CS420

Introduction to the Theory of Computation

Lecture 9: Push-down automata

Tiago Cogumbreiro

HW4 Errata

Exercise 3

The grammar should be:

$$
\begin{aligned}
C &\to BbA \\
A &\to BAC \mid \epsilon \\
B &\to \epsilon \mid AbA
\end{aligned}
$$

Please update your copy of the PDF

Today we will learn...

- Pushdown automata (PDA)
- Formalizing PDAs
- Union of PDAs
- Examples

> Section 2.2

Intuition

Define an automata family $\iff$ CFG

NFA recap

Each transition performs one input operations: read/skip an input

Examples

- **Read one input:** $q_1 \xrightarrow{\texttt{a}} q_2$
- **Skip one input:** $q_1 \xrightarrow{\epsilon} q_2$

Nondeterministic PushDown Automata (PDA)

- Extend NFAs with an *unbounded* **stack**
- Recognizes the same language as CFGs

PDA Execution

Each transition:

- input op, pre-stack op, post-stack op
- Format: $q \xrightarrow{\texttt{\$INPUT},\texttt{\$PRE}\to\texttt{\$POST}} q'$

Possible operations

$INPUT	*$PRE*	*$POST*
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP
	EMPTY?	CLEAR

Example

$$q_{\mathtt{a}} \xrightarrow{\texttt{READ a},\texttt{SKIP}\to\texttt{PUSH a}} q_{\mathtt{a}}$$

Nondeterministic PushDown Automata (PDA)

- Extend NFAs with an *unbounded* **stack**
- Recognizes the same language as CFGs

PDA Execution

Each transition:

- input op, pre-stack op, post-stack op
- Format: $q \xrightarrow{\texttt{\$INPUT,\$PRE}\to\texttt{\$POST}} q'$

Example

$q_{\mathtt{a}} \xrightarrow{\texttt{READ a,SKIP}\to\texttt{PUSH a}} q_{\mathtt{a}}$

Possible operations

$INPUT	*$PRE*	*$POST*
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP
	EMPTY?	CLEAR

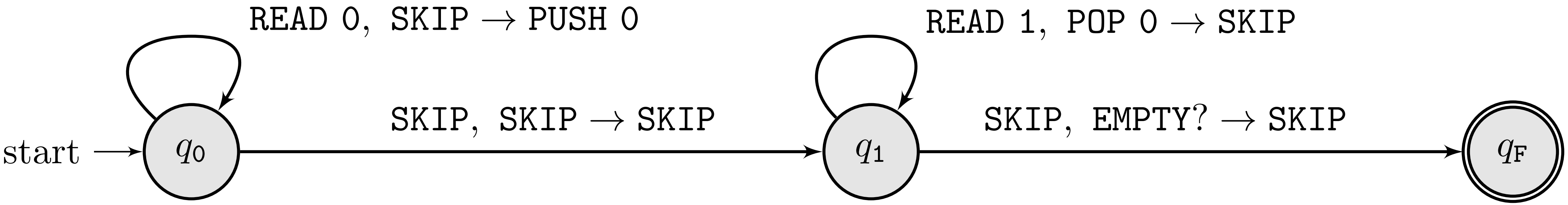
Attention!

> The comma does not denote parallel edges.
> Instead, we stack multiple transitions **vertically**.

PDA example (intuition)

Give a PDA that recognizes $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$

1. $q_{\mathtt{a}} \xrightarrow{\mathtt{READ\ a,SKIP} \to \mathtt{PUSH\ a}} q_{\mathtt{a}}$
2. $q_{\mathtt{a}} \xrightarrow{\mathtt{SKIP,SKIP} \to \mathtt{SKIP}} q_{\mathtt{b}}$
3. $q_{\mathtt{b}} \xrightarrow{\mathtt{READ\ b,POP\ a} \to \mathtt{SKIP}} q_{\mathtt{b}}$
4. $q_{\mathtt{b}} \xrightarrow{\mathtt{SKIP,EMPTY?} \to \mathtt{SKIP}} q_{\mathtt{y}}$



Writing transitions

Possible operations

$INPUT	$PRE	$POST
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP
	EMPTY?	CLEAR

Exercises

1. Pop 0 and clear the stack:

Writing transitions

Possible operations

$INPUT	*$PRE*	*$POST*
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP
	EMPTY?	CLEAR

Exercises

1. Pop 0 and clear the stack: `SKIP`, `POP 0` $\rightarrow$ `CLEAR`
2. Check if read 0 and stack is empty (two solutions):

Writing transitions

Possible operations

\$INPUT	\$PRE	\$POST
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP
	EMPTY?	CLEAR

Exercises

1. Pop 0 and clear the stack: $\texttt{SKIP}, \texttt{POP 0} \to \texttt{CLEAR}$
2. Check if read 0 and stack is empty (two solutions): $\texttt{READ 0}, \texttt{EMPTY?} \to \texttt{SKIP}$
   $\texttt{READ 0}, \texttt{EMPTY?} \to \texttt{CLEAR}$
3. Check if stack is empty (2 solutions):

Writing transitions

Possible operations

$INPUT	*$PRE*	*$POST*
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP
	EMPTY?	CLEAR

Exercises

1. Pop 0 and clear the stack: $\mathtt{SKIP}, \mathtt{POP\ 0} \to \mathtt{CLEAR}$
2. Check if read 0 and stack is empty (two solutions): $\mathtt{READ\ 0}, \mathtt{EMPTY?} \to \mathtt{SKIP}$
   $\mathtt{READ\ 0}, \mathtt{EMPTY?} \to \mathtt{CLEAR}$
3. Check if stack is empty (2 solutions):
   $\mathtt{SKIP}, \mathtt{EMPTY?} \to \mathtt{CLEAR}$
   $\mathtt{SKIP}, \mathtt{EMPTY?} \to \mathtt{SKIP}$
4. Check if 0 is on top and leave stack untouched:

Writing transitions

Possible operations

\$INPUT	\$PRE	\$POST
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP
	EMPTY?	CLEAR

Exercises

1. Pop 0 and clear the stack: $\texttt{SKIP}, \texttt{POP 0} \to \texttt{CLEAR}$
2. Check if read 0 and stack is empty (two solutions): $\texttt{READ 0}, \texttt{EMPTY?} \to \texttt{SKIP}$
   $\texttt{READ 0}, \texttt{EMPTY?} \to \texttt{CLEAR}$
3. Check if stack is empty (2 solutions):
   $\texttt{SKIP}, \texttt{EMPTY?} \to \texttt{CLEAR}$
   $\texttt{SKIP}, \texttt{EMPTY?} \to \texttt{SKIP}$
4. Check if 0 is on top and leave stack untouched:
   $\texttt{SKIP}, \texttt{POP 0} \to \texttt{PUSH 0}$
5. Read 2 and leave stack untouched:

Writing transitions

Possible operations

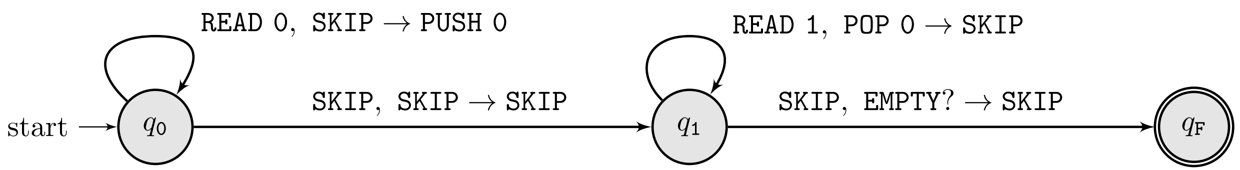
$INPUT	$PRE	$POST
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP
	EMPTY?	CLEAR

Exercises

1. Pop 0 and clear the stack: $\texttt{SKIP}, \texttt{POP 0} \rightarrow \texttt{CLEAR}$
2. Check if read 0 and stack is empty (two solutions): $\texttt{READ 0}, \texttt{EMPTY?} \rightarrow \texttt{SKIP}$
   $\texttt{READ 0}, \texttt{EMPTY?} \rightarrow \texttt{CLEAR}$
3. Check if stack is empty (2 solutions):
   $\texttt{SKIP}, \texttt{EMPTY?} \rightarrow \texttt{CLEAR}$
   $\texttt{SKIP}, \texttt{EMPTY?} \rightarrow \texttt{SKIP}$
4. Check if 0 is on top and leave stack untouched:
   $\texttt{SKIP}, \texttt{POP 0} \rightarrow \texttt{PUSH 0}$
5. Read 2 and leave stack untouched:
   $\texttt{READ 2}, \texttt{SKIP} \rightarrow \texttt{SKIP}$

