CS131 Compilers: Writing Assignment 1

Due March 13, 2022

田皓原 - 2020533013

This assignment asks you to prepare written answers to questions on regular languages, finite automata, and lexical analysis. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work and you should indicate in your submission who you worked with, if applicable. Written assignments are submitted on **Gradescope**. You should use the Latex template provided at the course web site to write your solution and use the *tikz* package to draw automata, and **all the submission that use other method to draw the graph will be deducted all the scores of the question**.

I worked with: nobody

1. ($1 \times 3 = 3$ **pts**) For each of the follow prompts, write any non-empty sentence:

   a. Name one reason why you would like to learn in this class.

   b. Write a question you would like the professor to answer 1 on any topic, from personal opinions to the class material and the teaching assistant to answer 1 in the discussion part.

   c. What do you expect from this class.

   Answer

   (a) When talking about duplicate checking of code, TA of cs100 said that you would never simiply change the names of variables and regard it as totally different from the other code, when you actually have learnt Compilers. (now i know they are all 'identifier's after lexical analysis.) I found it interesting in learning the underlying, deepening my understanding.

   (b) I don't have rudimentary knowledge base yet in computer underlying, what basic knowledge am I supposed to master, as a beginer, or is there some book or website recommended that I can learn some related simple basis by myself. Thanks!

   (c) Just learn more about compilers, how it works, how to implement, how to optimize.

2. ($1 \times 3 = 3$ **pts**) Regular expression.

   a. Strings over the alphabet $\Sigma = \{a, b\}$ that contains at most two a' s, and at least three b' s.

   b. Strings over the alphabet $\Sigma = \{a, b\}$ that do not end in a double letter(e.g. aa or bb).

   c. Strings over the alphabet $\Sigma = \{a, b, c\}$ that a will not appear adjacent.

   Answer

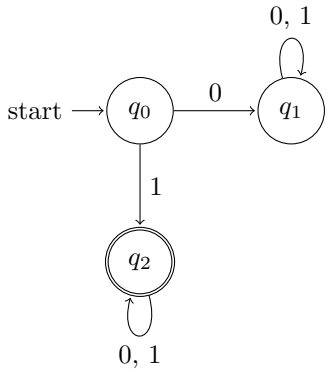
   (a) $(bbb|abbb|babb|bbab|bbbb^*a|aabbb|ababb|abbab|abbbb^*a|baabb|babab|babbb^*a|bbaab|bbabb^*a|bbbb^*ab^*a)b^*$

   (b) $(a|b)^*(ab|ba)|a|b|\epsilon$

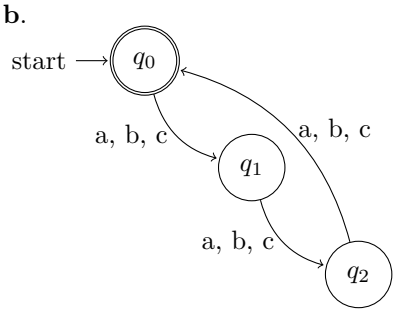
(c) $(b|c)^*(a(b|c)^+)^*a?(b|c)^*$

3. $(2 \times 3 = 6$ **pts**) Describe the strings which the following automata accept.

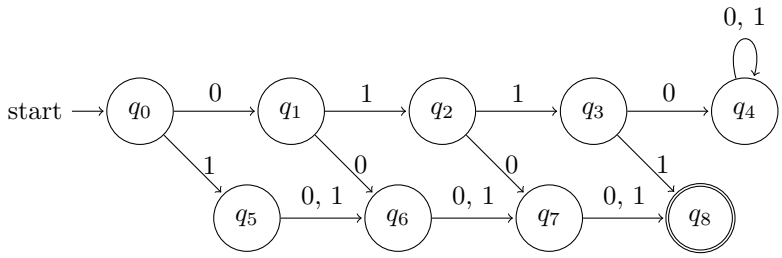
a.



b.



c.



Answer

(a) Strings over the alphabet $\Sigma = \{0, 1\}$ that start with 1.

