Automata Theory :: Regular Expressions

Jörg Endrullis

Vrije Universiteit Amsterdam

Regular Expressions

We define the **regular expressions** over an alphabet Σ:

- $\emptyset$ is a regular expression
- λ is a regular expression
- a is a regular expression for every $a \in \Sigma$
- $r_1 + r_2$ is a regular expression for all regular expr. r_1 and r_2
- $r_1 \cdot r_2$ is a regular expression for all regular expr. r_1 and r_2
- r^* is a regular expression for all regular expressions r

Regular Expressions

We define the **regular expressions** over an alphabet Σ:

- $\varnothing$ is a regular expression
- λ is a regular expression
- a is a regular expression for every $a \in \Sigma$
- $r_1 + r_2$ is a regular expression for all regular expr. r_1 and r_2
- $r_1 \cdot r_2$ is a regular expression for all regular expr. r_1 and r_2
- r^* is a regular expression for all regular expressions r

A regular expression is syntax, describing a language.

Every **regular expression** r defines a **language** $L(r)$:

$$L(\varnothing) = \varnothing \qquad L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(\lambda) = \{\lambda\} \qquad L(r_1 \cdot r_2) = L(r_1)L(r_2)$$

$$L(a) = \{a\} \quad \text{for } a \in \Sigma \qquad L(r^*) = L(r)^*$$

Example

Example

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

Example

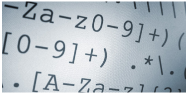
Example

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

Regular expressions are used to search and manipulate text.

For example:

- `grep` in Linux
- script languages such as Perl



Every major programming language has regular expressions.

Example

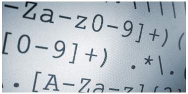
Example

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

Regular expressions are used to search and manipulate text.

For example:

- `grep` in Linux
- script languages such as Perl



Every major programming language has regular expressions.

Exercise

Find a regular expression r over $\Sigma = \{a, b\}$ such that $L(r)$ consists of all words that contain the pattern *bab*:

Example

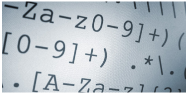
Example

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

Regular expressions are used to search and manipulate text.

For example:

- `grep` in Linux
- script languages such as Perl



Every major programming language has regular expressions.

Exercise

Find a regular expression r over $\Sigma = \{a, b\}$ such that $L(r)$ consists of all words that contain the pattern *bab*:

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

Example

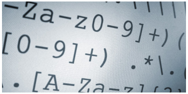
Example

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

Regular expressions are used to search and manipulate text.

For example:

- `grep` in Linux
- script languages such as Perl



Every major programming language has regular expressions.

Exercise

Find a regular expression r over $\Sigma = \{a,b\}$ such that $L(r)$ consists of all words that contain the pattern *bab*:

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

Regular Expressions $\iff$ Regular Languages

Theorem

A language L is **regular**
$\iff$ there is a **regular expression** r with $L(r) = L$.

Regular Expressions $\iff$ Regular Languages

Theorem

A language L is **regular**

$\iff$ there is a **regular expression** r with $L(r) = L$.

Proof.

We need to prove two directions:

- $(\Leftarrow)$ Translate regular expressions into NFAs.
- $(\Rightarrow)$ Translate NFAs into regular expressions.

$\square$

($\Leftarrow$) From Regular Expressions to NFAs

Construction ($\Leftarrow$)

For every regular expression r, we build an NFA M such that

- $L(M) = L(r)$,
- M has precisely one final and one (different) starting state

($\Leftarrow$) From Regular Expressions to NFAs

Construction ($\Leftarrow$)

For every regular expression r, we build an NFA M such that

- $L(M) = L(r)$,
- M has precisely one final and one (different) starting state

We construct M by induction (recursion) on r.

$\varnothing$:

$r_1 + r_2$:

λ:

$r_1 \cdot r_2$:

a:

r^*:

(⇐) From Regular Expressions to NFAs

Construction (⇐)

For every regular expression r, we build an NFA M such that

- $L(M) = L(r)$,
- M has precisely one final and one (different) starting state

We construct M by induction (recursion) on r.

$\varnothing$: 

λ:

a:

$r_1 + r_2$:

$r_1 \cdot r_2$:

r^*:

($\Leftarrow$) From Regular Expressions to NFAs

Construction ($\Leftarrow$)

For every regular expression r, we build an NFA M such that

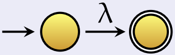
- $L(M) = L(r)$,
- M has precisely one final and one (different) starting state

We construct M by induction (recursion) on r.

$\varnothing$:

$r_1 + r_2$:

λ:



$r_1 \cdot r_2$:

a:

r^*:

(⇐) From Regular Expressions to NFAs

Construction (⇐)

For every regular expression r, we build an NFA M such that

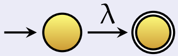
- $L(M) = L(r)$,
- M has precisely one final and one (different) starting state

We construct M by induction (recursion) on r.