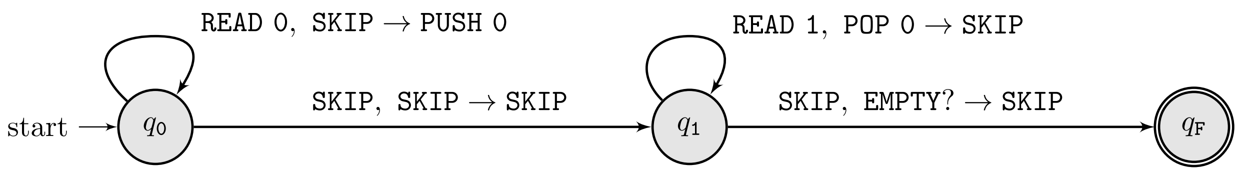
Simplifying the notation

Simplifying the notation

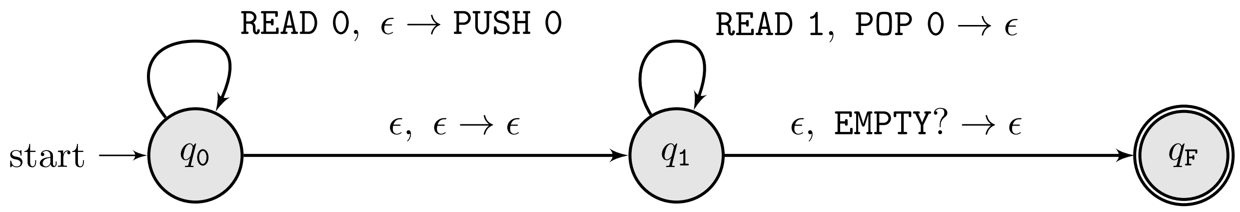


We can replace SKIP by ϵ

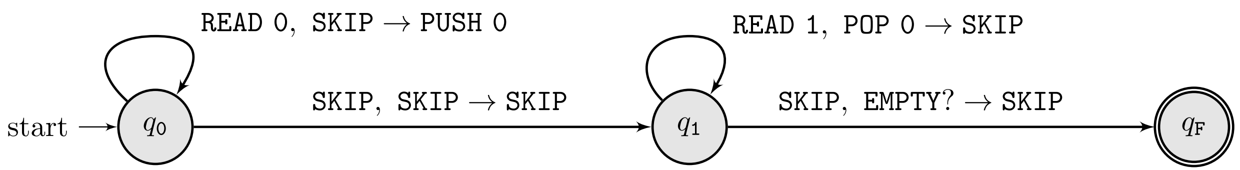
Simplifying the notation



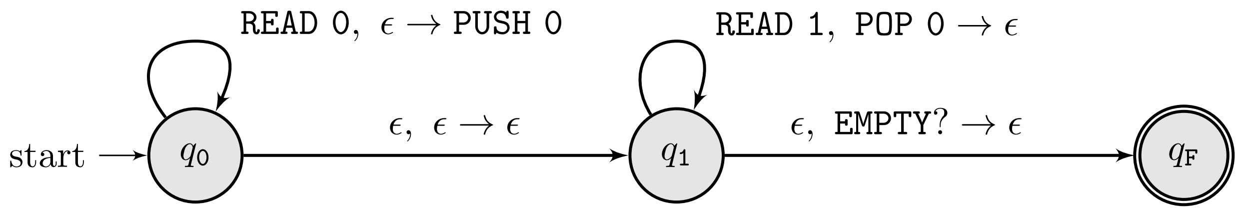
We can replace SKIP by ϵ



Simplifying the notation

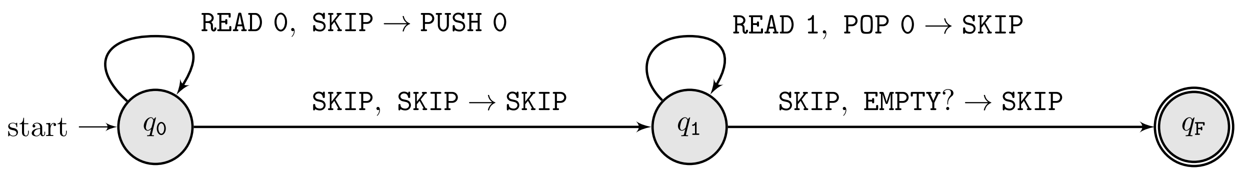


We can replace SKIP by ϵ

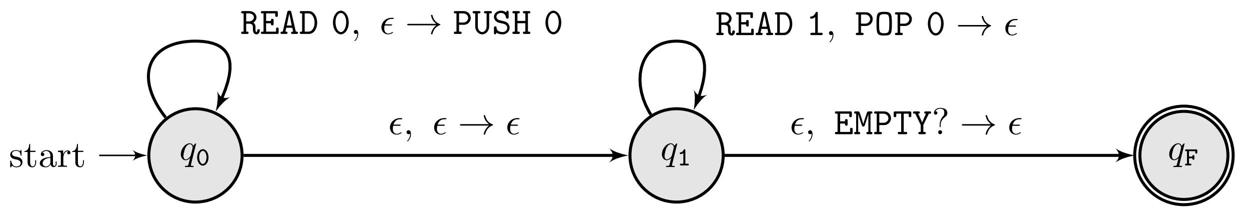


We can omit READ

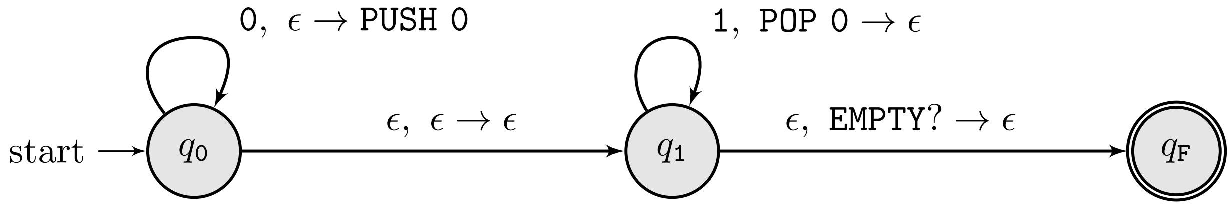
Simplifying the notation



We can replace SKIP by ϵ

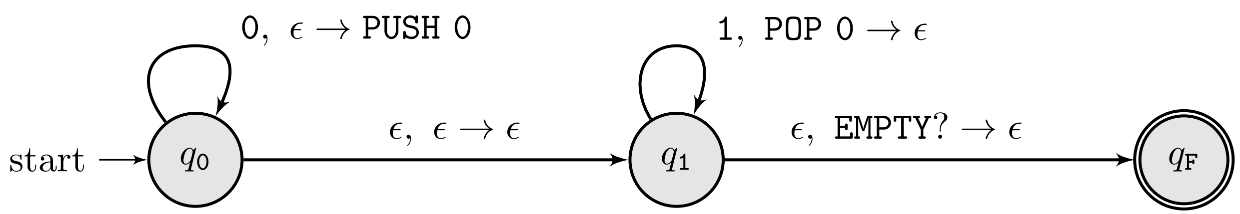


We can omit READ



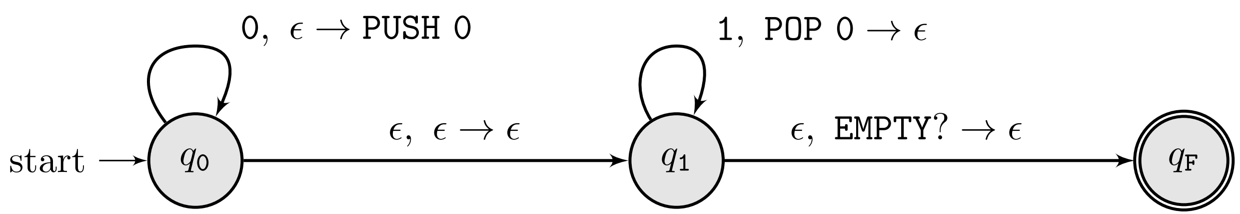
Since read always appears in the same position, we can omit it, as we do in regular DFAs/NFAs.

Simplifying the notation

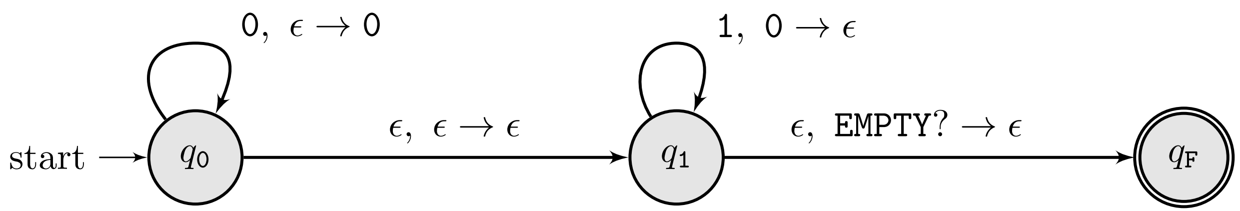


We can omit PUSH/POP

Simplifying the notation



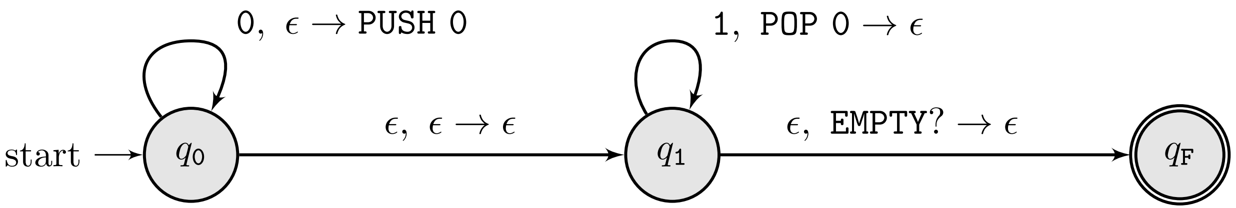
We can omit PUSH/POP



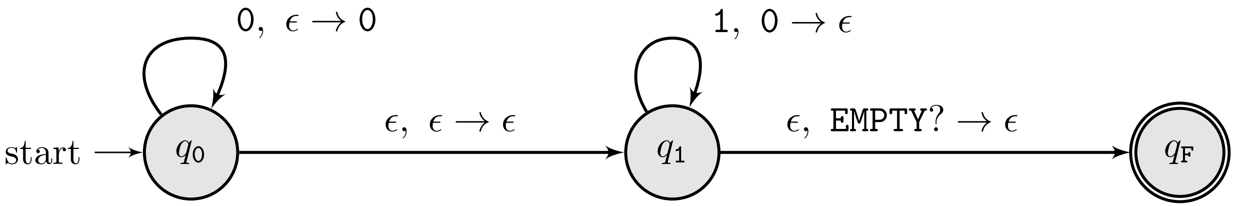
> Since push/pop always appear in the same position, we can omit them.

We can replace EMPTY?/CLEAR by $

Simplifying the notation

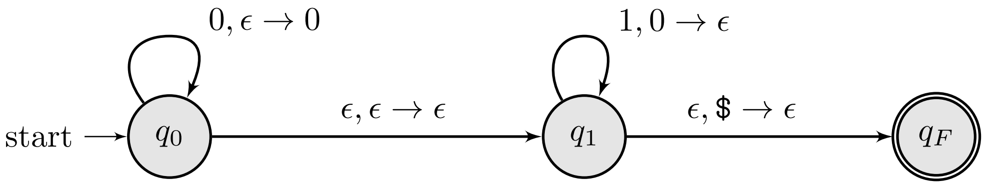


We can omit PUSH/POP



Since push/pop always appear in the same position, we can omit them.

We can replace EMPTY?/CLEAR by $



Since empty?/clear always appear in the same position.

Writing transitions

Possible operations

$INPUT	$PRE	$POST
READ (n)	POP (n)	PUSH (n)
SKIP (ϵ)	SKIP (ϵ)	SKIP (ϵ)
	EMPTY? ($)	CLEAR ($)

Exercises

1. Check if read 0 and stack is empty

Writing transitions

Possible operations

\$INPUT	*\$PRE*	*\$POST*
READ (n)	POP (n)	PUSH (n)
SKIP (ϵ)	SKIP (ϵ)	SKIP (ϵ)
	EMPTY? (\$)	CLEAR (\$)

Exercises

1. Check if read 0 and stack is empty
   $0, \$ \to \epsilon$ or $0, \$ \to \$$
2. Pop 0 and clear the stack

Writing transitions

Possible operations

$INPUT	$PRE	$POST
READ (n)	POP (n)	PUSH (n)
SKIP (ϵ)	SKIP (ϵ)	SKIP (ϵ)
	EMPTY? ($)	CLEAR ($)

Exercises

1. Check if read 0 and stack is empty
   $0, \$ \to \epsilon$ or $0, \$ \to \$$
2. Pop 0 and clear the stack $\epsilon, 0 \to \$$
3. Check if stack is empty (2 solutions)

Writing transitions

Possible operations

$INPUT	$PRE	$POST
READ (n)	POP (n)	PUSH (n)
SKIP (ϵ)	SKIP (ϵ)	SKIP (ϵ)
	EMPTY? ($)	CLEAR ($)

Exercises

1. Check if read 0 and stack is empty
   $0, \$ \rightarrow \epsilon$ or $0, \$ \rightarrow \$$
2. Pop 0 and clear the stack $\epsilon, 0 \rightarrow \$$
3. Check if stack is empty (2 solutions)
   $\epsilon, \$ \rightarrow \$$
   $\epsilon, \$ \rightarrow \epsilon$
4. Check if 0 is on top of the stack and leave stack untouched:

Writing transitions

Possible operations

$INPUT	*$PRE*	*$POST*
READ (n)	POP (n)	PUSH (n)
SKIP (ϵ)	SKIP (ϵ)	SKIP (ϵ)
	EMPTY? ($)	CLEAR ($)

Exercises

1. Check if read 0 and stack is empty
   $0, \$ \to \epsilon$ or $0, \$ \to \$$
2. Pop 0 and clear the stack $\epsilon, 0 \to \$$
3. Check if stack is empty (2 solutions)
   $\epsilon, \$ \to \$$
   $\epsilon, \$ \to \epsilon$
4. Check if 0 is on top of the stack and leave stack untouched: $\epsilon, 0 \to 0$
5. Read 2, leave stack untouched

