3 = \{a^n b^n \mid n \geq 0\}$, show acceptance





Overview

1. The states q_1, q_2, q_3, q_4 are always in the converted PDA
2. States q_1 and q_2 push the sentinel and initial variable
3. The edge between q_3 and q_4 is always $\epsilon, \$ \to \epsilon$
4. There is always a self loop for each letter in the alphabet of $a, a \to \epsilon$
5. The only difficulty is **generating the substitution rules**



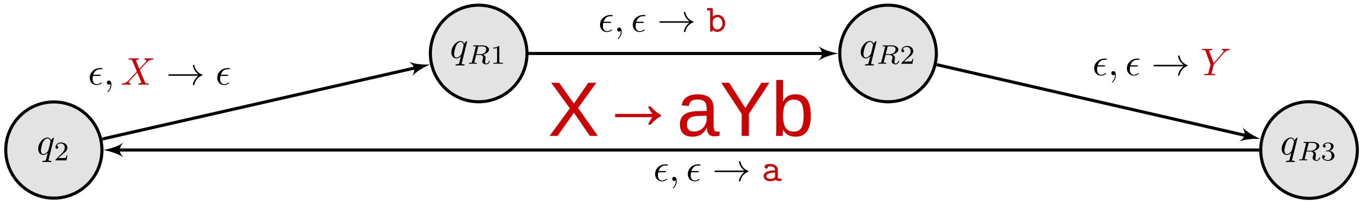
How to encode $\boldsymbol{S} \to \boldsymbol{aSb}$?

(multi push)

Encoding multi-push productions

By example $X \to aYb$

1. reverse the production, example:
   $X \to aYb$ yields bYa.
2. Create one state R_i for each variable/terminal in the reversed string, each transition pushes a variable/terminal of the **reversed** string



> **Note:** In the book (and in my diagrams) I merge the first two transitions. This is equivalent to the above method; you can use either, as long as you do it correctly.

Exercise 3

Exercise 3

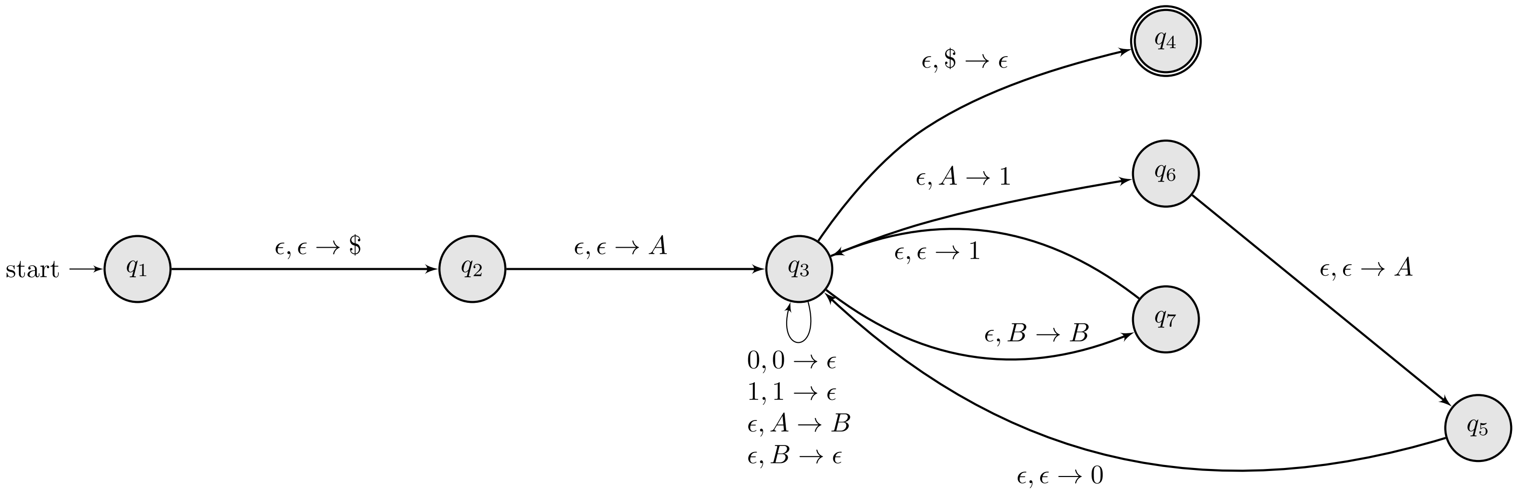
Convert the following grammar into a PDA

