CMSC330 Spring 2023 Quiz 3 Solutions

Proctoring TA: ____________________ **Name:** ____________________

UID: ____________________

Problem 1: Basics

[Total 3 pts]

Please circle **True** or **False** for the following statements:

DFA's can contain epsilon transitions **True** [✓ **False**]
colorblue Taken from Notes and Lecture. Since epsilon transitions are optional using them causes uncertainty or non-determinism as to how to traverse the graph.

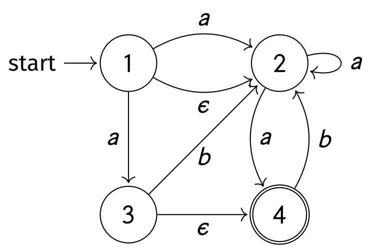
All DFA's are NFA's **True** [✓ **False**]
Stated in Lecture. DFAs are a subset of NFAs

All NFA's can be converted to Regex [✓ **True**] **False**
Said in lecture and said in notes. Every NFA has a Regex equivalence. You can translate NFA to DFA to regex systematically, but you can go directly from NFA to regex, although this process is much harder

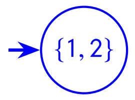
Problem 2: NFA to DFA

[Total 8 pts]

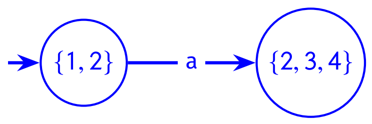
Convert the following NFA to a DFA. Square/circle your final DFA.



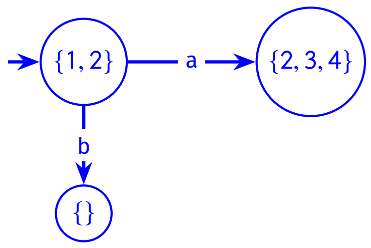
We will use the algorithm in the notes/stated in lecture and will show each iteration of the `while loop` but first, the setup.
The starting state of the DFA is calculated by doing an ϵ-closure on the starting of the NFA



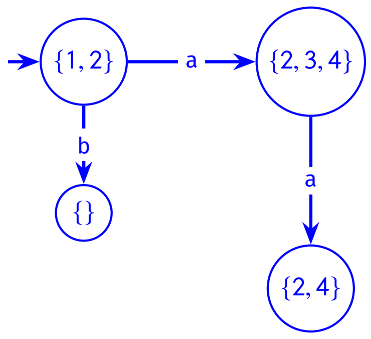
We now need to move on each character of the alphabet and perform ϵ`-closure` on the result. We will start with the `"a"` symbol.

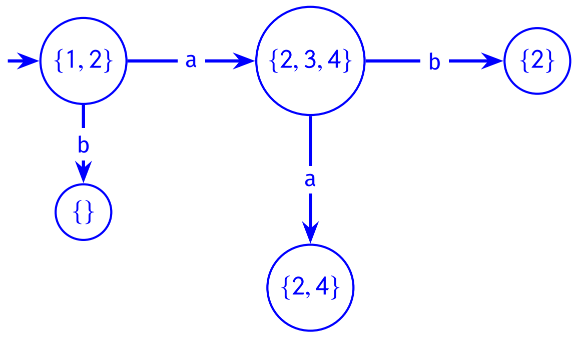


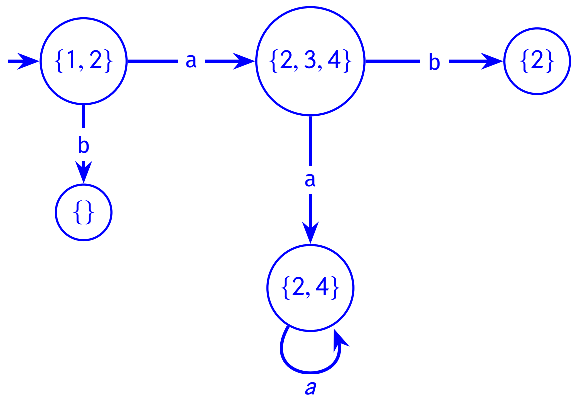
Now on the character of "b"