Writing transitions

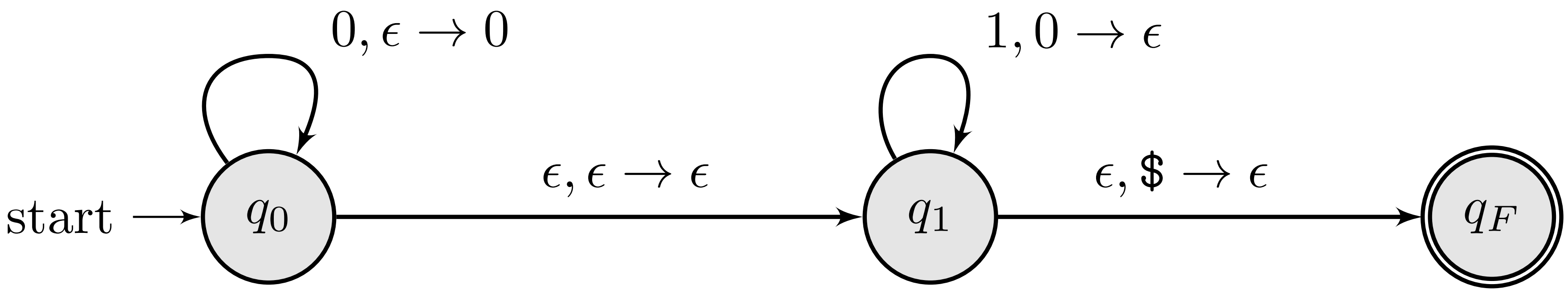
Possible operations

$INPUT	$PRE	$POST
READ (n)	POP (n)	PUSH (n)
SKIP (ϵ)	SKIP (ϵ)	SKIP (ϵ)
	EMPTY? ($)	CLEAR ($)

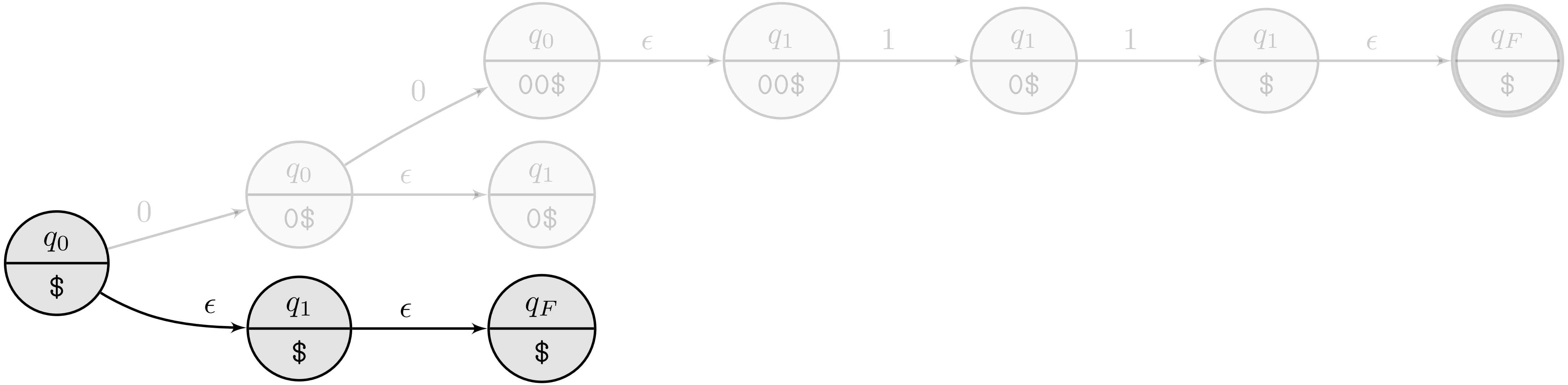
Exercises

1. Check if read 0 and stack is empty
   $0, \$ \to \epsilon$ or $0, \$ \to \$$
2. Pop 0 and clear the stack $\epsilon, 0 \to \$$
3. Check if stack is empty (2 solutions)
   $\epsilon, \$ \to \$$
   $\epsilon, \$ \to \epsilon$
4. Check if 0 is on top of the stack and leave stack untouched: $\epsilon, 0 \to 0$
5. Read 2, leave stack untouched
   $2, \epsilon \to \epsilon$

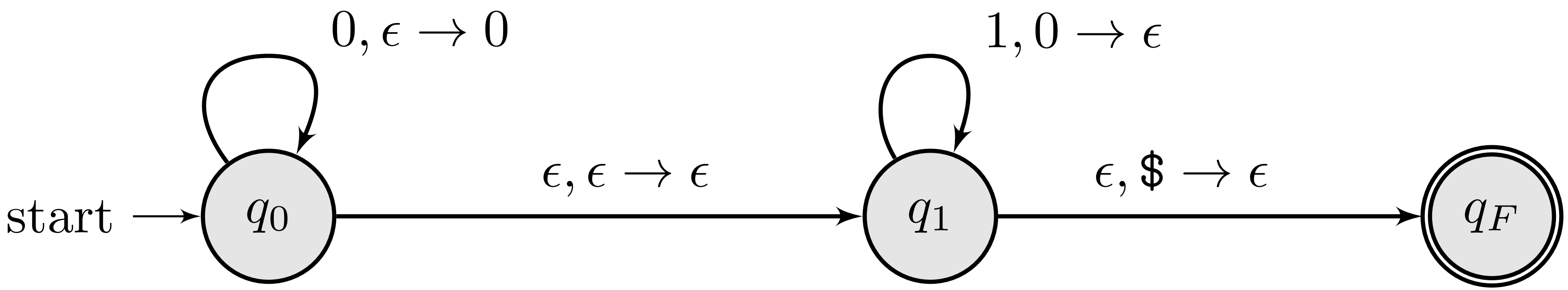
Acceptance example



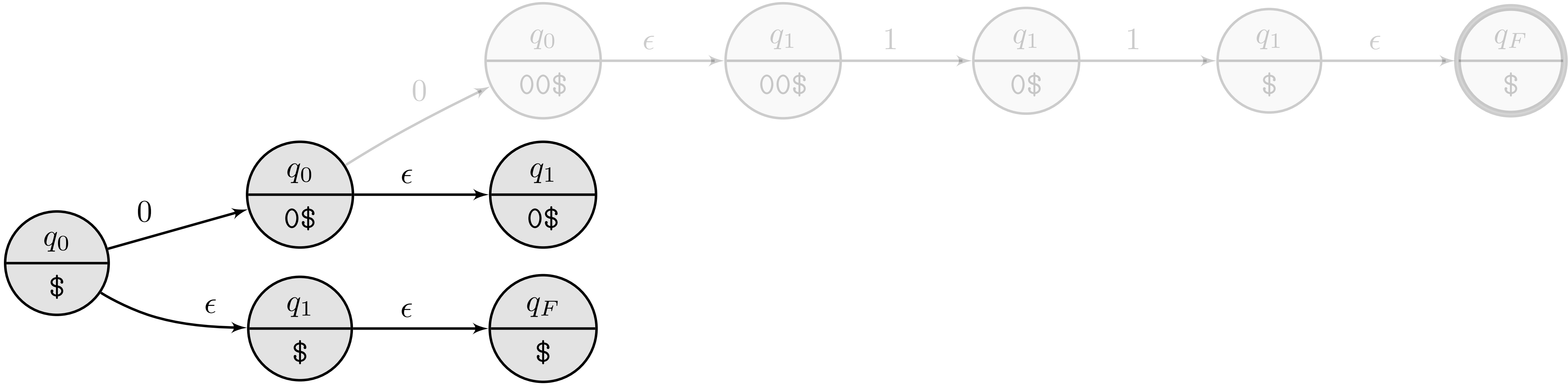
Accepting [ϵ0011]



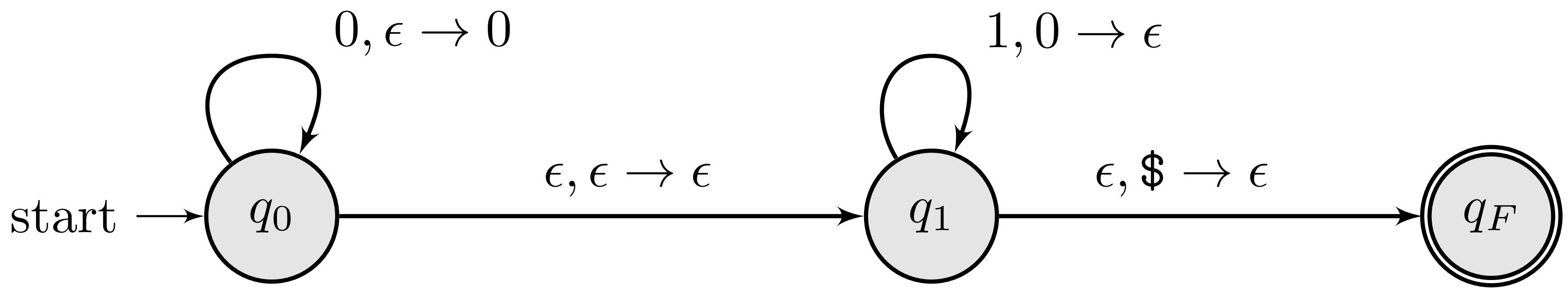
Acceptance example



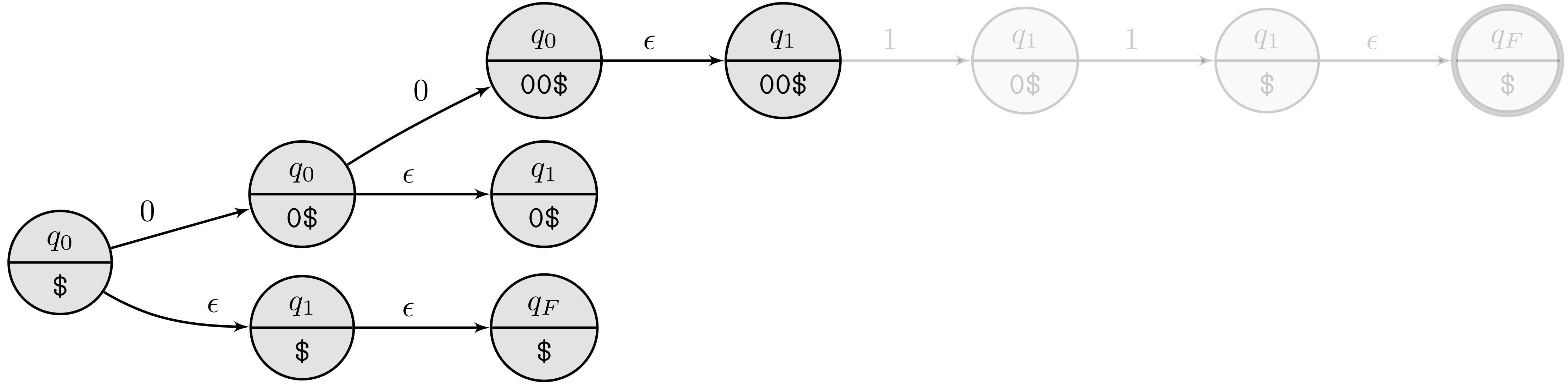
Accepting [**0ϵ**011]



