Theoretical Computer Science

Lab Session 11

April 15, 2021

Agenda

- Regular Expressions (RegExp);
- FSA to RegExp;
- RegExp to (N)FSA.

Regular Expressions (RegExp)

Regular Expressions (RegExp): Definition

Inductive definition of RegExps over an alphabet Σ:
Basis.

- Symbol $\emptyset$ is a regular expression (denoting the language $\emptyset$);
- Symbol ϵ is a RegExp (denoting the language $\{\epsilon\}$);
- Each symbol σ of Σ is a RegExp
  $L(\sigma) = \{\sigma\}$ for each $\sigma \in \Sigma$.

Induction. Let r and s be two RegExps, then

- $(r.s)$ is a RegExp (for simplicity, the dot is often omitted);
  $L((r.s)) = \{ww' \mid (w \in L(r) \wedge w' \in L(s))\}$,
- $(r|s)$ is a RegExp
  $L((r \mid s)) = L(r) \cup L(s)$;
- r^* is a RegExp
  $L(r^*) = L(r)^*$

Regular Expressions (RegExp): Definition

If A and B are RegExp, then

- $A.B = \{xy | x \in A \text{ and } y \in B\}$
- $A|B = \{x | x \in A \text{ or } x \in B\}$
- $A^* = \{x_1.x_2.x_3 \ldots x_k | k \geq 0 \text{ and each } x_i \in A\}$

are also RegExp.
It is also possible to use the following notation:
$A^+ = \{x_1.x_2.x_3 \ldots x_k | k \geq 1 \text{ and each } x_i \in A\}$

RegExp: Precedence order

Parentheses in an expression may be omitted. If they are, evaluation is done in the precedence order:

1. Kleene star *
2. Concatenation .
3. Union |

RegExp: Exercises (1)

Consider the alphabet $\Sigma = \{0, 1\}$. Give English descriptions of the languages of the following regular expressions:

1. $\emptyset$;
2. $\emptyset^*$;
3. $(0^*1^*)^*000(0 \mid 1)^*$;
4. $(1 \mid \epsilon)(00^*1)^*0^*$;

RegExp: Solutions (1)

English descriptions of the languages over the alphabet $\Sigma = \{0, 1\}$ of the regular expressions:

1. $\emptyset$ - an empty set;
2. $\emptyset^*$ - a set containing only the empty string;
3. $(0^*1^*)^*000(0 \mid 1)^*$ - a set of strings containing 3 consecutive 0's;
4. $(1 \mid \epsilon)(00^*1)^*0^*$ - a set of strings that do not have adjacent 1's;

RegExp: Exercises (2)

Consider the alphabet $\Sigma = \{a, b\}$. Build Regular Expressions for:

5. the set of strings that consist of alternating *a*'s and *b*'s;
6. the set of strings that contain an odd number of *a*'s;
7. the set of strings that end with *b* and do not contain the substring *aa*;
8. the set of strings in which both the number of *a*'s and the number of *b*'s are even.

RegExp: Solutions (2)

5. Regular Expression for the set of strings that consist of alternating *a*'s and *b*'s

$$(\epsilon \mid a)(ba)^*(\epsilon \mid b)$$

RegExp: Solutions (2)

6. Regular Expression for the set of strings that contain an odd number of a's

$$(b \mid ab^*a)^*ab^*$$

RegExp: Solutions (2)

7. Regular Expression for the set of strings that end with *b* and do not contain the substring *aa*

$$(b \mid ab)^*(b \mid ab)$$

RegExp: Solutions (2)

8. Regular Expression for the set of strings in which both the number of *a*'s and the number of *b*'s are even

$$(aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ab \mid ba))^*$$

FSA to RegExp

Kleene's Algorithm: from FSA to Regular Expression

It transforms a given finite state automaton (FSA) into a regular expression.
Description: Given an FSA $M = (Q, A, \delta, q_0, F)$, with $Q = \{q_0, \ldots, q_n\}$ its set of states, the algorithm computes

- the sets R_{ij}^{k} of all strings that take M from state q_i to q_j without going through any state numbered higher than k.
- each set R_{ij}^{k} is represented by a regular expression.
- the algorithm computes them step by step for $k = -1, 0, \ldots, n$.
- since there is no state numbered higher than n, the regular expression R_{0j}^{n} represents the set of all strings that take M from its start state q_0 to q_j.
- If $F = \{q_i, \ldots, q_f\}$ is the set of accept states, the regular expression $R_{0i}^{n} \mid \ldots \mid R_{0f}^{n}$ represents the language accepted by M.

