CSC 320 - Lecture 06

#regular #expressions #shrinking #DFA #equivalence-method

Regular Expressions

Remember. Class of Languages recognized by DFAs $=$ Class of Languages recognized by NFAs $=$ Regular Languages.

Claim 2. If a language is regular, then it can be described by a regular expression.

Idea. We know that if L is a regular language then L_1 is accepted by a finite automaton.

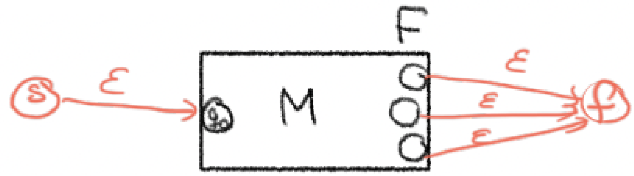
Plan. Describe procedure that converts finite automaton M into regular expression.

- Transform = M into a *hybrid* automaton G - between automaton and regular expression(s).
- Shrink until **obtaining regular expression** that recognizes same language as original automaton M.

Preparation

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$, create new generalized (nondeterministic finite automaton).

1. Add new start state s and ε-transition from s to q_0.
2. G has one accept state.
    1. Add new accept state f.
    2. ε-transitions from all states in F to f ($L(G) = L(M)$).
3. Transform label on each transition into regular expression:
    1. Eg. If label is a, b change it into $a \cup b$ (language does not change).



For each pair of states q_a and q_b with more than one transition from q_a to q_b, combine to one transition.

For each pair of states q_a and q_b with no transition from q_a to q_b, add a transition and label it with $\emptyset$.

Generalized Automaton G

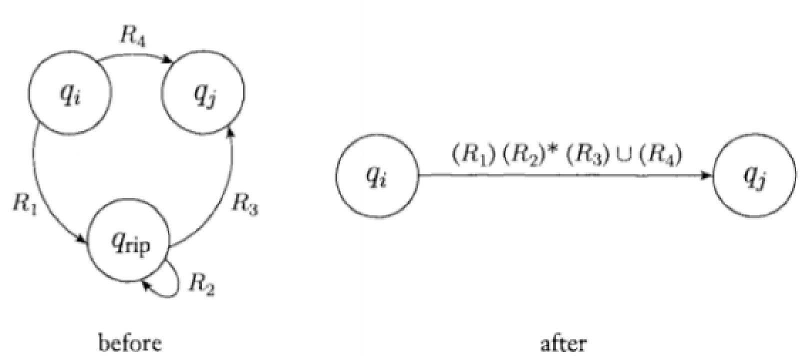
G almost like NFA, except:

- Transitions in the state diagram labeled with regular expressions.
- Exactly one start state.
- Exactly one accept state.
- Exactly one transition from every state to every other state (including same state).
  - Exceptions: No transition from accept state, no transition to the start state.

Transforming G Into Regular Expression R

Next we shrink/simplify G:

- Remove states from Q (that is neither s or f), one by one, ensuring that G does not change language it recognizes.

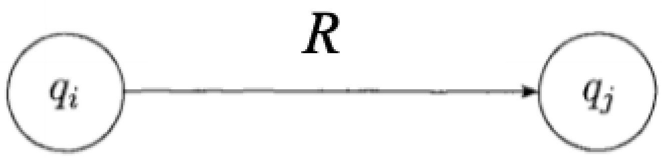


Shrinking

Consider $G = (Q \cup \{s, f\}, \Sigma, \delta, s, f)$

For $q_i, q_j \in Q \cup \{s, f\}$: reg$(q_i, q_j) = R$

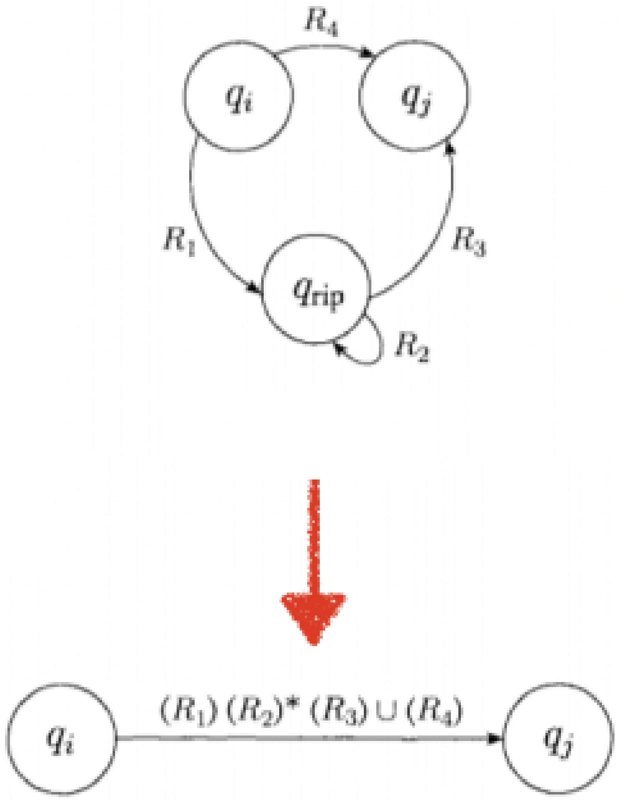
- R is regular expression with $\delta(q_i, R) = q_j$.



