CS420

Introduction to the Theory of Computation

Lecture 5: Regular expressions

Tiago Cogumbreiro

Today we will learn...

- Regular expressions
- Soundness: Converting a regular expression into an NFA
- Completeness: Converting an NFA into a regular expression

> Section 1.3

Regular expressions

- An automata describes the **process** of recognizing a language
- For the purpose of characterizing an automata in terms of its recognized language, we do not care how many states, how many transitions)
- When we know the problem, we can devise a domain specific language (DSL) to **abstract** away the internals of a process

Regular expression versus automaton

- A regular expressions specifies what language can be recognized (WHAT)
- An automaton describes a computational mechanism of recognizing a language (HOW)

Regular expressions

Inductive definition

$$R ::= a \mid \epsilon \mid \emptyset \mid R_1 + R_2 \mid R_1 \cdot R_2 \mid R^\star$$

Informal description

A regular expression R is one of the following cases:

- a for language $\{[a]\}$, consists of string $[a]$
- ϵ for language $\{\epsilon\}$, consists of the empty string
- $\emptyset$ for language $\{\}$, ie, the language that does not recognize any string
- $R_1 + R_2$ for the language that results from the union of R_1 with R_2
- $R_1 \cdot R_2$ for the language that results from the concatenation of R_1 with R_2
- $R^\star$ for the language that results from applying the kleene operation on R

Example 1

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \epsilon$$

Example 1

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \epsilon$$

String a and string ϵ.

Formally (as sets)

Example 1

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \epsilon$$

String a and string ϵ.

Formally (as sets)

$$\{a\} \cup \{\epsilon\} = \{a, \epsilon\}$$

Example 1

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \epsilon$$

String a and string ϵ.

Formally (as sets)

$$\{a\} \cup \{\epsilon\} = \{a, \epsilon\}$$

As an NFA

```
union(char(a), empty)
```

Example 1

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \epsilon$$

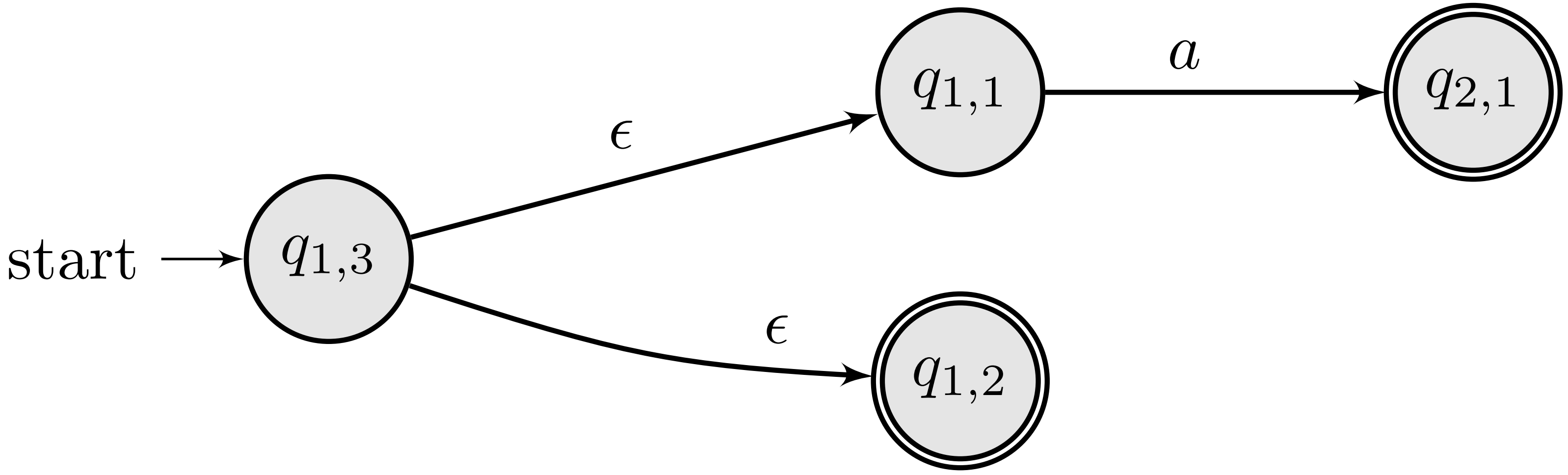
String a and string ϵ.

Formally (as sets)

$$\{a\} \cup \{\epsilon\} = \{a, \epsilon\}$$

As an NFA

```
union(char(a), empty)
```



Example 2

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a \cdot b) + (b \cdot a)$$

Example 2

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a \cdot b) + (b \cdot a)$$

String ab and ba.

Formally (as sets)

Example 2

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a \cdot b) + (b \cdot a)$$

String ab and ba.

Formally (as sets)

$$\{ab\} \cup \{ba\} = \{ab, ba\}$$

Example 2

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a \cdot b) + (b \cdot a)$$

String ab and ba.

Formally (as sets)

$$\{ab\} \cup \{ba\} = \{ab, ba\}$$

As an NFA

```
union(
  concat(char(a), char(b))
  concat(char(b), char(a)))
```

Example 2

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a \cdot b) + (b \cdot a)$$

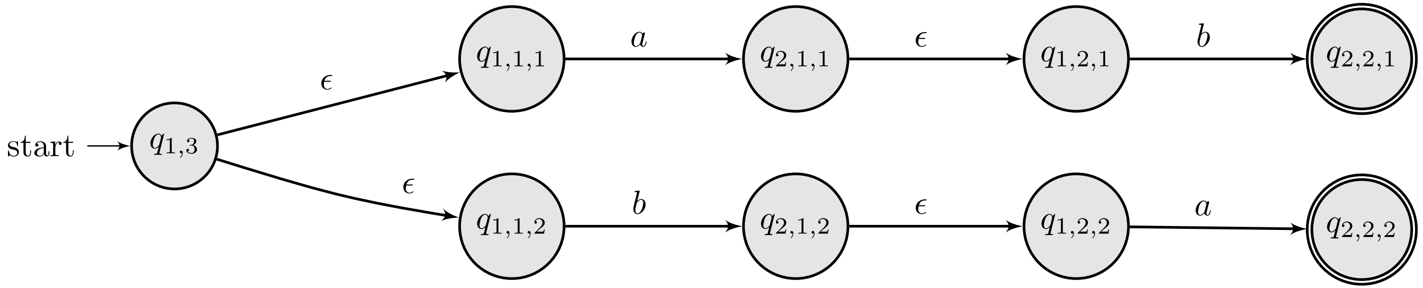
String ab and ba.

Formally (as sets)

$$\{ab\} \cup \{ba\} = \{ab, ba\}$$

As an NFA

```
union(
  concat(char(a), char(b))
  concat(char(b), char(a)))
```



Example 3

Regular expression

Let $\Sigma = \{a, b\}$.

$$b^\star \cdot a \cdot b^\star$$

Example 3

Regular expression

Let $\Sigma = \{a, b\}$.

$$b^\star \cdot a \cdot b^\star$$

Strings with exactly a single a.

Formally (as sets)

Example 3

Regular expression

Let $\Sigma = \{a, b\}$.

$$b^\star \cdot a \cdot b^\star$$

Strings with exactly a single a.

Formally (as sets)

$$\{b\}^\star \cdot \{a\} \cdot \{b\}^\star$$

Example 3

Regular expression

Let $\Sigma = \{a, b\}$.

$$b^\star \cdot a \cdot b^\star$$

Strings with exactly a single a.

Formally (as sets)

$$\{b\}^\star \cdot \{a\} \cdot \{b\}^\star$$

As an NFA

Example 3

Regular expression

Let $\Sigma = \{a, b\}$.

$$b^\star \cdot a \cdot b^\star$$

Strings with exactly a single a.

Formally (as sets)

$$\{b\}^\star \cdot \{a\} \cdot \{b\}^\star$$

As an NFA

```
concat(
  concat(
    star(char(b)),
    char(a)
  ),
  star(char(b)))
```

Example 3

Regular expression

Let $\Sigma = \{a, b\}$.

$$b^\star \cdot a \cdot b^\star$$

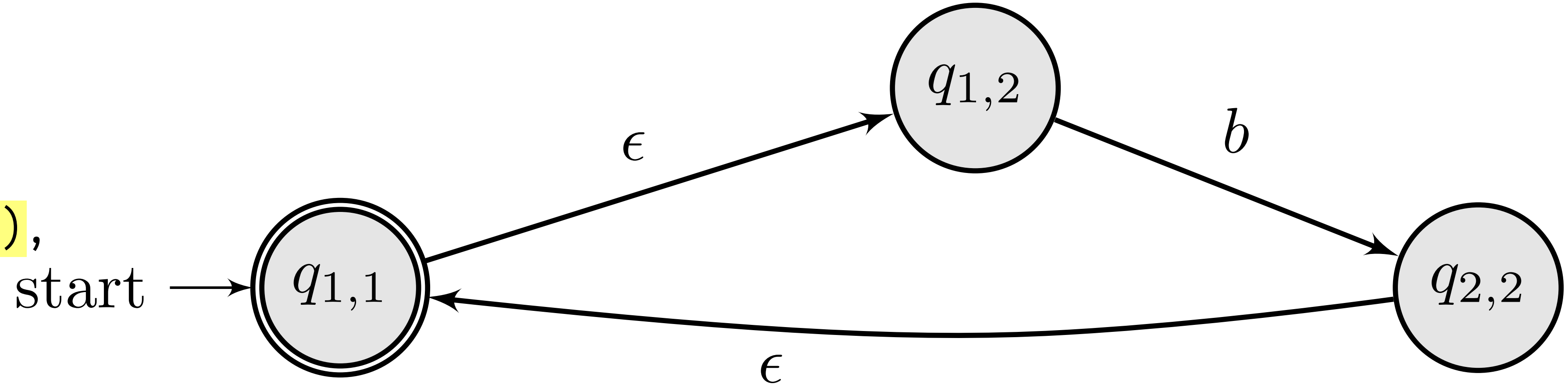
Formally (as sets)

$$\{b\}^\star \cdot \{a\} \cdot \{b\}^\star$$

Strings with exactly a single a.

