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

\[ R ::= a \mid \epsilon \mid \emptyset \mid R_1 + R_2 \mid R_1 \cdot R_2 \mid R^* \]

Informal description

A regular expression \( R \) is one of the following cases:

- \( a \) for language \( \{[a]\} \), consists of string \( [a] \)
- \( \epsilon \) 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 \( R_2 \)
- \( R^* \) 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 $\epsilon$.  

Formally (as sets)
Example 1

Regular expression

Let \( \Sigma = \{a, b\} \).

\[ a + \epsilon \]

String \( a \) and string \( \epsilon \).

Formally (as sets)

\[ \{a\} \cup \{\epsilon\} = \{a, \epsilon\} \]
Example 1

Regular expression

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

$\quad a + \epsilon$

Formally (as sets)

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

String $a$ and string $\epsilon$.

As an NFA

union(char($a$), empty)
Example 1

Regular expression

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

\[ a + \epsilon \]

String $a$ and string $\epsilon$.

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

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

Formally (as sets)

String ab and ba.
Example 2

Regular expression

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

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

String $ab$ and $ba$.

Formally (as sets)

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

```javascript
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 $\Sigma = \{a, b\}$.

$$b^* \cdot a \cdot b^*$$
Example 3

Regular expression

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

$$b^* \cdot a \cdot b^*$$

Formally (as sets)

Strings with exactly a single $a$. 
Example 3

Regular expression

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

$$b^* \cdot a \cdot b^*$$

Strings with exactly a single $a$.

Formally (as sets)

$$\{b\}^* \cdot \{a\} \cdot \{b\}^*$$
Example 3

Regular expression

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

$\begin{align*}
    b^* \cdot a \cdot b^*
\end{align*}$

Strings with exactly a single $a$.

As an NFA

Formally (as sets)

$\{b\}^* \cdot \{a\} \cdot \{b\}^*$
Example 3

Regular expression

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

$$b^* \cdot a \cdot b^*$$

Strings with exactly a single $a$.

Formally (as sets)

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

As an NFA

```java
concat(
    concat(
        star(char(b)),
        char(a)
    ),
    star(char(b))
)
```
Example 3

Regular expression

Let \( \Sigma = \{a, b\} \).

\[
\begin{align*}
\epsilon \cdot \{b\}^* \cdot \{a\} \cdot \{b\}^* \\
&= \text{Strings with exactly a single } a.
\end{align*}
\]

Formally (as sets)

As an NFA
Example 3

Regular expression

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

$$b^* \cdot a \cdot b^*$$

Formally (as sets)

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

Strings with exactly a single $a$.

As an NFA

$$\text{concat(}
\text{concat(}
\text{star(char(b)), char(a))},
\text{star(char(b))))}$$
Example 3

Regular expression

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

$$b^* \cdot a \cdot b^*$$

Strings with exactly a single $a$.

Formally (as sets)

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

As an NFA

concat(
  concat(
    star(char(b)),
    char(a)
  ),
  star(char(b)))
Example 4

Regular expression

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

$$(a + b)^* \cdot b \cdot (a + b)^*$$
Example 4

Regular expression

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

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

Strings with at least one $b$.

Formally (as sets)
Example 4

Regular expression

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

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

Strings with at least one $b$.

Formally (as sets)

$$(\{a\} \cup \{b\})^* \cdot \{b\} \cdot (\{a\} \cup \{b\})^*$$

$= \{a, b\}^* \cdot \{b\} \cdot \{a, b\}^*$$
Example 4

Regular expression

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

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

Strings with at least one $b$.

As an NFA

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

Formally (as sets)

$$(\{a\} \cup \{b\})^* \cdot \{b\} \cdot (\{a\} \cup \{b\})^*$$

$= \{a, b\}^* \cdot \{b\} \cdot \{a, b\}^*$$
Example 4

Regular expression

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

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

Strings with at least one $b$.

Formally (as sets)

$$(\{a\} \cup \{b\})^* \cdot \{b\} \cdot (\{a\} \cup \{b\})^*$$

$= \{a, b\}^* \cdot \{b\} \cdot \{a, b\}^*$$

As an NFA

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

Diagram:

- Start state $q_{1,1}$ transitions to $q_{1,3,2}$ on $\epsilon$.
- $q_{1,3,2}$ transitions to $q_{2,1,2}$ on $a$ and to $q_{1,2,2}$ on $\epsilon$.
- $q_{1,2,2}$ transitions to $q_{2,2,2}$ on $b$.
- $q_{1,1}$ transitions to $q_{1,1,2}$ on $\epsilon$.
- $q_{1,1,2}$ transitions to $q_{2,1,2}$ on $\epsilon$.
- $q_{2,1,2}$ transitions to $q_{2,1,2}$ on $a$.
- $q_{2,2,2}$ transitions to $q_{2,2,2}$ on $b$.

Diagram:

```
<input image here>
```
Example 4

Regular expression

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

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

Strings with at least one $b$.

Formally (as sets)

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

As an NFA

$\text{concat(}
\text{star(union(char(a), char(b))),}
\text{concat(}
\text{char(b),}
\text{star(union(char(a), char(b))))})$
Example 4

Regular expression

Let \( \Sigma = \{a, b\} \).

\[ (a + b)^* \cdot b \cdot (a + b)^* \]

Strings with at least one \( b \).

Formally (as sets)

\[ \left( \{a\} \cup \{b\} \right)^* \cdot \{b\} \cdot \left( \{a\} \cup \{b\} \right)^* \]

\[ = \{a, b\}^* \cdot \{b\} \cdot \{a, b\}^* \]

As an NFA

\[
\begin{align*}
\text{concat(} & \text{star(union(char(a), char(b))}, \\
& \text{concat(} \\
& \text{char(b),} \\
& \text{star(union(char(a), char(b))))}))
\end{align*}
\]
Example 5

Regular expression

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

$$a + \emptyset$$
Example 5

Regular expression

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

$\quad a + \emptyset$

Formally (as sets)

String a.
Example 5

Regular expression

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

Formally (as sets)

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

String a.

$a + \emptyset$
Example 5

Regular expression

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

$a + \emptyset$

Formally (as sets)

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

String $a$.

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

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

$a \cdot \emptyset$

Formally (as sets)

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

The empty set.
Example 6

Regular expression

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

$$a \cdot \emptyset$$

The empty set.

Formally (as sets)

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

**Why?** Because,

$$L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$$
Example 6

Regular expression

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

$$a \cdot \emptyset$$

The empty set.

As an NFA

concat(char(a), nil)

Formally (as sets)

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

Why? Because,

$$L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$$
Example 6

Regular expression

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

$\quad a \cdot \emptyset$

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

$$a^* + b^*$$
Example 7

Regular expression

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

$$a^* + b^*$$

String where all letters are the same.

Formally (as sets)

$$\{a\}^* \cup \{b\}^*$$
Example 7

Regular expression

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

$a^* + b^*$

String where all letters are the same.

Formally (as sets)

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

As an NFA

union(
  star(char(a)),
  star(char(b)))
Example 7

Regular expression

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

$$a^* + b^*$$

String where all letters are the same.

Formally (as sets)

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

As an NFA

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

\[\text{Diagram of NFA}\]
Formalizing the regular expressions

The language of a regular expression