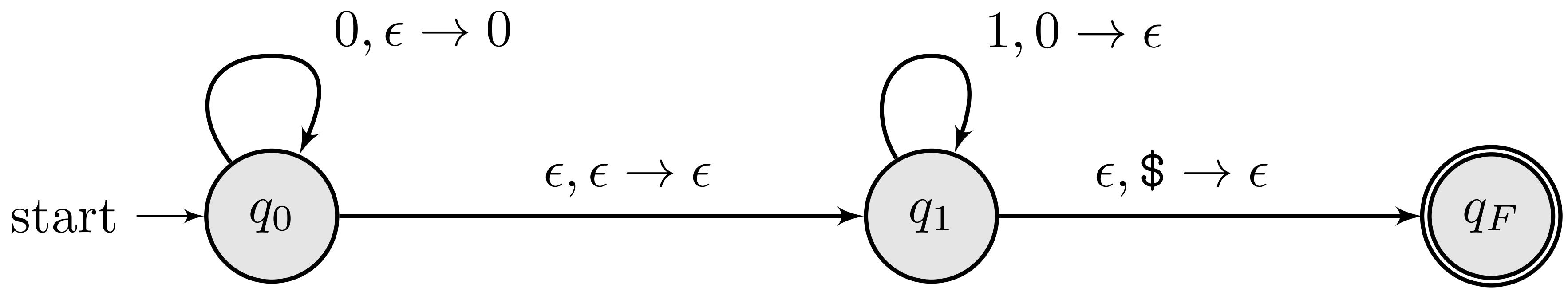
Acceptance example



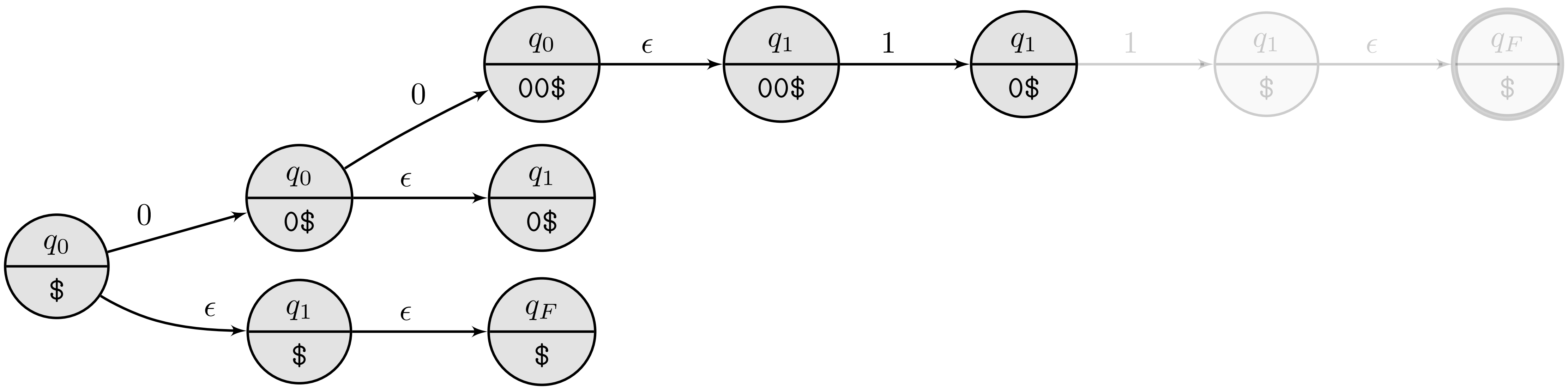
Accepting [0**0ϵ**11]



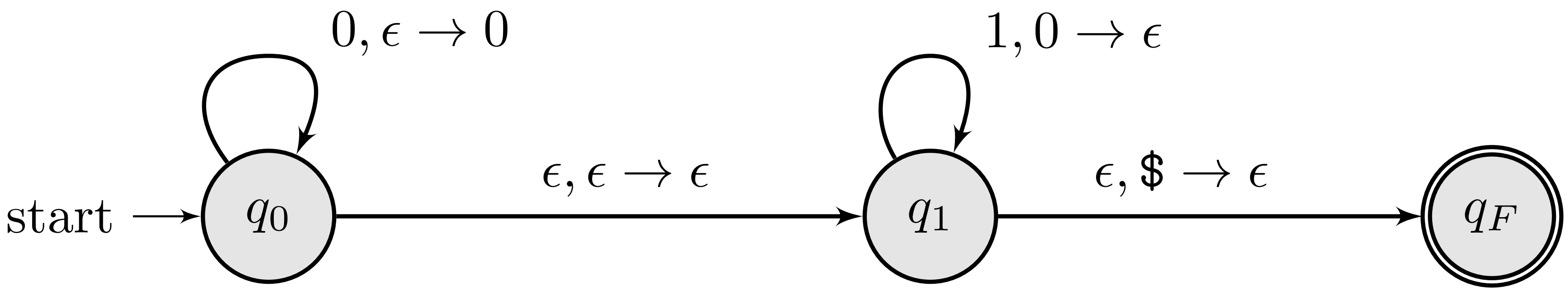
Acceptance example



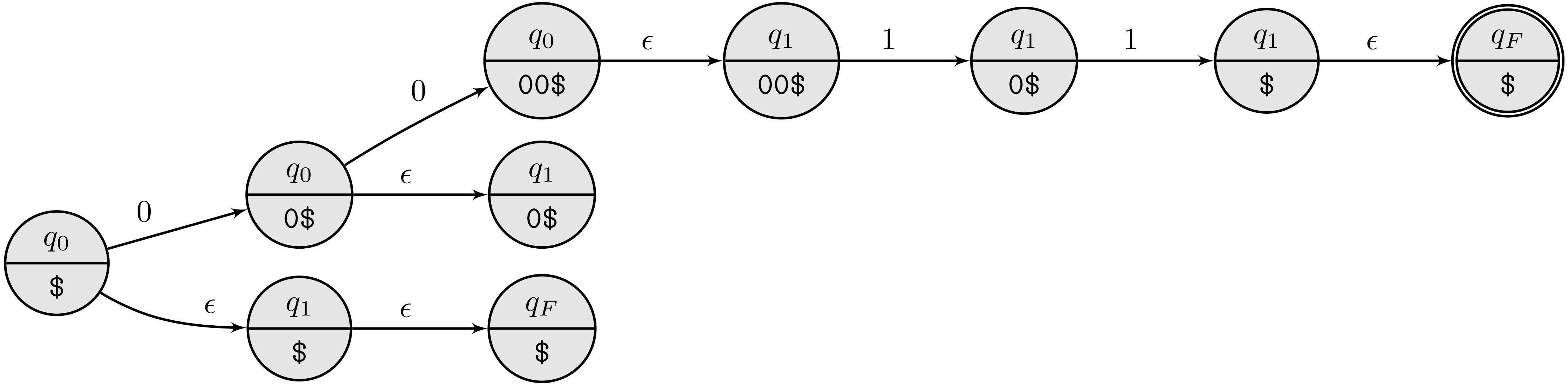
Accepting [00**1ϵ**1]



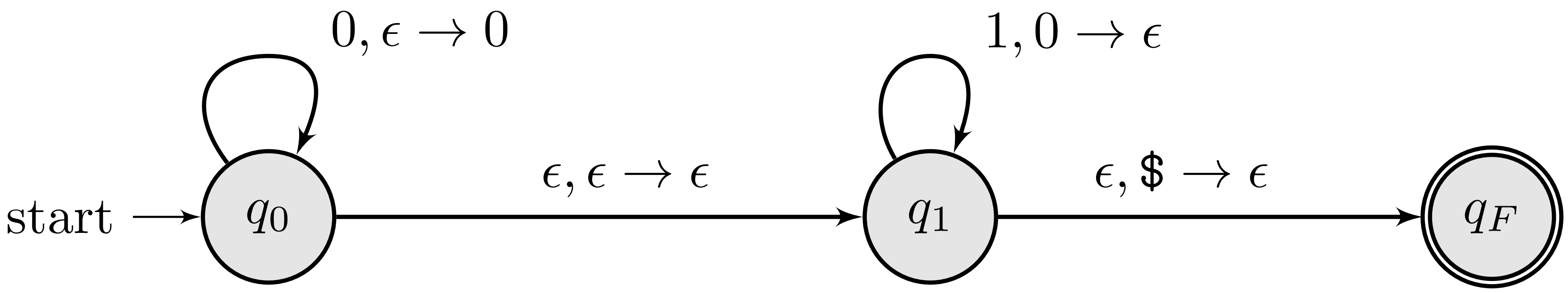
Acceptance example



Accepting [001**1ϵ**]

