Introduction to Logic and Computer Science
Foundation of Computer Science
Logic in Computer Science

Lecture #12: Finite automaton and regular language

Tamás Mihálydeák, László Aszalós

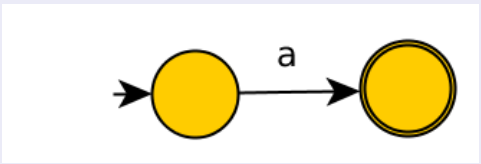
This work was supported by the construction EFOP-3.4.3-16-2016-00021.
The project was supported by the European Union, co-financed by the European Social Fund.

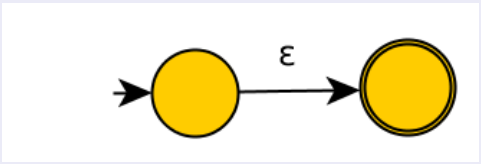
1. Automaton based on regular expression
2. Regular expression based on automaton
3. Construction

Automaton based on regular expression

By following the inductive definition we construct a non-deterministic automaton. Inductive hypothesis: for R_1 and for R_2 we have accepting automata A_1 and A_2

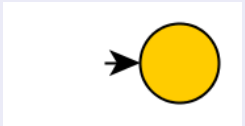
$R = a$, where $a \in \Sigma \cup \{\varepsilon\}$





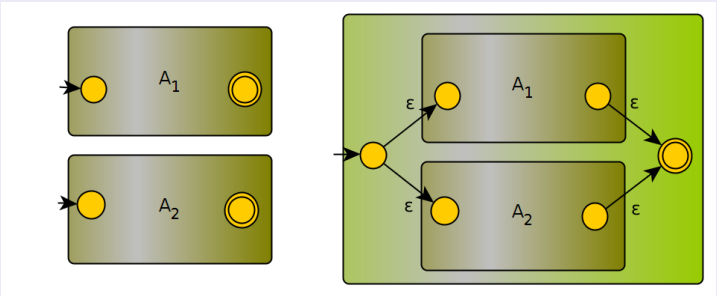
$R = \emptyset$

There is no final state, it does not accept anything!

