University of Waterloo
CS 360 — Introduction to the Theory of Computing
Fall 2017
Handout on State Elimination

In this handout we describe an alternate way of going from a (possibly nondeterministic) finite automaton to a regular expression. This method is somewhat easier to use than the one described in John Watrous' course notes, pp. 43–44. This handout is based on one by Eric Bach, University of Wisconsin.

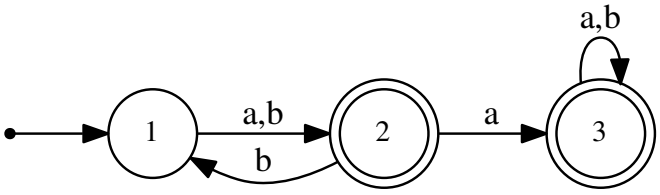
The idea is to expand our notion of NFA-ϵ to allow transitions on arbitrary regular expressions, not simply single symbols or ϵ. We successively eliminate states, replacing transitions that enter and leave a state with a more complicated regular expression, until eventually there are only two states left: a start state, an accepting state, and a single transition connecting them, labeled with a regular expression.

To begin with, the automaton should have

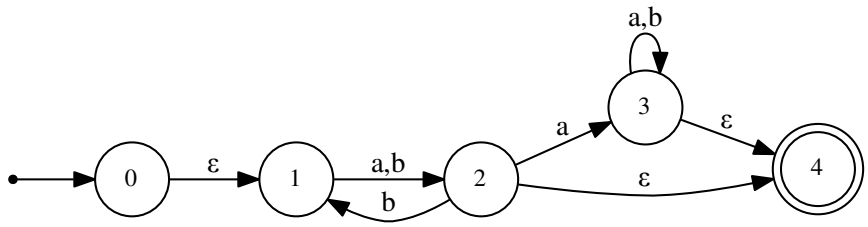
- a start state s that has no transitions into s (including self-loops), and which is not accepting.
  - If your automaton does not obey this property, add a new, non-accepting start state s', and add an ϵ-transition from s' to s.
- The automaton should also have a single accepting state q with no transitions leaving q (including self-loops).
  - If your automaton does not obey this property, add a new accepting state q', change all other accepting states to non-accepting, and add ϵ-transitions from them to q'.

Note that these changes don't change the language recognized by the automaton.

For example, if your automaton is:

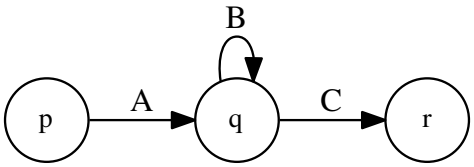


then you should modify it as follows:

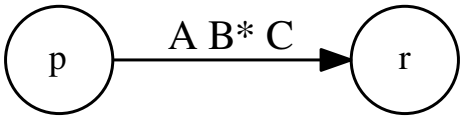


Next, repeat the following steps, which eliminate a state, until there are exactly two states left with a single transition connecting them:

1. Pick a non-start, non-accepting state q to eliminate. The state q will have i transitions in and j transitions out. Each will be labelled with a regular expression. For each of the ij combinations of transitions into and out of q, replace



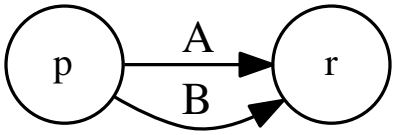
with



and then delete state q. Of course, you might need to insert some parens around A, B, and C.

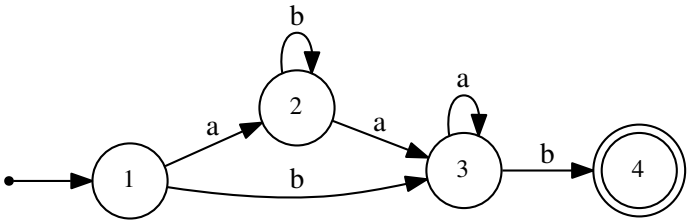
2. If many different transitions go between a pair of states, replace them with a single transition labeled by the union ($\cup$) of the individual transition labels.

For example, replace

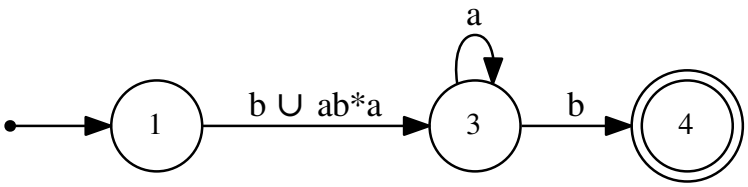


with

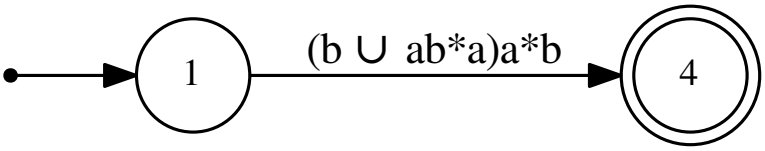
Here's an example of the method. Start with the NFA



and eliminate state 2 to get



Now eliminate state 3 to get



So a regular expression for the language recognized by the original automaton is $(b \cup ab^*a)a^*b$. In general, if you choose a different order to eliminate the states, you could get a different regular expression.