(b) Strings over the alphabet $\Sigma = \{a, b, c\}$ whose length is a multiple of three.

(c) Strings over the alphabet $\Sigma = \{0, 1\}$ whose length is four except 0110.

4. $(2 \times 3 = 6$ **pts**) Convert the following regular expression to minimized DFA, the process is required.

a. $a(b|c)^*$

b. $((\epsilon|a)|b^*)^*$

c. $(a|b)^*abb(a|b)^*$

Answer

(a) $a(b|c)^*\#$

1 2 3 4

$followpos(1) = \{2, 3, 4\}$

$followpos(2) = \{2, 3, 4\}$

$followpos(3) = \{2, 3, 4\}$

$followpos(4) = \{\}$

$S_0 = firstpos(root) = \{1\}$

mark S_0

$a : followpos(1) = \{2, 3, 4\} = S_1,\ move(S_0, a) = S_1$

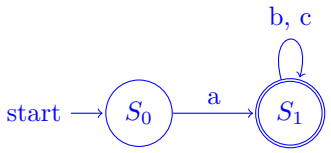
mark S_1

$b : followpos(2) = \{2, 3, 4\} = S_1,\ move(S_1, b) = S_1$

$c : followpos(3) = \{2, 3, 4\} = S_1,\ move(S_1, c) = S_1$

initial state: S_0

accepting state; S_1



(b) $((\epsilon|a)b^*)^*\#$

1 2 3

$followpos(1) = \{1, 2, 3\}$

$followpos(2) = \{1, 2, 3\}$

$followpos(3) = \{\}$

$S_0 = firstpos(root) = \{1, 2, 3\}$

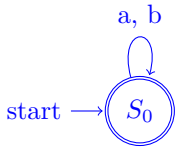
mark S_0

$a : followpos(1) = \{1, 2, 3\} = S_0,\ move(S_0, a) = S_0$

$b : followpos(2) = \{1, 2, 3\} = S_0,\ move(S_0, b) = S_0$

initial state:S_0

accepting state:S_0



(c) $(a|b)^*abb(a|b)^*\#$

1 2 345 6 7 8

$followpos(1) = \{1, 2, 3\}$

$followpos(2) = \{1, 2, 3\}$

$followpos(3) = \{4\}$

$followpos(4) = \{5\}$

$followpos(5) = \{6, 7, 8\}$

$followpos(6) = \{6, 7, 8\}$

$followpos(7) = \{6, 7, 8\}$

$followpos(8) = \{\}$

$S_0 = firstpos(root) = \{1, 2, 3\}$

mark S_0

$a : followpos(1) \cup followpos(3) = \{1, 2, 3, 4\} = S_1,\ move(S_0, a) = S_1$

$b : followpos(2) = \{1, 2, 3\} = S_0,\ move(S_0, b) = S_0$

mark S_1

$a : followpos(1) \cup followpos(3) = \{1, 2, 3, 4\} = S_1,\ move(S_1, a) = S_1$

$b : followpos(2) \cup followpos(4) = \{1, 2, 3, 5\} = S_2,\ move(S_1, b) = S_2$

mark S_2

$a : followpos(1) \cup followpos(3) = \{1, 2, 3, 4\} = S_1,\ move(S_2, a) = S_1$

$b : followpos(2) \cup followpos(5) = \{1, 2, 3, 6, 7, 8\} = S_3,\ move(S_2, b) = S_3$

mark S_3

$a : followpos(1) \cup followpos(3) \cup followpos(6) = \{1, 2, 3, 4, 6, 7, 8\} = S_4,\ move(S_3, a) = S_4$

$b : followpos(2) \cup followpos(7) = \{1, 2, 3, 6, 7, 8\} = S_3,\ move(S_3, b) = S_3$

mark S_4

$a : followpos(1) \cup followpos(3) \cup followpos(6) = \{1, 2, 3, 4, 6, 7, 8\} = S_4,\ move(S_4, a) = S_4$

$b : followpos(2) \cup followpos(4) \cup followpos(7) = \{1, 2, 3, 5, 6, 7, 8\} = S_5,\ move(S_4, b) = S_5$