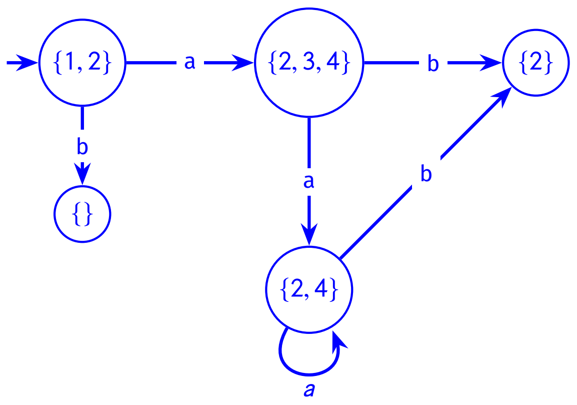


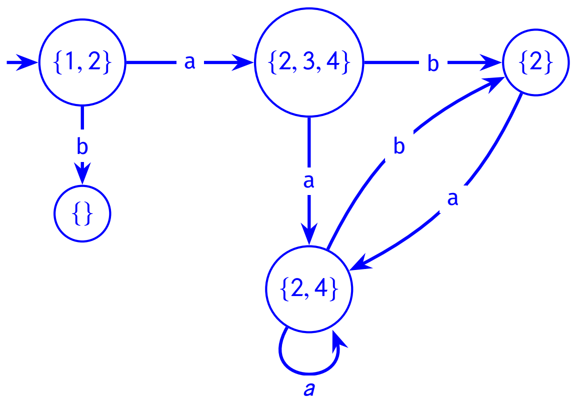
Now we do the same thing on the new states we made. Important to note that the node {} can be treated as the garbage state.

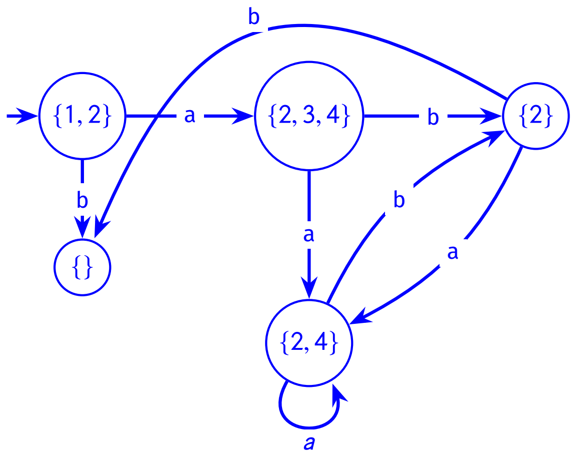




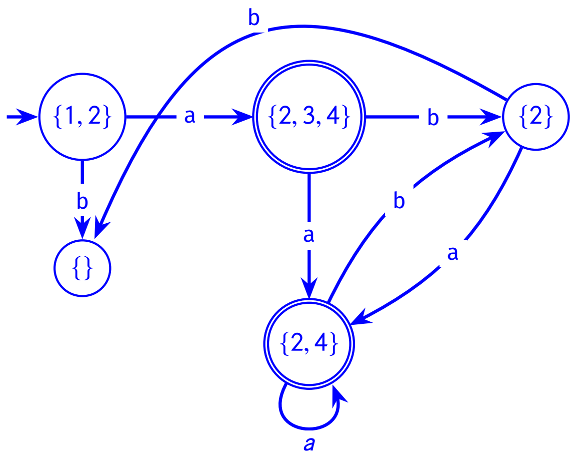




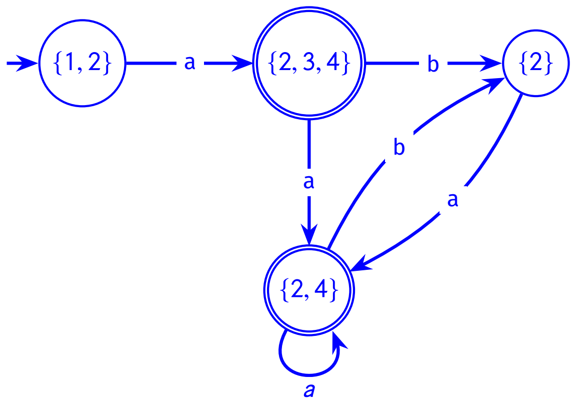




Now we just need to mark the final states:

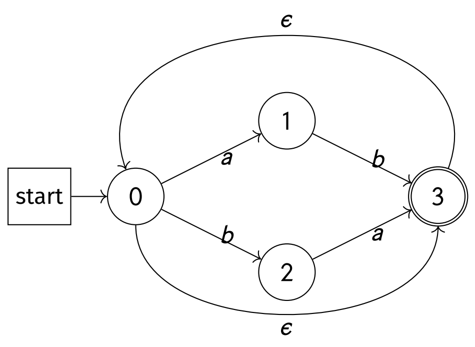


Without the garbage state (optional):