- \( L(a) = \)
Formalizing the regular expressions

The language of a regular expression

- $L(a) = \{a\}$
- $L(\epsilon) =$
Formalizing the regular expressions

The language of a regular expression

- $L(a) = \{a\}$
- $L(\epsilon) = \{\epsilon\}$
- $L(\emptyset) =$
Formalizing the regular expressions

The language of a regular expression

- \( L(a) = \{a\} \)
- \( L(\epsilon) = \{\epsilon\} \)
- \( L(\emptyset) = \emptyset \)
- \( L(R_1 + R_2) = \)
Formalizing the regular expressions

The language of a regular expression

- $L(a) = \{a\}$
- $L(\epsilon) = \{\epsilon\}$
- $L(\emptyset) = \emptyset$
- $L(R_1 + R_2) = L(R_1) \cup L(R_2)$
- $L(R_1 \cdot R_2) =$
Formalizing the regular expressions

The language of a regular expression

- \( L(a) = \{a\} \)
- \( L(\epsilon) = \{\epsilon\} \)
- \( L(\emptyset) = \emptyset \)
- \( L(R_1 + R_2) = L(R_1) \cup L(R_2) \)
- \( L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2) \)
- \( L(R^*) = \)
Formalizing the regular expressions

The language of a regular expression

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

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 \( L(\text{NFA}(R)) = L_1 \).

Given an alphabet \( \Sigma \)

- \( \text{NFA}(a) = \)
All Regexes have an equivalent NFA

Lemma 1.55

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

Given an alphabet \( \Sigma \)

- \( \text{NFA}(a) = \text{char}_\Sigma(a) \)
- \( \text{NFA}(\epsilon) = \)
All Regexes have an equivalent NFA

Lemma 1.55

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

Given an alphabet $\Sigma$

- $\text{NFA}(a) = \text{char}_\Sigma(a)$
- $\text{NFA}(\epsilon) = \text{empty}_\Sigma$
- $\text{NFA}(\emptyset) =$
All Regexes have an equivalent NFA

Lemma 1.55

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

Given an alphabet $\Sigma$

- $\text{NFA}(a) = \text{char}_\Sigma(a)$
- $\text{NFA}(\epsilon) = \text{empty}_\Sigma$
- $\text{NFA}(\emptyset) = \text{nil}_\Sigma$
- $\text{NFA}(R_1 \cup R_2) =$
All Regular expressions have an equivalent NFA

Lemma 1.55

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

Given an alphabet $\Sigma$

- $\text{NFA}(a) = \text{char}_\Sigma(a)$
- $\text{NFA}(\epsilon) = \text{empty}_\Sigma$
- $\text{NFA}(\emptyset) = \text{nil}_\Sigma$
- $\text{NFA}(R_1 \cup R_2) = \text{union}(\text{NFA}(R_1), \text{NFA}(R_2))$
- $\text{NFA}(R_1 \cdot R_2) =$
All Regexes have an equivalent NFA

Lemma 1.55

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

Given an alphabet $\Sigma$

- $\text{NFA}(a) = \text{char}_\Sigma(a)$
- $\text{NFA}(\epsilon) = \text{empty}_\Sigma$
- $\text{NFA}(\emptyset) = \text{nil}_\Sigma$
- $\text{NFA}(R_1 \cup R_2) = \text{union}(\text{NFA}(R_1), \text{NFA}(R_2))$
- $\text{NFA}(R_1 \cdot R_2) = \text{concat}(\text{NFA}(R_1), \text{NFA}(R_2))$
- $\text{NFA}(R^*) =$
All Regexes have an equivalent NFA

Lemma 1.55

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

Given an alphabet $\Sigma$

- $\text{NFA}(a) = \text{char}_\Sigma(a)$
- $\text{NFA}(\epsilon) = \text{empty}_\Sigma$
- $\text{NFA}(\emptyset) = \text{nil}_\Sigma$
- $\text{NFA}(R_1 \cup R_2) = \text{union}(\text{NFA}(R_1), \text{NFA}(R_2))$
- $\text{NFA}(R_1 \cdot R_2) = \text{concat}(\text{NFA}(R_1), \text{NFA}(R_2))$
- $\text{NFA}(R^*) = \text{star}(\text{NFA}(R))$
All Regexes have an equivalent NFA

Lemma 1.55