Kleene's Algorithm

The initial regular expressions, for $k = -1$, are computed as

$$R_{ij}^{-1} = a_1 \mid \ldots \mid a_m \text{ if } i \neq j, \text{where } \delta(q_i, a_1) = \ldots = \delta(q_i, a_m) = q_j$$

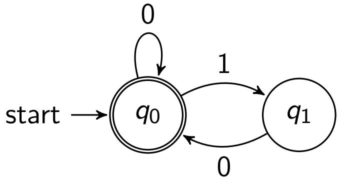
$$R_{ij}^{-1} = a_1 \mid \ldots \mid a_m \mid \epsilon \text{ if } i = j, \text{where } \delta(q_i, a_1) = \ldots = \delta(q_i, a_m) = q_j$$

After that, in each step the expressions R_{ij}^{k} are computed from the previous ones by

$$R_{ij}^{k} = R_{ik}^{k-1} \left(R_{kk}^{k-1}\right)^{*} R_{kj}^{k-1} \mid R_{ij}^{k-1}$$

Kleene's Algorithm: Example

Give a regular expression that describes the language accepted by:

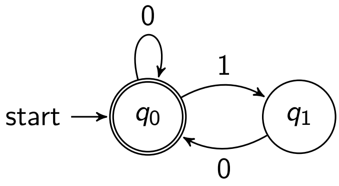


Initial Regular Expression (Step -1)

$$\begin{aligned} R_{00}^{-1} &= 0 \mid \epsilon \\ R_{01}^{-1} &= 1 \\ R_{10}^{-1} &= 0 \\ R_{11}^{-1} &= \epsilon \end{aligned}$$

Example

Give a regular expression that describes the language accepted by:



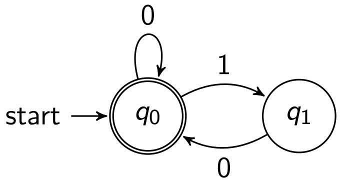
Step 0

$$R_{ij}^{k} = R_{ik}^{k-1}\left(R_{kk}^{k-1}\right)^{*} R_{kj}^{k-1} \mid R_{ij}^{k-1}$$

$$\begin{aligned}
R_{00}^{0} &= (0 \mid \epsilon)(0 \mid \epsilon)^{*}(0 \mid \epsilon) \mid (0 \mid \epsilon) = 0^{*} \\
R_{01}^{0} &= (0 \mid \epsilon)(0 \mid \epsilon)^{*}1 \mid 1 = 0^{*}1 \\
R_{10}^{0} &= 0(0 \mid \epsilon)^{*}(0 \mid \epsilon) \mid 0 = 00^{*} \\
R_{11}^{0} &= 0(0 \mid \epsilon)^{*}1 \mid \epsilon = 00^{*}1 \mid \epsilon
\end{aligned}$$

Example

Give a regular expression that describes the language accepted by:

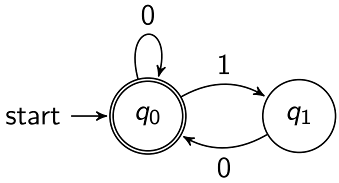


Step 1

$$R^1_{00} = \quad (0^*1)(00^*1 \mid \epsilon)^*(00^*) \mid 0^* = (0^*1)(00^*1)^*(00^*) \mid 0^*$$

Example

Give a regular expression that describes the language accepted by:



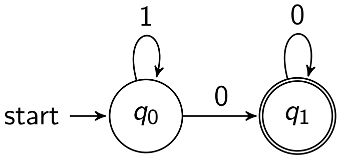
Step 1

$$R^1_{00} = \quad (0^*1)(00^*1 \mid \epsilon)^*(00^*) \mid 0^* = (0^*1)(00^*1)^*(00^*) \mid 0^*$$

Do we need to compute the rest? (i.e R^1_{01}, R^1_{10} and R^1_{11})

Exercise

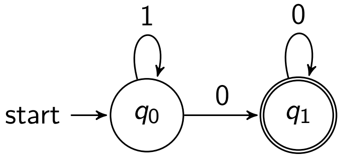
Give a regular expression that describes the language accepted by:

1. 

Solution.

Exercise solution

Give a regular expression that describes the language accepted by:

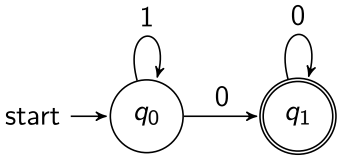


Initial Regular Expression (Step -1)

$$\begin{aligned} R_{00}^{-1} &= 1 \mid \epsilon \\ R_{01}^{-1} &= 0 \\ R_{10}^{-1} &= \emptyset \\ R_{11}^{-1} &= 0 \mid \epsilon \end{aligned}$$

Exercise solution

Give a regular expression that describes the language accepted by:

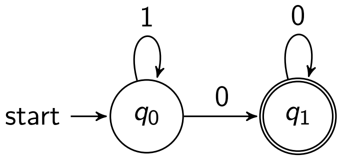


Step 0

$$\begin{aligned}
R_{00}^0 &= (1 \mid \epsilon)(1 \mid \epsilon)^*(1 \mid \epsilon) \mid (1 \mid \epsilon) = 1^* \\
R_{01}^0 &= (1 \mid \epsilon)(1 \mid \epsilon)^*0 \mid 0 = 1^*0 \\
R_{10}^0 &= \emptyset(1 \mid \epsilon)^*(1 \mid \epsilon) \mid \emptyset = \emptyset \\
R_{11}^0 &= \emptyset(1 \mid \epsilon)^*0 \mid (0 \mid \epsilon) = 0 \mid \epsilon
\end{aligned}$$

Exercise solution

Give a regular expression that describes the language accepted by:

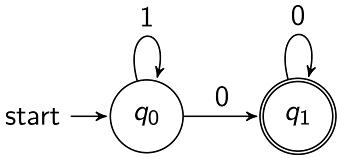


Step 1

$$R_{01}^{1} = \quad (1^*0)(0 \mid \epsilon)^*(0 \mid \epsilon) \mid 1^*0 = 1^*00^* \mid 1^*0$$

Exercise solution

Give a regular expression that describes the language accepted by:



Step 1

$$R^1_{01} = \quad (1^*0)(0 \mid \epsilon)^*(0 \mid \epsilon) \mid 1^*0 = 1^*00^* \mid 1^*0$$

Do we need to compute the rest? (i.e R^1_{00}, R^1_{10} and R^1_{11})

From Regular Expression to (N)FSA.

NFSA with ϵ-transition

To apply the Thompson's algorithm we should introduce a notion of ϵ-moves in FSA.
We extend the class of NFSAs by allowing instantaneous ϵ-transitions:

- The automaton may be allowed to change its state without reading the input symbol.
- In diagrams, such transitions are depicted by labeling the appropriate arcs with ϵ.
- Note that this does not mean that ϵ has become an input symbol. On the contrary, we assume that the symbol ϵ does not belong to any alphabet.
- ϵ-NFSAs add a convenient feature but they do not extend the class of languages that can be represented. Both NFSAs and ϵ-NFSAs recognize exactly the same languages.

NFSA with ϵ-transition: formal definition

An ϵ-FSA is a tuple $\langle Q, I, \delta, q_0, F \rangle$, where

- Q is a set of states
- I is the alphabet of input symbols
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states
- $\delta : Q \times I_\epsilon \to \mathbb{P}(Q)$ is the transition function

I_ϵ is defined as $I \cup \{\epsilon\}$

The Thompson's Construction

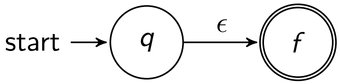
- It is an algorithm for transforming a regexp into an equivalent (N)FSA.
- This (N)FSA can be used to match strings against the regular expression.

The Algorithm

The algorithm works recursively by splitting an expression into its constituent subexpressions, from which the (N)FSA will be constructed using a set of rules (see below)

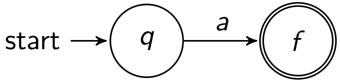
Rule: the Empty Expression

The empty-expression ϵ is converted to:



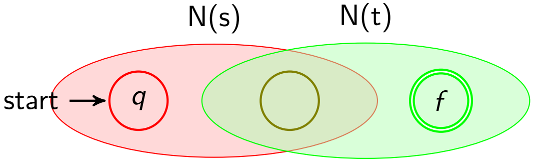
Rule: a symbol

A symbol a of the input alphabet is converted to



Rule: Concatenation Expression

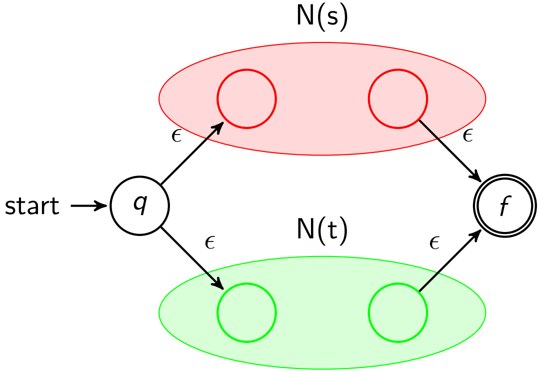
The concatenation expression *st* is converted to



$N(s)$ and $N(t)$ are the (N)FSA of the subexpression s and t, respectively.
The initial state of N(s) is the initial state of the whole NFSA. The final state of N(s) becomes the initial state of N(t). The final state of N(t) is the final state of the whole NFSA.

