CS420

Introduction to the Theory of Computation

Lecture 16: Push-down automata

Tiago Cogumbreiro

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: $q \xrightarrow{\texttt{\$INPUT,\$PRE}\rightarrow\texttt{\$POST}} q'$

Example

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

Possible operations

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

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: $q \xrightarrow{\texttt{\$INPUT,\$PRE}\to\texttt{\$POST}} q'$

Possible operations

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

Example

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

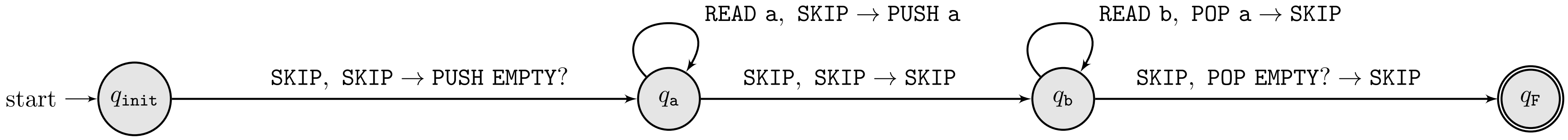
Attention!

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

PDA example (intuition)

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

1. $q_{\texttt{init}} \xrightarrow{\texttt{SKIP,SKIP}\to\texttt{PUSH EMPTY?}} q_{\texttt{a}}$
2. $q_{\texttt{a}} \xrightarrow{\texttt{READ a,SKIP}\to\texttt{PUSH a}} q_{\texttt{a}}$
3. $q_{\texttt{a}} \xrightarrow{\texttt{SKIP,SKIP}\to\texttt{SKIP}} q_{\texttt{b}}$
4. $q_{\texttt{b}} \xrightarrow{\texttt{READ b,POP a}\to\texttt{SKIP}} q_{\texttt{b}}$
5. $q_{\texttt{b}} \xrightarrow{\texttt{SKIP,EMPTY?}\to\texttt{SKIP}} q_{\texttt{F}}$



Exercising transitions

Writing transitions

Possible operations

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

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel EMPTY?):

Writing transitions

Possible operations

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

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel `EMPTY?`):
   READ 0, EMPTY? $\rightarrow$ **SKIP**
2. Test if stack is empty:

Writing transitions

Possible operations

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

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel EMPTY?):
   READ 0, EMPTY? $\to$ SKIP
2. Test if stack is empty:
   SKIP, EMPTY? $\to$ SKIP
3. Test if a is on top and leave stack untouched:

Writing transitions

Possible operations

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

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel EMPTY?):
   READ 0, EMPTY? $\to$ **SKIP**
2. Test if stack is empty:
   SKIP, EMPTY? $\to$ **SKIP**
3. Test if a is on top and leave stack untouched:
   SKIP, POP a $\to$ **PUSH a**
4. Read b and leave stack untouched:

Writing transitions

Possible operations

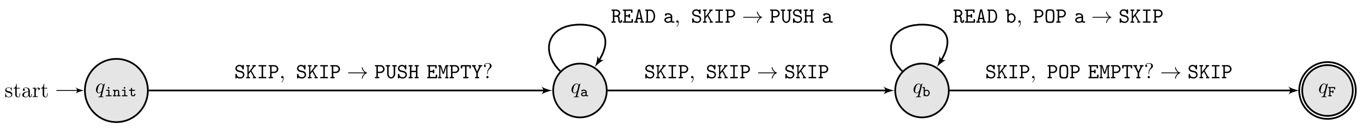
$INPUT	*$PRE*	*$POST*
READ n	POP n	PUSH n
SKIP (ϵ)	SKIP	SKIP

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel EMPTY?):
   READ 0, EMPTY? → SKIP
2. Test if stack is empty:
   SKIP, EMPTY? → SKIP
3. Test if a is on top and leave stack untouched:
   SKIP, POP a → PUSH a
4. Read b and leave stack untouched:
   READ b, SKIP → SKIP