$\varnothing$:

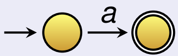
$r_1 + r_2$:

λ:



$r_1 \cdot r_2$:

a:



r^*:

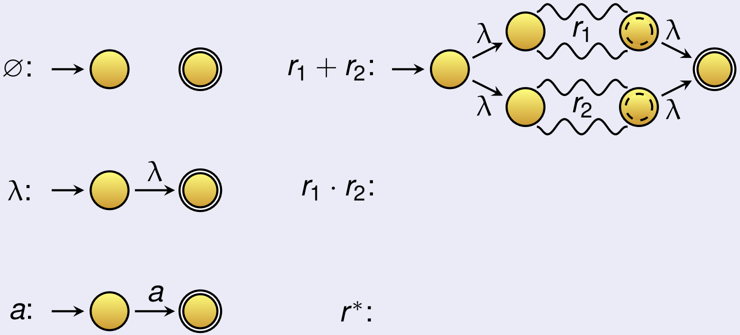
(⇐) From Regular Expressions to NFAs

Construction (⇐)

For every regular expression r, we build an NFA M such that

- $L(M) = L(r)$,
- M has precisely one final and one (different) starting state

We construct M by induction (recursion) on r.



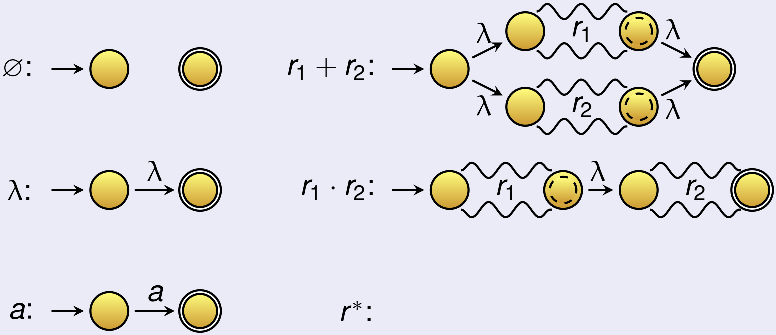
(⇐) From Regular Expressions to NFAs

Construction (⇐)

For every regular expression r, we build an NFA M such that

- $L(M) = L(r)$,
- M has precisely one final and one (different) starting state

We construct M by induction (recursion) on r.



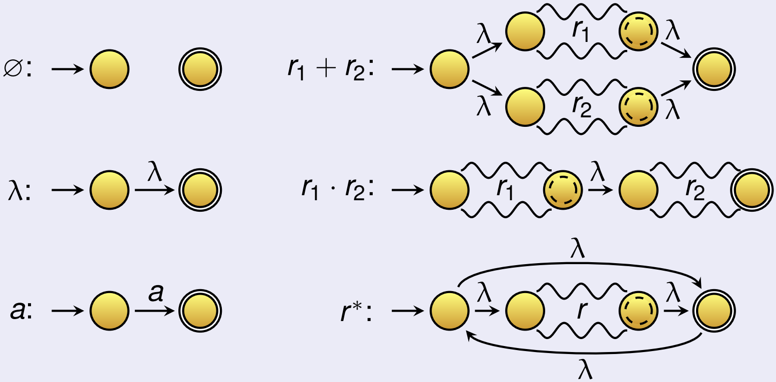
($\Leftarrow$) From Regular Expressions to NFAs

Construction ($\Leftarrow$)

For every regular expression r, we build an NFA M such that

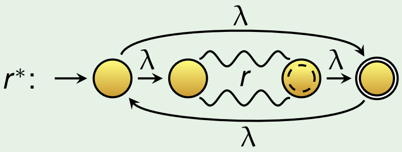
- $L(M) = L(r)$,
- M has precisely one final and one (different) starting state

We construct M by induction (recursion) on r.



Exercise

Understanding the start case



Note that:

- $((a^*) \cdot b)^*$ shows that the new starting state is needed
- $(a \cdot (b^*))^*$ shows that the new final state is needed

What goes wrong without introducing the new start/final state?

($\Rightarrow$) From NFAs to Regular Expressions (1)

Construction ($\Rightarrow$)

For every NFA M, we construct a regular expression r with

$$L(r) = L(M)$$

($\Rightarrow$) From NFAs to Regular Expressions (1)

Construction ($\Rightarrow$)

For every NFA M, we construct a regular expression r with

$$L(r) = L(M)$$

Step 1:

We transform M such that there is

- precisely one initial state
- precisely one final state

($\Rightarrow$) From NFAs to Regular Expressions (2)

Step 2:

We remove all double arrows.