$$A \to 0A1 \mid B$$
$$B \to 1B \mid \epsilon$$

Exercise 3

Convert the following grammar into a PDA

$$A \to 0A1 \mid B$$
$$B \to 1B \mid \epsilon$$



Converting a PDA into a CFG

Converting a PDA into a CFG

1. modify the PDA into a **simplified** PDA:
   - has a single accepting state
   - empties the stack before accepting
   - every transition is in one of these forms:
     - skips popping and pushes one symbol onto the stack: $\epsilon \to c$
     - pops one symbol off the stack and skips pushing: $c \to \epsilon$

Converting a PDA into a CFG

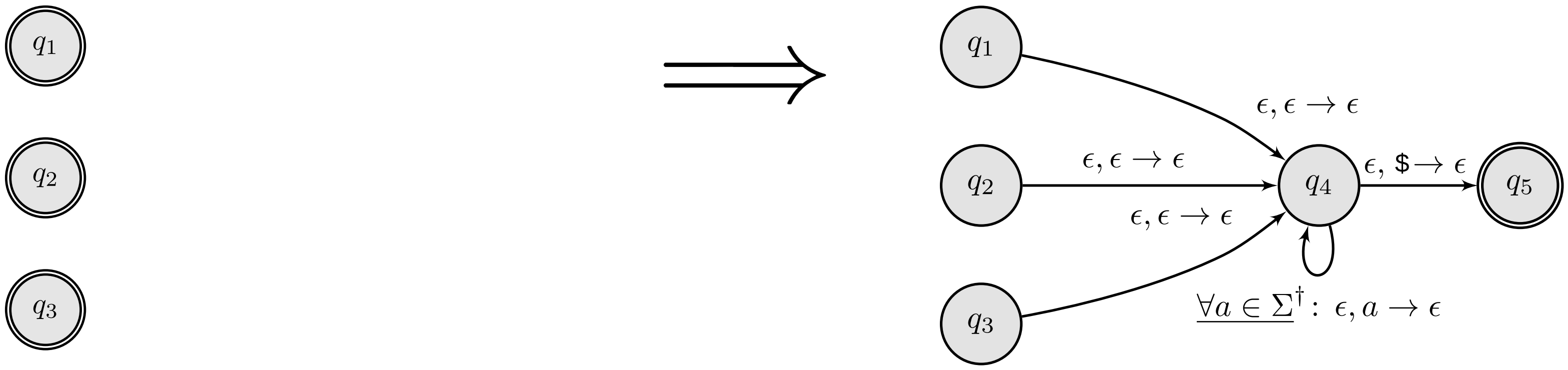
1. modify the PDA into a **simplified** PDA:
   - has a single accepting state
   - empties the stack before accepting
   - every transition is in one of these forms:
     - skips popping and pushes one symbol onto the stack: $\epsilon \to c$
     - pops one symbol off the stack and skips pushing: $c \to \epsilon$
2. given a simplified PDA build a CFG
   - $A_{qq} \to \epsilon$ if $q \in Q$
   - $A_{pq} \to A_{pr} A_{rq}$ if $p, q \in Q$
   - $A_{\underline{pq}} \to \mathtt{a} A_{rs} \mathtt{b}$ if $(r, \mathtt{u}) \in \delta(\underline{p}, \mathtt{a}, \epsilon)$ and $(\underline{q}, \epsilon) \in \delta(s, \mathtt{b}, \mathtt{u})$

Simplifying a PDA

Simplifying a PDA

Transformation 1: Has a single accepting state

Transformation 2: Empties the stack before accepting



† Notation $\underline{\forall a \in \Sigma}$ means that there will be one edge $\epsilon, a \to \epsilon$ per $a \in \Sigma$

Simplifying a PDA

Transformation 3

Every transition is in one of these forms:

