Prep Work 4 - Regex Conversions

CS 234

due February 17, before class

0 Introduction

This assignment has 1 part: conversions.

This assignment is to be completed individually, but feel free to collaborate according to the course's external collaboration policy (which can be found in the syllabus).

The deliverables consist of one `.pdf` file. The deliverables should be submitted electronically by the deadline. Put any attribution text in the `.pdf` file.

Every file should be named like `FLast_cs234_pX.ext` where `F` is your first initial, `Last` is your last name, `X` is the assignment number, and `ext` is the appropriate file extension. For example, Alexandra Silva's `.pdf` file should be given the name `ASilva_cs234_p4.pdf`. (Alexandra Silva is researcher who works with regular expressions and automata. I've also coauthored a paper with her!)

1 Conversions

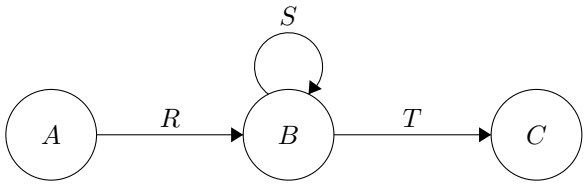
Read chapter 6 in the textbook. Then complete the following tasks in your `.pdf` submission. Clearly label your responses with the task number.

First, here are tasks concerning converting regexes to ϵNFAs.

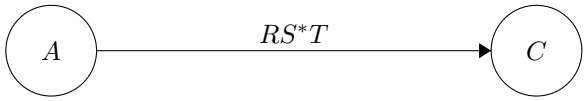
1. Look at figure 6.2. In your own words, how does it make an automaton for the union ($+$) of the languages of NFA1 and NFA2?
2. Look at figure 6.3. In your own words, how does it make an automaton for the concatenation ($\cdot$) of the languages of NFA1 and NFA2?
3. Look at figure 6.4. In your own words, how does it make an automaton for the Kleene star ($*$) of the language of NFA1?
4. Try converting $a^* + bc$ to an ϵNFA using the method described in section 6.2.

The textbook gives many good examples for state elimination, but the actual algorithm it specifies (algorithm 6.1) is a little light on detail. To help you out, here is a more detailed algorithm:

1. Let the transitions on the automaton be labelled with *regexes* instead of just letters. Initially, each regex will just be the single letter that is already present.
2. For each final state, obtain a regex via the following method:
   (a) Use the following procedure to eliminate each state besides the start state and the selected final state (the start state and final state may be the same state).
      i. Pick a state B to eliminate, and consider the set of its incoming transitions I and outgoing transitions O. (Do not include self-loops in these sets.) For each incoming-outgoing pair in $I \times O$, do the following:
         A. Suppose B is the state to eliminate, the chosen incoming transition is from A, and the chosen outgoing transition is to C (A and C might be the same state). Such a situation abstractly looks like the following automaton. (If B has no self-loops, then $S = 1$.)



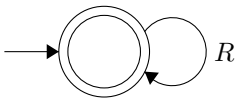
B. Add a new transition going from A to C that transitions with RS^*T. (If A and C are the same state, this new transition is a self-loop.) If there already is an transition from A to C, add this to the existing regex on that transition.



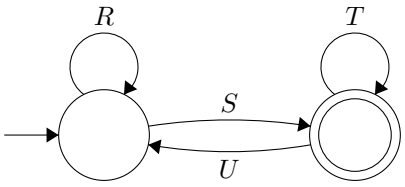
ii. Once that is done for every possible pair of incoming-outgoing transitions, remove B from the automaton.

(b) Now all states besides the start and the selected final state are gone. Take one of the following regexes depending on what the form of the resulting automaton is.

i. If the final and start state are the same, then the resulting automaton looks like the following, and R^* is its associated regex. (If there is no self-loop, $R = 1$ so that R^* is also 1.)



ii. Otherwise, the resulting automaton looks like the following, and $(R + ST^*U)^*ST^*$ is its associated regex. (As usual, if there is no self-loop, it is considered to have a self-loop with regex 1.)



3. Sum together all obtained regexes to get the regex for the language of the original automaton.

The remaining tasks concern converting DFAs to regexes.

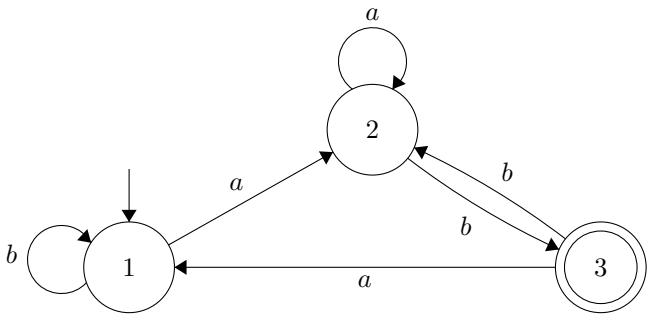
1. In your own words, how does the transition labelled RS^*T added in step 2.a.i.B preserve the set of paths from A to C?

2. In your own words, why is R^* the associate regex for the automaton in step 2.b.i?

3. In your own words, why is $(R + ST^*U)ST^*$ the associate regex for the automaton in step 2.b.ii?

4. In your own words, why do we sum together the regexes obtained for each final state in step 3?

5. If you were to run the algorithm by hand on the following automaton, you would only need to eliminate state 2. What are all the pairs of incoming-outgoing transitions to consider for state 2?



6. Run the algorithm by hand on the above automaton to find a regex for its language.