Theory of Computation

Lab Session 11

April 07, 2016

Innopolis University

News: Written essay

- Essay Submission is via Moodle. Go to (`https://moodle.university.innopolis.ru/mod/assign/view.php?id=584`) and lookup your team.
- Only one team member should post a submission. However, it is the whole team responsibility to ensure an on time submission.
- If you submit your essay by 23:59 on April-08-2016, then you can earn up to 100% of the essay report score.
- If you submit your essay after 23:59 on April-08-2016 and by 23:59 on April 12, then you can earn up to 80% of the essay report score.
- If you submit your essay after 23:59 on April-12, you receive zero points.

News: Oral live presentation

- Each team will do a live presentation on April 23, 2016.
- Plan for 20 minutes of talk-time followed by 10 minutes of questions.
- It is optional, but recommended to prepare a presentation with slides.
- You may use the whiteboard, if needed.
- Anyway, rehearse your presentation beforehand.

News: Video presentation

- A video presentation is optional and it can earn up to 5 bonus points.
- It should be uploaded to a video sharing site (e.g. Youtube.com).
- The link should appear in the submitted survey.

Agenda

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

Regular Expressions (RegExp)

Regular Expressions (RegExp): Definition

Inductive definition of RegExps over an alphabet A:

Basis.

- $\emptyset$ is a regular expression (denoting the language $\emptyset$);
- The empty string is a RegExp (denoting the language $\{\epsilon\}$);
- Each symbol of A is a RegExp (denoting $\{a\}$, $a \in A$).

Induction. Let r and s be two RegExps, then

- $(r.s)$ is a RegExp (denoting r concatenated with s). For simplicity, the dot is often omitted;
- $(r|s)$ is a RegExp (denoting r union s);
- $(r)^*$ is a RegExp (denoting the smallest superset of r containing ϵ and closed under).

RegExp: Exercises

Build Regular Expressions for:

1. the set of strings that consists of alternating a's and b's;
2. the set of strings that consists of an odd number of a's;
3. the set of strings that ends with b and not contains the substring aa;
4. the set of strings which both the number of a's and the number of b's are even.

Consider the alphabet $A = \{a, b\}^*$ for previous exercises.

Solution.

RegExp: Exercises (1)

Build Regular Expressions for the set of strings that consists of alternating a's and b's

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

RegExp: Exercises (2)

Build Regular Expressions for the set of strings that consists of an odd number of a's

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

RegExp: Exercises (3)

Build Regular Expressions for the set of strings that ends with b and not contains the substring aa

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

RegExp: Exercises (4)

Build Regular Expressions for the set of strings which both the number of a's and the number of b's are even

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

From Regular Expression to (N)FSA.

The Thompson's Construction

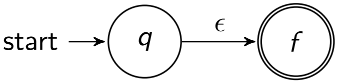
- It is an algorithm for transforming a regexp into an equivalent (N)FSA.
- This (N)FSA can be used to match strings against the regular expression.

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)

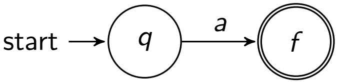
Rule: the empty expression

The empty-expression ϵ is converted to:



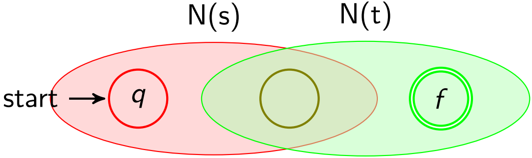
Rule: a symbol

A symbol a of the input alphabet is converted to



Rule: concatenation expression

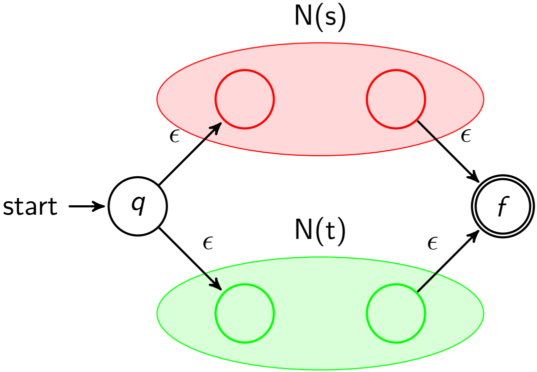
The concatenation expression st is converted to



$N(s)$ and $N(t)$ are the (N)FSA of the subexpression s and t, respectively.