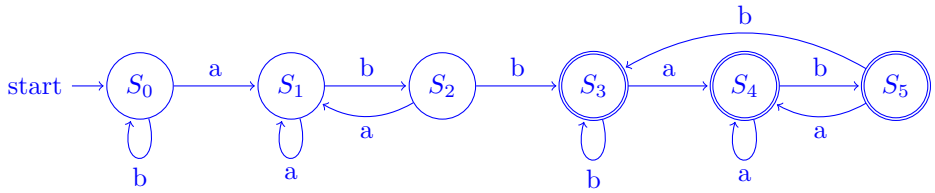
mark S_5

$a : followpos(1) \cup followpos(3) \cup followpos(6) = \{1, 2, 3, 4, 6, 7, 8\} = S_4,\ move(S_5, a) = S_4$

$b : followpos(2) \cup followpos(5) \cup followpos(7) = \{1, 2, 3, 6, 7, 8\} = S_3,\ move(S_5, b) = S_3$

initial state:S_0

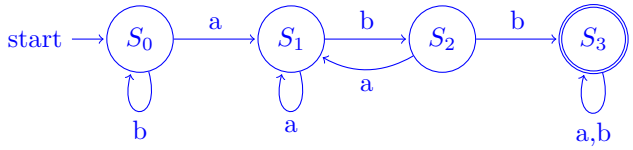
accepting state:S_3, S_4, S_5



Affere minimizing,

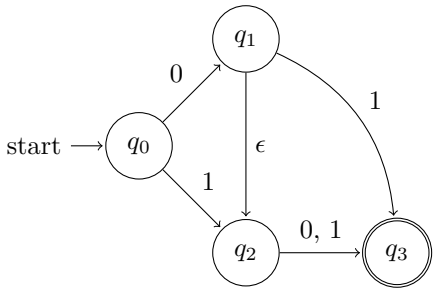
initial state:S_0

accepting state:S_3



5. ($2 \times 3 = 6$ **pts**) Convert the following NFAs to minimized DFAs, the process is not required.

a

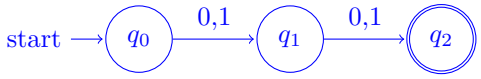


b. Question 2.a.

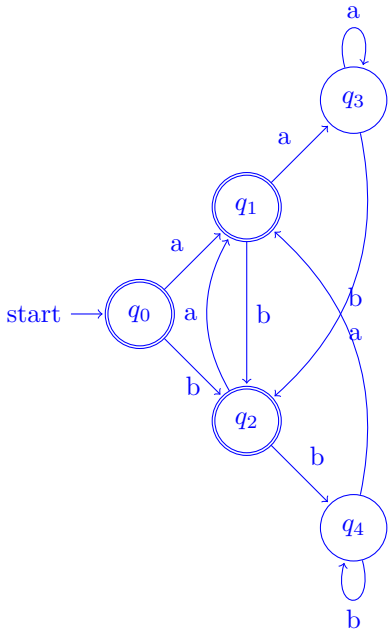
c. Question 2.c.

Answer

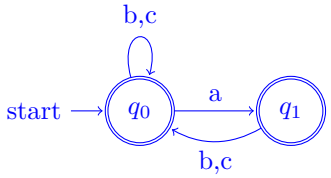
(a)



(b)



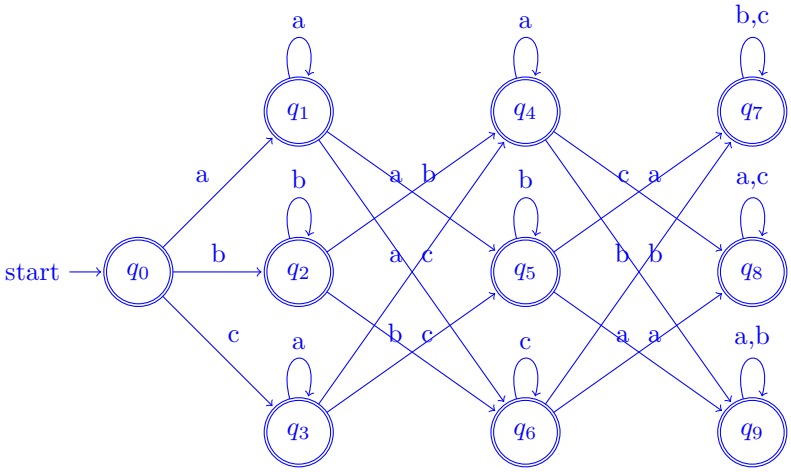
(c)