- skips popping and pushes one symbol onto the stack: $\epsilon \to c$
- pops one symbol off the stack and skips pushing: $c \to \epsilon$

Case 1

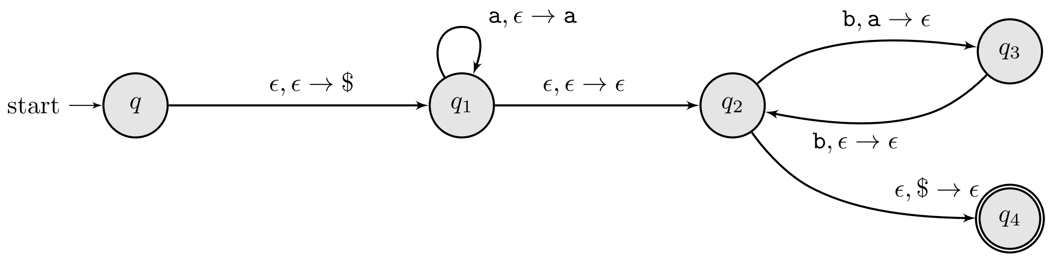
Case 2

Example 4

Simplified PDA

- single accepting state
- empties the stack before accepting
- every transition is in one of these forms:
  - $\epsilon \to c$
  - $c \to \epsilon$

Is it simplified?

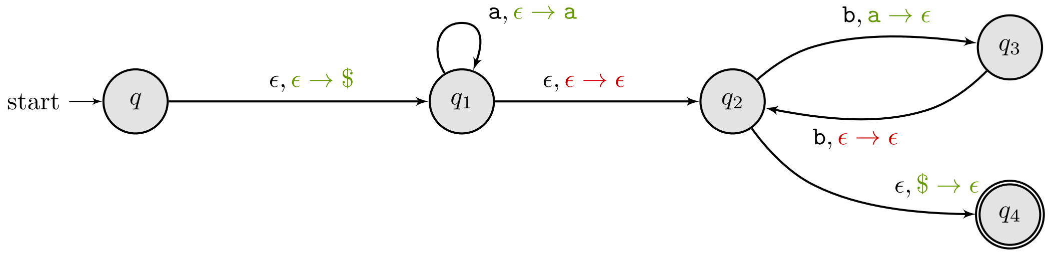


Example 4

Simplified PDA

- single accepting state
- empties the stack before accepting
- every transition is in one of these forms:
  - $\epsilon \to c$
  - $c \to \epsilon$

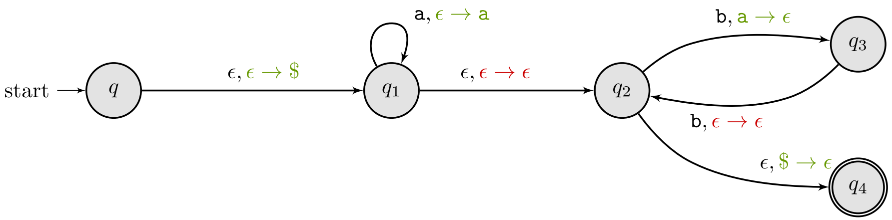
Is it simplified?



No!