As an NFA

```
concat(
  concat(
    star(char(b)),
    char(a)
  ),
  star(char(b)))
```



Example 3

Regular expression

Let $\Sigma = \{a, b\}$.

$$b^\star \cdot a \cdot b^\star$$

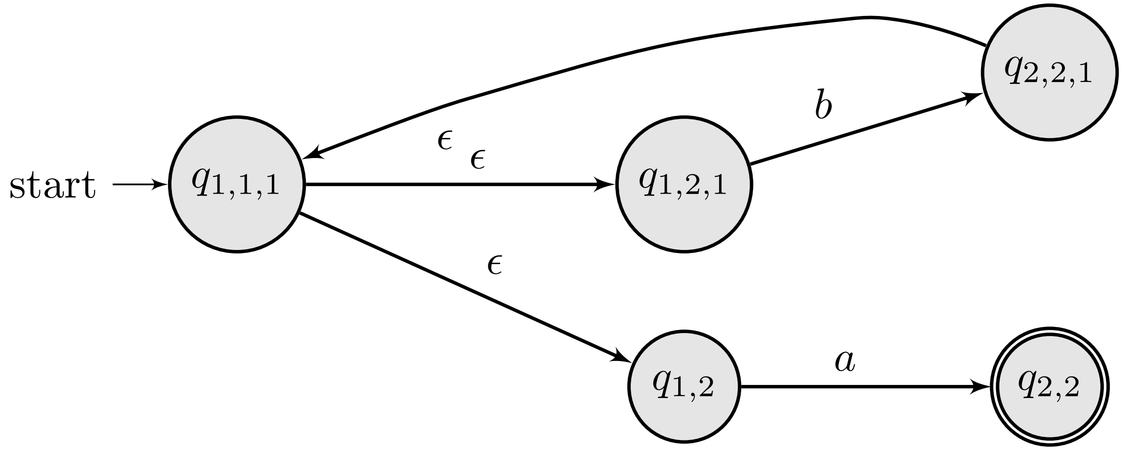
Formally (as sets)

$$\{b\}^\star \cdot \{a\} \cdot \{b\}^\star$$

Strings with exactly a single a.

As an NFA

```
concat(
  concat(
    star(char(b)),
    char(a)
  ),
  star(char(b)))
```



Example 3

Regular expression

Let $\Sigma = \{a, b\}$.

$$b^\star \cdot a \cdot b^\star$$

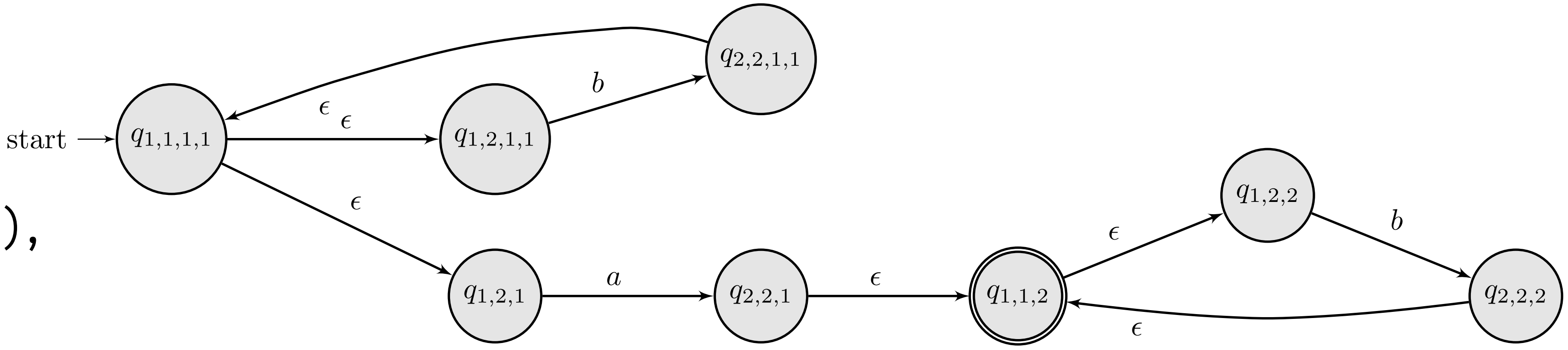
Formally (as sets)

$$\{b\}^\star \cdot \{a\} \cdot \{b\}^\star$$

Strings with exactly a single a.

As an NFA

```
concat(
  concat(
    star(char(b)),
    char(a)
  ),
  star(char(b)))
```



Example 4

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a + b)^\star \cdot b \cdot (a + b)^\star$$

Example 4

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a + b)^\star \cdot b \cdot (a + b)^\star$$

Strings with at least one b.

Formally (as sets)

Example 4

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a + b)^{\star} \cdot b \cdot (a + b)^{\star}$$

Strings with at least one b.

Formally (as sets)

$$\left(\{a\} \cup \{b\}\right)^{\star} \cdot \{b\} \cdot \left(\{a\} \cup \{b\}\right)^{\star}$$
$$= \{a, b\}^{\star} \cdot \{b\} \cdot \{a, b\}^{\star}$$

Example 4

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a + b)^\star \cdot b \cdot (a + b)^\star$$

Strings with at least one b.

Formally (as sets)

$$\left(\{a\} \cup \{b\}\right)^\star \cdot \{b\} \cdot \left(\{a\} \cup \{b\}\right)^\star$$
$$= \{a, b\}^\star \cdot \{b\} \cdot \{a, b\}^\star$$

As an NFA

```
concat(
  star(union(char(a), char(b))),
  concat(
    char(b),
    star(union(char(a), char(b)))))
```

Example 4

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a + b)^\star \cdot b \cdot (a + b)^\star$$

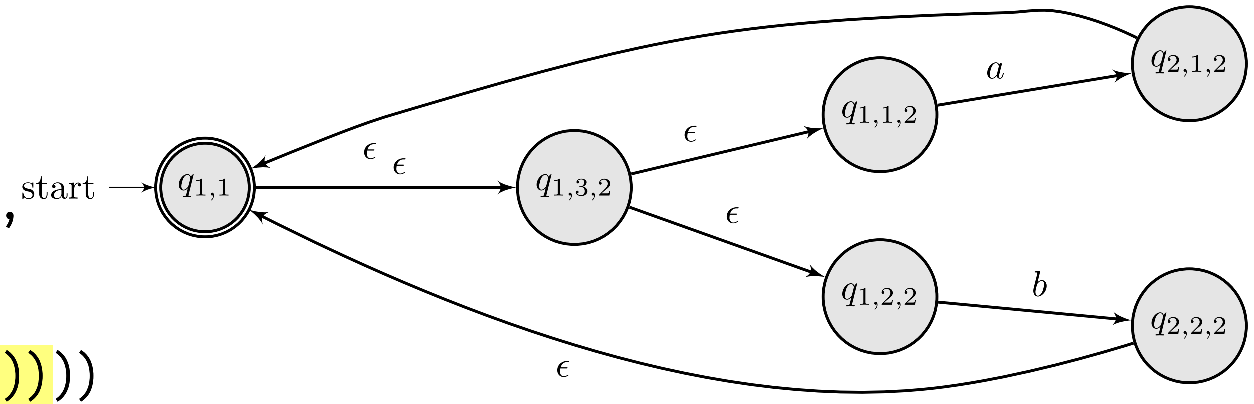
Strings with at least one b.

Formally (as sets)

$$(\{a\} \cup \{b\})^\star \cdot \{b\} \cdot (\{a\} \cup \{b\})^\star$$
$$= \{a, b\}^\star \cdot \{b\} \cdot \{a, b\}^\star$$

As an NFA

```
concat(
  star(union(char(a), char(b))),
  concat(
    char(b),
    star(union(char(a), char(b)))))
```



Example 4

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a + b)^\star \cdot b \cdot (a + b)^\star$$

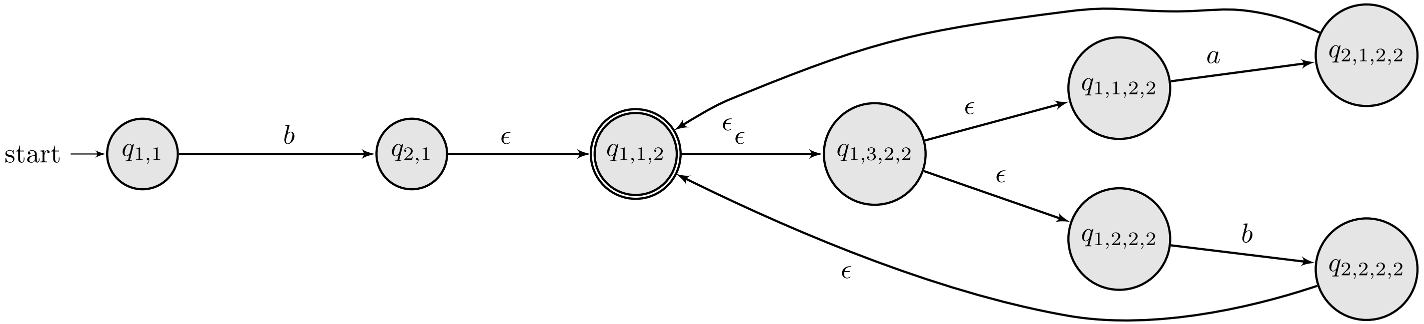
Strings with at least one b.

Formally (as sets)

$$(\{a\} \cup \{b\})^\star \cdot \{b\} \cdot (\{a\} \cup \{b\})^\star$$
$$= \{a, b\}^\star \cdot \{b\} \cdot \{a, b\}^\star$$

As an NFA

```
concat(
  star(union(char(a), char(b))),
  concat(
    char(b),
    star(union(char(a), char(b)))))
```



Example 4

Regular expression

Let $\Sigma = \{a, b\}$.

$$(a+b)^\star \cdot b \cdot (a+b)^\star$$

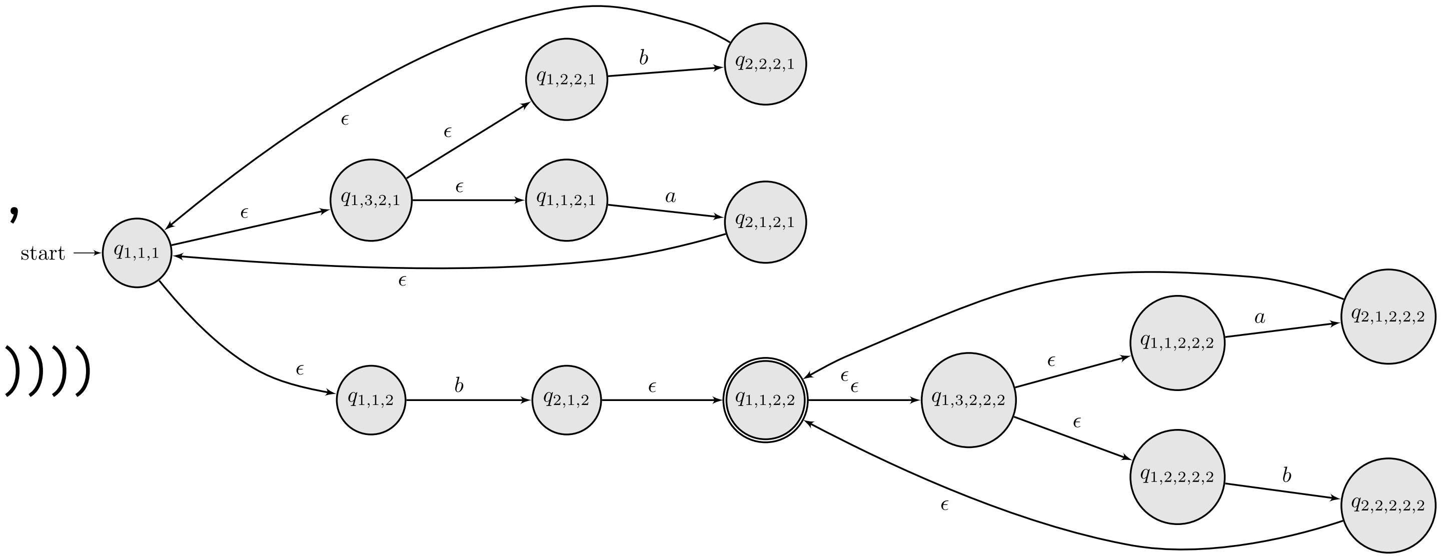
Strings with at least one b.