Rule: union expression

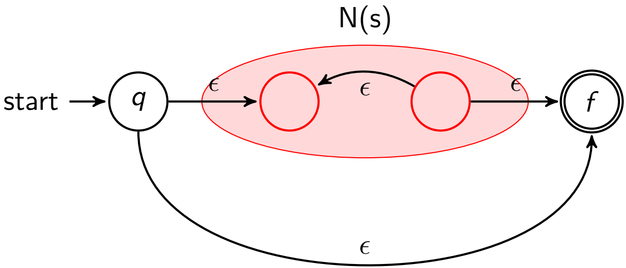
The union expression $s|t$ is converted to



$N(s)$ and $N(t)$ are the (N)FSA of the subexpression s and t, respectively.

Rule: Kleene star expression

The Kleene star expression s^* is converted to



$N(s)$ is the (N)FSA of the subexpression s.

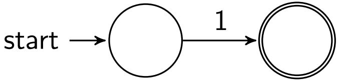
Exercise (1)

Build a (N)FSA for $(1 \mid 01)^*$

Exercise (1)

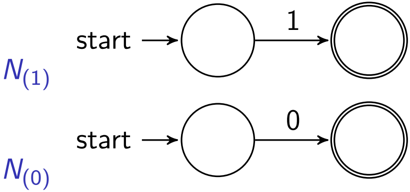
Build a (N)FSA for $(1 \mid 01)^*$

$N_{(1)}$



Exercise (1)

Build a (N)FSA for $(1 \mid 01)^*$



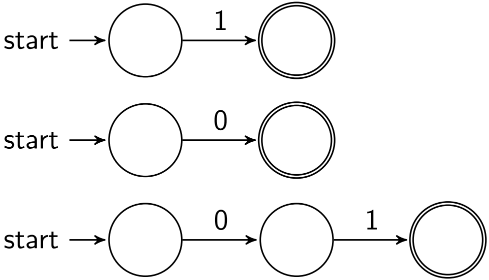
Exercise (1)

Build a (N)FSA for $(1 \mid 01)^*$

$N_{(1)}$

$N_{(0)}$

$N_{(01)}$



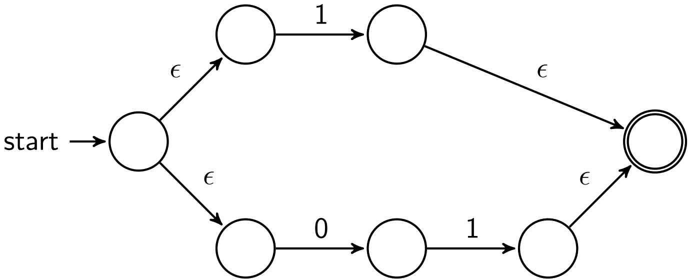
Exercise (1)

Build a (N)FSA for $(1 \mid 01)^*$

Exercise (1)

Build a (N)FSA for $(1 \mid 01)^*$

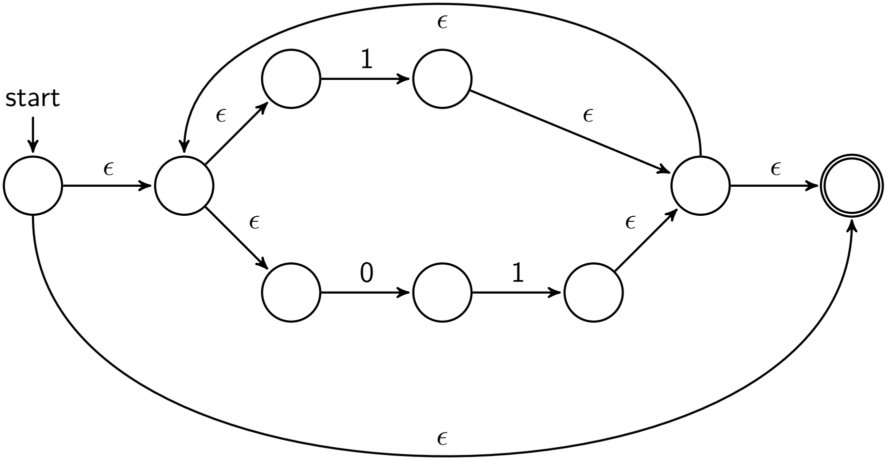
$N_{(1|01)}$



Exercise (1)