Choose a state $q_{rip} \in Q$ and transform machine $G = (Q \cup \{s, f\}, \Sigma, \delta, s, f)$ to $G' = (Q', \Sigma, \delta', s, f)$ with $Q' = Q \cup \{s, f\} - \{q_{rip}\}$ and update δ to δ'. (Remove q_{rip}).

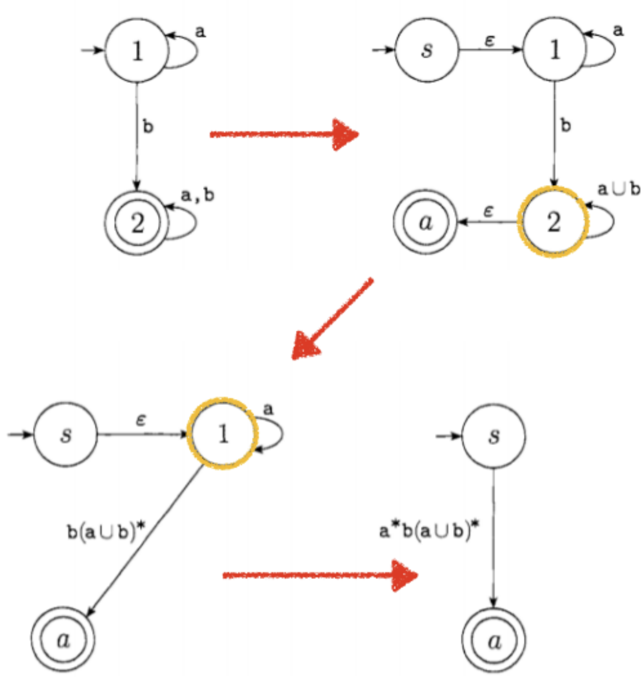
- For each $q_i \in Q'$ and $q_j \in Q'$: reg$'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4))$ where...
  - $R_1 =$ reg(q_i, q_{rip})
  - $R_2 =$ reg(q_{rip}, q_{rip})
  - $R_3 =$ reg(q_{rip}, q_j)
  - $R_4 =$ reg(q_i, q_j)

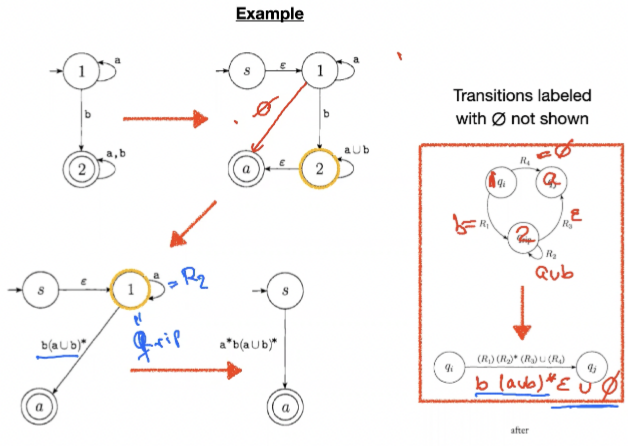
Note. We repeat the step until s and f are the only states left.