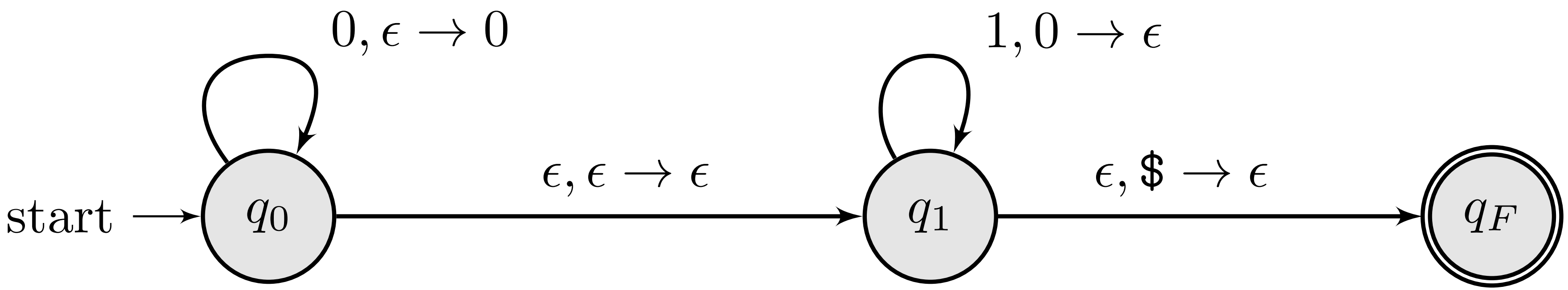


Acceptance example

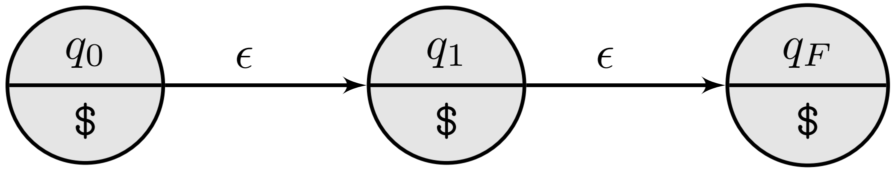


Accepting: 11

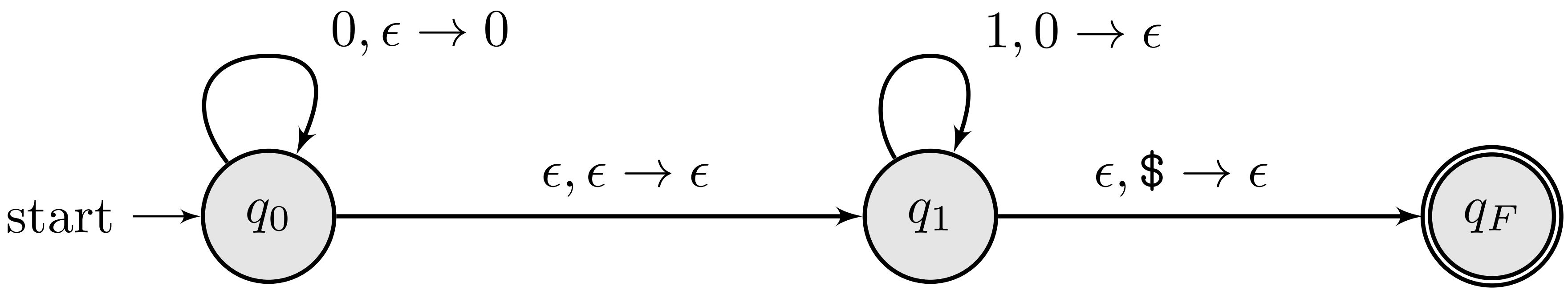
Acceptance example



Accepting: 11

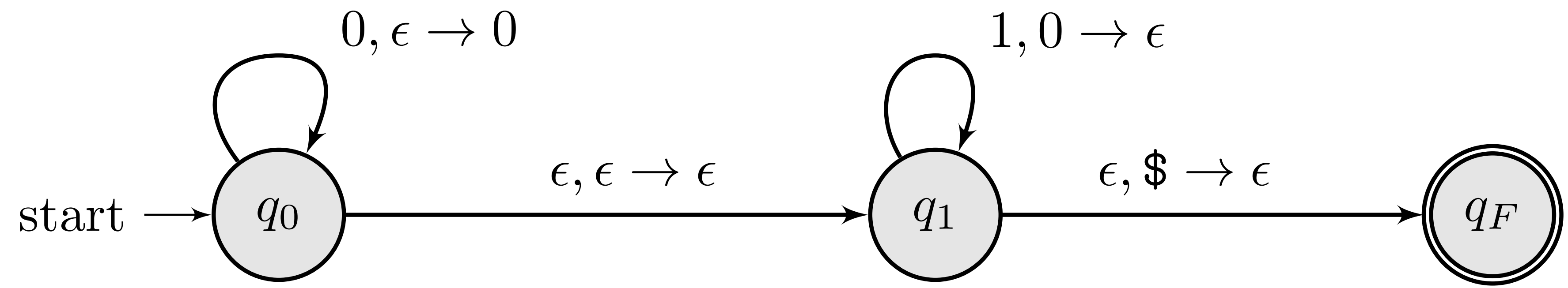


Acceptance example

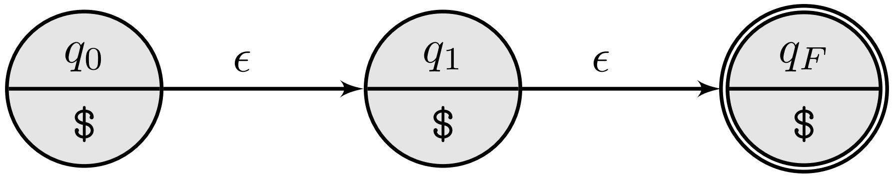


Accepting: $\boldsymbol{\epsilon}$

Acceptance example



Accepting: ϵ



Formalizing a PDA

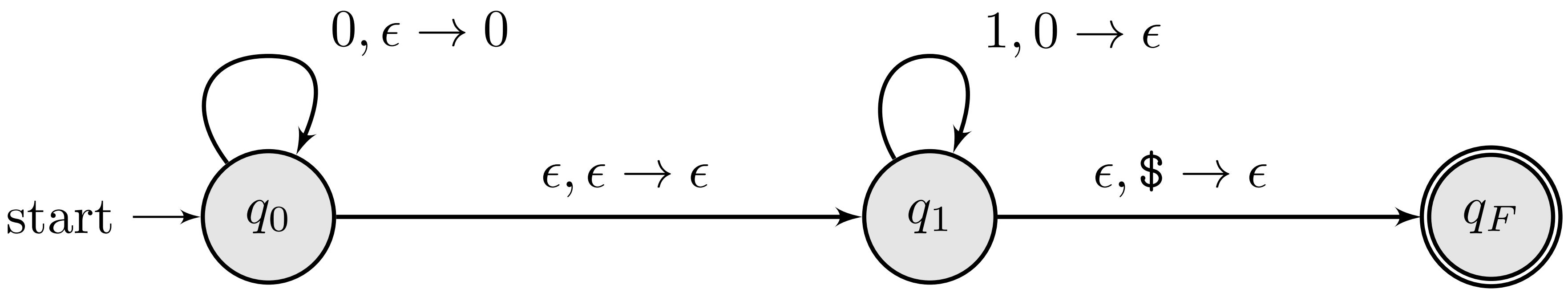
Formalizing a PDA

Definition 2.13

A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

1. Q is a finite set called states
2. Σ is a finite set called input alphabet
3. Γ is a finite set called stack alphabet
4. $\delta\colon Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function
5. $q_0 \in Q$ is the start state
6. $F \subseteq Q$ is the set of accepted states

Example



Let $(Q, \Sigma, \Gamma, \delta, q_1, \{q_F\})$ be defined as:

1. $Q = \{q_0, q_1, q_F\}$
2. $\Sigma = \{0, 1\}$
3. $\Gamma = \{0, \$\}$

where δ is defined by branches

$$\begin{aligned}
\delta(q_0, 0, \epsilon) &= \{(q_0, 0)\}\\
\delta(q_0, \epsilon, \epsilon) &= \{(q_1, \epsilon)\}\\
\delta(q_1, 1, 0) &= \{(q_1, \epsilon)\}\\
\delta(q_1, \epsilon, \$) &= \{(q_1, \$)\}\\
\delta(q, c, s) &= \{\}
\end{aligned}$$

Give a PDA for the following grammar

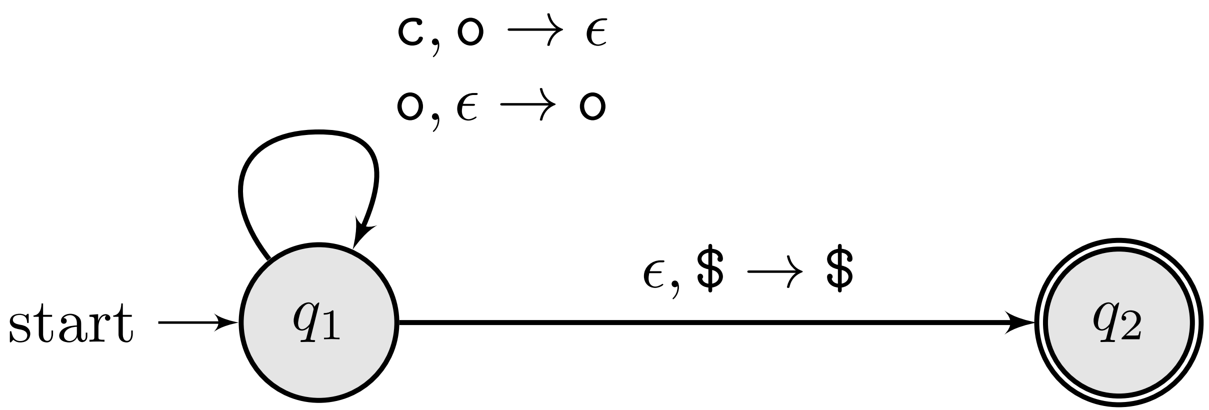
Ballanced parenthesis

$$\boxed{C \to \underline{\mathtt{o}}\, C\, \underline{\mathtt{c}} \mid CC \mid \epsilon}$$

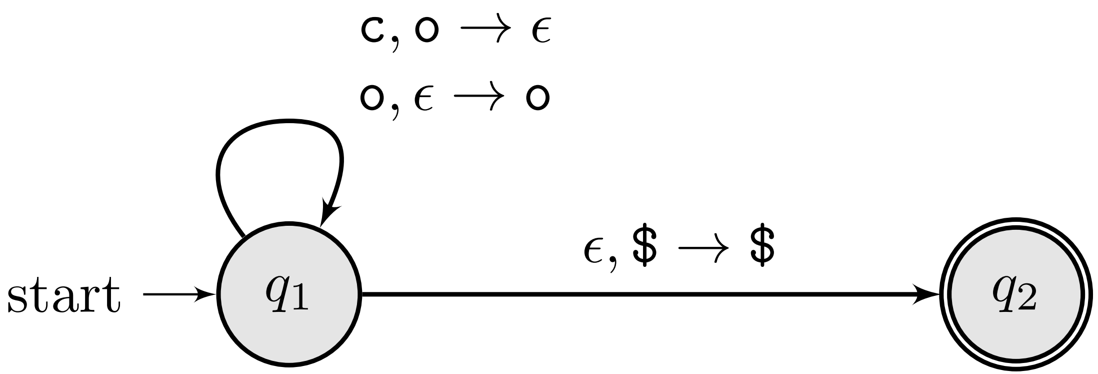
Give a PDA for the following grammar

Ballanced parenthesis

$$\boxed{C \to \underline{\mathtt{o}}\, C\, \underline{\mathtt{c}} \mid CC \mid \epsilon}$$