Formally (as sets)

$$(\{a\} \cup \{b\})^\star \cdot \{b\} \cdot (\{a\} \cup \{b\})^\star$$
$$= \{a, b\}^\star \cdot \{b\} \cdot \{a, b\}^\star$$

As an NFA

```
concat(
  star(union(char(a), char(b))),
  concat(
    char(b),
    star(union(char(a), char(b)))))
```



Example 5

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \emptyset$$

Example 5

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \emptyset$$

String a.

Formally (as sets)

Example 5

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \emptyset$$

String a.

Formally (as sets)

$$\{a\} \cup \emptyset = \{a\}$$

Example 5

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \emptyset$$

String a.

Formally (as sets)

$$\{a\} \cup \emptyset = \{a\}$$

As an NFA

```
union(char(a), nil)
```

Example 5

Regular expression

Let $\Sigma = \{a, b\}$.

$$a + \emptyset$$

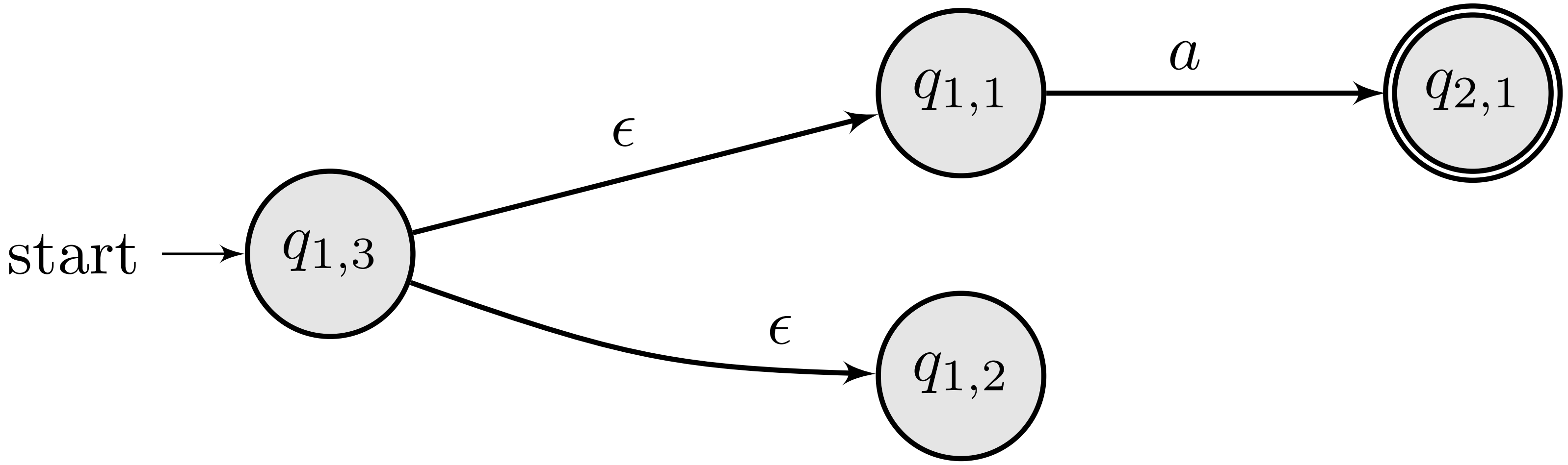
String a.

Formally (as sets)

$$\{a\} \cup \emptyset = \{a\}$$

As an NFA

```
union(char(a), nil)
```



Example 6

Regular expression

Let $\Sigma = \{a, b\}$.

$$a \cdot \emptyset$$

Example 6

Regular expression

Let $\Sigma = \{a, b\}$.

$$a \cdot \emptyset$$

The empty set.

Formally (as sets)

$\{a\} \cdot \emptyset =$

Example 6

Regular expression

Let $\Sigma = \{a, b\}$.

$$a \cdot \emptyset$$

The empty set.

Formally (as sets)

$\{a\} \cdot \emptyset = \emptyset$

> **Why?** Because,
> $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$

Example 6

Regular expression

Let $\Sigma = \{a, b\}$.

$$a \cdot \emptyset$$

The empty set.

Formally (as sets)

$$\{a\} \cdot \emptyset = \emptyset$$

> **Why?** Because,
> $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$

As an NFA

```
concat(char(a), nil)
```

Example 6

Regular expression

Let $\Sigma = \{a, b\}$.

$$a \cdot \emptyset$$

The empty set.

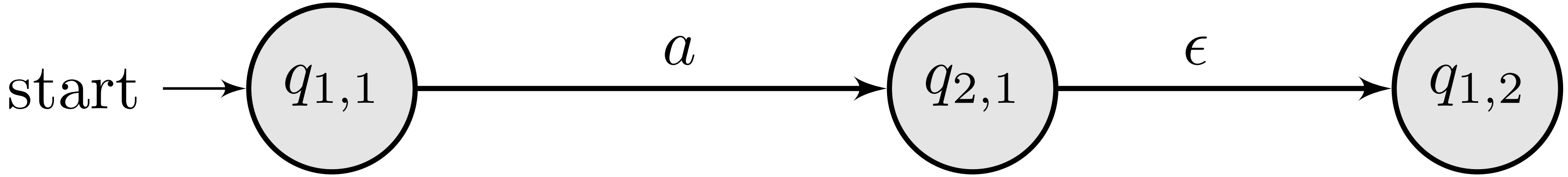
Formally (as sets)

$$\{a\} \cdot \emptyset = \emptyset$$

> **Why?** Because,
> $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$

As an NFA

`concat(char(a), nil)`



> Note the absence of accepted states.

Example 7

Regular expression

Let $\Sigma = \{a, b\}$.

$$a^\star + b^\star$$

Example 7

Regular expression

Let $\Sigma = \{a, b\}$.

$$a^\star + b^\star$$

String where all letters are the same.

Formally (as sets)

$$\{a\}^\star \cup \{b\}^\star$$

Example 7

Regular expression

Let $\Sigma = \{a, b\}$.

$$a^\star + b^\star$$

String where all letters are the same.

Formally (as sets)

$$\{a\}^\star \cup \{b\}^\star$$

As an NFA

```
union(
  star(char(a)),
  star(char(b)))
```

Example 7

Regular expression

Let $\Sigma = \{a, b\}$.

$$a^\star + b^\star$$

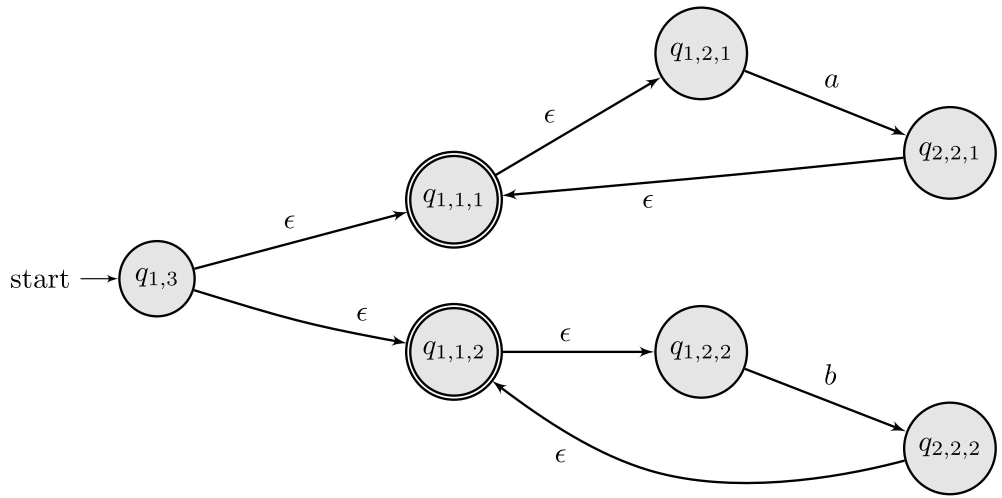
String where all letters are the same.

Formally (as sets)

$$\{a\}^\star \cup \{b\}^\star$$

As an NFA

```
union(
  star(char(a)),
  star(char(b)))
```



Formalizing the regular expressions

The language of a regular expression

- $L(\underline{a}) =$

Formalizing the regular expressions

The language of a regular expression

- $L(\underline{a}) = \{a\}$
- $L(\underline{\epsilon}) =$

Formalizing the regular expressions

The language of a regular expression

- $L(\underline{a}) = \{a\}$
- $L(\underline{\epsilon}) = \{\epsilon\}$
- $L(\underline{\emptyset}) =$

Formalizing the regular expressions

The language of a regular expression

- $L(\underline{a}) = \{a\}$
- $L(\underline{\epsilon}) = \{\epsilon\}$
- $L(\underline{\emptyset}) = \emptyset$
- $L(\underline{R_1 + R_2}) =$

Formalizing the regular expressions

The language of a regular expression

- $L(\underline{a}) = \{a\}$
- $L(\underline{\epsilon}) = \{\epsilon\}$
- $L(\underline{\emptyset}) = \emptyset$
- $L(\underline{R_1 + R_2}) = L(R_1) \cup L(R_2)$
- $L(\underline{R_1 \cdot R_2}) =$

Formalizing the regular expressions

The language of a regular expression

- $L(\underline{a}) = \{a\}$
- $L(\underline{\epsilon}) = \{\epsilon\}$
- $L(\underline{\emptyset}) = \emptyset$
- $L(\underline{R_1 + R_2}) = L(R_1) \cup L(R_2)$
- $L(\underline{R_1 \cdot R_2}) = L(\underline{R_1}) \cdot L(\underline{R_2})$
- $L(\underline{R^\star}) =$

Formalizing the regular expressions

The language of a regular expression

- $L(\underline{a}) = \{a\}$
- $L(\underline{\epsilon}) = \{\epsilon\}$
- $L(\underline{\emptyset}) = \emptyset$
- $L(\underline{R_1 + R_2}) = L(\underline{R_1}) \cup L(\underline{R_2})$
- $L(\underline{R_1 \cdot R_2}) = L(\underline{R_1}) \cdot L(\underline{R_2})$
- $L(\underline{R^\star}) = L(\underline{R})^\star$

> We underline and color red regular expressions, so as to distinguish regular expressions from set-theory expressions (in black). Set theory is our **meta**-theory.

What is a regular expression?

- A regular expression is just a syntactic term
- Specifies the language accepted by some automaton
- We say that R represents a language

Why not use set theory? Because less is more

What is a regular expression?