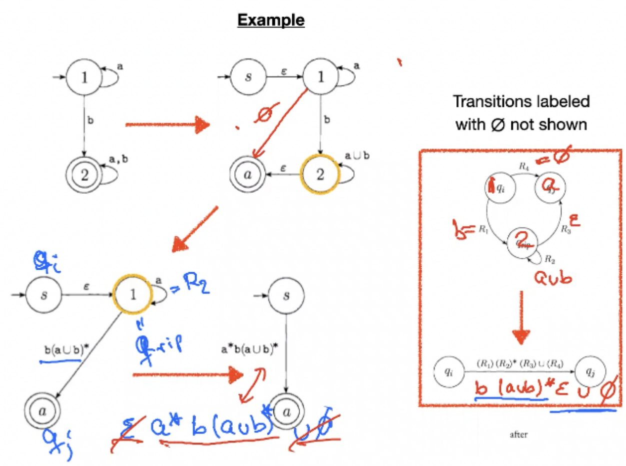


Note. Transitions labeled with $\emptyset$ not shown.

Example







Note. $b(a \cup b)^* \varepsilon \cup \emptyset$ and $a^* b(a \cup b)^*$

Finishing Up...

Show. Regular expression R labeling transition from s to f is regular expression that describes $L(M)$.

Still To Do. Prove that $L(R) = L(M)$.

- We show that in every step of state removal $L(G) = L(G')$.
- In other words we show that ripping out a state doesn't change the language.

We Show: $L(G) = L(G')$

Note. $L(G) \subseteq L(G')$

1. When removing state q_{rip}: Every string accepted by G through transitions that **did not** pass through q_{rip} remains in language.
2. Consider string accepted by G via transitions passing through q_{rip}.

    1. Removing q_{rip} from accepting sequence of states in G yields accepting sequence of states in G.
        1. Let $\ldots, q_i, q_{rip}, \ldots, q_{rip}, q_j, \ldots$ be sequence of states during computation in G.
        2. reg$'(q_i, q_j)$ includes any substring recognized through $\ldots, q_i, q_j, \ldots$
        3. Thus, $\ldots, q_i, q_j, \ldots$ is accepting computation in G'.
3. $L(G') \subseteq L(G)$
    1. G' accepts string w
    2. w accepted by G' corresponds to the concatenation of labels on path in G.
    3. w must have been accepted by G.

Summary

Theorem. A language is regular if and only if there exists some regular expression that describes it.

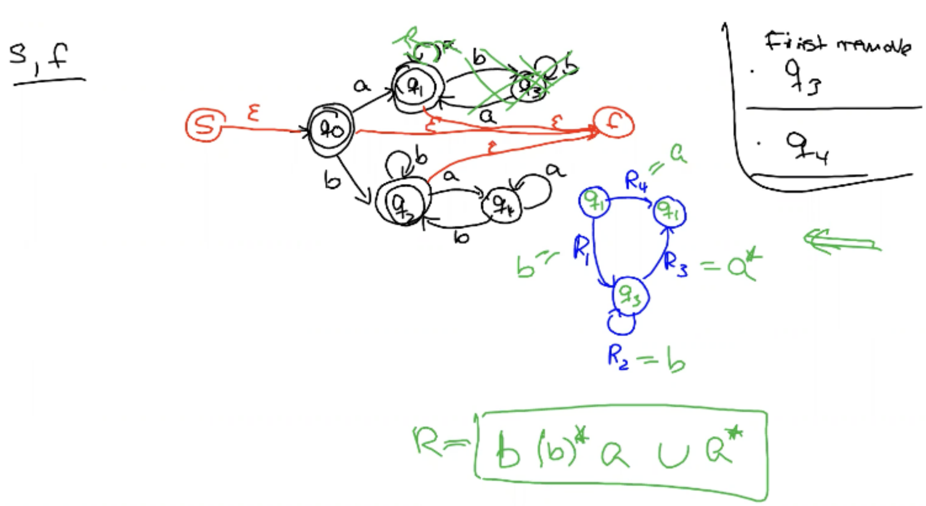
1. If a language is described by a regular expression, then it is regular

> Regular Expression $\longrightarrow$ DFA $\longrightarrow$ Regular Language

2. If a language is regular, then it can be described by a regular expression.

> Regular Language $\longrightarrow$ DFA $\longrightarrow$ GNFA $\longrightarrow$ Regular Expression

Example



Coming Up

- DFA State Minimization (Myhill-Nerode).
- Non-Regular Languages and the Pumping Lemma.

Is There A Systematic Way to Reduce the Number of States in A Finite Automaton?

- **Recall**. Two finite automata are equivalent if they both recognize the same language.
- **Goal**. Find an algorithm that allows us to reduce the number of states of a finite automaton while maintaining its language.
- We can do this for deterministic finite automata.

Note. This only works for DFAs!

Note. You can also use a technique called Equivalence Method to minimize DFAs.

Previous Lecture

[Lecture05]

Next Lecture

[Lecture07]