Example 4

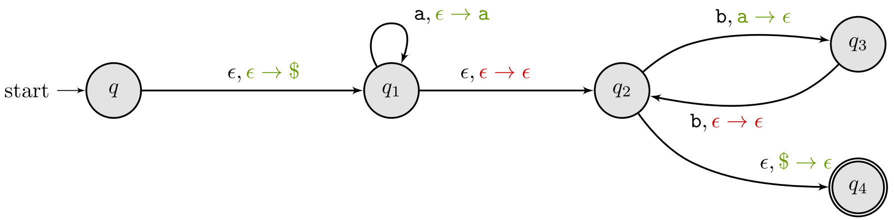
Not Simplified



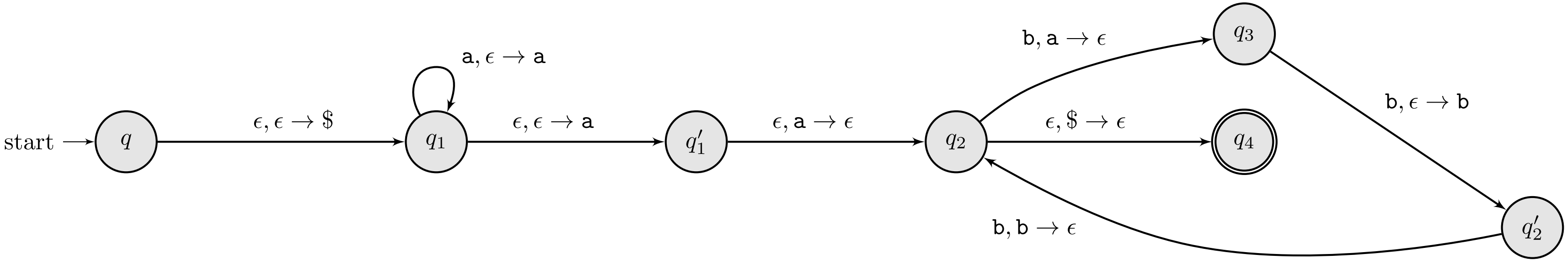
Simplified

Example 4

Not Simplified



Simplified



Simplified PDA to CFG

Simplified PDA to CFG

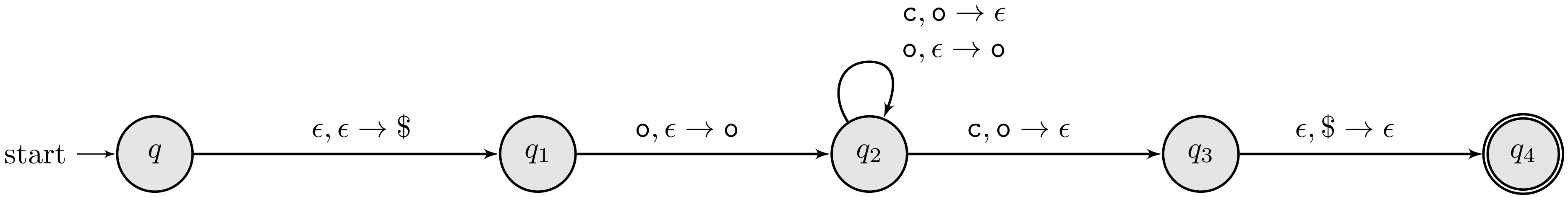
Given a simplified PDA build a CFG

1. $A_{qq} \to \epsilon$ if $q \in Q$
2. $A_{pq} \to A_{pr} A_{rq}$ if $p, r, q \in Q$
3. $A_{\underline{pq}} \to \mathtt{a} A_{rs} \mathtt{b}$ if $(r, \mathtt{u}) \in \delta(\underline{p}, \mathtt{a}, \epsilon)$ and $(\underline{q}, \epsilon) \in \delta(s, \mathtt{b}, \mathtt{u})$


```
for p=={a, ϵ -> u}==>r in transitions:     # for every transition
    for s=={b, u -> ϵ}==>q in transitions: # for every transition
        yield A_pq -> a A_rs b
```

Example 5

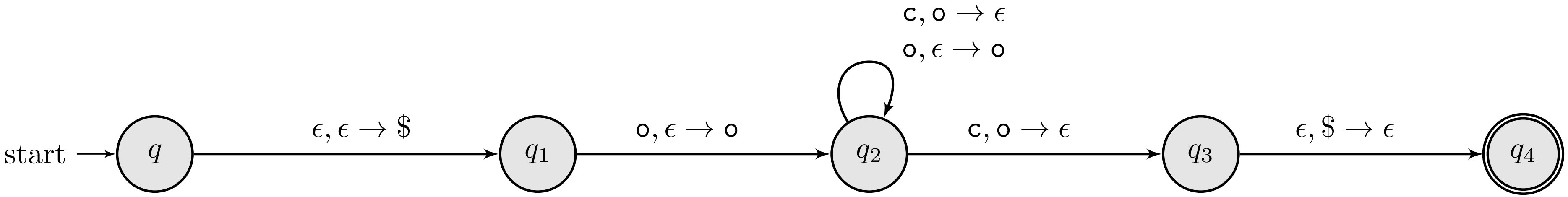
- Balanced parenthesis that are wrapped inside an outermost parenthesis.