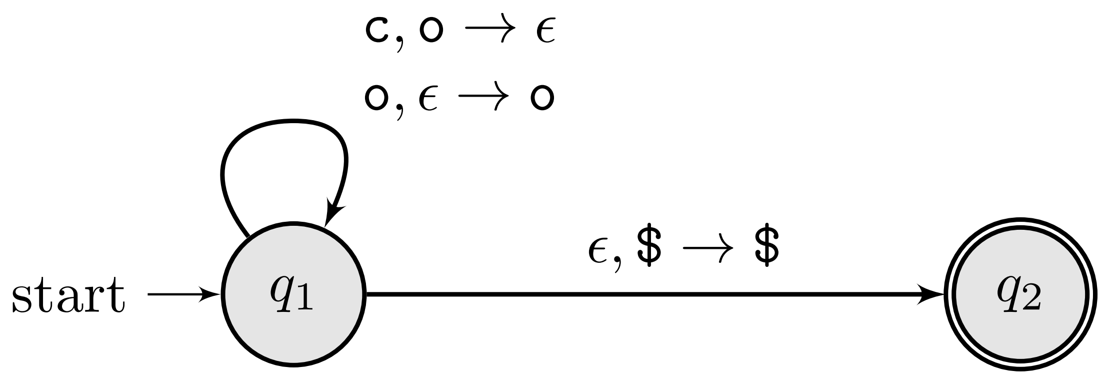


Acceptance

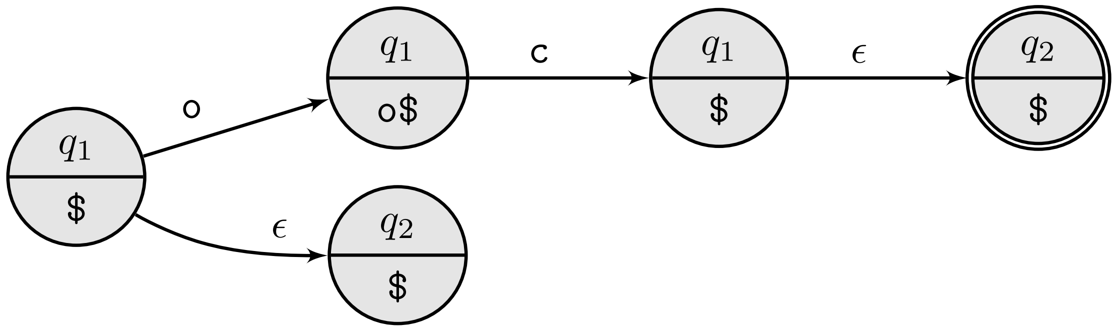


Acceptance: oc

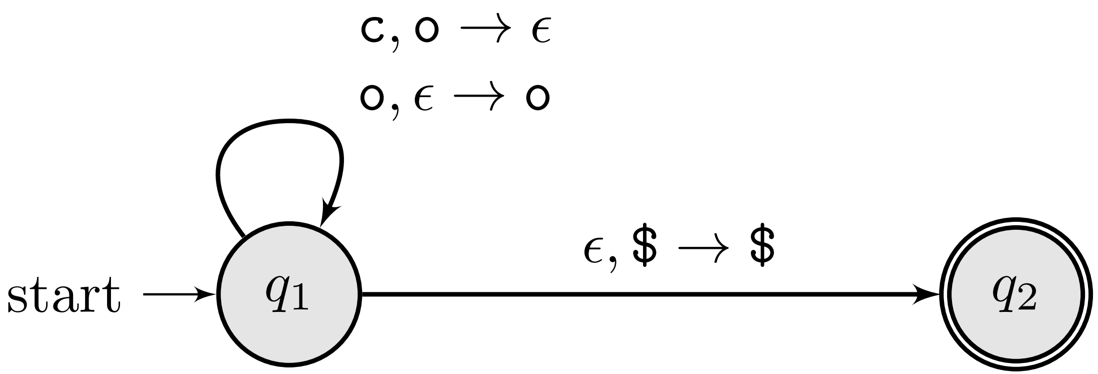
Acceptance



Acceptance: OC

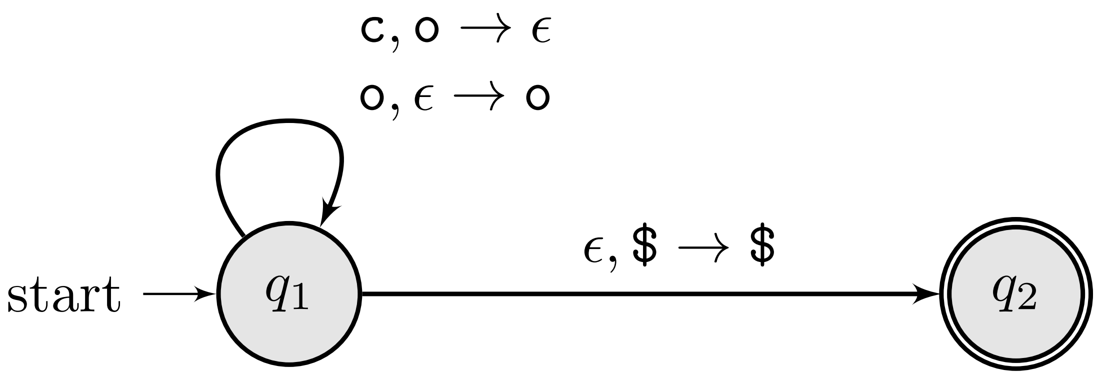


Acceptance

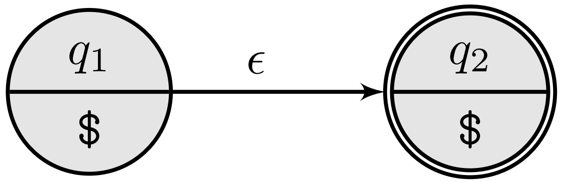


Acceptance: ϵ

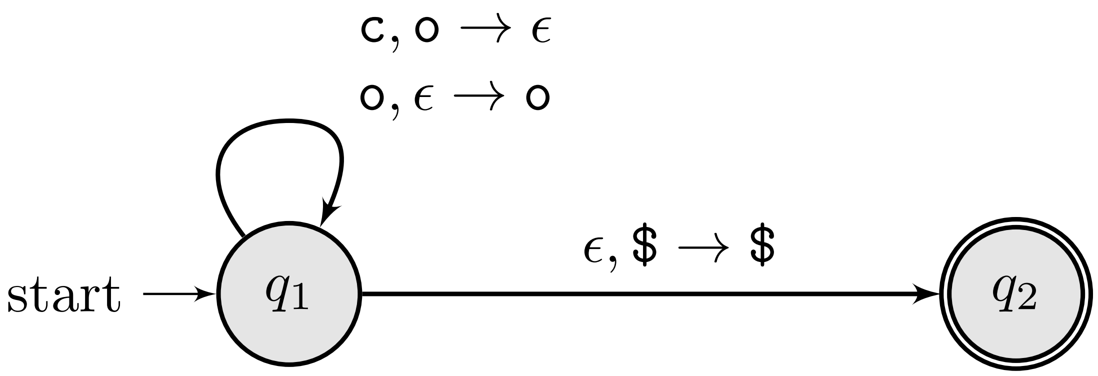
Acceptance



Acceptance: ϵ

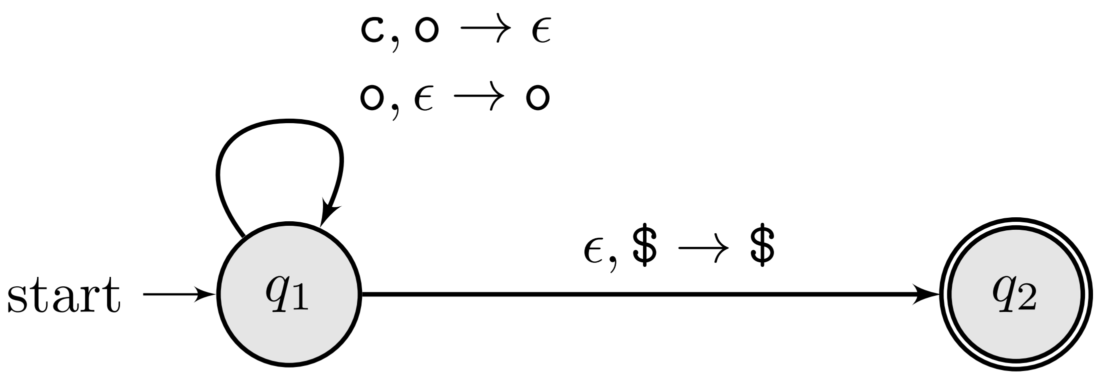


Acceptance

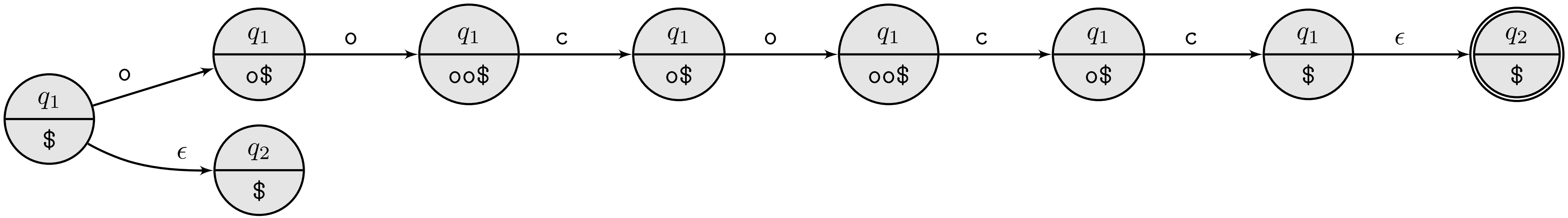


Acceptance: OOCOCC

Acceptance



Acceptance: OOCOCC



Formalizing stack operation

Let $S(o_1, o_2, s)$ be defined as follows, where $S : \Gamma_\epsilon \times \Gamma_\epsilon \times \mathrm{Stack}(\Gamma) \to \mathrm{Stack}(\Gamma)$ and $\mathrm{Stack}(\Gamma) = \mathrm{List}(\Gamma - \$)$:

Pop operation

$$
\begin{aligned}
s \triangleright \epsilon &= s \\
[] \triangleright \$ &= [] \\
n :: s \triangleright n &= s
\end{aligned}
$$

Examples

$$
\begin{gathered}
[0, 1] \triangleright \epsilon = [0, 1] \\
[0, 1] \triangleright \$ \text{ is undefined!} \\
[] \triangleright \$ = \$ \\
[0, 1] \triangleright 0 = [1] \\
[0, 1] \triangleright 1 \text{ is undefined!}
\end{gathered}
$$

Push operation

$$
\begin{aligned}
s \triangleleft \epsilon &= s \\
s \triangleleft \$ &= [] \\
s \triangleleft n &= n :: s
\end{aligned}
$$

Examples

$$
\begin{gathered}
[0, 1] \triangleleft \epsilon = [0, 1] \\
[0, 1] \triangleleft \$ = [] \\
[] \triangleleft \$ = [] \\
[0, 1] \triangleleft 0 = [0, 0, 1] \\
[0, 1] \triangleleft 1 = [1, 0, 1]
\end{gathered}
$$

Stack operation exercises

Examples

Expression	Result
$ab \triangleright c =$	
$ab \triangleleft c =$	
$ab \triangleright a =$	
$ab \triangleleft a =$	
$ab \triangleright \$ =$	
$ab \triangleleft \$ =$	
$\epsilon \triangleright \$ =$	
$\epsilon \triangleleft \$ =$	
$\epsilon \triangleright a =$	
$\epsilon \triangleleft a =$	

Stack operation exercises

Examples

Expression	Result
$ab \triangleright c =$	**undef**
$ab \triangleleft c =$	cab
$ab \triangleright a =$	b
$ab \triangleleft a =$	aab
$ab \triangleright \$ =$	undef
$ab \triangleleft \$ =$	ϵ
$\epsilon \triangleright \$ =$	ϵ
$\epsilon \triangleleft \$ =$	ϵ
$\epsilon \triangleright a =$	**undef**
$\epsilon \triangleleft a =$	a

Formalizing acceptance

$$\frac{(q',o') \in \delta(q,y,o)}{(q,s) \xrightarrow{y} (q', s \triangleright o \triangleleft o')}$$

Rule 0. We can go from state q and stack s into state q' and stack s' with input $y \in \Sigma_\epsilon$ if we can construct s' from a push o and a pop o' on stack s.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, let the **steps through** relation, notation $q \curvearrowright_M w$, be defined as:

$$\frac{q \in F}{(q,s) \curvearrowright_M []}$$

Rule 1. State q steps through $[]$ if q is a final state.

$$\frac{(q,s) \xrightarrow{y} (q',s') \qquad (q',s') \curvearrowright_M w}{(q,s) \curvearrowright_M y \cdot w}$$

Rule 2. If we can go from q to q' with y and q' steps through w, then q steps through $y \cdot w$.

Acceptance. We say that M accepts w if, and only if, $q_0, [] \curvearrowright_M w$.

Example of acceptance

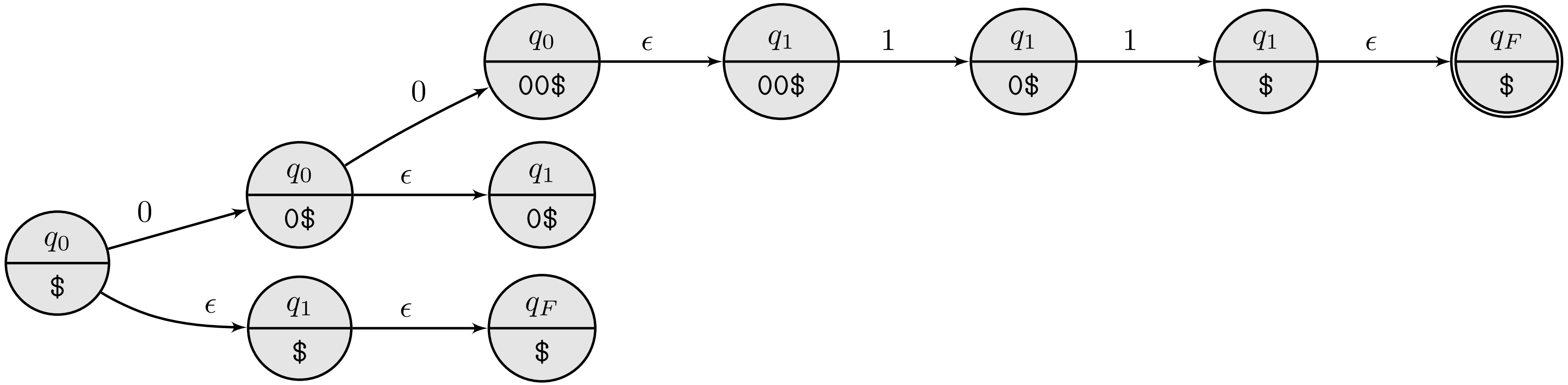
We can build a chain of states as follows

$$(q_0, []) \xrightarrow{0} (q_0, [0]) \xrightarrow{0} (q_1, [0,0]) \xrightarrow{\epsilon} (q_1, [0,0]) \xrightarrow{1} (q_1, [0]) \xrightarrow{1} (q_1, []) \xrightarrow{\epsilon} (q_F, [])$$

Since q_F is a final state, we have that

$$(q_0, []) \curvearrowright [0, 0, 1, 1]$$

Recall



Example 2.16

> A sequence of a-s then b-s and finally c-s with as many a-s as there are b-s or as there are c-s.

$$\{a^i b^j c^k \mid i = j \vee i = k\}$$

Example 2.16

> A sequence of a-s then b-s and finally c-s with as many a-s as there are b-s or as there are c-s.