6. (10 **pts**) Let L be a language over $\Sigma = \{a, b, c\}$.
String w is in L if and only if w is of the form $w = x^n s$ where x is a or b or c, $n \geq 0$, and s is a string that does not contain x as a substring. Here, x^n denotes x being repeated n times. You can imagine w as a string with a head consisting of a character repeated 0 or more times and said character does not appears in the tail of w.
Examples of strings in L: $\underline{c}abab$, $\underline{bbb}$, $\underline{bb}$aaac.
Examples of strings not in L: $\underline{b}$aaa$\underline{b}$c, $\underline{c}$ab$\underline{c}$ab.

Draw an NFA for L. Your solution should not have more than **10** states.

Answer



7. (10 **pts**) Draw the NFA for the set of all strings over the alphabet $\Sigma = \{a, b\}$, where both a and b occur even times.

   Examples of strings that should be accepted by this NFA: abbabbbbaa, baabaaabaaaaab.

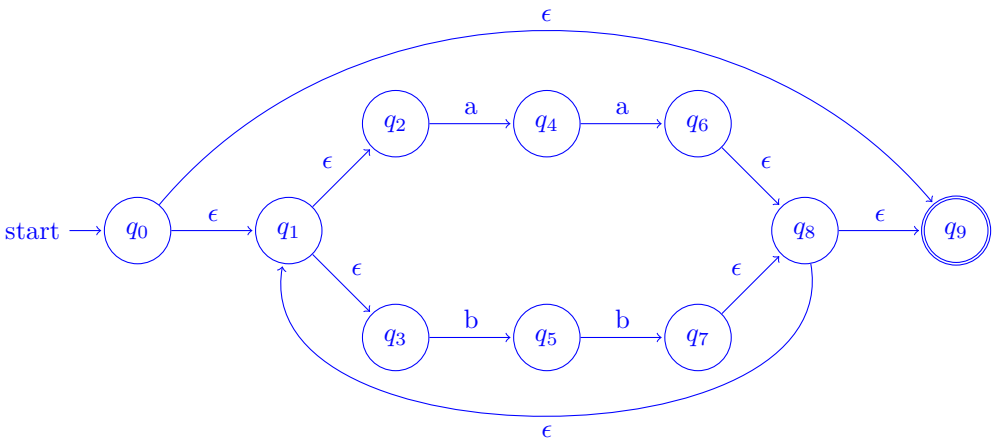
   Examples of strings that should **not** be accepted: ababb, abbbabba.

   Answer

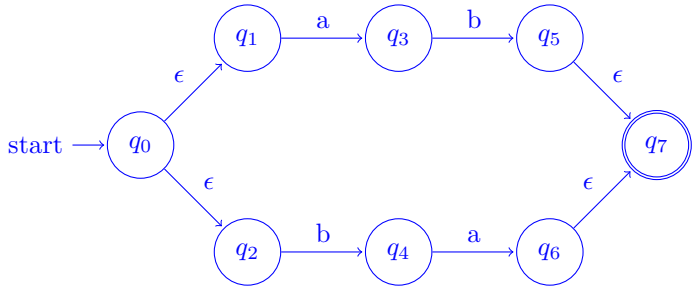
   The regular expression should be $(aa|bb)^*((ab|ba)(aa|bb)^*(ab|ba)(aa|bb)^*)^*$

   Since the NFA can be so big, I would like to show it with some equivalent symbols.