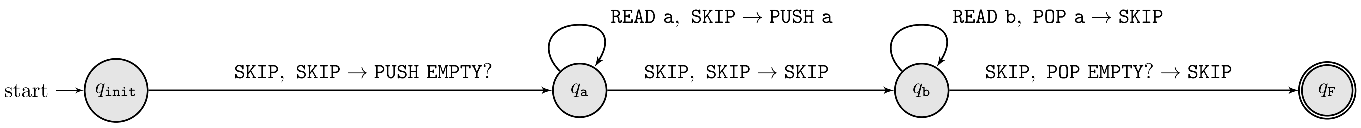
Simplifying the notation

Simplifying the notation

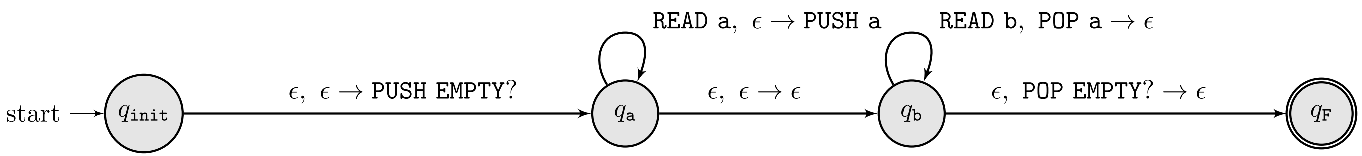


We can replace SKIP by ϵ

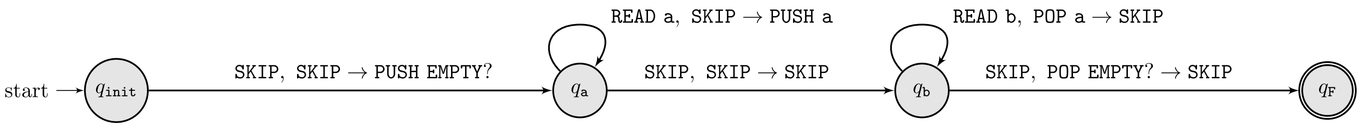
Simplifying the notation



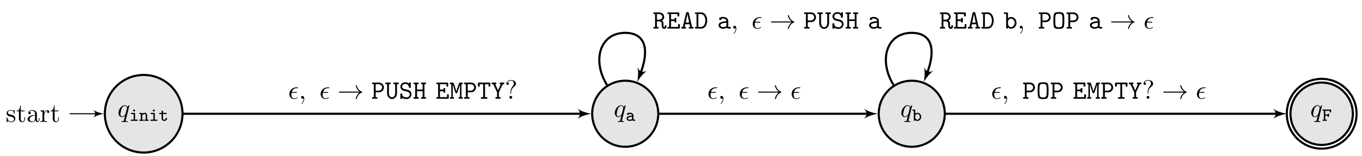
We can replace SKIP by ϵ



Simplifying the notation

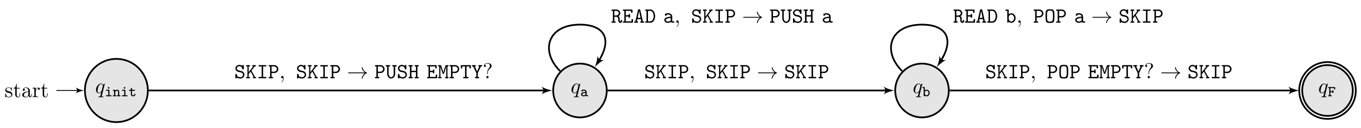


We can replace SKIP by ϵ

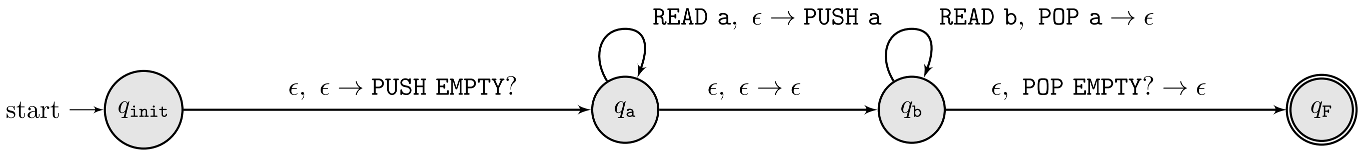


We can omit READ

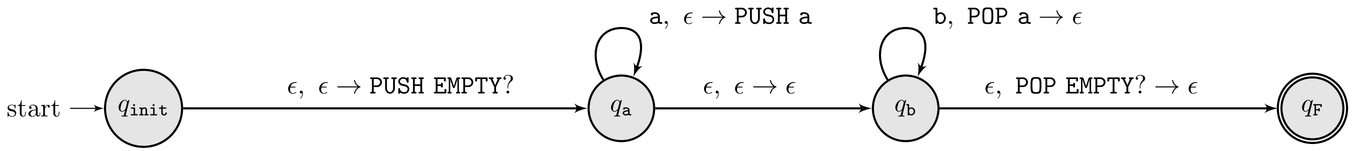
Simplifying the notation



We can replace SKIP by ϵ

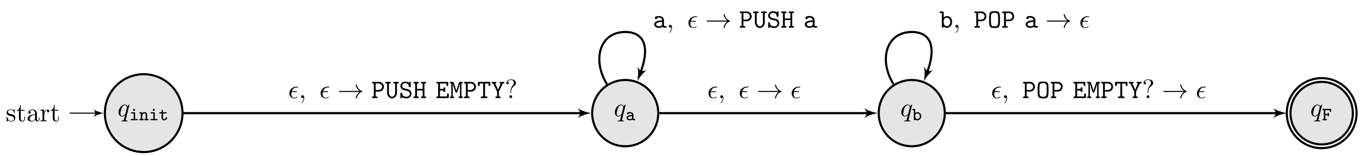


We can omit READ



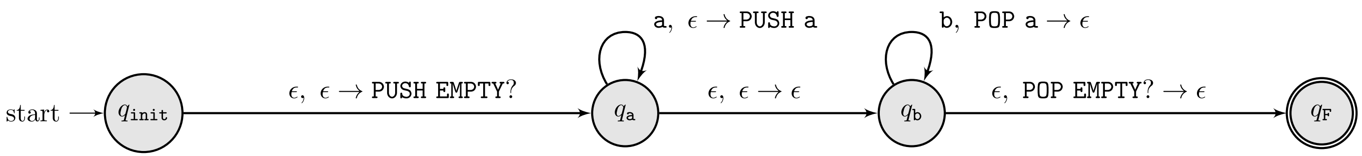
Since read always appears in the same position, we can omit it, as we do in regular DFAs/NFAs.

Simplifying the notation

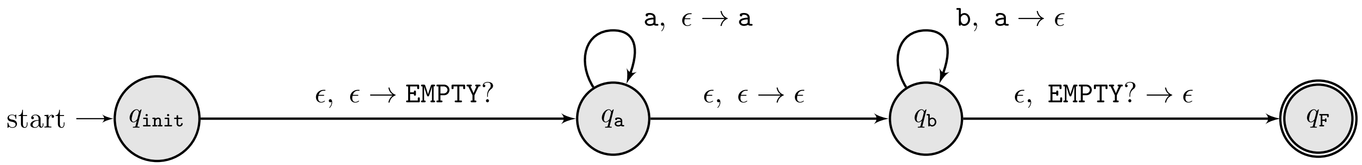


We can omit PUSH/POP

Simplifying the notation



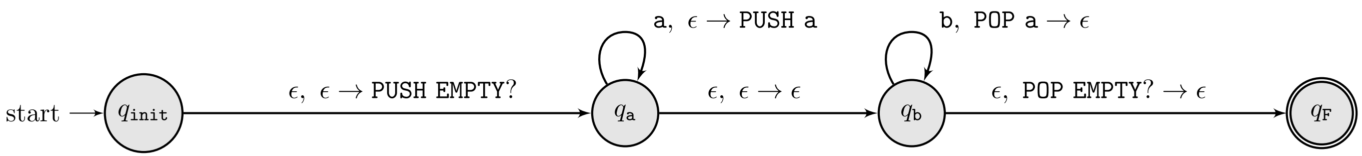
We can omit PUSH/POP



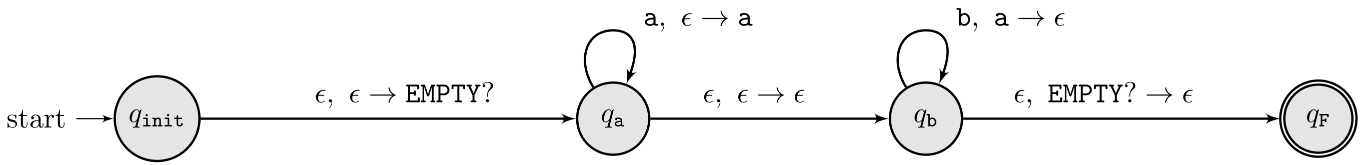
> Since push/pop always appear in the same position, we can omit them.

We can replace sentinel EMPTY? by $

Simplifying the notation

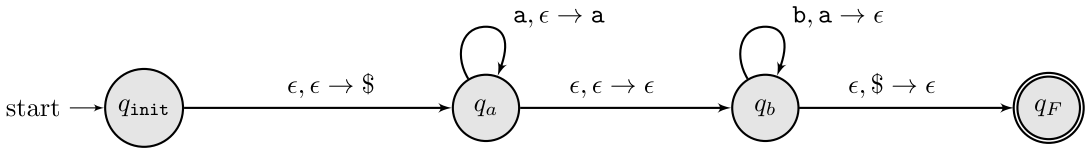


We can omit PUSH/POP



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

We can replace sentinel EMPTY? by $



Since empty? always appear in the same position.

Exercising transitions

(abbreviated notation)

Writing transitions

Possible operations

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

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $\$$)

Writing transitions

Possible operations

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

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $\$$)
   $0, \$ \to \$$
2. Test if stack is empty while leaving the stack unchanged (assume sentinel $\$$)

Writing transitions

Possible operations

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

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $\$$)
   $0, \$ \to \$$
2. Test if stack is empty while leaving the stack unchanged (assume sentinel $\$$)
   $\epsilon, \$ \to \$$
3. Test if 0 is on top of the stack and replace it by 1:

Writing transitions

Possible operations

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

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $\$$)
   $0, \$ \to \$$
2. Test if stack is empty while leaving the stack unchanged (assume sentinel $\$$)
   $\epsilon, \$ \to \$$
3. Test if 0 is on top of the stack and replace it by 1:
   $\epsilon, 0 \to 1$
4. Read 2, leave stack untouched

Writing transitions

Possible operations

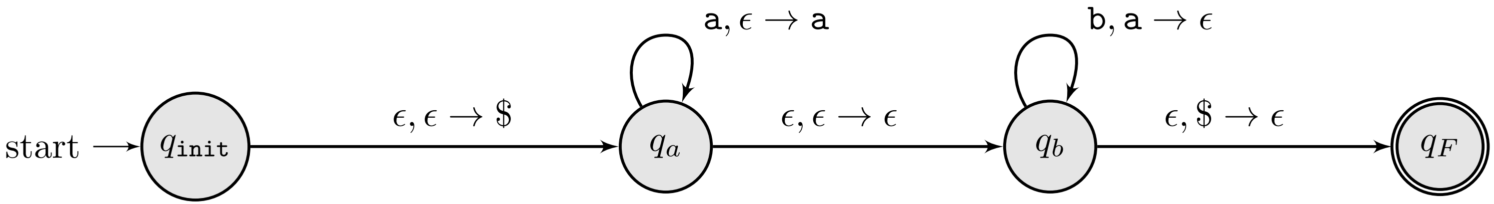
\$INPUT	\$PRE	\$POST
READ (n)	POP (n)	PUSH (n)
SKIP (ϵ)	SKIP (ϵ)	SKIP (ϵ)

Exercises

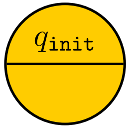
1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $\$$)
   $0, \$ \to \$$
2. Test if stack is empty while leaving the stack unchanged (assume sentinel $\$$)
   $\epsilon, \$ \to \$$
3. Test if 0 is on top of the stack and replace it by 1:
   $\epsilon, 0 \to 1$
4. Read 2, leave stack untouched
   $2, \epsilon \to \epsilon$

Acceptance example

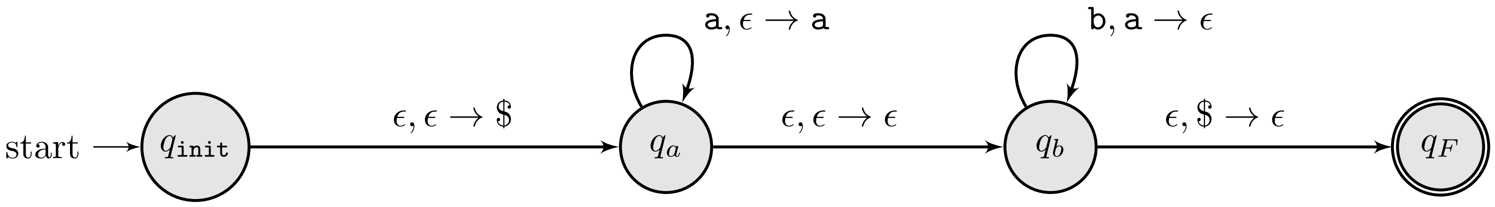
Acceptance example



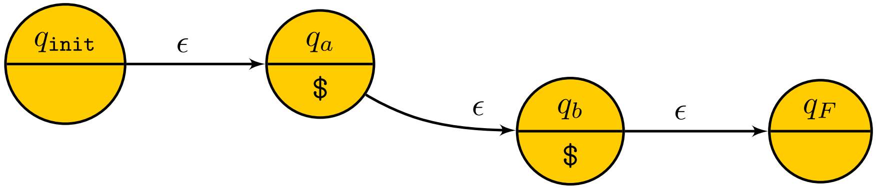
Accepting [**ϵ**aabb]