Build a (N)FSA for $(1 \mid 01)^*$

$N_{(1|01)^*}$



Exercises

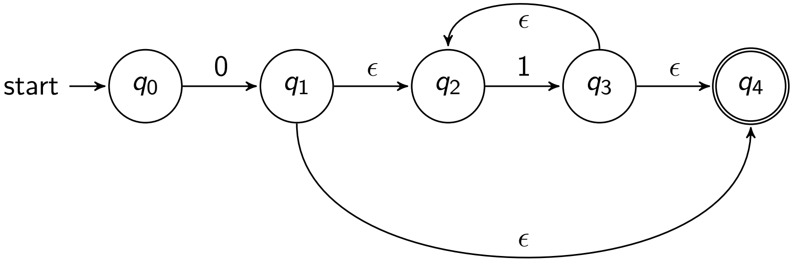
Build a (N)FSA for:

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

Solution.

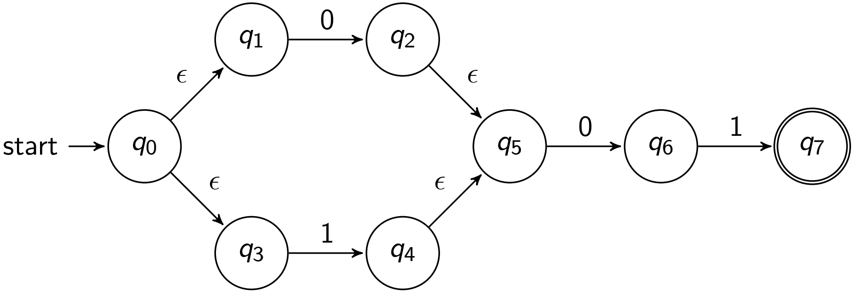
Exercises (1)

Build a (N)FSA for 01^*



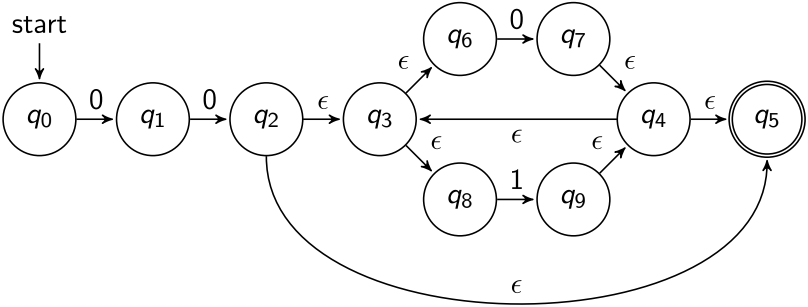
Exercises (2)

Build a (N)FSA for (0 | 1)01



Exercises (3)

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



FSA to RegExp

Kleene's algorithm: from FSA to Regular Expression

It transforms a given deterministic finite state automaton (FSA) into a regular expression.

Description: Given a 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 $F = \{q_1, \ldots, q_f\}$ is the set of accept states, the regular expression $R_{01}^n \mid \ldots \mid R_{0f}^n$ represents the language accepted by M.