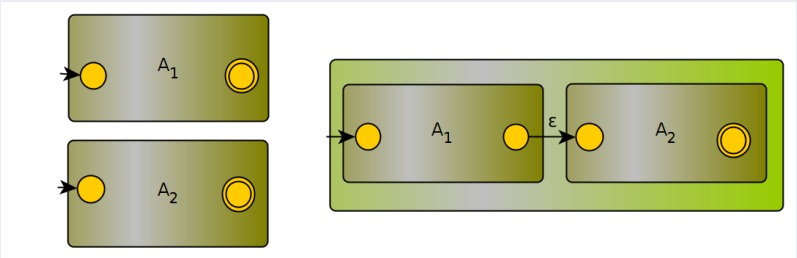


Automaton based on regular expression

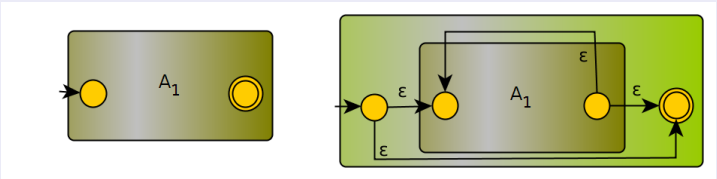
$R = R_1 \cup R_2$



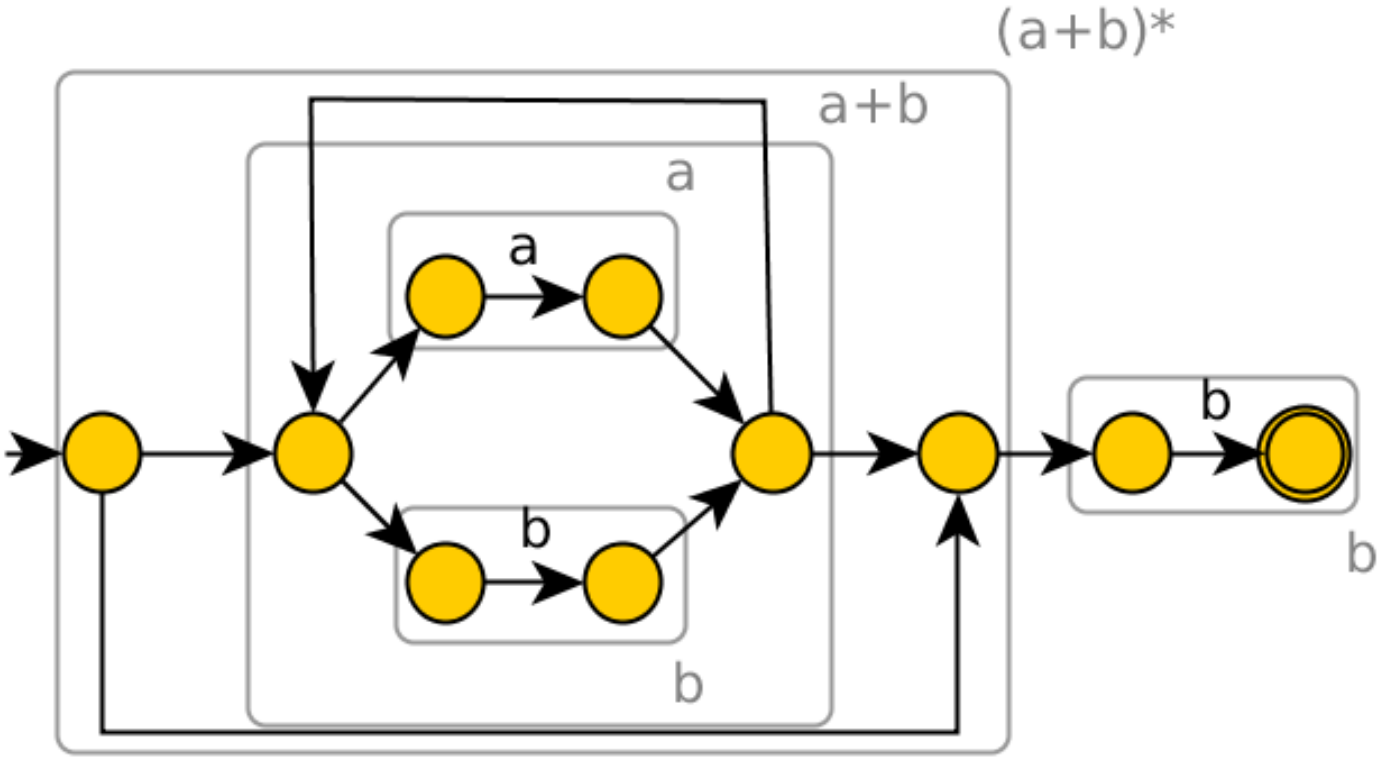
$R = R_1 \circ R_2$



$R = R_1^\star$



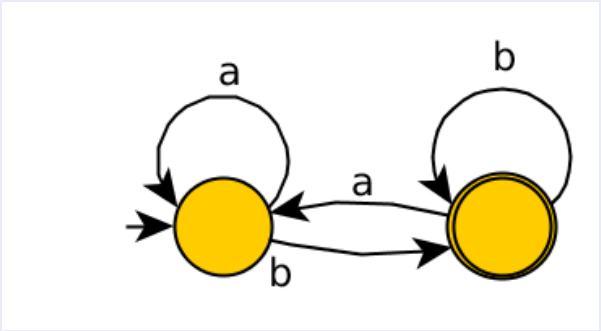
Example: $(a + b)^\star b$



Example: $(a + b)^\star b$

Remark

Following this step, Thompson's algorithm produced a deterministic version of the non-deterministic finite automaton and then minimized it. Applying this algorithm, we get the following automaton:



The implementation of this algorithm can be find in the lexers *lex* and *flex*.

Regular expression based on automaton

Theorem

For any deterministic automaton A there exist a regular expression R, such that $L_A = L_R$.