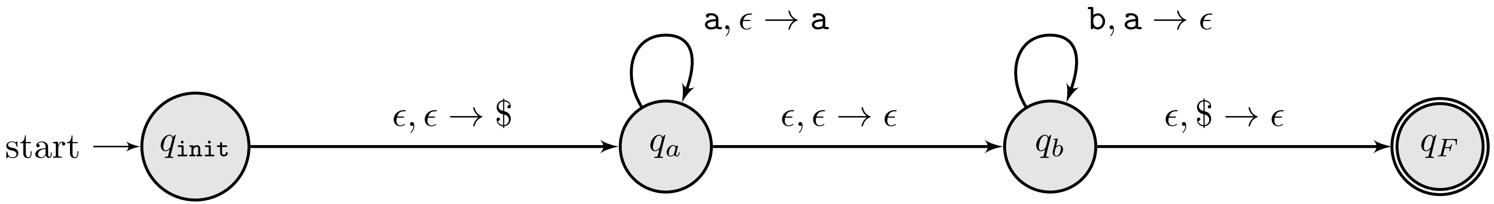
Acceptance example



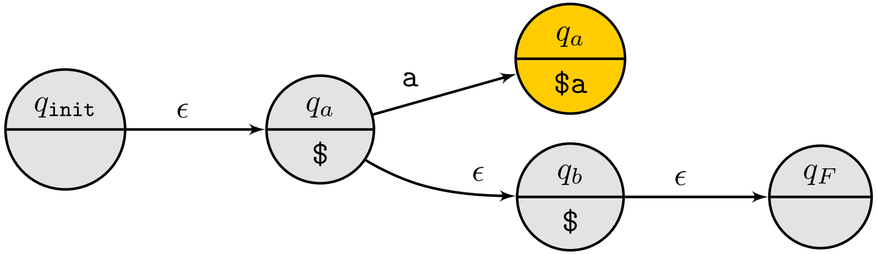
Accepting [**a**abb]



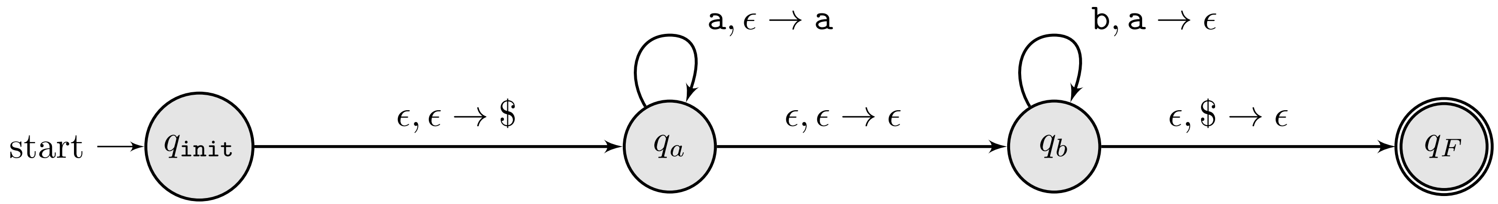
Acceptance example



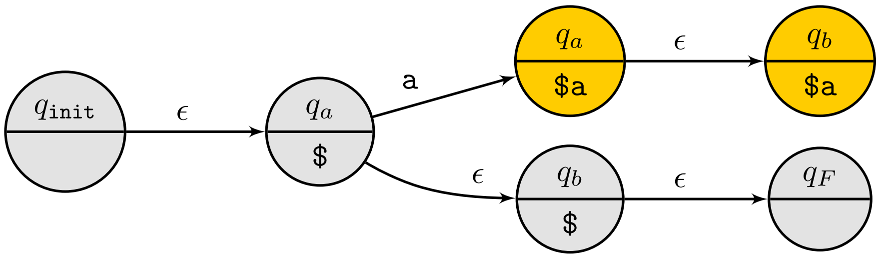
Accepting [a**ϵ**abb]



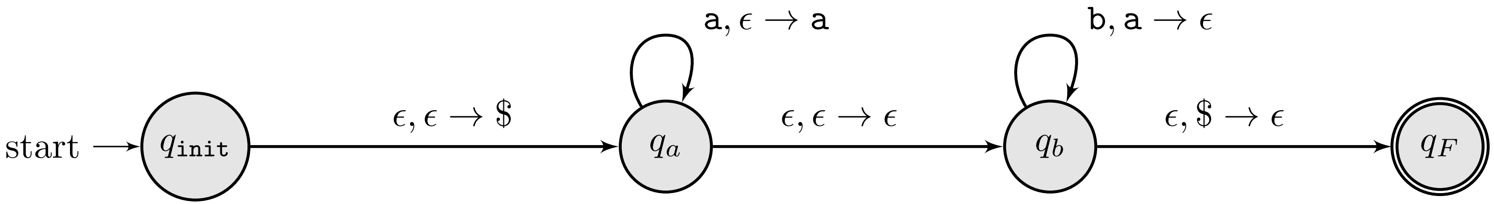
Acceptance example



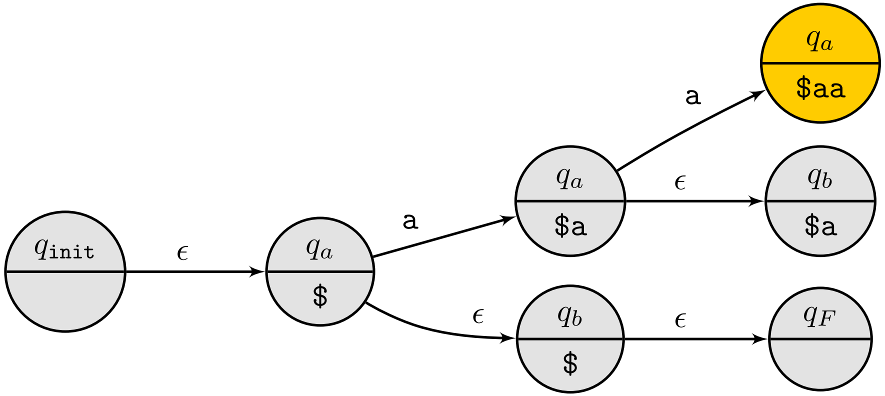
Accepting [a**a**bb]



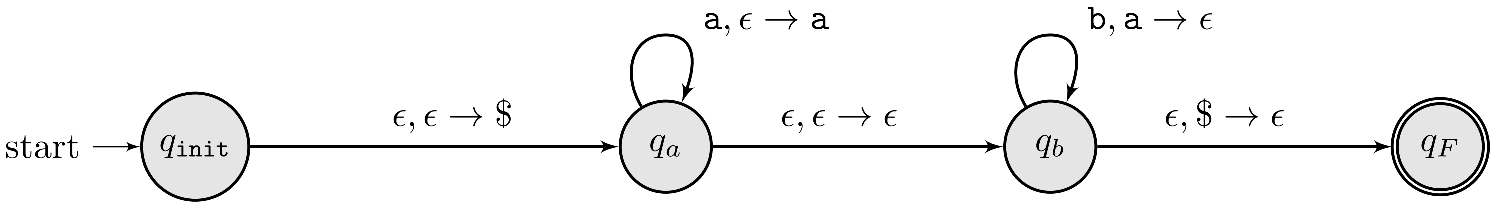
Acceptance example



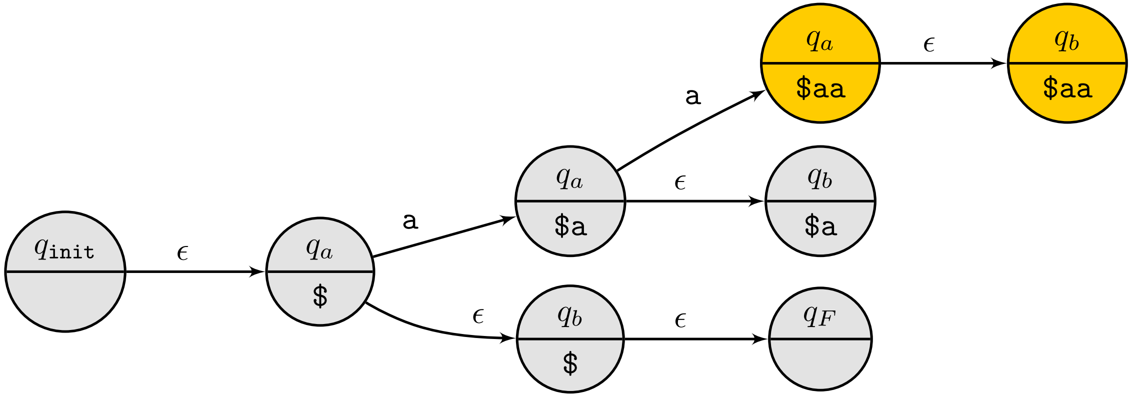
Accepting [aa**ϵ**bb]



