Converting Finite Automata to Regular Expressions

Petr Zemek

Brno University of Technology
Faculty of Information Technology

2009-10-07

Basic Terms

- Finite automata (NFAs, DFAs)
- Regular expressions (REGEXPs)
- . . .

Why?

Two possible transformations:

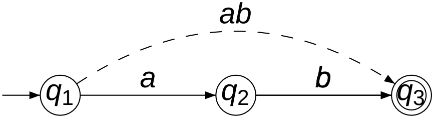
- Regular expression $\longrightarrow$ Finite automaton

  $\checkmark$

- Finite automaton $\longrightarrow$ Regular expression

  Uhm. . . Why?

Transitive Closure Method

- Rather theoretical approach.
- 

- Sketch of the method:
  1. Let $Q = \{q_1, q_2, \ldots, q_m\}$ be the set of all automatons states.
  2. Suppose that regular expression R_{ij} represents the set of all strings that transition the automaton from q_i to q_j.
  3. Wanted regular expression will be the union of all R_{sf}, where q_s is the starting state and q_f is one the final states.

- The main problem is how to construct R_{ij} for all states q_i, q_j.

How to construct R_{ij}?

- Suppose R_{ij}^k represents the set of all strings that transition the automaton from q_i to q_j without passing through any state higher than q_k. We can construct R_{ij} by successively constructing $R_{ij}^1, R_{ij}^2, \ldots, R_{ij}^{|Q|} = R_{ij}$.
- R_{ij}^k is recursively defined as:

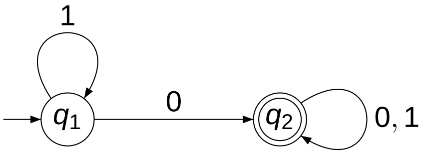
$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$$

- Assuming we have initialized R_{ij}^0 to be:

$$R_{ij}^0 = \begin{cases} r & \text{if } i \neq j \text{ and r transitions NFA from } q_i \text{ to } q_j \\ r + \varepsilon & \text{if } i = j \text{ and r transitions NFA from } q_i \text{ to } q_j \\ \emptyset & \text{otherwise} \end{cases}$$

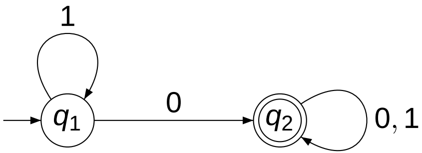
Example (1/5)

Transform the following NFA to the corresponding REGEXP using Transitive Closure Method:



Example (2/5)

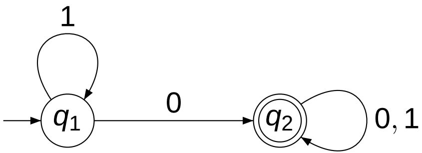
1) Initialize R^0_{ij}:



$$\begin{array}{l|l} R^0_{11} & \varepsilon + 1 \\ R^0_{12} & 0 \\ R^0_{21} & \emptyset \\ R^0_{22} & \varepsilon + 0 + 1 \end{array}$$

Example (3/5)

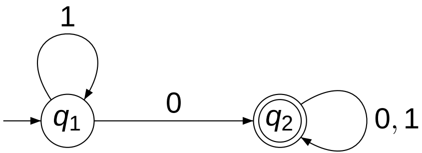
2) Compute R_{ij}^1:



	By direct substitution	Simplified
R_{11}^1	$\varepsilon + 1 + (\varepsilon + 1)(\varepsilon + 1)^*(\varepsilon + 1)$	1^*
R_{12}^1	$0 + (\varepsilon + 1)(\varepsilon + 1)^*0$	1^*0
R_{21}^1	$\emptyset + \emptyset(\varepsilon + 1)^*(\varepsilon + 1)$	$\emptyset$
R_{22}^1	$\varepsilon + 0 + 1 + \emptyset(\varepsilon + 1)^*0$	$\varepsilon + 0 + 1$

Example (4/5)

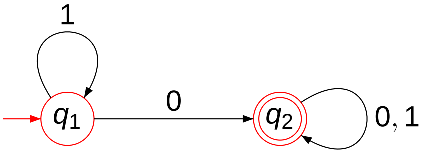
3) Compute R^2_{ij}:



	By direct substitution	Simplified
R^2_{11}	$1^* + 1^*0(\varepsilon + 0 + 1)^*\emptyset$	1^*
R^2_{12}	$1^*0 + 1^*0(\varepsilon + 0 + 1)^*(\varepsilon + 0 + 1)$	$1^*0(0 + 1)^*$
R^2_{21}	$\emptyset + (\varepsilon + 0 + 1)(\varepsilon + 0 + 1)^*\emptyset$	$\emptyset$
R^2_{22}	$\varepsilon + 0 + 1 + (\varepsilon + 0 + 1)(\varepsilon + 0 + 1)^*(\varepsilon + 0 + 1)$	$(0 + 1)^*$

Example (5/5)

4) Get the resulting regular expression:



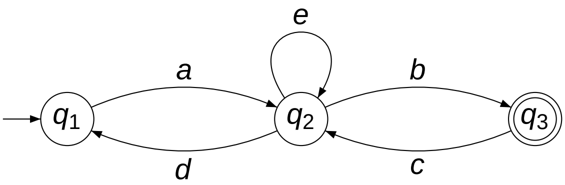
$\Rightarrow R_{12}^2 = R_{12} = 1^*0(0+1)^*$ is the REGEXP corresponding to the NFA.