Kleene's algorithm

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

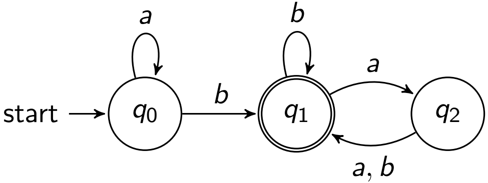
$$R_{ij}^{-1} = a_1 \mid \ldots \mid a_m \text{ if } i \neq j, \text{where } \delta\left(q_i, a_1\right) = \ldots = \delta\left(q_i, a_m\right) = q_j$$
$$R_{ij}^{-1} = a_1 \mid \ldots \mid a_m \mid \epsilon \text{ if } i = j, \text{where } \delta\left(q_i, a_1\right) = \ldots = \delta\left(q_i, a_m\right)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

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

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

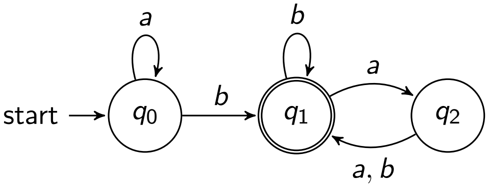
Kleene's algorithm: Exercise

Build a RegExp of the automaton below using the Kleene's algorithm



Kleene's algorithm: Exercise

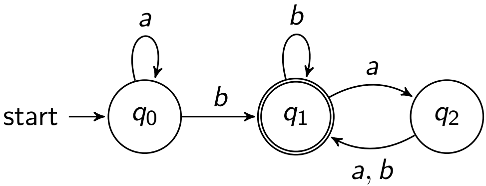
Initial Regular Expression (Step -1)



$$
\begin{aligned}
R_{00}^{-1} &= a \mid \epsilon \\
R_{01}^{-1} &= b \\
R_{02}^{-1} &= \emptyset \\
R_{10}^{-1} &= \emptyset \\
R_{11}^{-1} &= b \mid \epsilon
\end{aligned}
\qquad
\begin{aligned}
R_{12}^{-1} &= a \\
R_{20}^{-1} &= \emptyset \\
R_{21}^{-1} &= a \mid b \\
R_{22}^{-1} &= \epsilon
\end{aligned}
$$

Kleene's algorithm: Exercise

Step 0

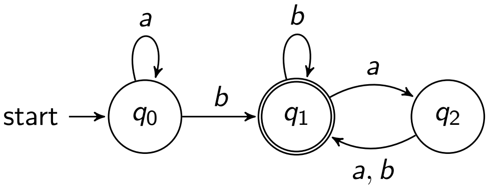


$$
\begin{aligned}
R_{00}^0 &= a^* \\
R_{01}^0 &= a^* b \\
R_{02}^0 &= \emptyset \\
R_{10}^0 &= \emptyset \\
R_{11}^0 &= b \mid \epsilon
\end{aligned}
\qquad
\begin{aligned}
R_{12}^0 &= a \\
R_{20}^0 &= \emptyset \\
R_{21}^0 &= a \mid b \\
R_{22}^0 &= \epsilon
\end{aligned}
$$

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

Kleene's algorithm: Exercise

Step 1



$$\begin{aligned} R^1_{00} &= a^* \\ R^1_{01} &= a^*b^*b \\ R^1_{02} &= a^*b^*ba \\ R^1_{10} &= \emptyset \\ R^1_{11} &= b^* \end{aligned}$$

$$\begin{aligned} R^1_{12} &= b^*a \\ R^1_{20} &= \emptyset \\ R^1_{21} &= (a \mid b)b^* \\ R^1_{22} &= (a \mid b)b^*a \mid \epsilon \end{aligned}$$