Problem 3: NFA and Regex

[Total 9 pts]



(a) What Regex corresponds exactly to the following NFA? (Only circle the regex that best matches the NFA) [2 pts]

`(abba)*`   `(ab)+`   `(ba)(ba)*`   `abba`   [✓ `(ab|ba)*`]

We could sole this a few different ways. First by looking at the structure, noticing there is a cycle and 2 distinct paths. Alternatively we could have come up with a bunch of strings the machine accepts that the regular expression does not. For example:
`(abba)*` - The machine accepts the string `"ab"` but this regex does note
`(ab)+` - The machine accepts the string `"ba"` but this regex does note
`(abba)*` - The machine accepts the string `"ab"` but this regex does note
`(abba)*` - The machine accepts the string `"ab"` but this regex does note

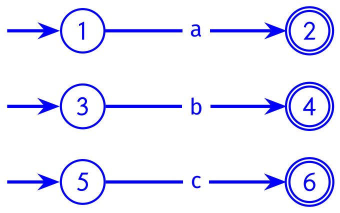
(b) Which of the following strings are accepted by the NFA? (You may choose more than one answer) [2 pts]

[✓ `Empty String`]   `a`   [✓ `abba`]   `baabb`   `bba`

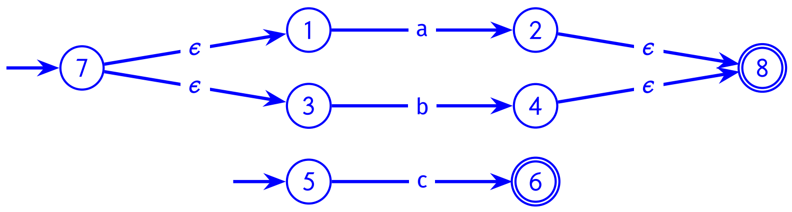
Here are the graph traces of the accepted strings. Since there are infinitely many paths for the others to try, we will leave it up to you to find a path:
`Empty String` - $[0 - \epsilon - > 3]$
`abba` - $[0 - a - > 1 - b - > 3 - \epsilon - > 0 - b - > 2 - a - > 3]$

[5 pts] (c) Convert the following Regex to NFA: (a|b)∗c

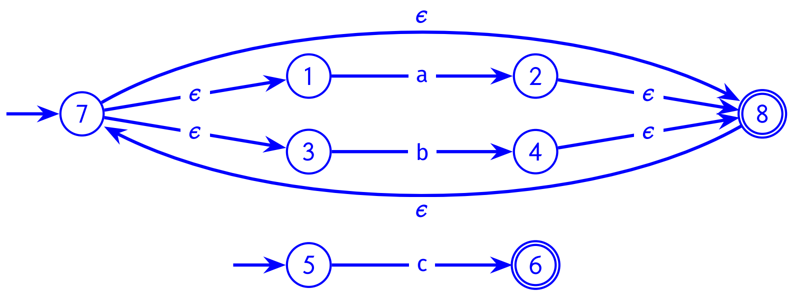
Let's build the smaller machines and then combine them to the larger machine. Let's begin by doing the "a", "b", and "c" machines.



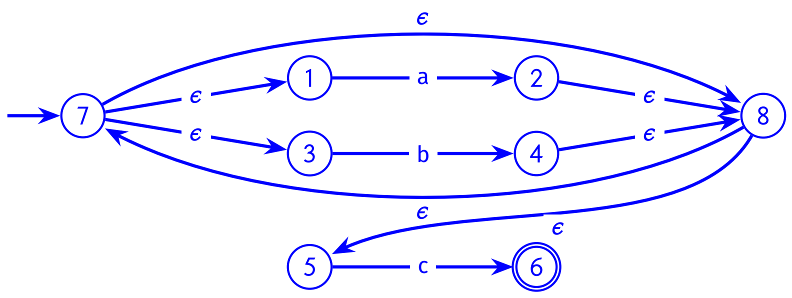
Now let's combine to get the "a|b" machine



Now let's modify to get the "(a|b)∗" machine



Now let's concatenate to get the "(a|b)∗c" machine



Optimized version (optional):