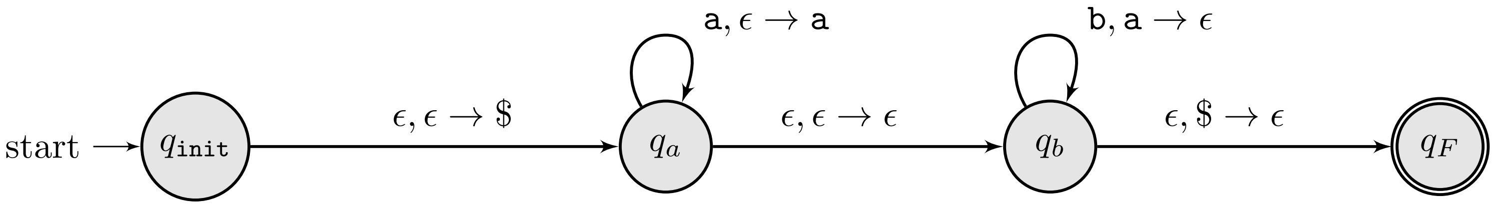
Acceptance example



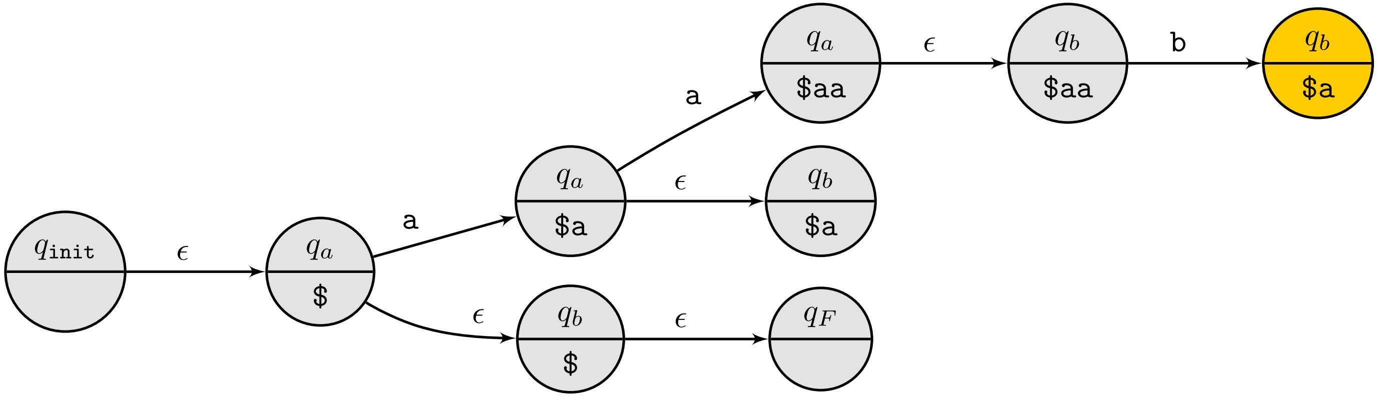
Accepting [aa**b**b]



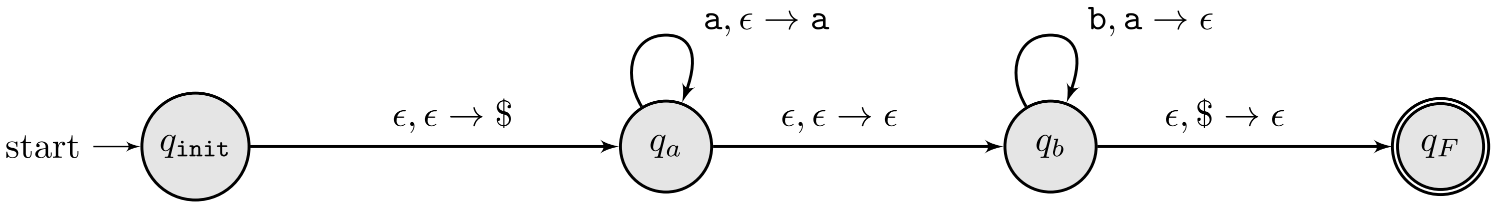
Acceptance example



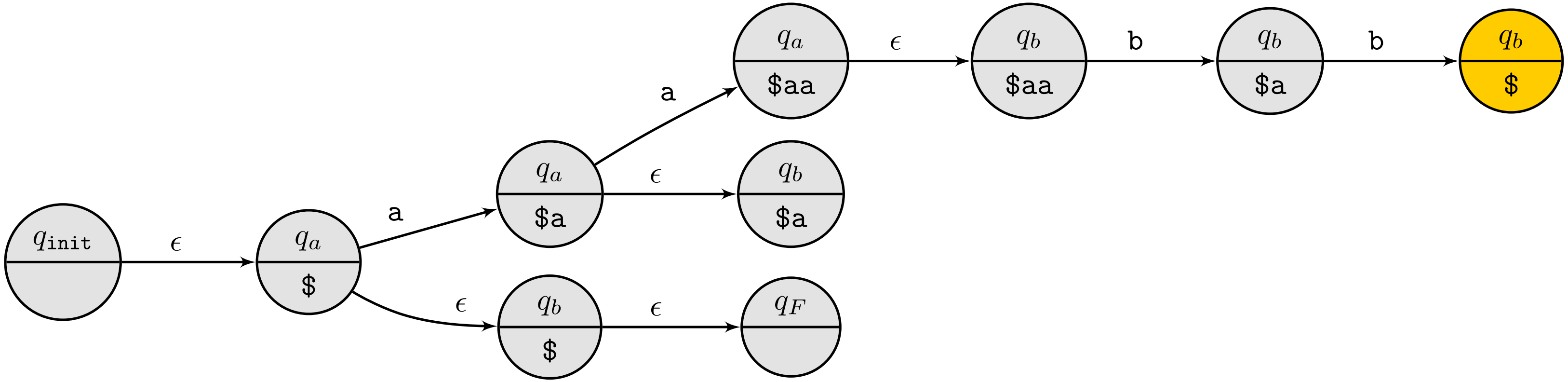
Accepting [aabϵb]



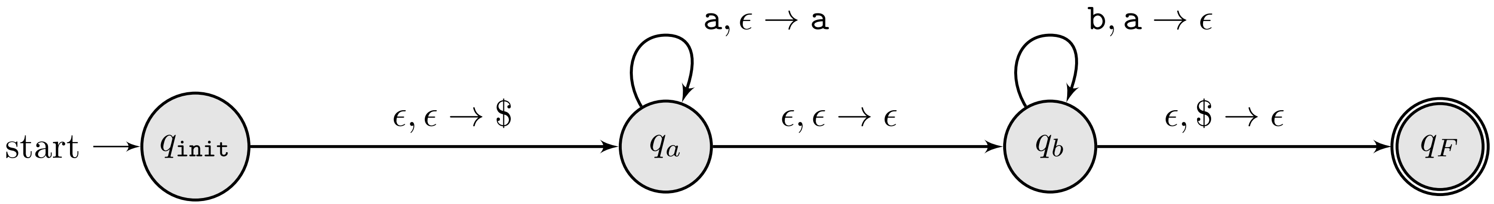
Acceptance example



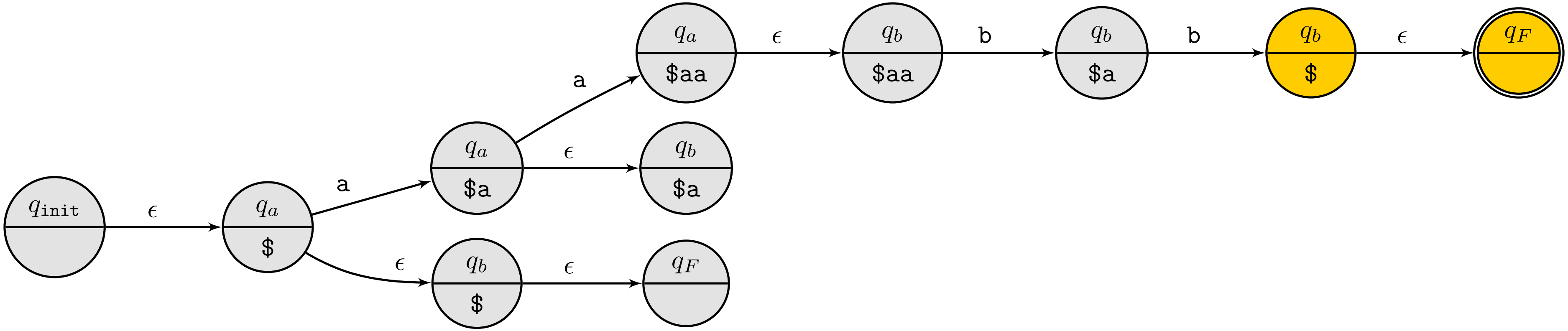
Accepting [aab**b**]