Rule: Union Expression

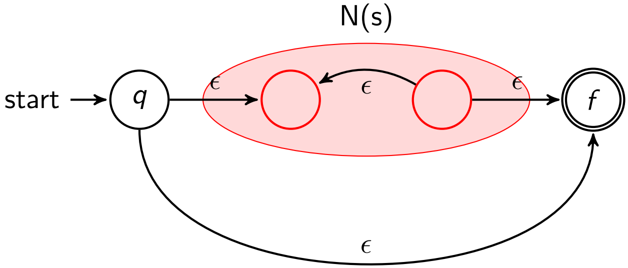
The union expression $s|t$ is converted to



$N(s)$ and $N(t)$ are the (N)FSA of the subexpression s and t, respectively.
State q goes via ϵ either to the initial state of N(s) or N(t). Their final states become intermediate states of the whole NFSA and merge via two ϵ-transitions into the final state of the NFSA.

Rule: Kleene Star Expression

The Kleene star expression s^* is converted to



$N(s)$ is the (N)FSA of the subexpression s.
An ϵ-transition connects initial and final state of the NFSA with the sub-NFSA N(s) in between. Another ϵ-transition from the inner final to the inner initial state of N(s) allows for repetition of expression s according to the star operator.

Exercises

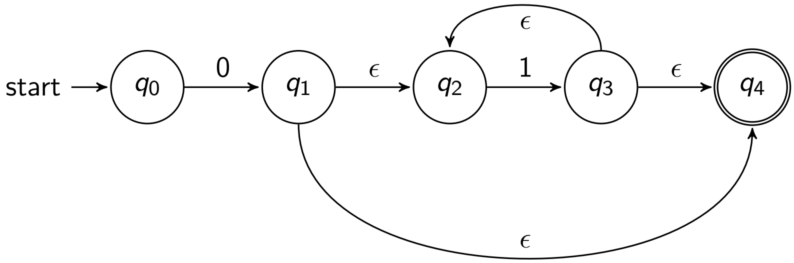
Build a (N)FSA for:

1. 01^*;
2. $(0 \mid 1)01$;
3. $00(0 \mid 1)^*$

Solution.

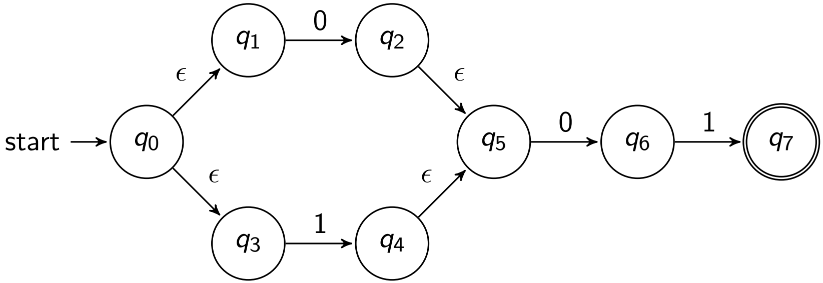
Solution (1)

(N)FSA for 01*


$$\begin{tikzcd}
\text{start} \arrow[r] & q_0 \arrow[r, "0"] & q_1 \arrow[r, "\epsilon"] \arrow[rrr, bend right, "\epsilon"'] & q_2 \arrow[r, "1"] & q_3 \arrow[l, bend right, "\epsilon"'] \arrow[r, "\epsilon"] & q_4
\end{tikzcd}$$

Solution (2)

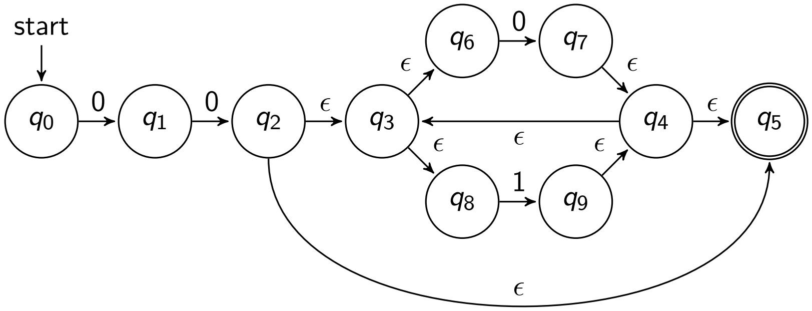
(N)FSA for (0 | 1)01



$$\begin{tikzcd}
 & q_1 \arrow[r, "0"] & q_2 \arrow[dr, "\epsilon"] & & & \\
\text{start} \arrow[r] & q_0 \arrow[ur, "\epsilon"] \arrow[dr, "\epsilon"'] & & q_5 \arrow[r, "0"] & q_6 \arrow[r, "1"] & \boxed{q_7} \\
 & q_3 \arrow[r, "1"] & q_4 \arrow[ur, "\epsilon"'] & & &
\end{tikzcd}$$

Solution (3)

(N)FSA for $00(0 \mid 1)^*$


start
q_0
0
q_1
0
q_2
ϵ
q_3
ϵ
q_6
0
q_7
ϵ
q_4
ϵ
q_5
ϵ
ϵ
ϵ
q_8
1
q_9
ϵ