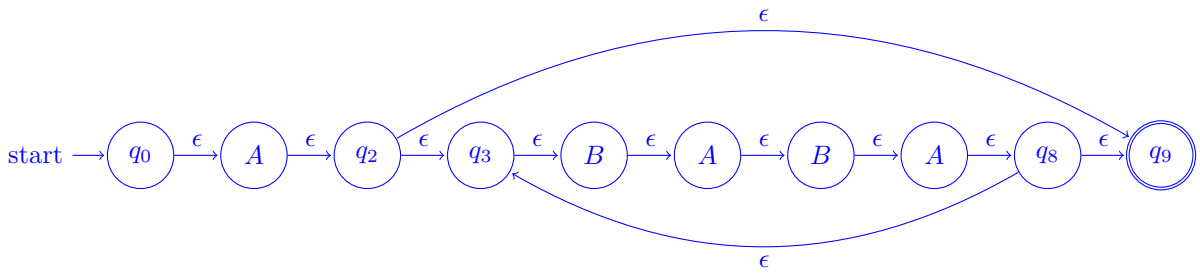
   Mark A equivalent to the following NFA (corresponding to regular expression $(aa|bb)^*$):



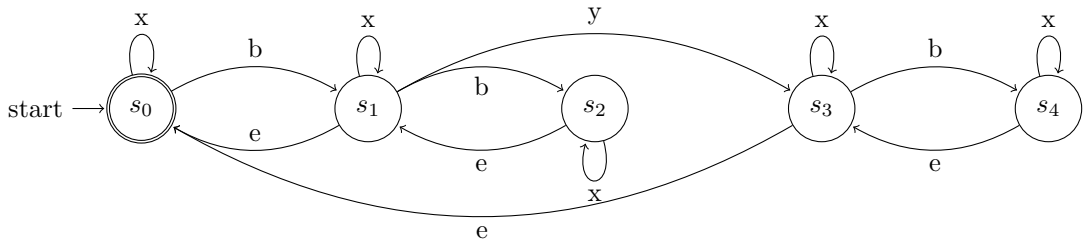
   Mark B equivalent to the following NFA (corresponding to regular expression $(ab|ba)$):



So we can get the NFA for $(aa|bb)^*((ab|ba)(aa|bb)^*(ab|ba)(aa|bb)^*)^*$ in terms of A and B:



8. (10 **pts**) Give the simplest description you can of the language described by the DFA below:



Answer

My idea:

I found that every state has a self loop with x, and all edge b go right, all edge e go left. Here I would like to declare a system that, each state has a state number, each edge b indicates $+1$ on the state numbers from one state to another, and edge e -1, edge x $+0$. By calculation, I found that this system is consistent with the rule. The state number from s_0 to s_4 are respectively $0, 1, 2, 1, 2$. y acts as a spliter of the string, and is defined as $+0$ in this system.

Then let's extract some features in this system:

(a) Since there is only one accepting state s_0 whose state number is 0, there must be equal number of edge b and e in the string.

(b) Since egde y is from s_1 to s_3, both have state number 1. So if we split the string into pieces with y, the number of b must be the number of e plus one before each y, and the number of e must be the number of b plus one after each y. Concluding that, the number of b equals 1 + the number of e before the first y, the number of e equals 1 + the number of b after the last y, and between two y's, the number of e and b are equal.

(c) Since the state number can be only $0, 1, 2$, so the number of b (denotes $|b|$) and the number of e (denotes $|e|$) in every substring from the begining should follow: $|e| \leq |b| \leq |e| + 2$.

(d) x can be anywhere in the string.

So, my final answer is:
Strings over the alphabet $\Sigma = \{x, y, b, e\}$ that,
from the begining till every character in the string, the number of b (denotes $|b|$) and the number of e (denotes $|e|$) should keep: $|e| \leq |b| \leq |e| + 2$;
$|b| = |e| + 1$ before the first y, $|e| = |b| + 1$ after the last y, $|e| = |b|$ between all y's; specially $|e| = |b|$ if there is no y;
x can be anywhere in the string.