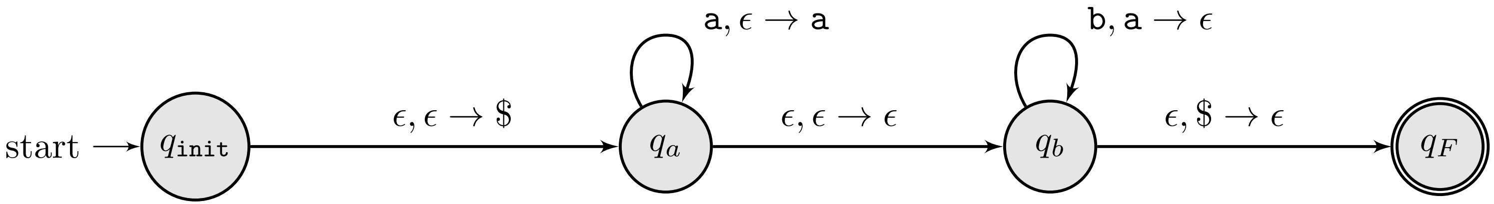
Acceptance example



Accepting [aabb**ϵ**]

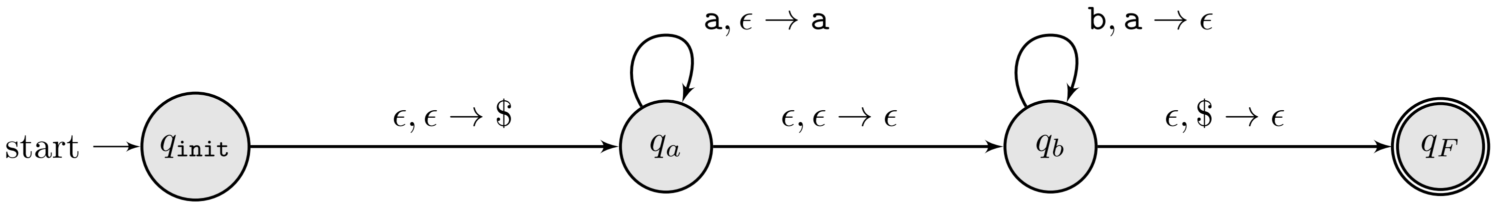


Acceptance example

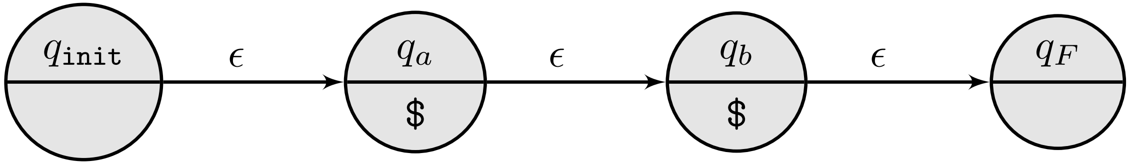


Accepting: bb

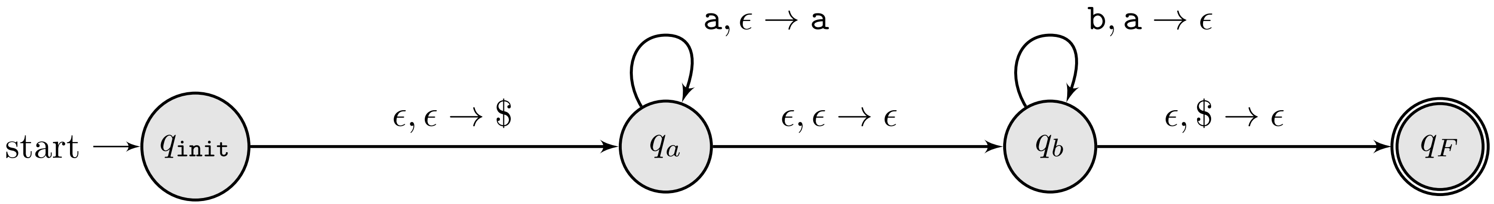
Acceptance example



Accepting: bb

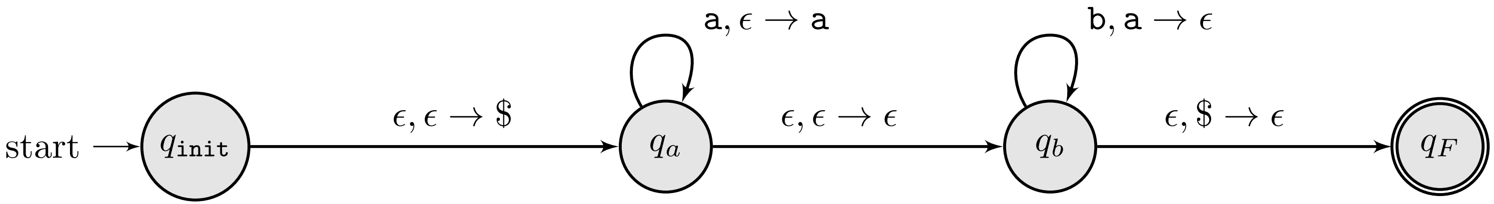


Acceptance example

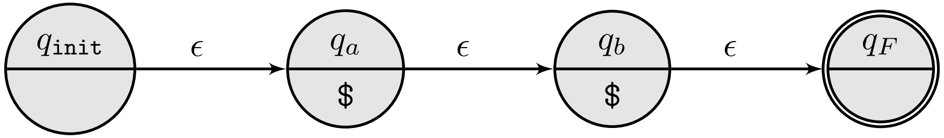


Accepting: $\boldsymbol{\epsilon}$

Acceptance example



Accepting: ϵ



Formalizing a PDA

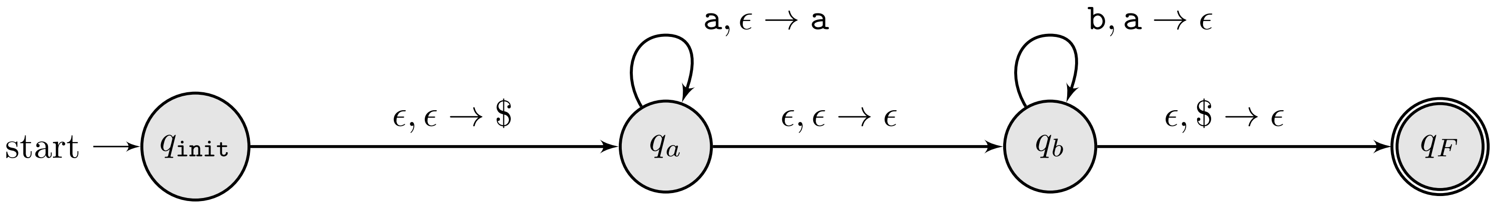
Formalizing a PDA

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. $Q = \{q_{init}, q_a, q_b, q_F\}$
2. $\Sigma = \{a, b\}$
3. $\Gamma = \{a, \$\}$

where δ is defined by branches

$$\delta(q_{init}, \epsilon, \epsilon) = \{(q_a, \$)\}$$
$$\delta(q_a, a, \epsilon) = \{(q_a, a)\}$$
$$\delta(q_a, \epsilon, \epsilon) = \{(q_b, \epsilon)\}$$
$$\delta(q_b, b, a) = \{(q_b, \epsilon)\}$$
$$\delta(q_b, \epsilon, \$) = \{(q_F, \$)\}$$
$$\delta(q, c, s) = \{\} \qquad \text{otherwise}$$

Exercise

Give a PDA for the following grammar

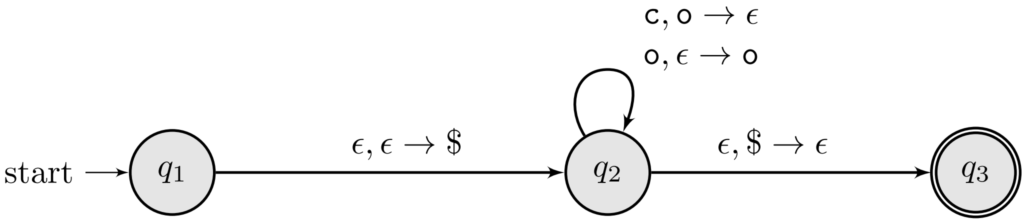
Ballanced parenthesis

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

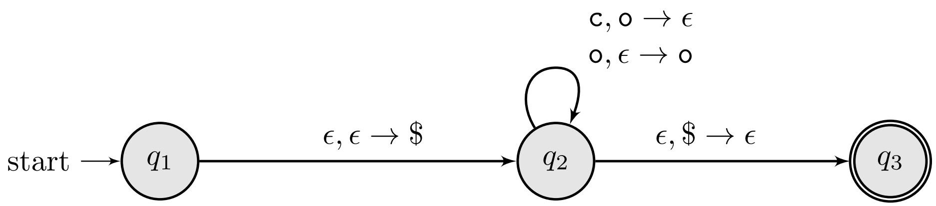
Give a PDA for the following grammar

Ballanced parenthesis

$$C \rightarrow \underline{\mathtt{o}}\ C\ \underline{\mathtt{c}} \mid CC \mid \epsilon$$