Is this PDA simplified?

Example 5

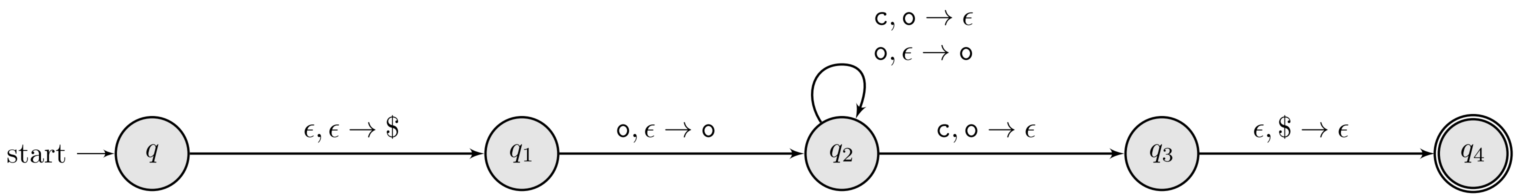
Balanced parenthesis that are wrapped inside an outermost parenthesis.



Is this PDA simplified?

Yes!

Example 5

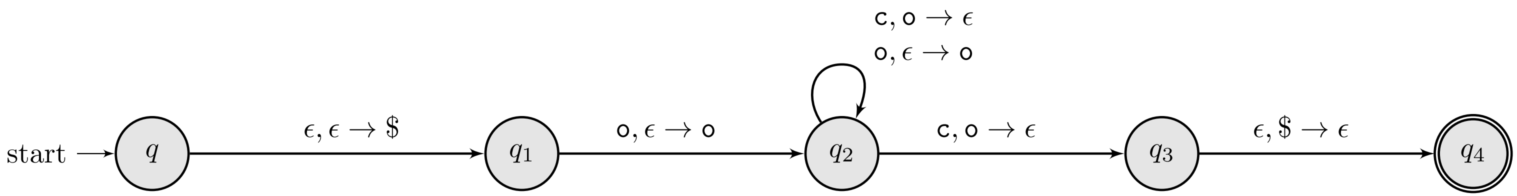


Step 1: $A_{qq} \to \epsilon$ if $q \in Q$

Step 2: $A_{pq} \to A_{pr} A_{rq}$ if $p, r, q \in Q$

$A_{11} \to \epsilon$	$A_{1,3} \to A_{1,2} A_{2,3}$	$A_{2,3} \to A_{2,1} A_{1,3}$	$A_{3,2} \to A_{3,1} A_{1,2}$
$A_{22} \to \epsilon$	$A_{1,3} \to A_{1,4} A_{4,3}$	$A_{2,3} \to A_{2,4} A_{4,3}$	$A_{3,2} \to A_{3,4} A_{4,2}$
$A_{33} \to \epsilon$	$A_{1,4} \to A_{1,2} A_{2,4}$	$A_{2,4} \to A_{2,1} A_{1,4}$	$A_{3,4} \to A_{3,1} A_{1,4}$
$A_{44} \to \epsilon$	$A_{1,4} \to A_{1,3} A_{3,4}$	$A_{2,4} \to A_{2,3} A_{3,4}$	$A_{3,4} \to A_{3,2} A_{2,4}$
$A_{1,2} \to A_{1,3} A_{3,2}$	$A_{2,1} \to A_{2,3} A_{3,1}$	$A_{3,1} \to A_{3,2} A_{2,1}$	$A_{4,1} \to A_{4,2} A_{2,1}$
$A_{1,2} \to A_{1,4} A_{4,2}$	$A_{2,1} \to A_{2,4} A_{4,1}$	$A_{3,1} \to A_{3,2} A_{2,1}$	$A_{4,1} \to A_{4,3} A_{3,1}$