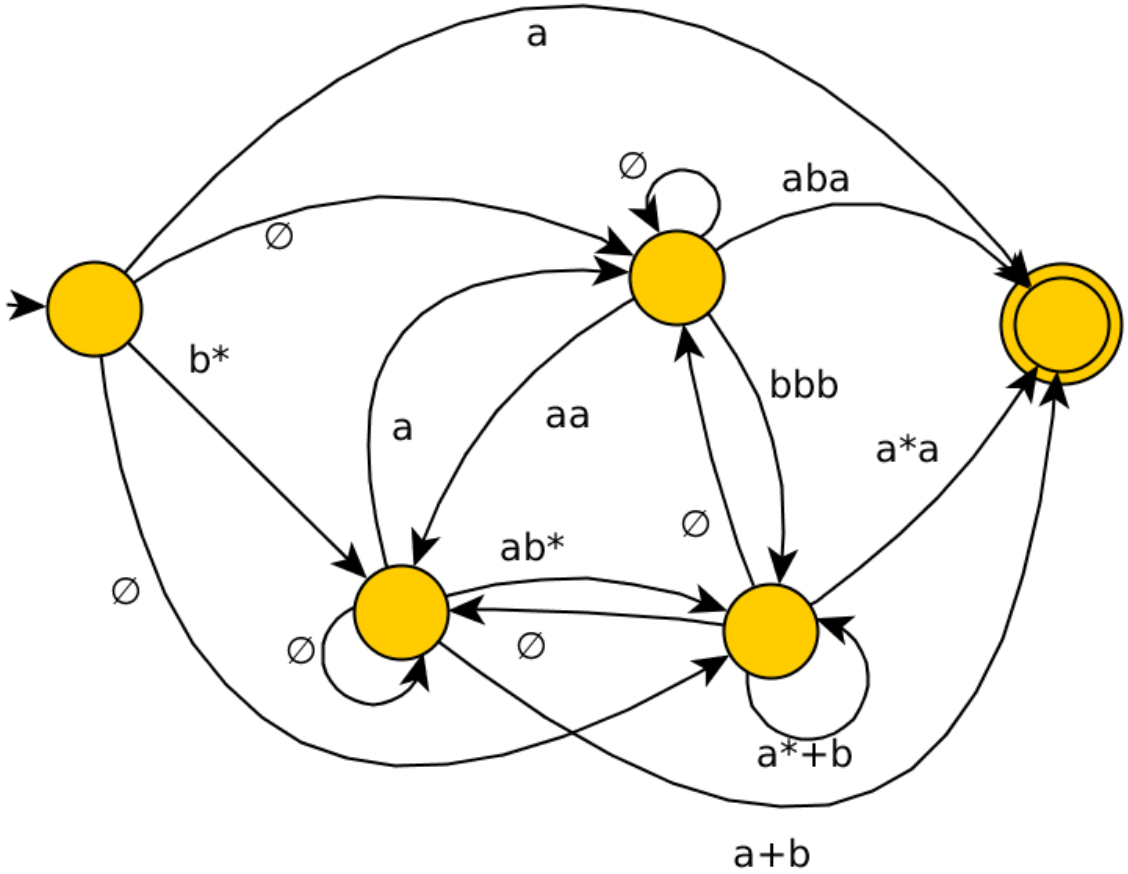
Remark

We need a more general automaton to do this.

Construction

- The transition function of the more general non-deterministic finite automaton can even contain a regular expression (instead of one letter).
- The starting state has no incoming edges, and has outgoing edges into every other state.
- It has one final state only. This state has no outgoing edge, and has ingoing edges from every other state. (The starting state and the final state are different.)
- All the states (except the starting and final) have outgoing edges to every states, even to itself.

Example: more general non-deterministic final automaton



More general NFA based on finite automaton

Construction

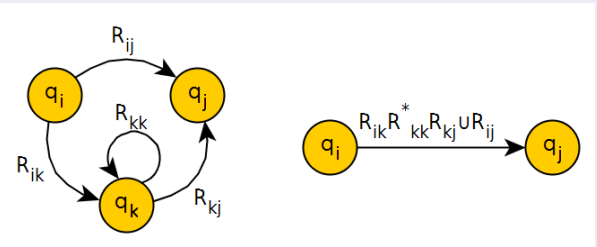
1. Create a new starting state and connect it with the old starting state with (outgoing) edge labelled ε
2. Create a new final state and connect it with the all the old final states with (incoming) edge labelled ε
3. If there are parallel edges between two states replace with one labelled by the union of the previous labels.
4. If there is no edge between two edges, add a new one with label $\emptyset$ (so it will be an unusable edge)

RE based on more general NFA

Construction

Let's assume that the automaton has k states.

- If $k = 2$, the only label contains the regular expression.
- If $k > 2$, then we construct a more general NFA with $k - 1$ states:
  - Let denote R_{ij} the label of the edge between q_i and q_j.
  - Let choose to delete the state q_k.
  - For any states i and j (different from k let replace R_{ij} with $R_{ik} \circ R_{kk}^{\star} \circ R_{kj} \cup R_{ij}$



Corollary

Theoram

The languages accepted by finite automata are regular.