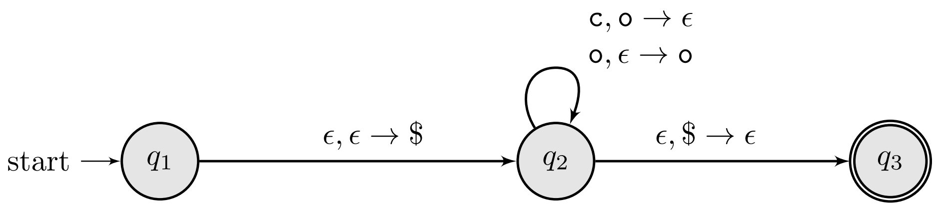


Acceptance

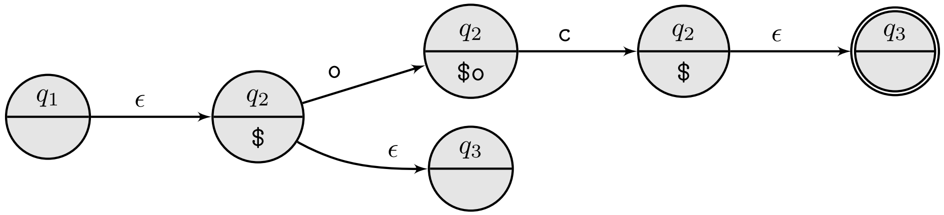


Acceptance: OC

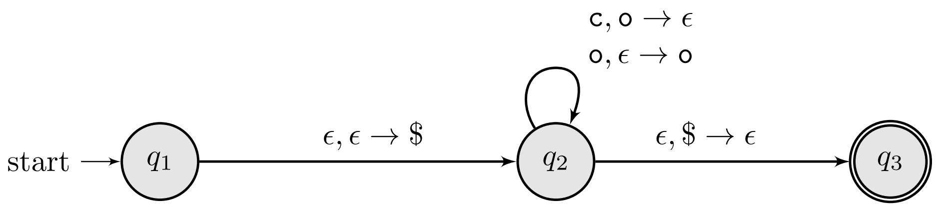
Acceptance



Acceptance: OC

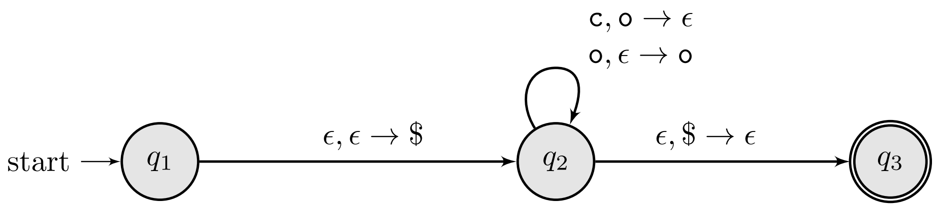


Acceptance

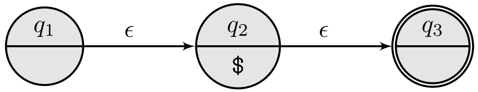


Acceptance: ϵ

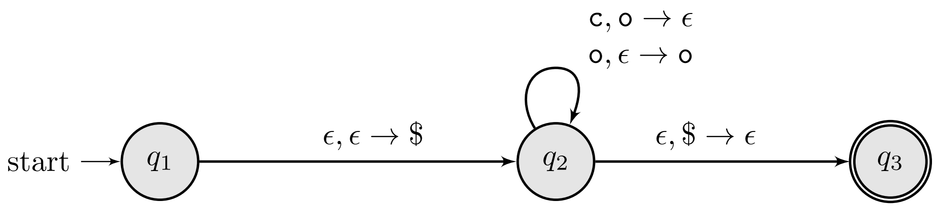
Acceptance



Acceptance: ϵ

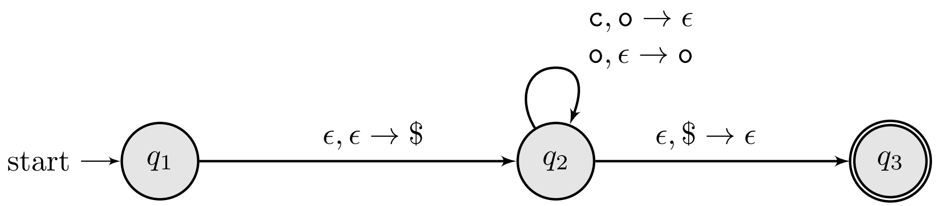


Acceptance

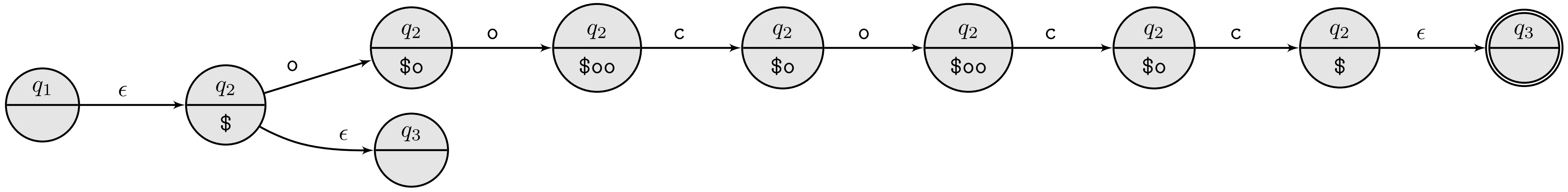


Acceptance: OOCOCC

Acceptance



Acceptance: OOCOCC



Formalization

Formalizing stack operation

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

Pop operation

$$s \rhd \epsilon = s$$
$$n :: s \rhd n = s$$

Examples

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

Push operation

$$s \lhd \epsilon = s$$
$$s \lhd n = n :: s$$

Examples

$$[0,1] \triangleleft \epsilon = [0,1]$$
$$[0,1] \triangleleft \$ = [0,1,\$]$$
$$[] \triangleleft \$ = [\$]$$
$$[0,1] \triangleleft 0 = [0,0,1]$$
$$[0,1] \triangleleft 1 = [1,0,1]$$

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
$ab \triangleleft \$ =$	$ab\$$
$\epsilon \triangleright \$ =$	**undef**
$\epsilon \triangleleft \$ =$	$\$$
$\epsilon \triangleright a =$	**undef**
$\epsilon \triangleleft a =$	a

Formalizing acceptance

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

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.

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

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

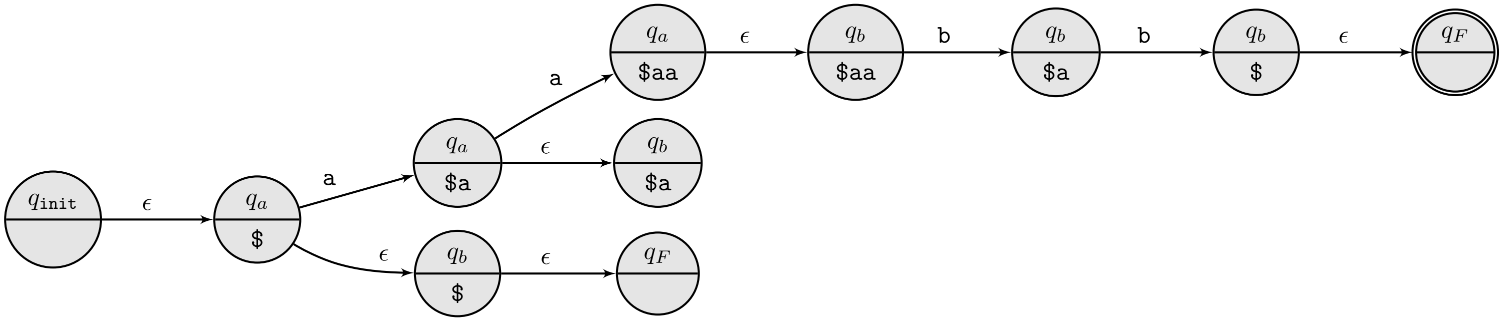
We can build a chain of states as follows

$$(q_{init}, []) \xrightarrow{\epsilon,\epsilon} (q_a, [\$]) \xrightarrow{a,\epsilon} (q_a, [a, \$]) \xrightarrow{a,\epsilon} (q_b, [a, a, \$]) \xrightarrow{\epsilon,\epsilon} (q_b, [a, a, \$]) \xrightarrow{b,a} (q_b, [a, \$]) \xrightarrow{b,a} (q_b, [\$]) \xrightarrow{\epsilon,\$} (q_F, [])$$

Since q_F is a final state, we have that

$$(q_{init}, []) \curvearrowright [a, a, b, b]$$

Recall



Example 2.16

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.