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

Given an alphabet \( \Sigma \)

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

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

NFA(\emptyset)

\[
\text{start} \rightarrow q_1
\]

NFA(\epsilon)

\[
\text{start} \rightarrow q_1
\]

NFA(1 + \epsilon)

\[
\begin{align*}
\text{start} & \rightarrow q_{1,3} \\
& \rightarrow q_{1,1} \\
& \rightarrow q_{2,1}
\end{align*}
\]

NFA((1 + \epsilon)^*)

\[
\begin{align*}
\text{start} & \rightarrow q_{1,1} \\
& \rightarrow q_{1,3,2} \\
& \rightarrow q_{1,3,2} \\
& \rightarrow q_{2,1,2}
\end{align*}
\]

NFA (0 \cdot \epsilon)

\[
\begin{align*}
\text{start} & \rightarrow q_{1,1} \\
& \rightarrow q_{2,1} \\
& \rightarrow q_{1,2}
\end{align*}
\]
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 $q_{\text{start}}$ and $q_{\text{end}}$ such that $q_{\text{start}}$ transitions via $\epsilon$ to the initial state of $N$, and every accepted state of $N$ transitions to $q_{\text{end}}$ via $\epsilon$. State $q_{\text{end}}$ becomes the new accepted state.

Input
Step 1: wrap the NFA

Given an NFA $N$, add two new states $q_{\text{start}}$ and $q_{\text{end}}$ such that $q_{\text{start}}$ transitions via $\epsilon$ to the initial state of $N$, and every accepted state of $N$ transitions to $q_{\text{end}}$ via $\epsilon$. State $q_{\text{end}}$ becomes the new accepted state.
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 \( q_{1,2} \)

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

\[
\text{compress}(q_{1,1} \xrightarrow{\epsilon} q_{1,2} \xrightarrow{1} q_{3,2}) = q_{1,1} \xrightarrow{\epsilon \cdot 1} q_{3,2}
\]
Step 3.1: compress state $q_{1,2}$

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

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

Each state that connects to $q_{1,2}$ must connect to every state that $q_{1,2}$ connects to. Some $q_{1,1}$ 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 $q_{2,2}$

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).

\[
\text{compress}(q_{1,1} \rightarrow q_{3,2} \rightarrow q_{3,2} \rightarrow q_{2,1}) + q_{1,1} \xrightarrow{0+\epsilon} q_{2,1} = q_{1,1} \xrightarrow{(10^*0) + (0+\epsilon)} q_{2,2}
\]

\[
\text{compress}(q_{1,1} \rightarrow q_{3,2} \rightarrow q_{3,2} \rightarrow q_{4,2}) = q_{3,2} \xrightarrow{10^*1} q_{2,1}
\]
Step 3.3: compress state $q_{4,2}$

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

\[
\text{compress}(q_{1,1} \xrightarrow{10^*1} q_{4,2} \xrightarrow{\epsilon} q_{2,1}) + q_{1,1} \xrightarrow{10^*1+0+\epsilon} q_{2,1} = q_{1,1} \xrightarrow{(10^*1\cdot\epsilon)(10^*0+0+\epsilon)} q_{2,2}
\]

Input

Output

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

```
start  → q₁,₁  ε  → q₁,₂  a  → q₂,₂  a, b  → q₂,₁
```

start  → q₁,₁  ε  → q₁,₂  b  → q₂,₂  ε  → q₂,₁
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

\[ \text{start} \rightarrow q_{1,1} \xrightarrow{\epsilon} q_{1,2} \xrightarrow{a} q_{2,2} \xrightarrow{b + a} q_{2,1} \]
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

![DFA Diagram Before]

After

![DFA Diagram 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

```
start -> q_{1,1} -> ab*b -> q_{3,2} -> q_{2,1}
q_{3,2} -> ab*b

q_{1,1} -> \epsilon
q_{4,2} -> \epsilon
q_{2,1} -> \epsilon
```
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