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

Kleene's algorithm: Exercise

Step 2

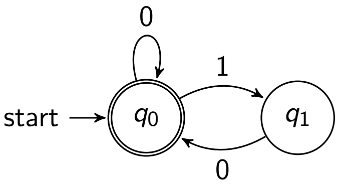
$$
\begin{aligned}
R_{00}^2 = &\quad a^*b^*ba((a \mid b)b^*a \mid \epsilon)^*\emptyset | a^* = a^* \\
R_{01}^2 = &\quad a^*b^*ba((a \mid b)b^*a \mid \epsilon)^*(a \mid b)b^* \mid a^*b^*b \\
R_{02}^2 = &\quad a^*b^*ba((a \mid b)b^*a \mid \epsilon)^*((a \mid b)b^*a \mid \epsilon) \mid a^*b^*ba \\
R_{10}^2 = &\quad b^*a((a \mid b)b^*a \mid \epsilon)^*\emptyset \mid \emptyset = \emptyset \\
R_{11}^2 = &\quad b^*a((a \mid b)b^*a \mid \epsilon)^*(a \mid b)b^* \mid b^* \\
R_{12}^2 = &\quad b^*a((a \mid b)b * a \mid \epsilon)^*((a \mid b)b^*a \mid \epsilon) \mid b^*a \\
R_{20}^2 = &\quad ((a \mid b)b^*a \mid \epsilon)((a \mid b)b^*a \mid \epsilon)^*\emptyset \mid \emptyset = \emptyset \\
R_{21}^2 = &\quad ((a \mid b)b^*a \mid \epsilon)((a \mid b)b^*a \mid \epsilon)^*(a \mid b)b^* \mid (a \mid b)b^* \\
R_{22}^2 = &\quad ((a \mid b)b^*a \mid \epsilon)((a \mid b)b^*a \mid \epsilon)^*((a \mid b)b^*a \mid \epsilon) \mid (a \mid b)b^*a \mid \epsilon
\end{aligned}
$$

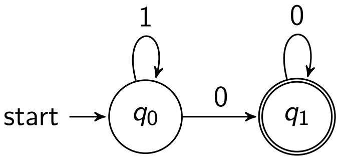
$$R_{ij}^k = R_{ik}^{k-1} \left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1} \mid R_{ij}^{k-1}$$

We are interested in R_{01}^2 since q_0 is the initial state and q_1 is the final state.

Exercises

Give a regular expression that describes the language accepted by:

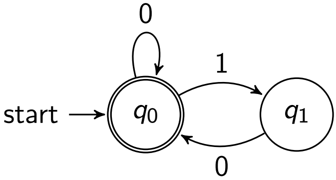
1. 

2. 

Solution.

Exercises (1)

Give a regular expression that describes the language accepted by:

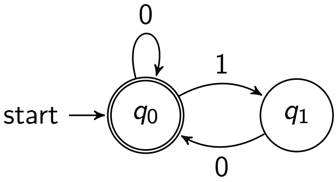


Initial Regular Expression (Step -1)

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

Exercises (1)

Give a regular expression that describes the language accepted by:

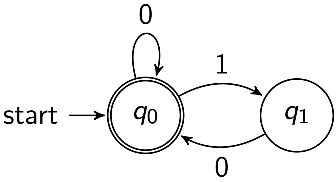


Step 0

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

Exercises (1)

Give a regular expression that describes the language accepted by:

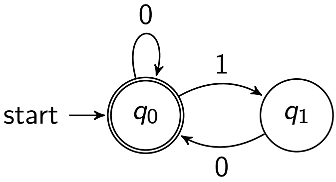


Step 1

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

Exercises (1)

Give a regular expression that describes the language accepted by:



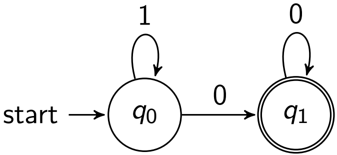
Step 1

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

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

Exercises (2)

Give a regular expression that describes the language accepted by:

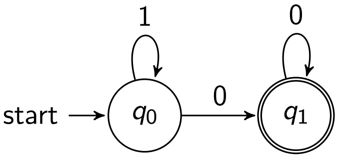


Initial Regular Expression (Step -1)

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

Exercises (2)

Give a regular expression that describes the language accepted by:

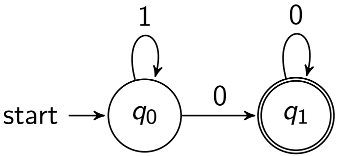


Step 0

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

Exercises (2)

Give a regular expression that describes the language accepted by:

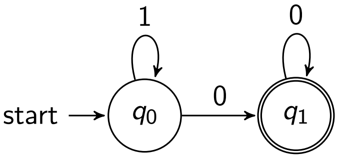


Step 1

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

Exercises (2)

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})