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

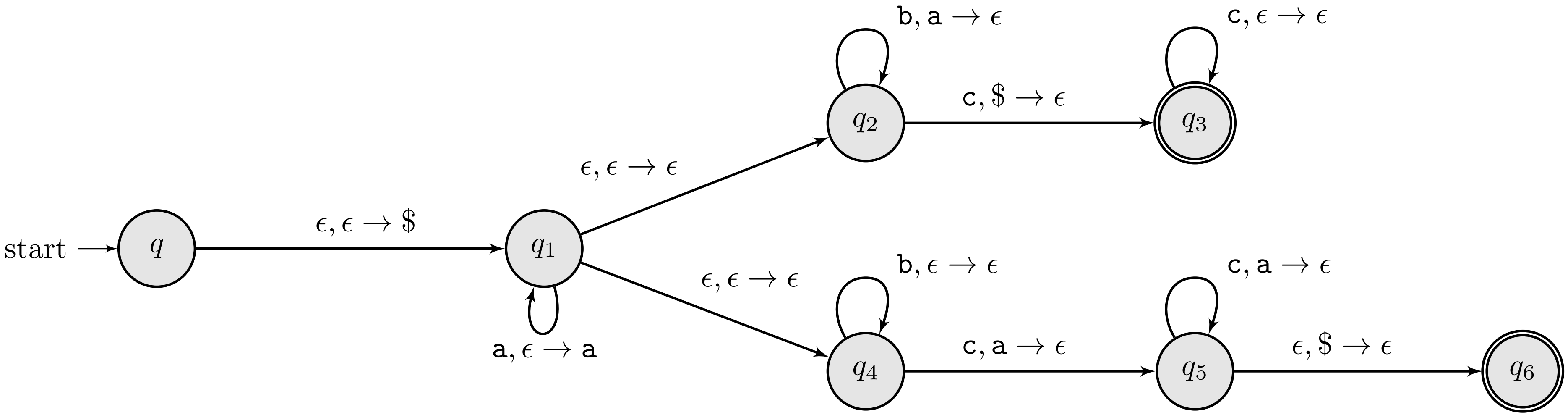
A solution

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

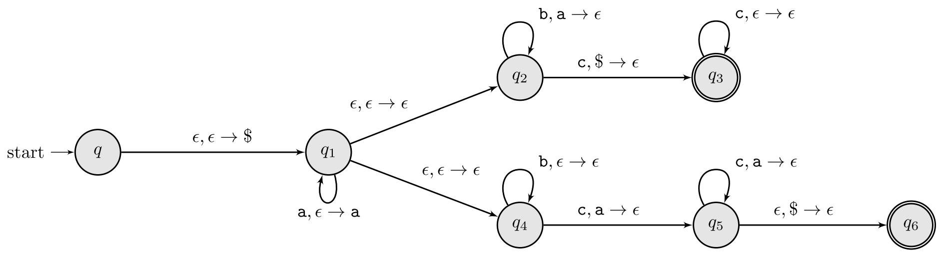
Step 2. Either:

- $(i = j)$ 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

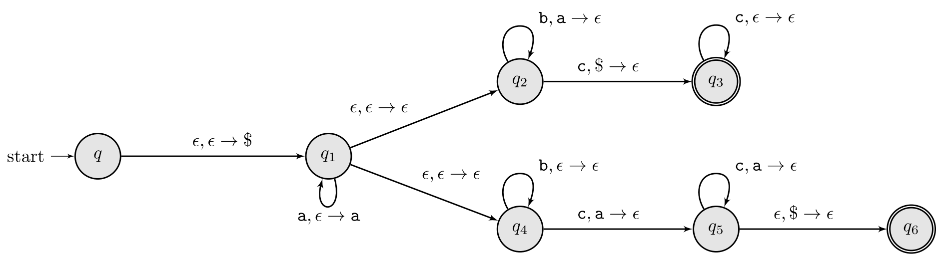
State diagram of Example 2.16

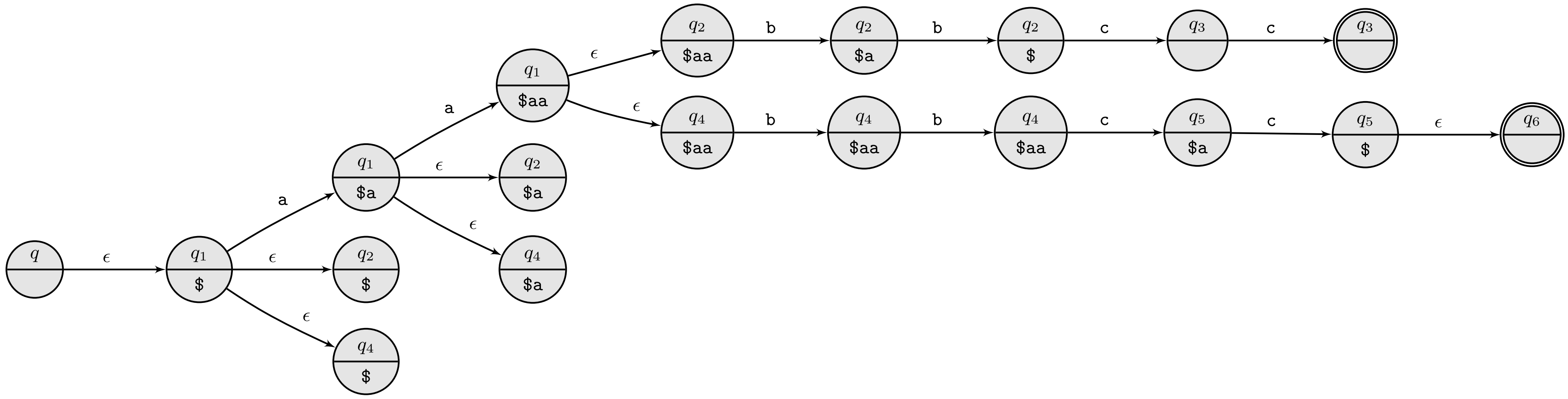


Example 2.16 accept [a, a, b, b, c, c]?

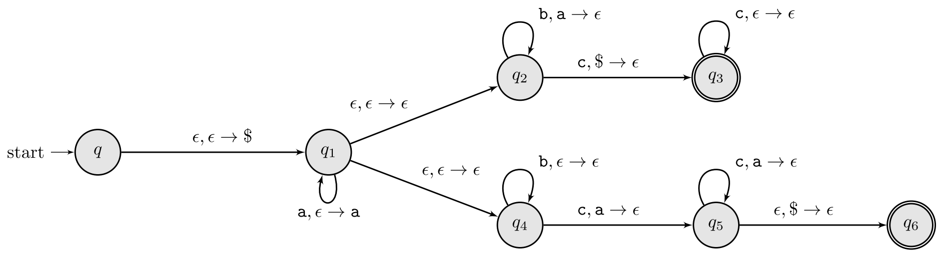


Example 2.16 accept `[a,a,b,b,c,c]`?

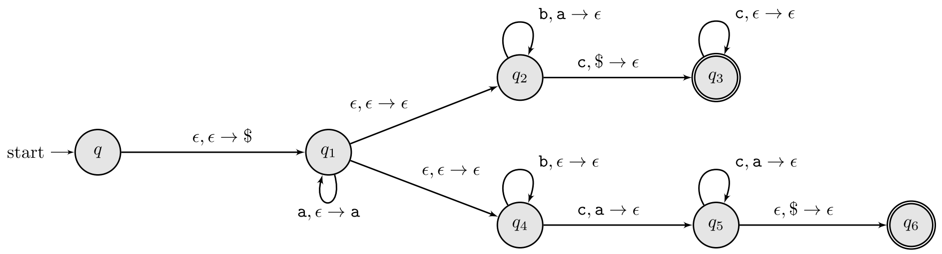


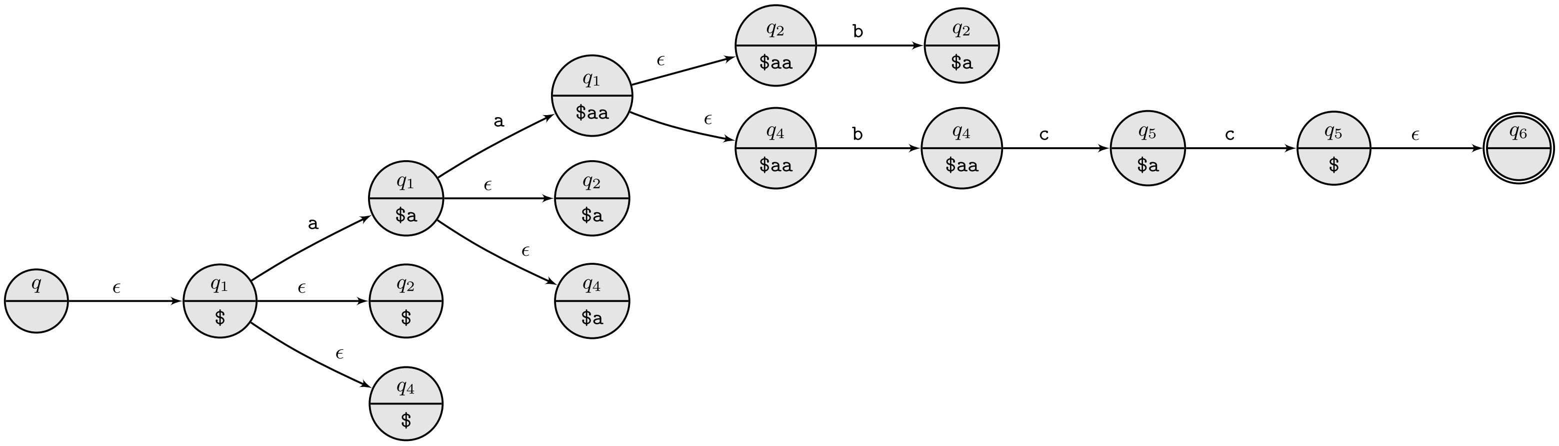


Example 2.16 accept [a,a,b,c,c]?