- A regular expression is just a syntactic term
- Specifies the language accepted by some automaton
- We say that R represents a language

Why not use set theory? Because less is more

- Having a syntactic term that represents a set of operations is a powerful abstraction

> We can understand what are the **minimal** operators needed to represent **all** DFAs/NFAs

Soundess

All Regexes have an equivalent NFA

`REGEX → NFA`

All Regexes have an equivalent NFA

Lemma 1.55

If $L(R) = L_1$, then $L(\mathrm{NFA}(R)) = L_1$.

Given an alphabet Σ

- $\mathrm{NFA}(\underline{a}) =$

All Regexes have an equivalent NFA

Lemma 1.55

If $L(R) = L_1$, then $L(\mathrm{NFA}(R)) = L_1$.

Given an alphabet Σ

- $\mathrm{NFA}(\underline{a}) = \mathrm{char}_\Sigma(a)$
- $\mathrm{NFA}(\underline{\epsilon}) =$

All Regexes have an equivalent NFA

Lemma 1.55

If $L(R) = L_1$, then $L(\text{NFA}(R)) = L_1$.

Given an alphabet Σ

- $\text{NFA}(\underline{a}) = \text{char}_\Sigma(a)$
- $\text{NFA}(\underline{\epsilon}) = \text{empty}_\Sigma$
- $\text{NFA}(\underline{\emptyset}) =$

All Regexes have an equivalent NFA

Lemma 1.55

If $L(R) = L_1$, then $L(\mathrm{NFA}(R)) = L_1$.

Given an alphabet Σ

- $\mathrm{NFA}(\underline{a}) = \mathrm{char}_\Sigma(a)$
- $\mathrm{NFA}(\underline{\epsilon}) = \mathrm{empty}_\Sigma$
- $\mathrm{NFA}(\underline{\emptyset}) = \mathrm{nil}_\Sigma$
- $\mathrm{NFA}(\underline{R_1 \cup R_2}) =$

All Regexes have an equivalent NFA

Lemma 1.55

If $L(R) = L_1$, then $L(\mathbf{NFA}(R)) = L_1$.

Given an alphabet Σ

- $\mathrm{NFA}(\underline{a}) = \mathrm{char}_\Sigma(a)$
- $\mathrm{NFA}(\underline{\epsilon}) = \mathrm{empty}_\Sigma$
- $\mathrm{NFA}(\underline{\emptyset}) = \mathrm{nil}_\Sigma$
- $\mathrm{NFA}(\underline{R_1 \cup R_2}) = \mathrm{union}(\mathrm{NFA}(R_1), \mathrm{NFA}(R_2))$
- $\mathrm{NFA}(\underline{R_1 \cdot R_2}) =$

All Regexes have an equivalent NFA

Lemma 1.55

If $L(R) = L_1$, then $L(\mathrm{NFA}(R)) = L_1$.

Given an alphabet Σ

- $\mathrm{NFA}(\underline{a}) = \mathrm{char}_\Sigma(a)$
- $\mathrm{NFA}(\underline{\epsilon}) = \mathrm{empty}_\Sigma$
- $\mathrm{NFA}(\underline{\emptyset}) = \mathrm{nil}_\Sigma$
- $\mathrm{NFA}(\underline{R_1 \cup R_2}) = \mathrm{union}(\mathrm{NFA}(R_1), \mathrm{NFA}(R_2))$
- $\mathrm{NFA}(\underline{R_1 \cdot R_2}) = \mathrm{concat}(\mathrm{NFA}(\underline{R_1}), \mathrm{NFA}(\underline{R_2}))$
- $\mathrm{NFA}(\underline{R^\star}) =$

All Regexes have an equivalent NFA

Lemma 1.55

If $L(R) = L_1$, then $L(\mathrm{NFA}(R)) = L_1$.

Given an alphabet Σ

- $\mathrm{NFA}(\underline{a}) = \mathrm{char}_\Sigma(a)$
- $\mathrm{NFA}(\underline{\epsilon}) = \mathrm{empty}_\Sigma$
- $\mathrm{NFA}(\underline{\emptyset}) = \mathrm{nil}_\Sigma$
- $\mathrm{NFA}(\underline{R_1 \cup R_2}) = \mathrm{union}(\mathrm{NFA}(R_1), \mathrm{NFA}(R_2))$
- $\mathrm{NFA}(\underline{R_1 \cdot R_2}) = \mathrm{concat}(\mathrm{NFA}(\underline{R_1}), \mathrm{NFA}(\underline{R_2}))$
- $\mathrm{NFA}(\underline{R^\star}) = \mathrm{star}(\mathrm{NFA}(\underline{R}))$

All Regexes have an equivalent NFA

Lemma 1.55

If $L(R) = L_1$, then $L(\mathrm{NFA}(R)) = L_1$.

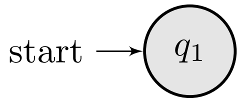
Given an alphabet Σ

- $\mathrm{NFA}(\underline{a}) = \mathrm{char}_\Sigma(a)$
- $\mathrm{NFA}(\underline{\epsilon}) = \mathrm{empty}_\Sigma$
- $\mathrm{NFA}(\underline{\emptyset}) = \mathrm{nil}_\Sigma$
- $\mathrm{NFA}(\underline{R_1 \cup R_2}) = \mathrm{union}(\mathrm{NFA}(R_1), \mathrm{NFA}(R_2))$
- $\mathrm{NFA}(\underline{R_1 \cdot R_2}) = \mathrm{concat}(\mathrm{NFA}(\underline{R_1}), \mathrm{NFA}(\underline{R_2}))$
- $\mathrm{NFA}(\underline{R^\star}) = \mathrm{star}(\mathrm{NFA}(\underline{R}))$

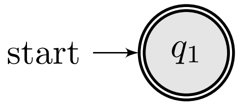
(Proof follows by induction on the structure of R.)

Recap

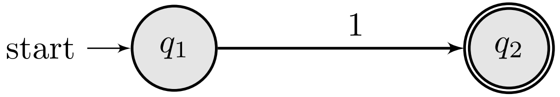
NFA($\emptyset$)



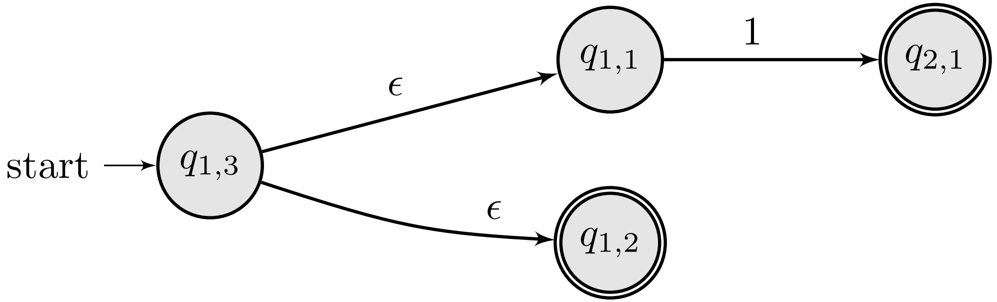
NFA(ϵ)



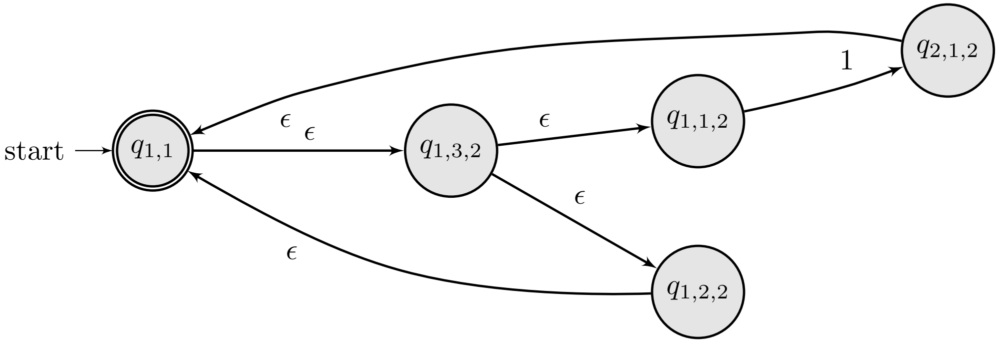
NFA(1)



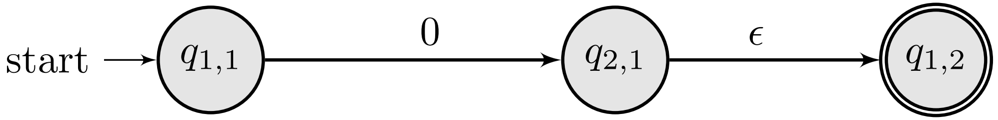
NFA($1 + \epsilon$)



NFA($(1 + \epsilon)^\star$)



NFA ($0 \cdot \epsilon$)



Completeness

All NFAs have an equivalent Regex

NFA $\rightarrow$ REGEX

Completeness

All NFAs have an equivalent Regex

Why is this result important?

Completeness

All NFAs have an equivalent Regex

Why is this result important?

> If we can derive an equivalent regular expression from any NFA, then our regular expression are enough to describe whatever can be described using finite automatons.

Overview:

Converting an NFA into a regular expression

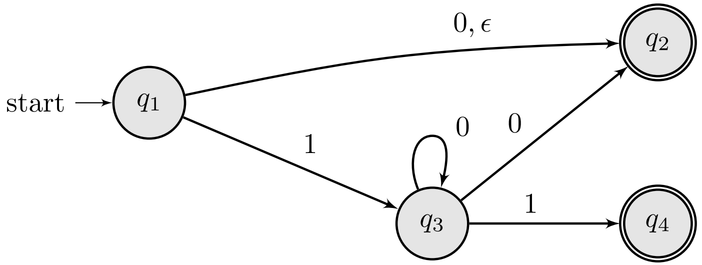
> There are many algorithms of converting an NFA into a Regex. Here is the algorithm we find in the book.

1. Wrap the NFA
2. Convert the NFA into a GNFA
3. Reduce the GNFA
4. Extract the Regex

Step 1: wrap the NFA

> Given an NFA N, add two new states q_{start} and q_{end} such that q_{start} transitions via ϵ to the initial state of N, and every accepted state of N transitions to q_{end} via ϵ. State q_{end} becomes the new accepted state.