$\{a^i b^j c^k \mid i = j \vee i = k\}$

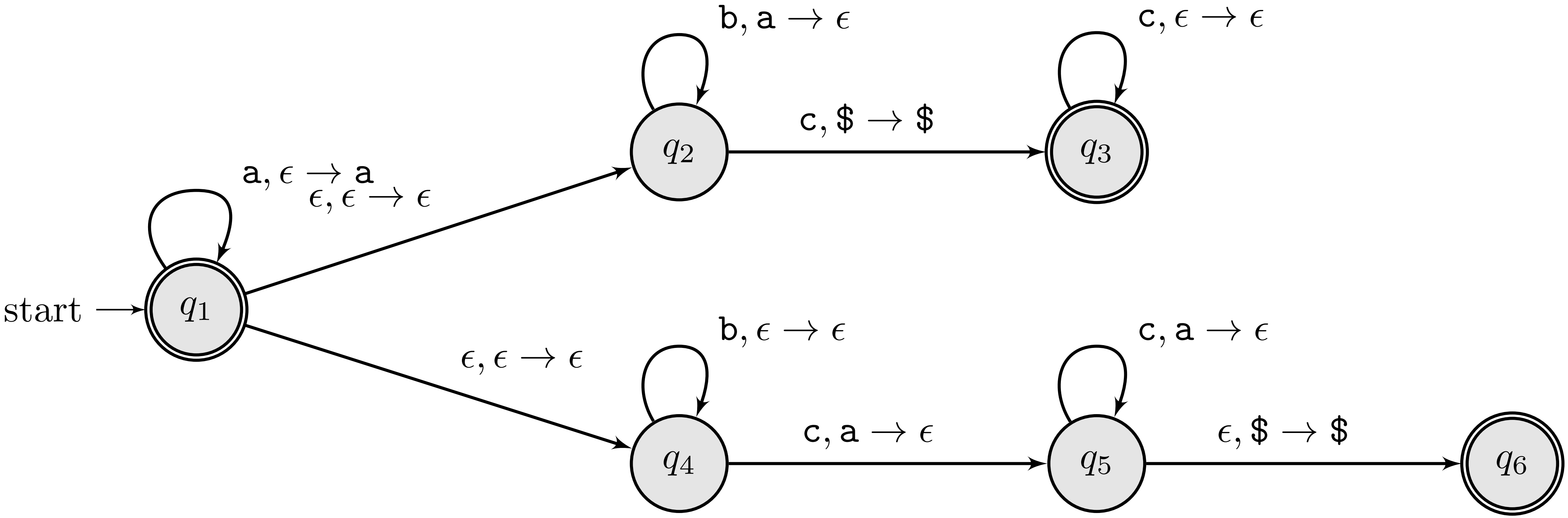
A solution

Step 1. read and push a total of N a's.

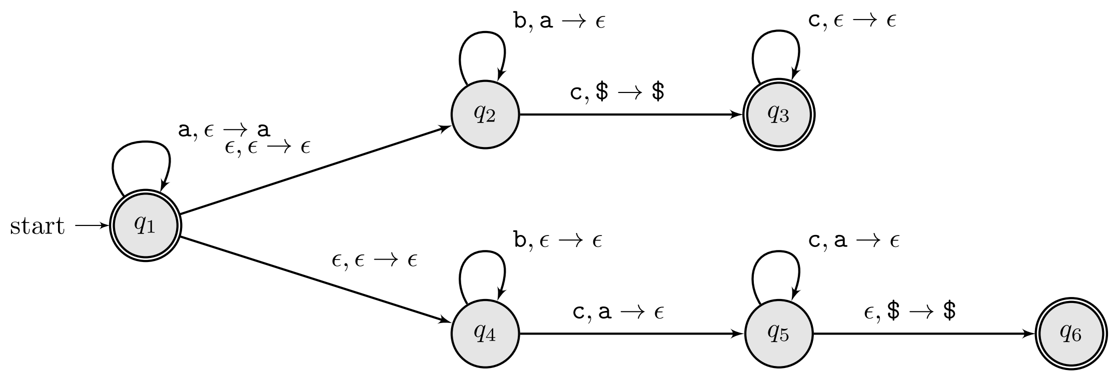
Step 2. Either:

- $(i = j)$ read N b's and pop a's; followed by reading an arbitrary number of c's
- $(i = k)$ read an arbitrary number of b's followed by read N c's and pop a's

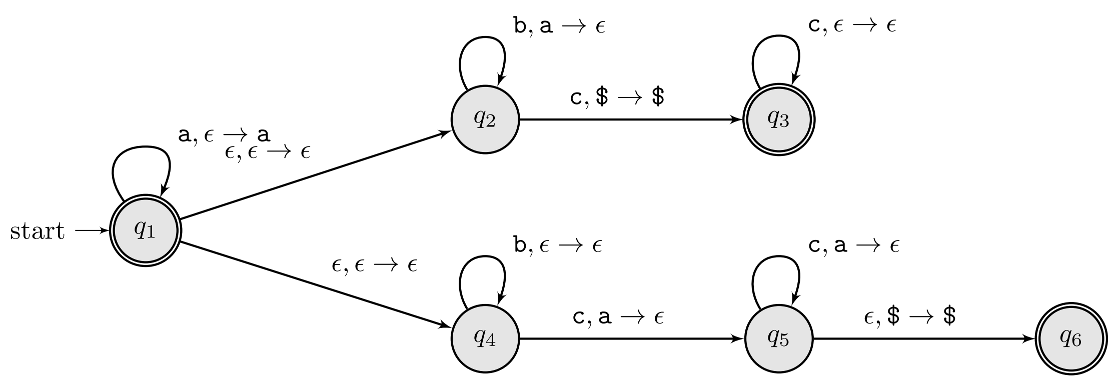
State diagram of Example 2.16

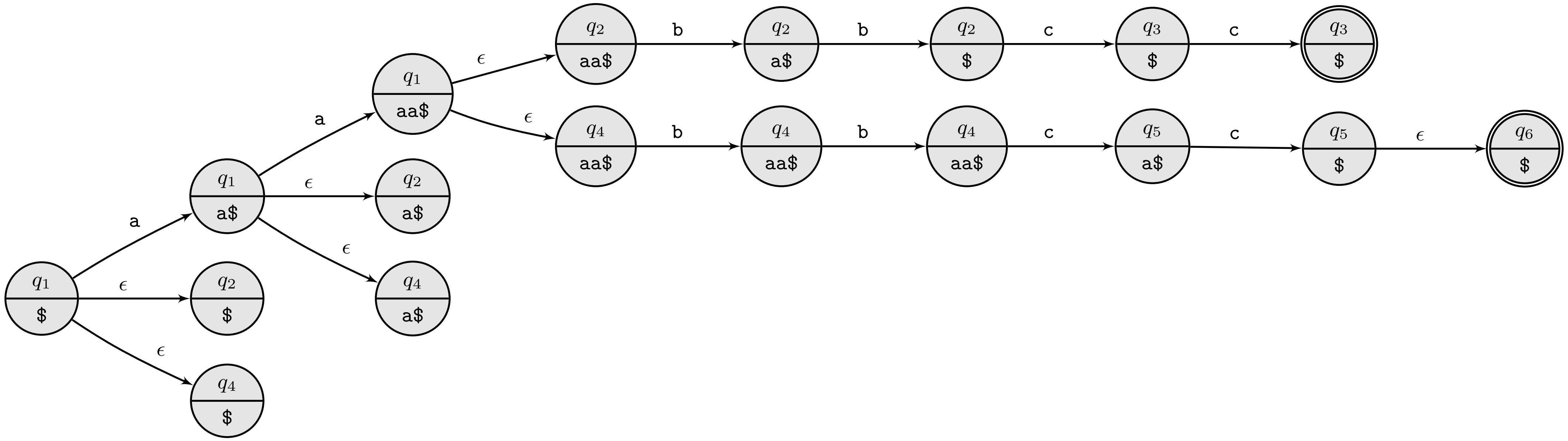


Example 2.16 accept [a,a,b,b,c,c]?

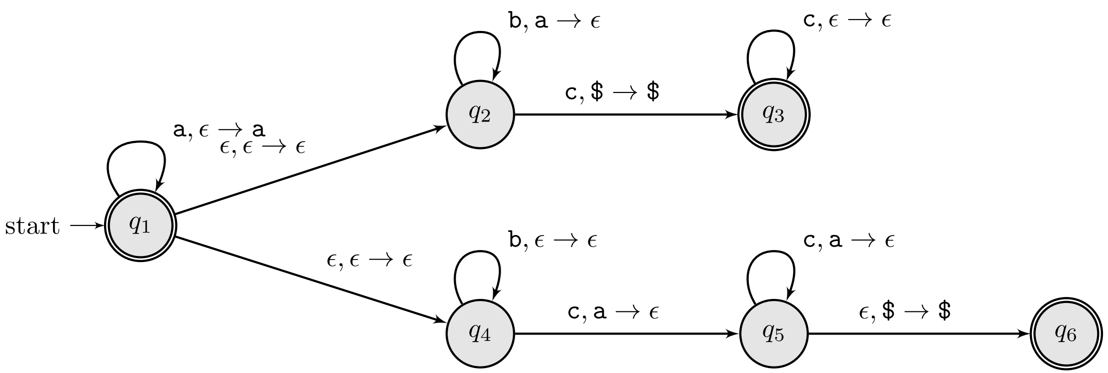


Example 2.16 accept `[a,a,b,b,c,c]`?

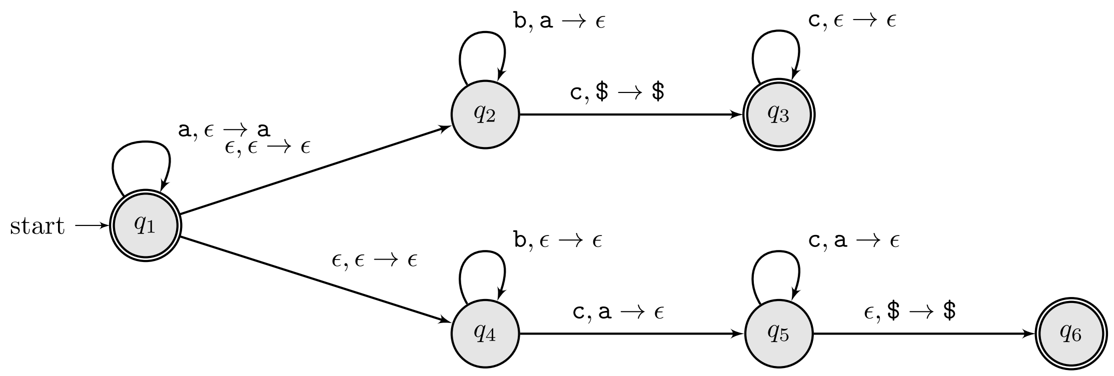


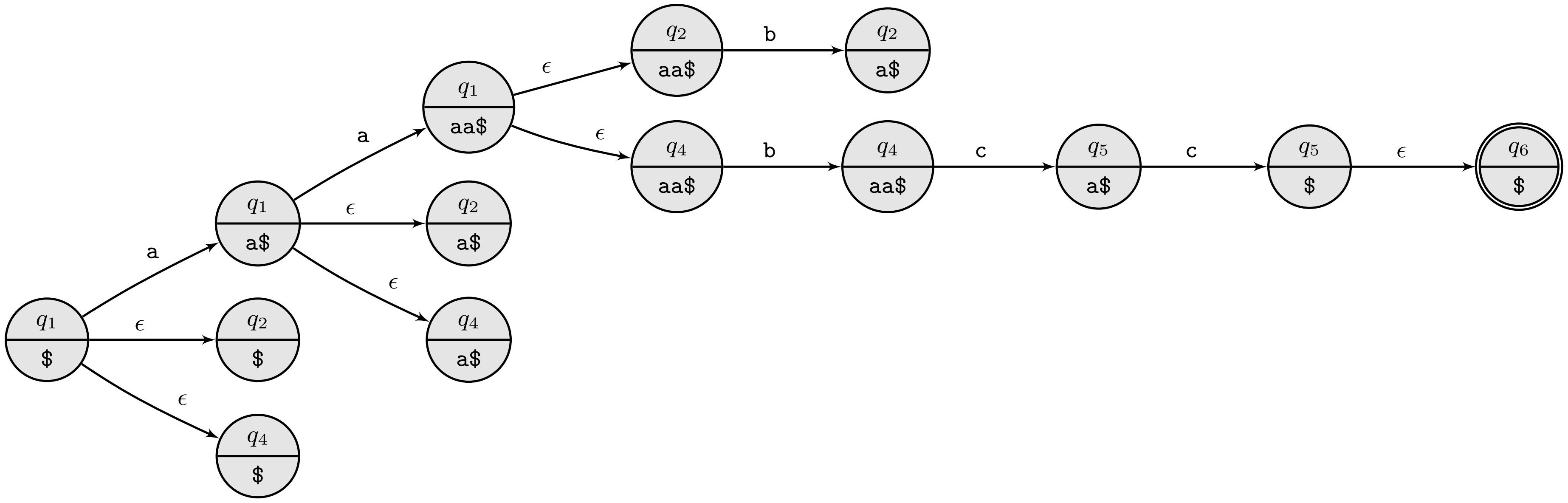


Example 2.16 accept `[a,a,b,c,c]`?