Example 5



Step 1: $A_{qq} \to \epsilon$ if $q \in Q$

Step 2: $A_{pq} \to A_{pr} A_{rq}$ if $p, r, q \in Q$

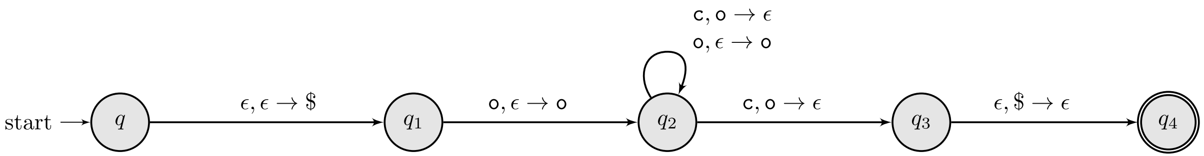
$$A_{4,2} \to A_{4,1} A_{1,2}$$

$$A_{4,2} \to A_{4,3} A_{3,2}$$

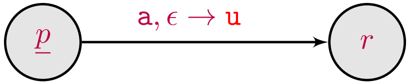
$$A_{4,3} \to A_{4,1} A_{1,3}$$

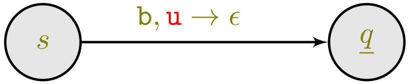
$$A_{4,3} \to A_{4,2} A_{2,3}$$

Example 5



Step 3: $A_{\underline{pq}} \to \mathtt{a}A_{rs}\mathtt{b}$ if $(r, \mathtt{u}) \in \delta(\underline{p}, \mathtt{a}, \epsilon)$ and $(\underline{q}, \epsilon) \in \delta(s, \mathtt{b}, \mathtt{u})$





Stack o

Push	Pop
`q1, read o, q2`	
`q2, read o, q2`	
	`q2, read c, q2`
	`q2, read c, q3`

New rules:

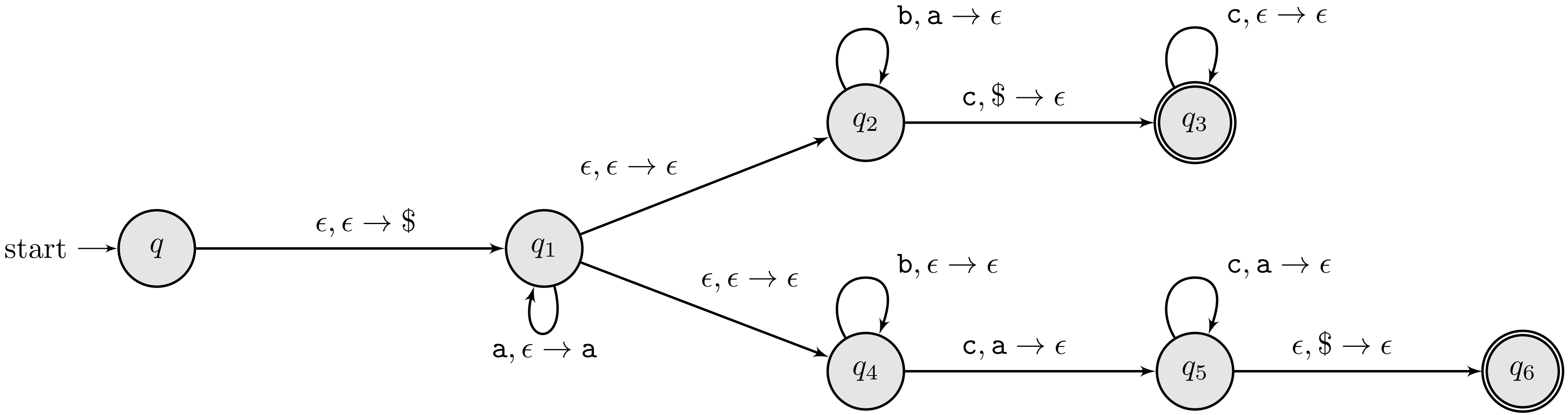
$$A_{1,2} \to oA_{22}c$$
$$A_{1,3} \to oA_{22}c$$
$$A_{2,2} \to oA_{22}c$$
$$A_{2,3} \to oA_{22}c$$

Intuition

- Create a table for each letter being pushed/poped.
- Pair each push with each pop.

Exercise 6

Simplify the PDA below



Exercise 6

Solution