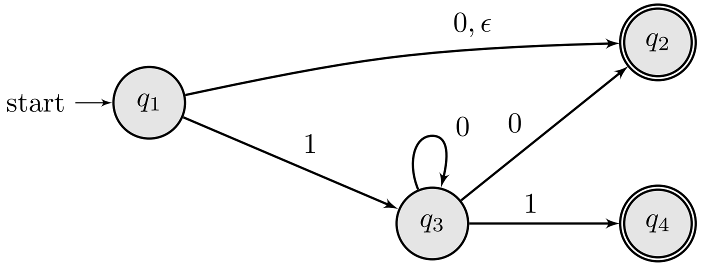
Input



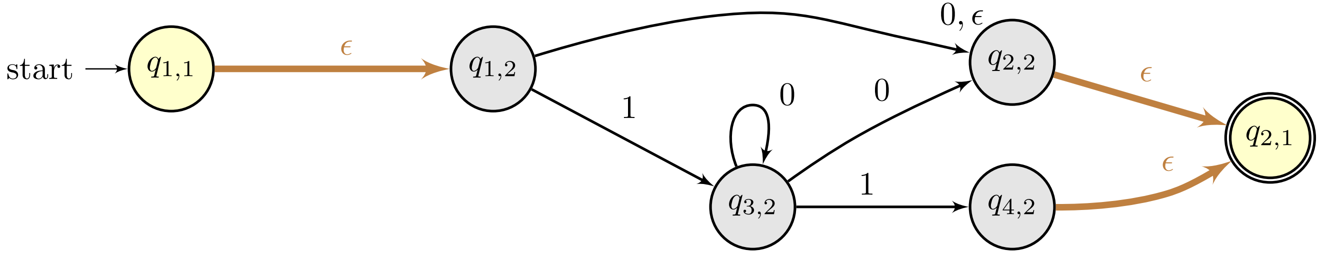
Step 1: wrap the NFA

> Given an NFA N, add two new states q_{start} and q_{end} such that q_{start} transitions via ϵ to the initial state of N, and every accepted state of N transitions to q_{end} via ϵ. State q_{end} becomes the new accepted state.

Input



Output

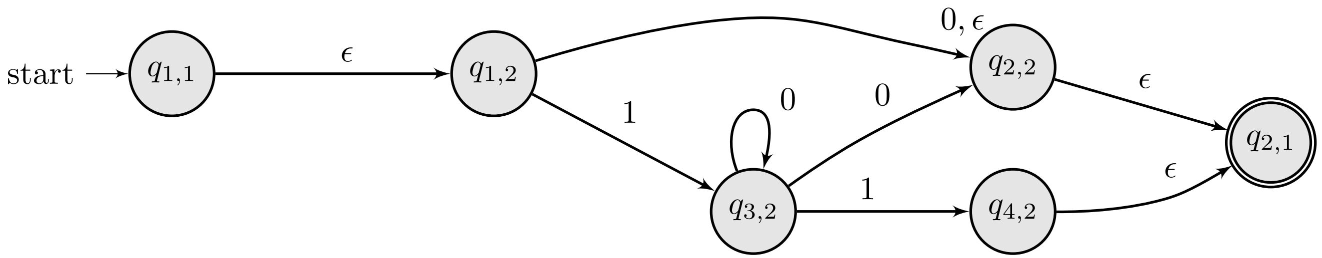


Step 2: Convert an NFA into a GNFA

A GNFA has regular expressions in the transitions, rather than the inputs.

> For every edge with $a_1, \ldots, a_n$ convert into $a_1 + \cdots + a_n$

Input

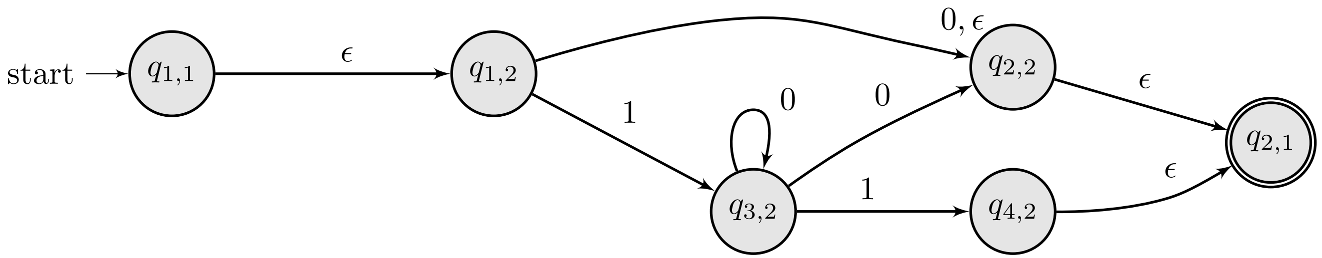


Step 2: Convert an NFA into a GNFA

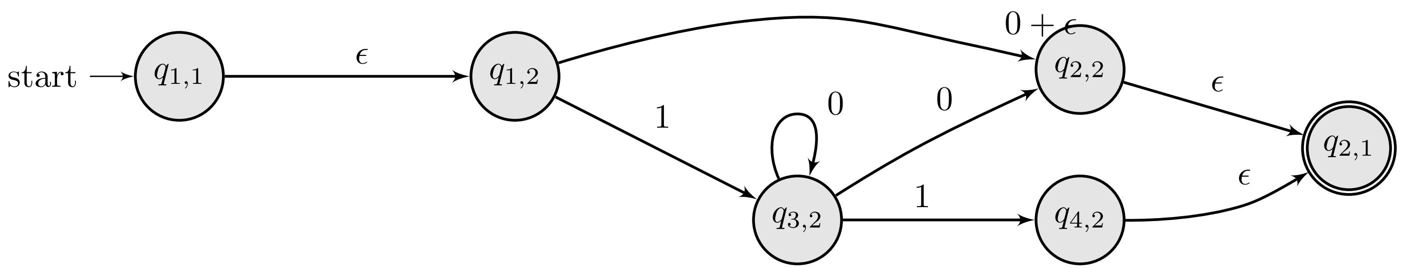
A GNFA has regular expressions in the transitions, rather than the inputs.

> For every edge with $a_1, \ldots, a_n$ convert into $a_1 + \cdots + a_n$

Input



Output



Step 3: Reduce the GNFA

While there are more than 2 states:

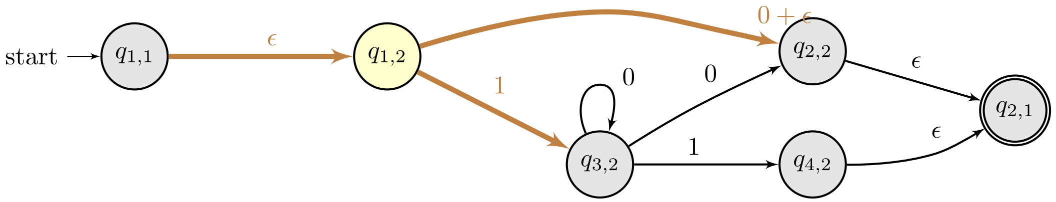
- pick a state and its incoming/outgoing edges, and convert it to transitions

Step 3.1: compress state $q_{1,2}$

$$\text{compress}(q_{1,1} \xrightarrow{\epsilon} q_{1,2} \xrightarrow{0+\epsilon} q_{2,2}) = q_{1,1} \xrightarrow{\epsilon\cdot(0+\epsilon)} q_{2,2}$$

$$\text{compress}(q_{1,1} \xrightarrow{\epsilon} q_{1,2} \xrightarrow{1} q_{3,2}) = q_{1,1} \xrightarrow{\epsilon\cdot 1} q_{3,2}$$

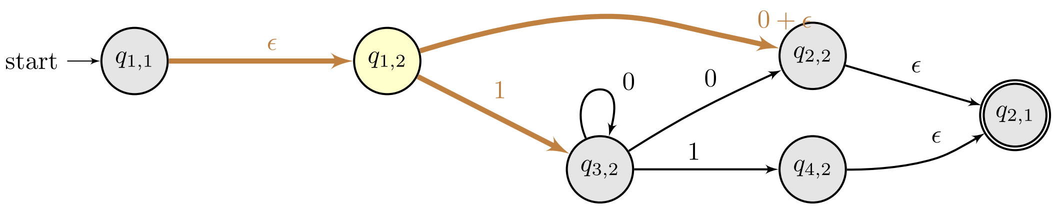
Input



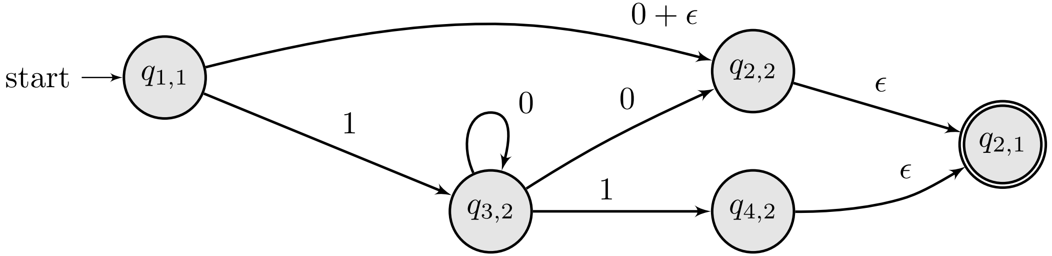
Step 3.1: compress state $q_{1,2}$

$$\text{compress}(q_{1,1} \xrightarrow{\epsilon} q_{1,2} \xrightarrow{0+\epsilon} q_{2,2}) = q_{1,1} \xrightarrow{\epsilon\cdot(0+\epsilon)} q_{2,2}$$
$$\text{compress}(q_{1,1} \xrightarrow{\epsilon} q_{1,2} \xrightarrow{1} q_{3,2}) = q_{1,1} \xrightarrow{\epsilon\cdot 1} q_{3,2}$$

Input



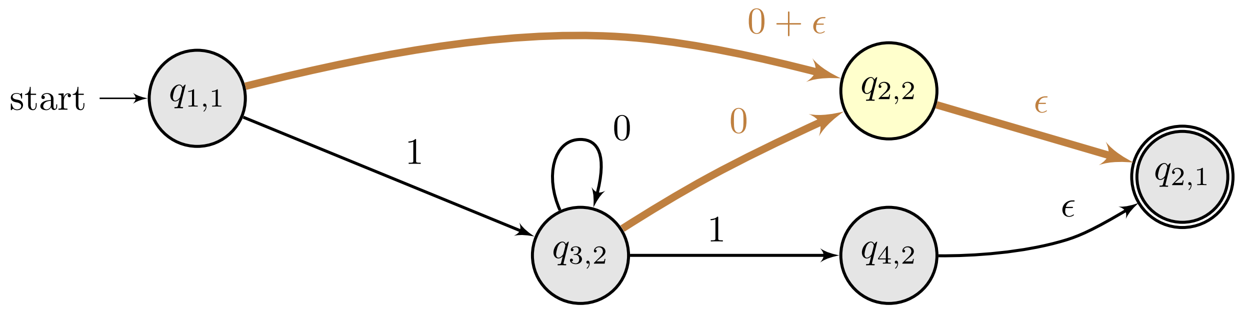
Output



> Each state that connects to $q_{1,2}$ must connect to every state that $q_{1,2}$ connects to. Som $q_{1,1}$ must connect with $q_{2,2}$ and $q_{1,1}$ must connect with $q_{3,2}$.