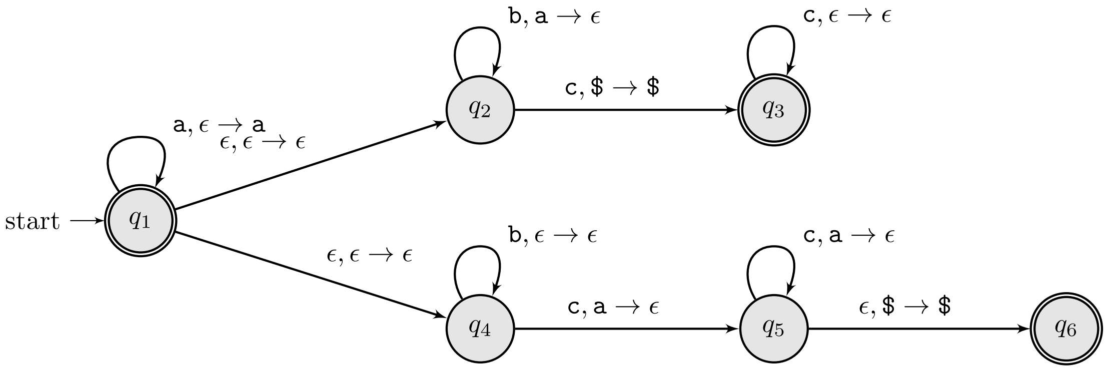


Example 2.16 accept [a,a,b,c,c]?

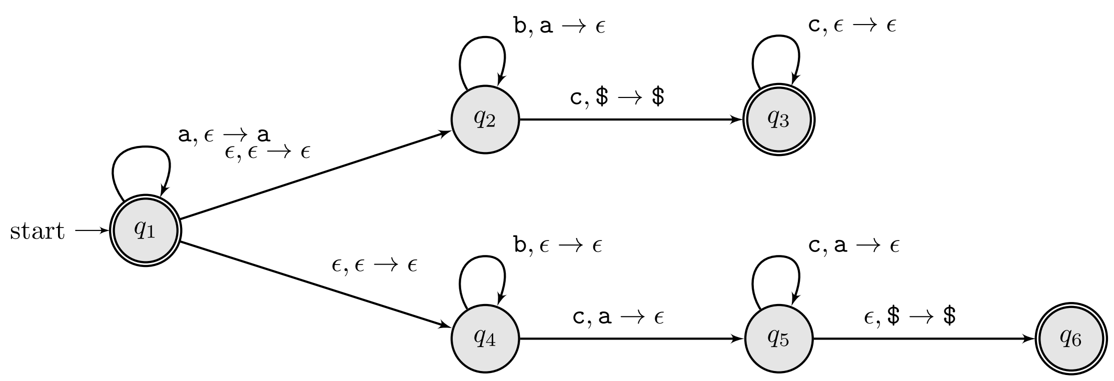


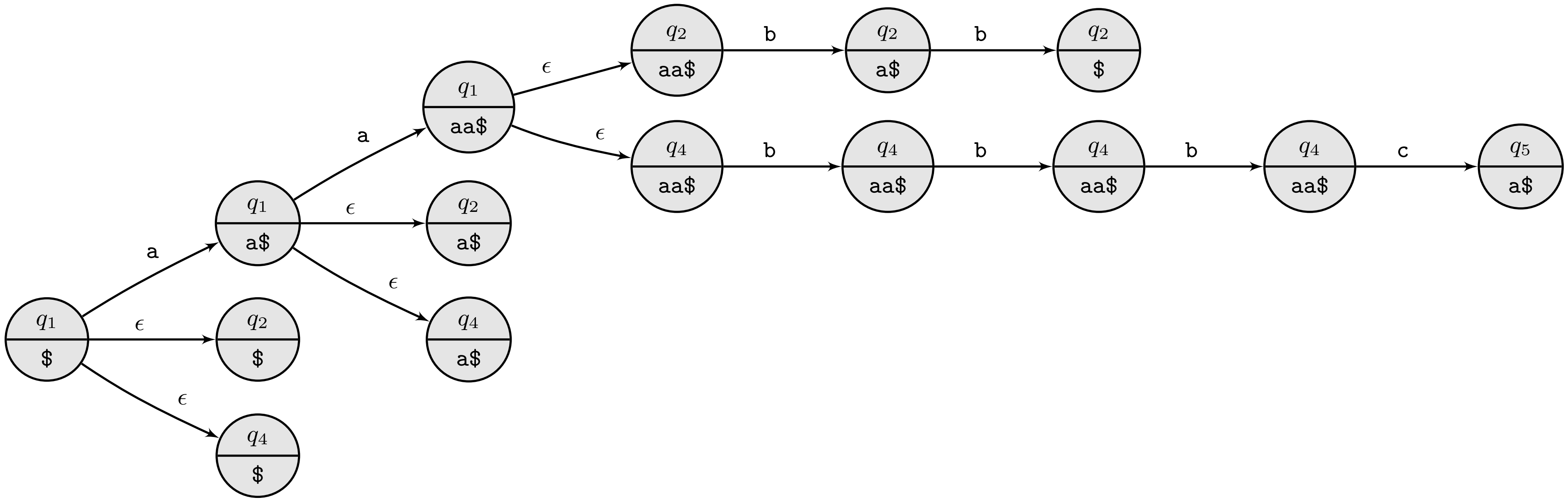


Example 2.16 accept [a,a,b,b,c]?

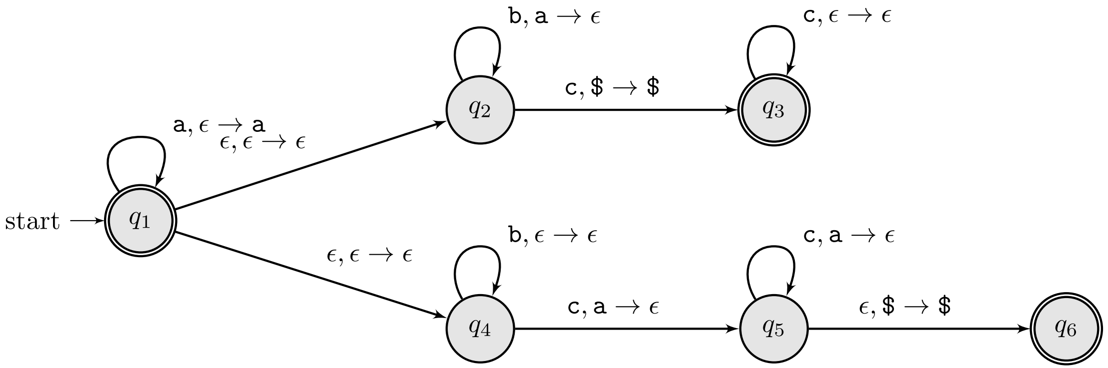


Example 2.16 accept [a,a,b,b,c]?

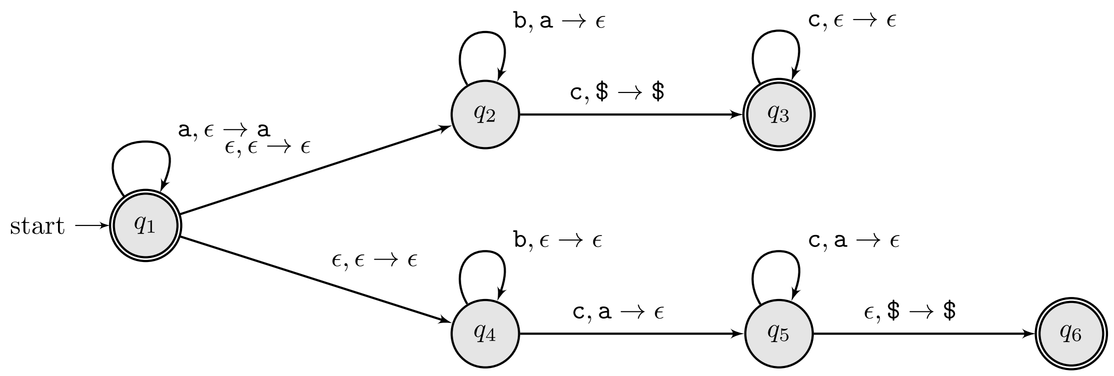


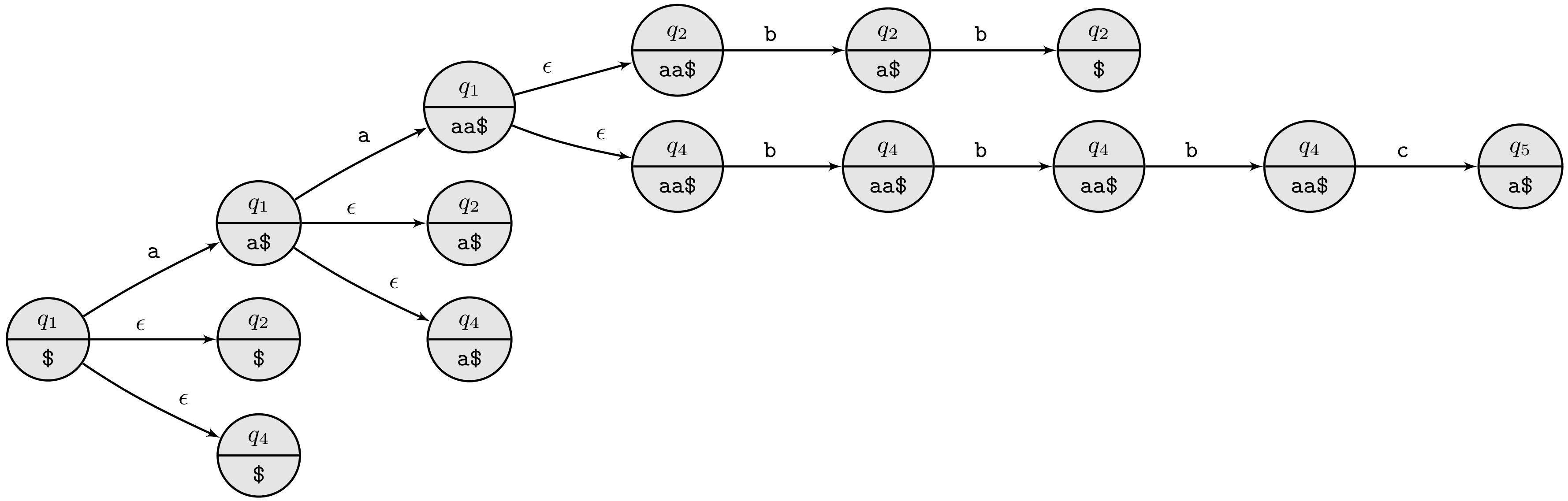


Example 2.16 accept [a, a, b, b, b, c, c]?



Example 2.16 accept [a,a,b,b,b,c,c]?





Union for PDAs?

Example 2.16

$$\{a^i b^j c^k \mid i = j \lor i = k\} = \{a^i b^j c^k \mid i = j\} \cup \{a^i b^j c^k \mid i = k\}$$

Example 2.16

$$\{a^i b^j c^k \mid i = j \lor i = k\} = \{a^i b^j c^k \mid i = j\} \cup \{a^i b^j c^k \mid i = k\}$$