State Removal Method

- Based on a transformation from NFA to GNFA (*generalized nondeterministic finite automaton*).
- Identifies patterns within the graph and removes states, building up regular expressions along each transition.
- Sketch of the method:
  1. Unify all final states into a single final state using ε-trans.
  2. Unify all multi-transitions into a single transition that contains union of inputs.
  3. Remove states (and change transitions accordingly) until there is only the starting a the final state.
  4. Get the resulting regular expression by direct calculation.
- The main problem is how to remove states correctly so the accepted language won't be changed.

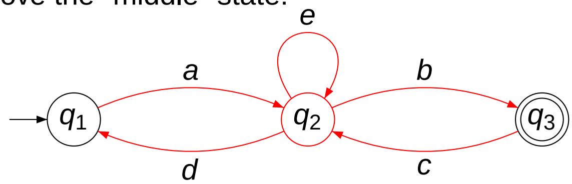
Example (1/3)

Transform the following NFA to the corresponding REGEXP using State Removal Method:

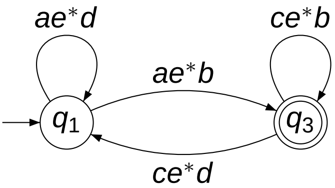


Example (2/3)

1) Remove the "middle" state:

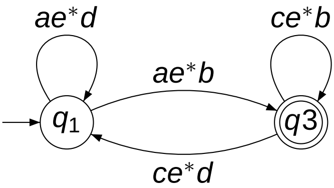


$\Downarrow$



Example (3/3)

2) Get the resulting regular expression r:



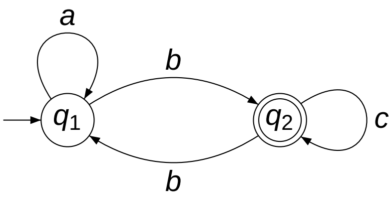
$$\Rightarrow r = (ae^*d)^*ae^*b(ce^*b + ce^*d(ae^*d)^*ae^*b)^*.$$

Brzozowski Algebraic Method

- Janusz Brzozowski, 1964
- Utilizes *equations over regular expressions*.
- Sketch of the method:
  1. Create a system of regular equations with one regular expression unknown for each state in the NFA.
  2. Solve the system.
  3. The regular expression corresponding to the NFA is the regular expression associated with the starting state.
- The main problem is how to create the system and how to solve it.

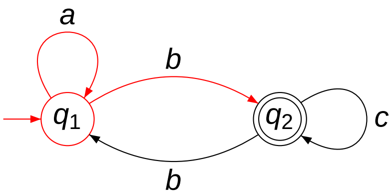
Example (1/5)

Transform the following NFA to the corresponding REGEXP using Brzozowski Method:



Example (2/5)

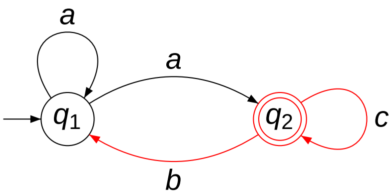
1) Create a characteristic regular equation for state 1:



$$X_1 = aX_1 + bX_2$$

Example (3/5)

2) Create a characteristic regular equation for state 2:



$$X_2 = \varepsilon + bX_1 + cX_2$$

Example (4/5)

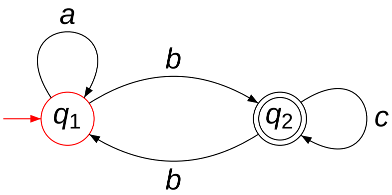
4) Solve the arisen system of regular expressions:

$$\begin{aligned} X_1 &= \phantom{\varepsilon + {}} aX_1 + bX_2 \\ X_2 &= \varepsilon + bX_1 + cX_2 \end{aligned}$$

Example (5/5)

Solution:

$$X_1 = (a + bc^*b)^*bc^*$$
$$X_2 = c^*[\varepsilon + b(a + bc^*b)^*bc^*]$$



$\Rightarrow X_1$ is the REGEXP corresponding to the NFA.

Comparison of presented methods

- Transitive Closure Method
  - \+ clear and simple implementation
  - \- tedious for manual use
  - \- tends to create very long regular expressions
- State Removal Method
  - \+ intuitive, useful for manual inspection
  - \- not as straightforward to implement as other methods
- Brzozowski Algebraic Method
  - \+ elegant
  - \+ generates reasonably compact regular expressions

References

- Christoph Neumann
  Converting Deterministic Finite Automata to Regular Expressions.
  http://neumannhaus.com/christoph/papers/2005-03-16.DFA_to_RegEx.pdf
- John E. Hopcroft and Jeffery D. Ullman
  Introduction to Automata Theory, Languages, and Computation.
  Addison-Wesley Publishing Company, Reading, MA, 1979.
- Peter Linz
  An introduction to Formal Languages and Automata.
  Jones and Bartlett Publishers, Sudbury, MA, third edition, 2001.
- Milan Češka and Tomáš Vojnar and Aleš Smrčka
  Studijní opora do předmětu Teoretická Informatika.
  https://www.fit.vutbr.cz/study/courses/TIN/public/Texty/oporaTIN.pdf
- Janusz A. Brzozowski
  Derivatives of regular expressions.
  J. ACM, 11(4):481–494, 1964.