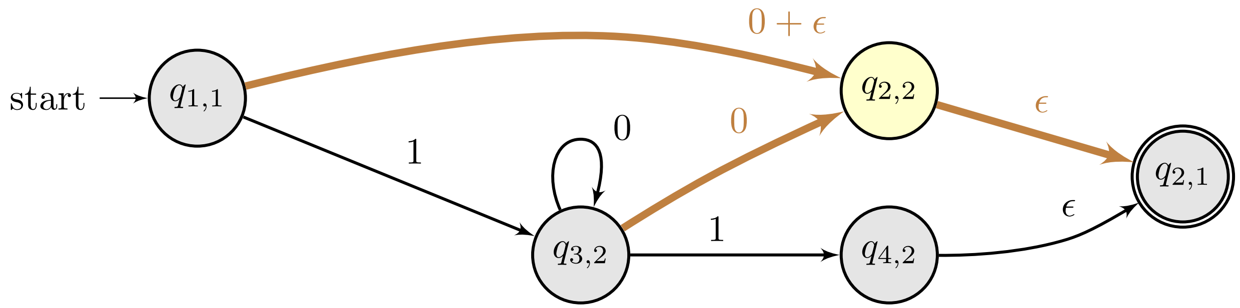
Step 3.2: compress state $q_{2,2}$

Input

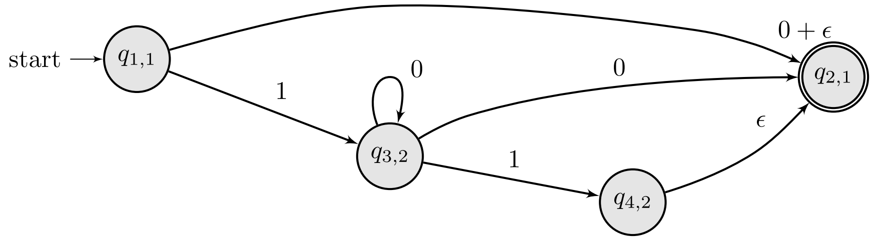


Step 3.2: compress state $q_{2,2}$

Input



Output



$$\text{compress}(q_{1,1} \xrightarrow{0+\epsilon} q_{2,2} \xrightarrow{\epsilon} q_{2,1}) = q_{1,1} \xrightarrow{(0+\epsilon)\cdot\epsilon} q_{2,2}$$

$$\text{compress}(q_{3,2} \xrightarrow{0} q_{2,2} \xrightarrow{\epsilon} q_{2,1}) = q_{3,2} \xrightarrow{0\cdot\epsilon} q_{2,1}$$

Each state that connects to $q_{2,2}$ must connect to every state that $q_{2,2}$ connects to. Som $q_{1,1}$ must connect with $q_{2,1}$ and $q_{3,2}$ must connect with $q_{2,1}$.

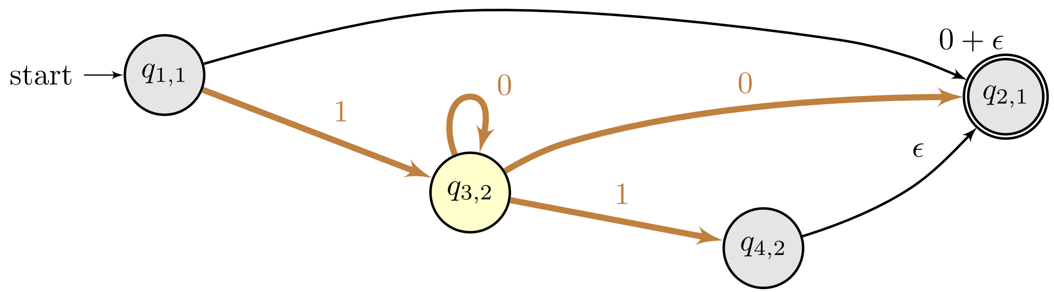
Step 3.3: compress state $q_{3,2}$

After compressing a state, we must merge the new node with any old node (in red).

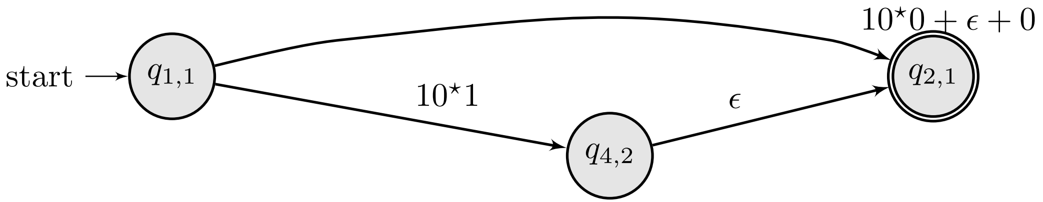
$$\text{compress}(q_{1,1} \xrightarrow{1} q_{3,2} \xrightarrow{0} q_{3,2} \xrightarrow{0} q_{2,1}) + q_{1,1} \xrightarrow{0+\epsilon} q_{2,1} = q_{1,1} \xrightarrow{(10^\star 0)+(0+\epsilon)} q_{2,2}$$

$$\text{compress}(q_{1,1} \xrightarrow{1} q_{3,2} \xrightarrow{0} q_{3,2} \xrightarrow{1} q_{4,2}) = q_{3,2} \xrightarrow{10^\star 1} q_{2,1}$$

Input



Output

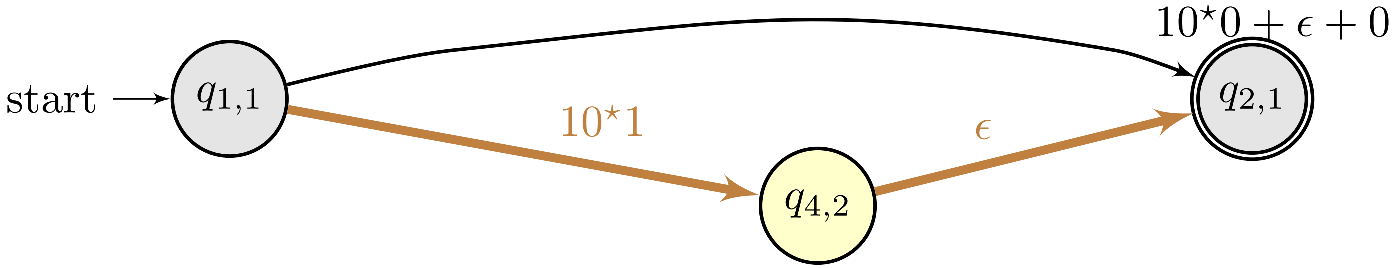


Step 3.3: compress state $q_{4,2}$

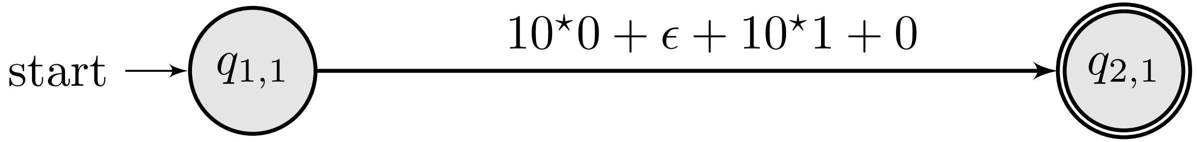
After compressing a state, we must merge the new node with any old node (in red).

$$\text{compress}\left(q_{1,1} \xrightarrow{10^\star 1} q_{4,2} \xrightarrow{\epsilon} q_{2,1}\right) + q_{1,1} \xrightarrow{10^\star 1 + 0 + \epsilon} q_{2,1} = q_{1,1} \xrightarrow{\left(10^\star 1 \cdot \epsilon\right) + \left(10^\star 0 + 0 + \epsilon\right)} q_{2,2}$$