($\Rightarrow$) From NFAs to Regular Expressions (2)

Step 2:

We remove all double arrows.

We use transition graphs with regular expressions as labels.

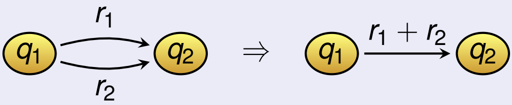
(⇒) From NFAs to Regular Expressions (2)

Step 2:

We remove all double arrows.

We use transition graphs with regular expressions as labels.

If there are 2 arrows from a state q_1 to q_2 with labels r_1 and r_2 replace them by one arrow with label $r_1 + r_2$:



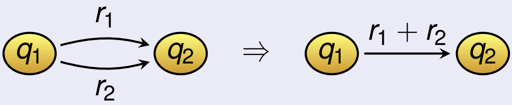
(⇒) From NFAs to Regular Expressions (2)

Step 2:

We remove all double arrows.

We use transition graphs with regular expressions as labels.

If there are 2 arrows from a state q_1 to q_2 with labels r_1 and r_2 replace them by one arrow with label $r_1 + r_2$:



Note that q_1 can be equal to q_2. Then the arrows are loops!

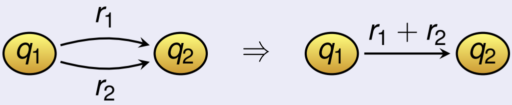
(⇒) From NFAs to Regular Expressions (2)

Step 2:

We remove all double arrows.

We use transition graphs with regular expressions as labels.

If there are 2 arrows from a state q_1 to q_2 with labels r_1 and r_2 replace them by one arrow with label $r_1 + r_2$:



Note that q_1 can be equal to q_2. Then the arrows are loops!

We remove all double arrows before continuing with Step 3.

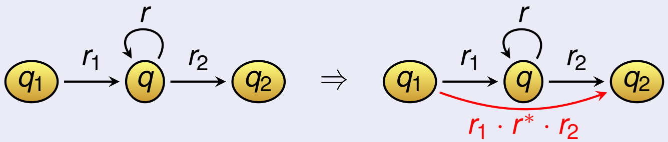
($\Rightarrow$) From NFAs to Regular Expressions (3)

Step 3: Pick one state q that is neither a starting nor a final state (if it exists). We remove q as follows.

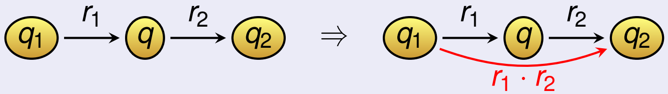
(⇒) From NFAs to Regular Expressions (3)

Step 3: Pick one state q that is neither a starting nor a final state (if it exists). We remove q as follows.

For all states q_1 and q_2 and arrows $q_1 \xrightarrow{r_1} q$ and $q \xrightarrow{r_2} q_2$, we add an arrow from q_1 to q_2 as follows:



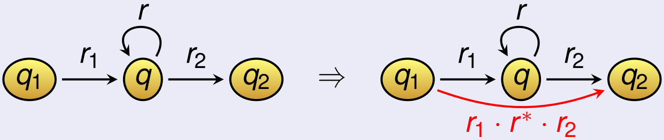
for the case that there is an arrow $q \xrightarrow{r} q$, and otherwise:



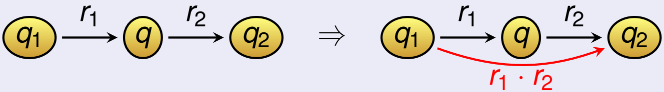
($\Rightarrow$) From NFAs to Regular Expressions (3)

Step 3: Pick one state q that is neither a starting nor a final state (if it exists). We remove q as follows.

For all states q_1 and q_2 and arrows $q_1 \xrightarrow{r_1} q$ and $q \xrightarrow{r_2} q_2$, we add an arrow from q_1 to q_2 as follows:



for the case that there is an arrow $q \xrightarrow{r} q$, and otherwise:

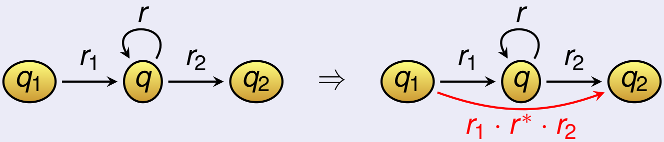


Note that q_1 can be equal to q_2.

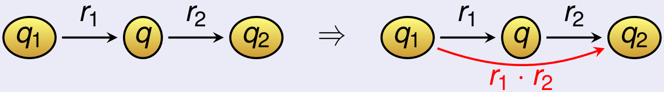
(⇒) From NFAs to Regular Expressions (3)

Step 3: Pick one state q that is neither a starting nor a final state (if it exists). We remove q as follows.