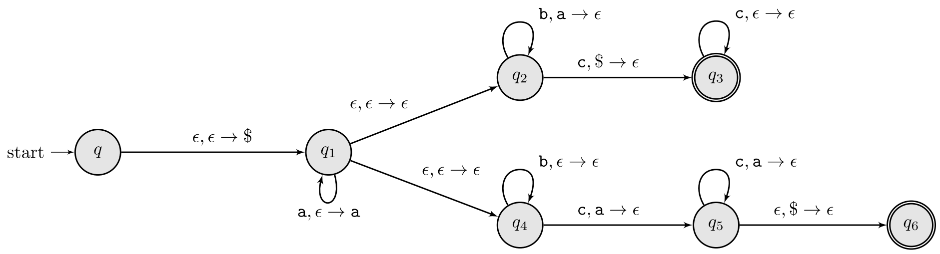


Example 2.16 accept [a,a,b,c,c]?

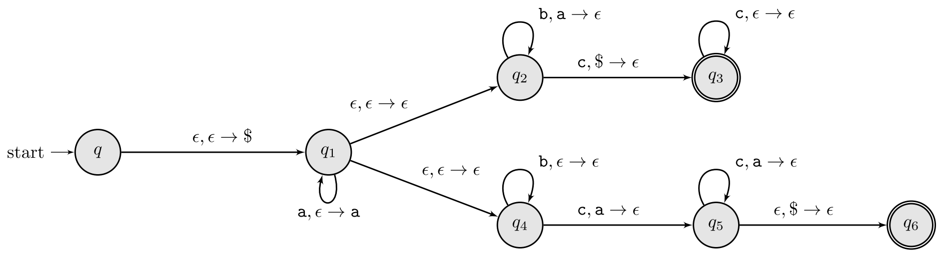


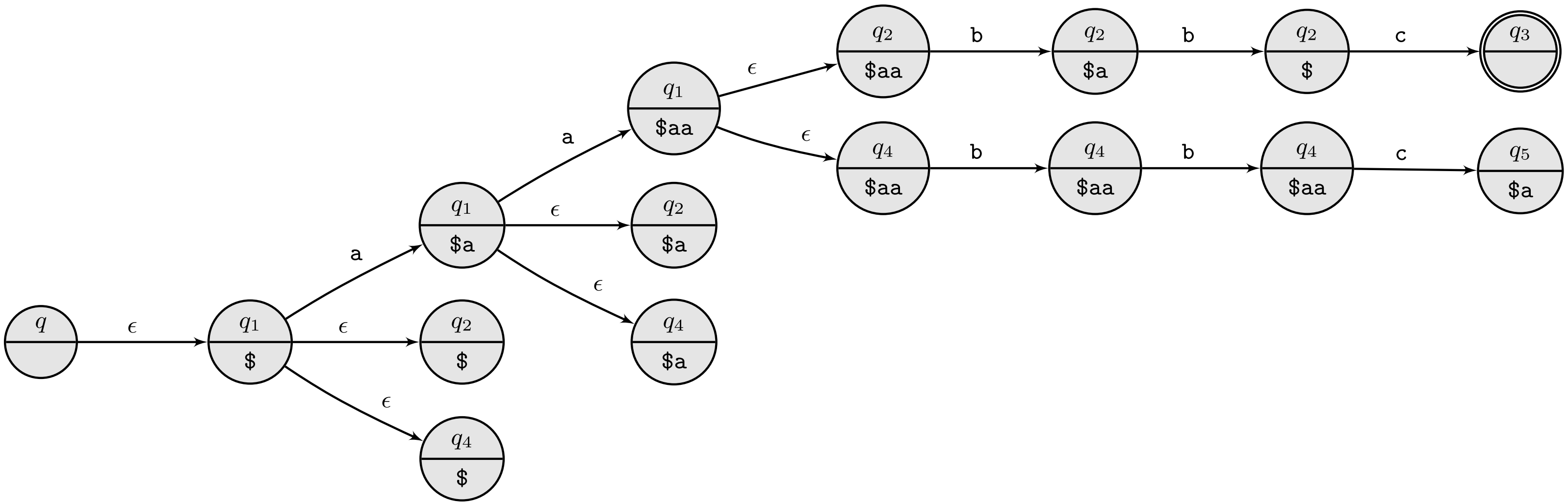


Example 2.16 accept [a,a,b,b,c]?

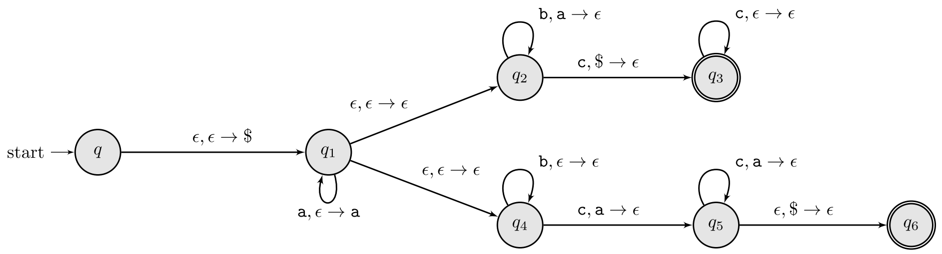


Example 2.16 accept [a,a,b,b,c]?

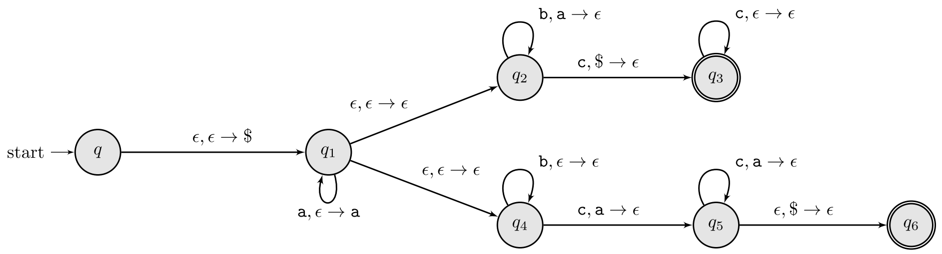


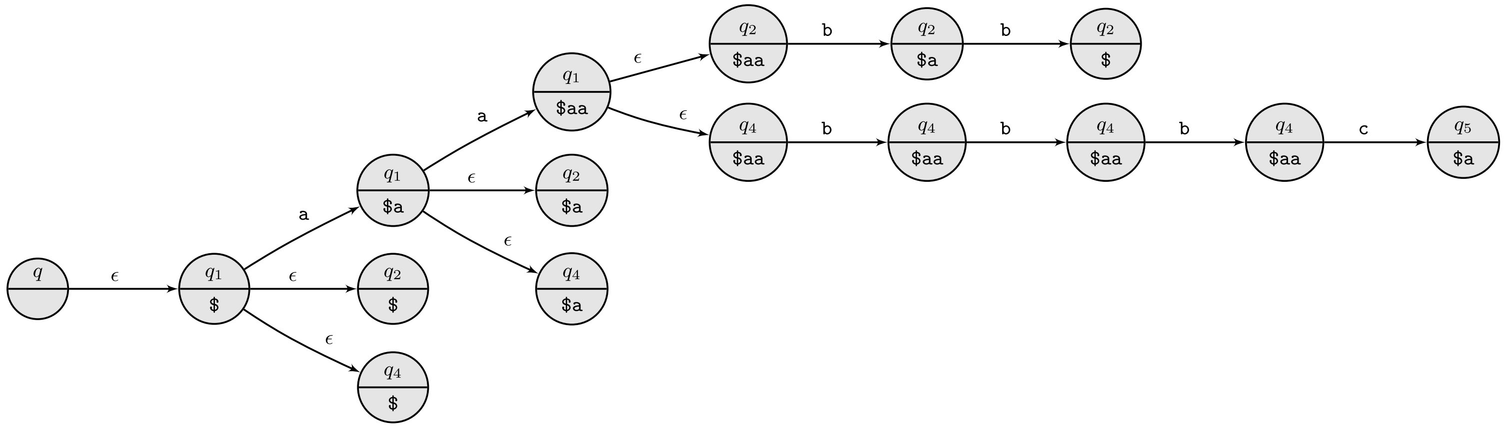


Example 2.16 rejects [a,a,b,b,b,c]?



Example 2.16 rejects [a,a,b,b,b,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

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