Input



Output

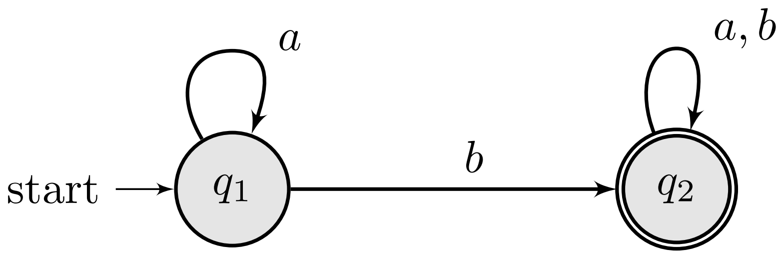


Result: $10^\star 1 + 10^\star 0 + 0 + \epsilon$

Exercise 1.66

Convert a DFA into a Regex

1. Convert the DFA into an NFA (same)

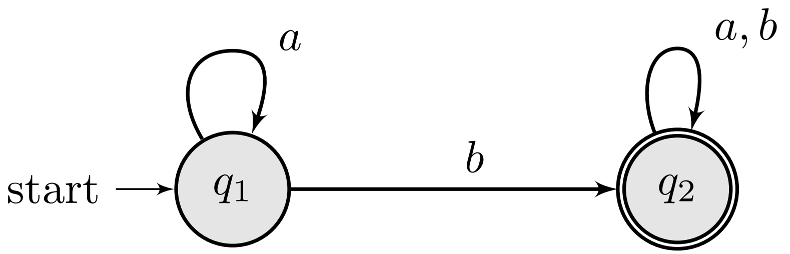


2. Wrap the NFA

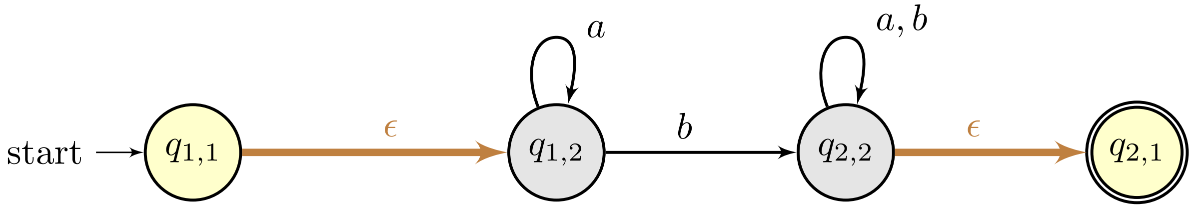
Exercise 1.66

Convert a DFA into a Regex

1. Convert the DFA into an NFA (same)



2. Wrap the NFA

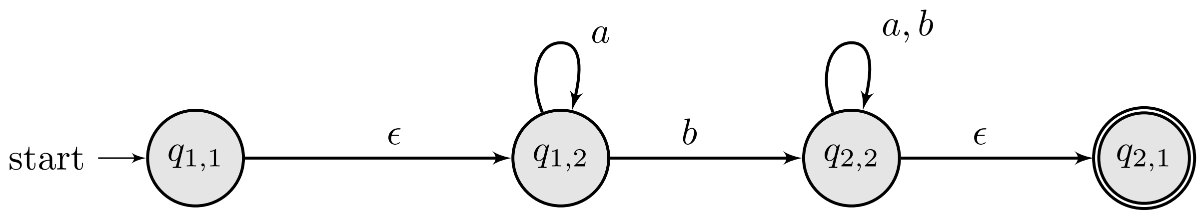


Exercise 1.66

Convert a DFA into a Regex

3. Convert NFA into GNFA

Before

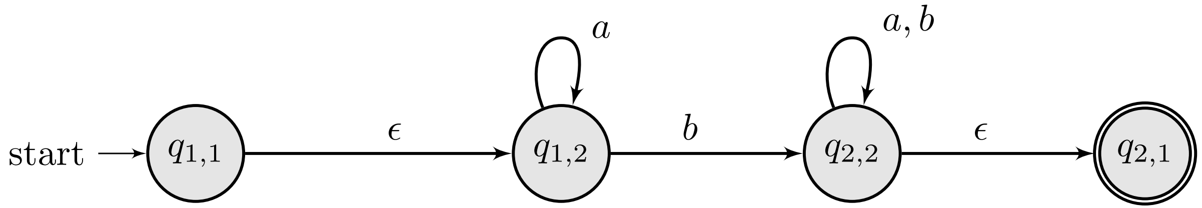


Exercise 1.66

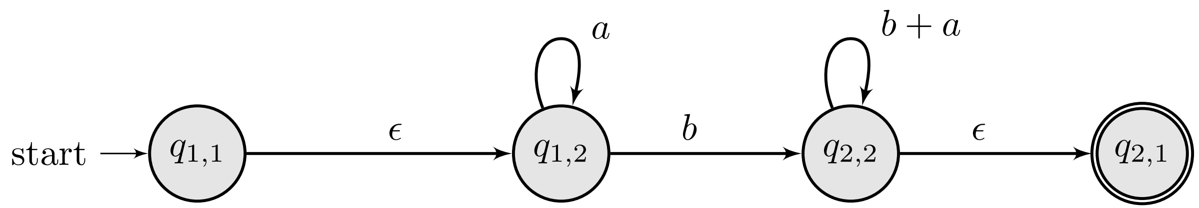
Convert a DFA into a Regex

3. Convert NFA into GNFA

Before



After

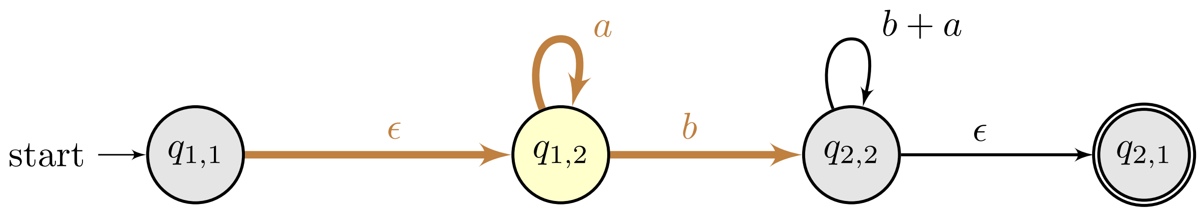


Exercise 1.66

Convert a DFA into a Regex

4. Compress state.

Before

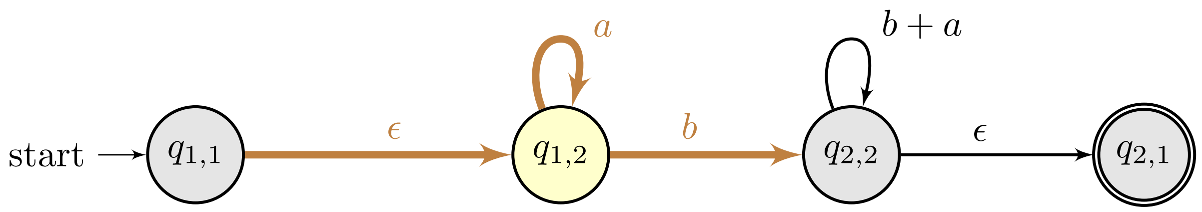


Exercise 1.66

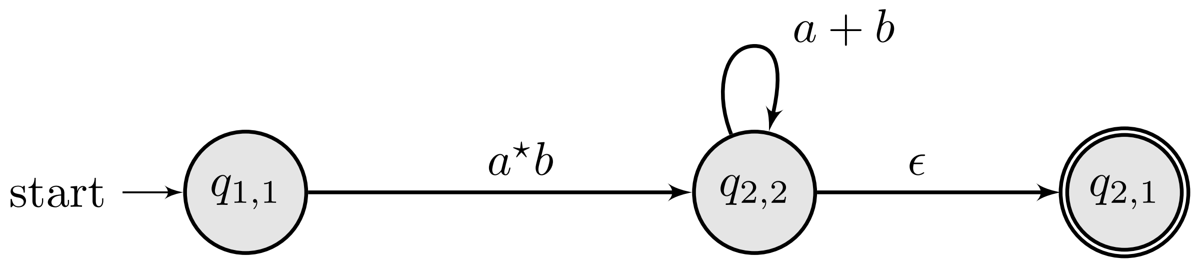
Convert a DFA into a Regex

4. Compress state.

Before



After

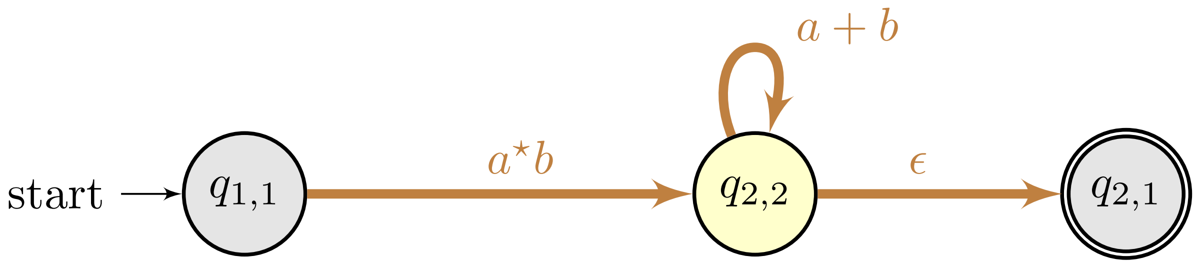


Exercise 1.66

Convert an DFA into a Regex

5. Compress state.