For all states q_1 and q_2 and arrows $q_1 \xrightarrow{r_1} q$ and $q \xrightarrow{r_2} q_2$, we add an arrow from q_1 to q_2 as follows:



for the case that there is an arrow $q \xrightarrow{r} q$, and otherwise:



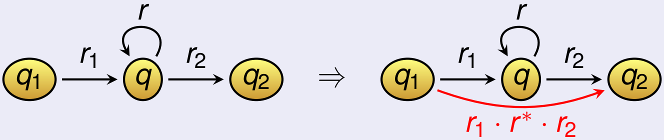
Note that q_1 can be equal to q_2.

Afterwards adding all these transitions we remove q.

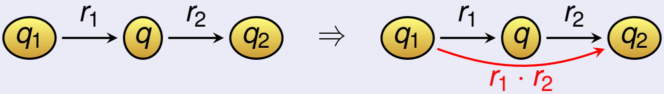
(⇒) From NFAs to Regular Expressions (3)

Step 3: Pick one state q that is neither a starting nor a final state (if it exists). We remove q as follows.

For all states q_1 and q_2 and arrows $q_1 \xrightarrow{r_1} q$ and $q \xrightarrow{r_2} q_2$, we add an arrow from q_1 to q_2 as follows:



for the case that there is an arrow $q \xrightarrow{r} q$, and otherwise:



Note that q_1 can be equal to q_2.

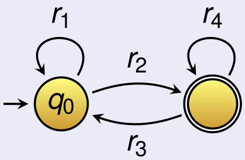
Afterwards adding all these transitions we remove q.

We repeat Step 2 and Step 3 until there is nothing to be done.

($\Rightarrow$) From NFAs to Regular Expressions (4)

Step 4:

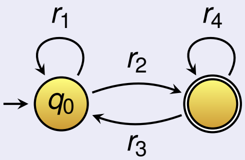
If $F \neq \{ q_0 \}$, then the transition graph is finally of the form:



($\Rightarrow$) From NFAs to Regular Expressions (4)

Step 4:

If $F \neq \{ q_0 \}$, then the transition graph is finally of the form:

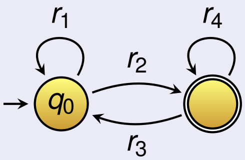


If an arrow r_i with $1 \leq i \leq 4$ does not exist, let $r_i = \varnothing$.

(⇒) From NFAs to Regular Expressions (4)

Step 4:

If $F \neq \{ q_0 \}$, then the transition graph is finally of the form:



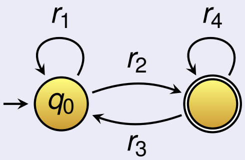
If an arrow r_i with $1 \leq i \leq 4$ does not exist, let $r_i = \varnothing$.

Then the regular expression is:

(⇒) From NFAs to Regular Expressions (4)

Step 4:

If $F \neq \{ q_0 \}$, then the transition graph is finally of the form:



If an arrow r_i with $1 \leq i \leq 4$ does not exist, let $r_i = \varnothing$.

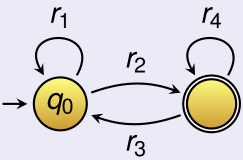
Then the regular expression is:

$$L(r_1^* \cdot r_2 \cdot (r_4 + r_3 \cdot r_1^* \cdot r_2)^*) \;=\; L(M)$$

($\Rightarrow$) From NFAs to Regular Expressions (4)

Step 4:

If $F \neq \{ q_0 \}$, then the transition graph is finally of the form:



If an arrow r_i with $1 \leq i \leq 4$ does not exist, let $r_i = \varnothing$.

Then the regular expression is:

$$L(r_1^* \cdot r_2 \cdot (r_4 + r_3 \cdot r_1^* \cdot r_2)^*) \ = \ L(M)$$

Question

What is the form of the transition graph and regular expression for the case that $F = \{ q_0 \}$?

Exercise

Find a regular expression r such that

$$L(r) = \{\, w \in \{a, b\}^* \mid n_a(w) \text{ even and } n_b(w) \text{ is odd} \,\}$$

where

- $n_a(w)$ is the number of a's in w, and
- $n_b(w)$ is the number of b's in w.

Exercise

Find a regular expression r such that

$$L(r) = \{\, w \in \{a, b\}^* \mid n_a(w) \text{ even and } n_b(w) \text{ is odd} \,\}$$

where

- $n_a(w)$ is the number of a's in w, and
- $n_b(w)$ is the number of b's in w.

Find a regular expression r over $\{a, b\}$ such that $L(r)$ consists of all words that do not contain the pattern *bab*.