Before

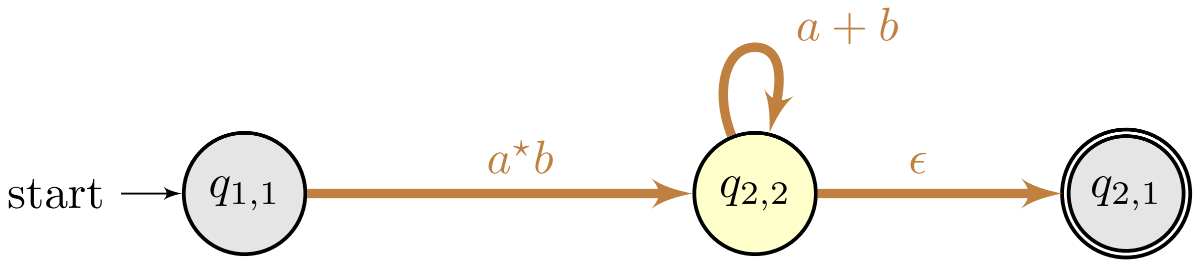


Exercise 1.66

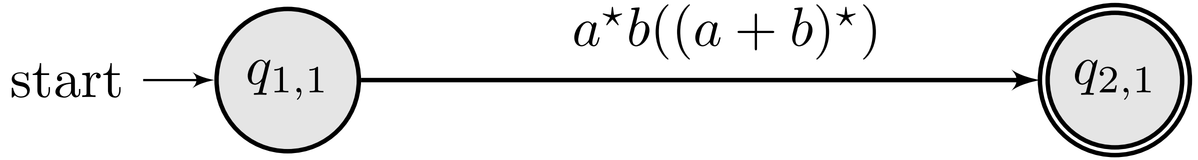
Convert an DFA into a Regex

5. Compress state.

Before



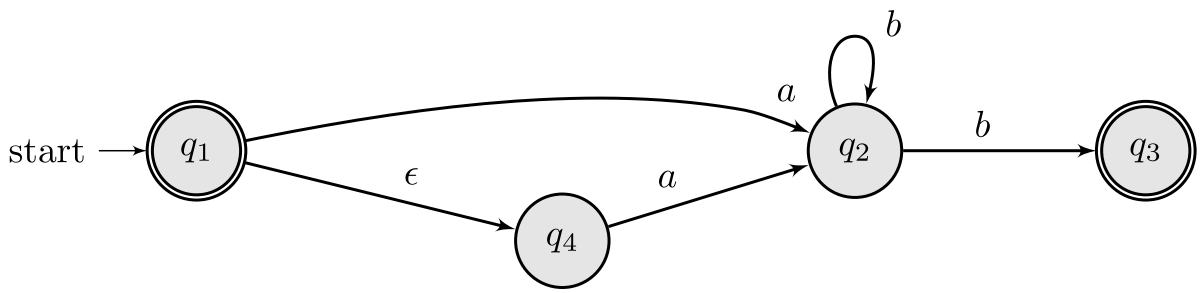
After



Exercise 8

Convert an NFA into a Regex

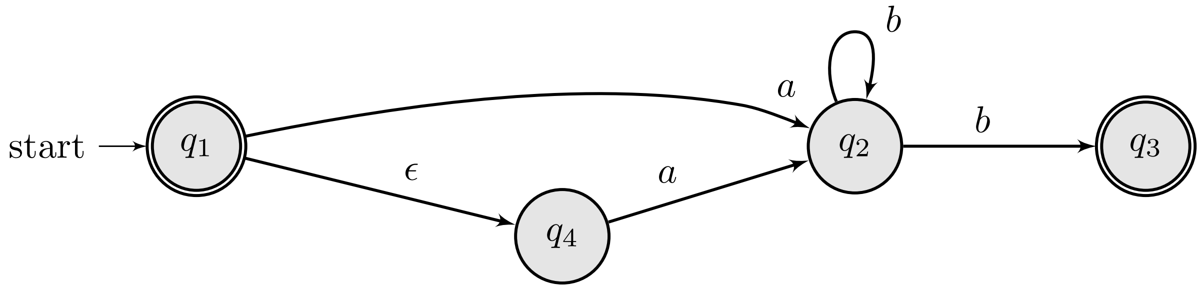
Before



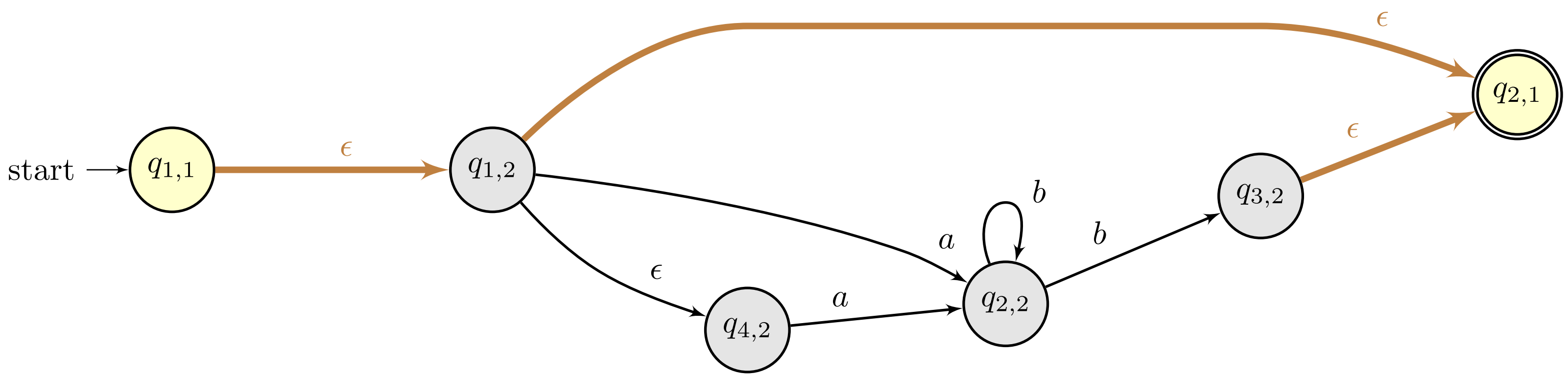
Exercise 8

Convert an NFA into a Regex

Before



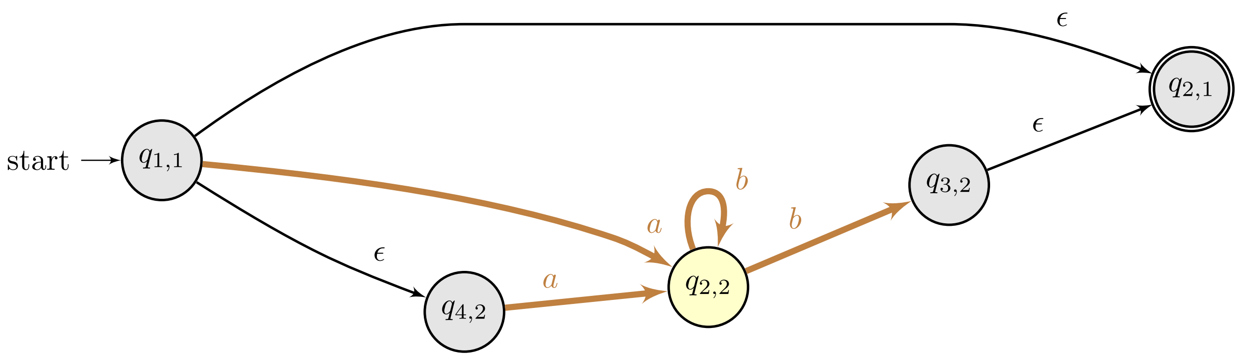
After



Exercise 8

Convert an NFA into a Regex

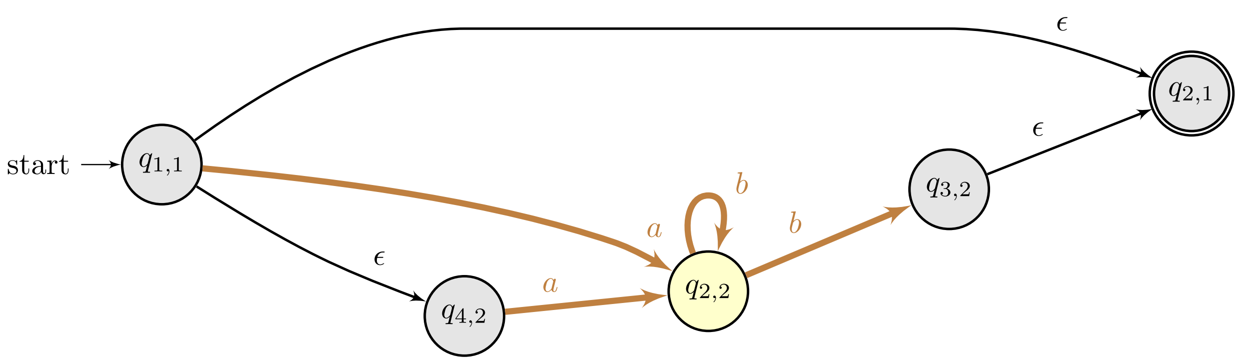
Before



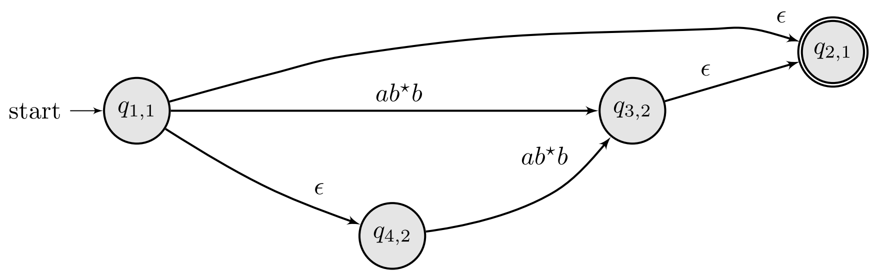
Exercise 8

Convert an NFA into a Regex

Before



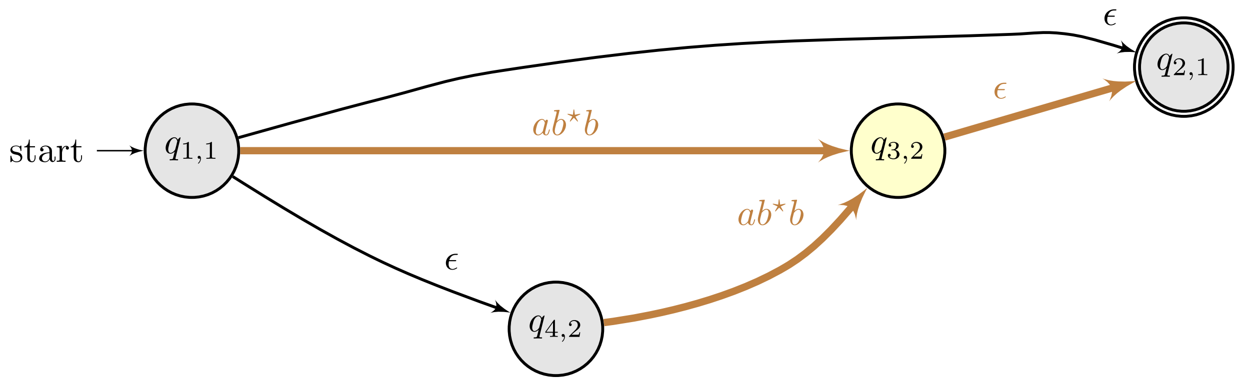
After



Exercise 8

Convert an NFA into a Regex

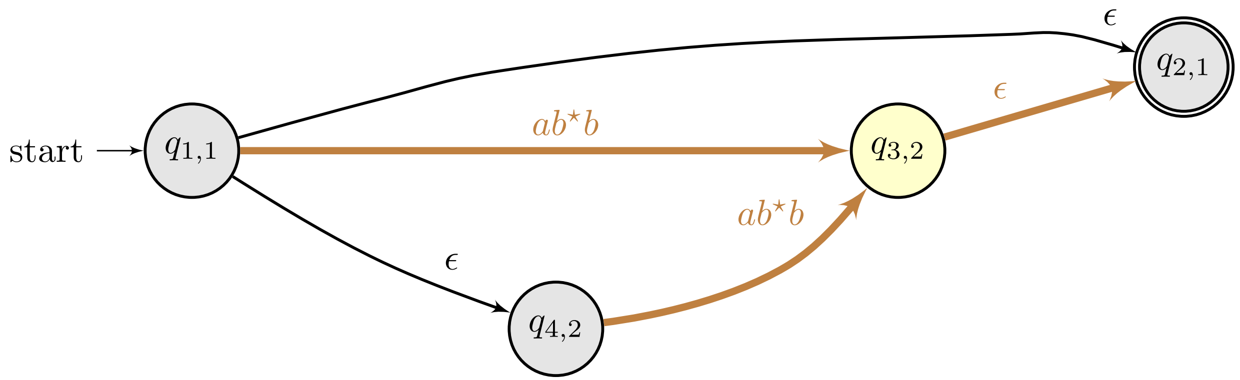
Before



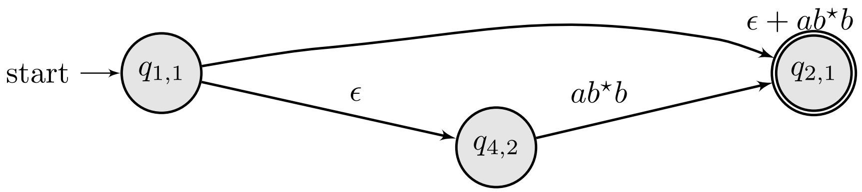
Exercise 8

Convert an NFA into a Regex

Before



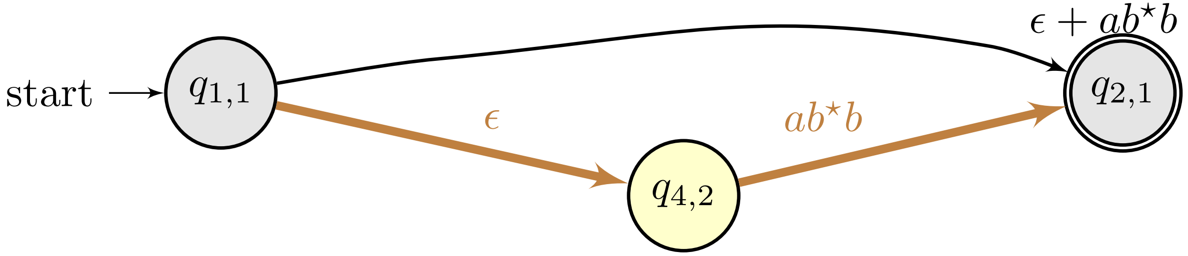
After



Exercise 8

Convert an NFA into a Regex

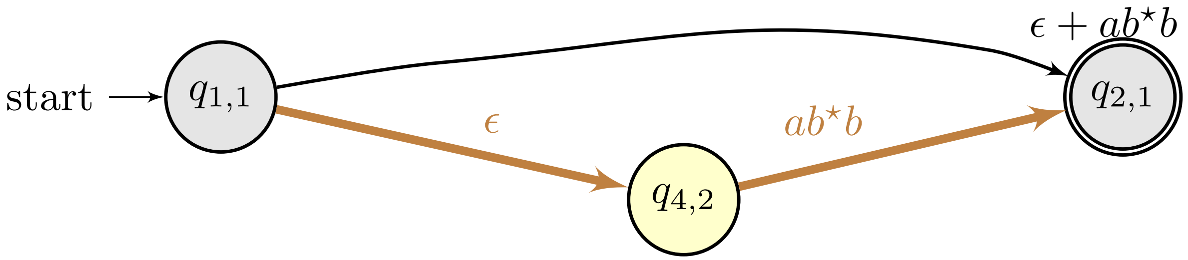
Before



Exercise 8

Convert an NFA into a Regex

Before



After