CS5003
Final

Foundations of C.S.

Spring, 2019

PRINT NAME: ____________________

$\mathcal{SIGN}$: ____________________

Do any **6** of the following 8 problems. Cross out completely any problem you do not want graded.

There is a five point bonus for each problem you do beyond the six. If you want a problem marked for a bonus, clearly mark it as a bonus problem. There is no partial credit for a bonus problem it must be completely correct.

1. Let L be the language with definition

$$L = \{a^i b^j \mid 0 \leq i \leq j \leq 2i\}.$$

1	
2	
3	
4	
5	
6	
7	
8	
total	

a) Give a recursive definition of L.

b) Construct a context free grammar whose language is L.

a) A recursive definition is:
BASIS: $\lambda \in L$
RECURSIVE STEP: If $u \in L$ *then* aub *and* aub^2 *are in* L.
CLOSURE: A string is in L *if if can be obtained from the basis by a finite number of applications of the recursive step.*
b)

$$G : S \rightarrow aSb \mid aSb^2 \mid \lambda$$

2. Let L be the subset of $\{a, b\}^*$ consisting of those strings which contain either ab or ba as a substring.

a) Give a regular expression for L.

The string either starts with a or starts with b, and is in the set as soon as the other letter appears. $(a^+b \cup b^+a)(a \cup b)^*$.

---

b) Give a regular grammar for L.

[Hint: It helps if your regular expression in part a is very simple. If it isn't, it is better to start from scratch.]

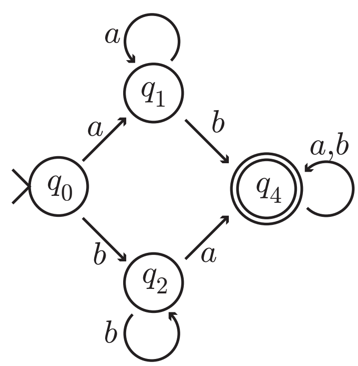
$$\begin{aligned} G : S &\rightarrow aA \mid bB \\ A &\rightarrow aA \mid bC \mid b \\ B &\rightarrow bB \mid aC \mid a \\ C &\rightarrow aC \mid bC \mid a \mid b \end{aligned}$$

Here A means prefix is all a's, needs a b, *B means* prefix is all b's, needs an a, *and C means, the prefix is in L.*

Note that since the empty string is not in the language, we do not need λ-rules.

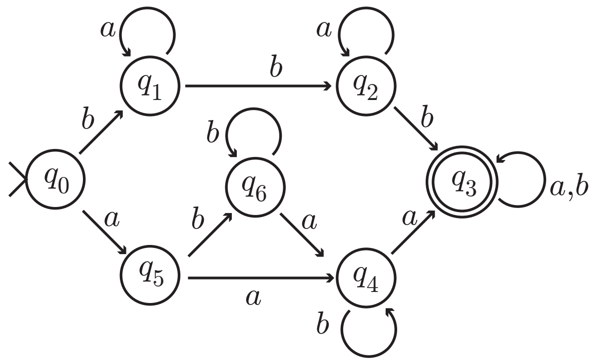
---

c) Find a Deterministic Finite Automaton which recognizes L.



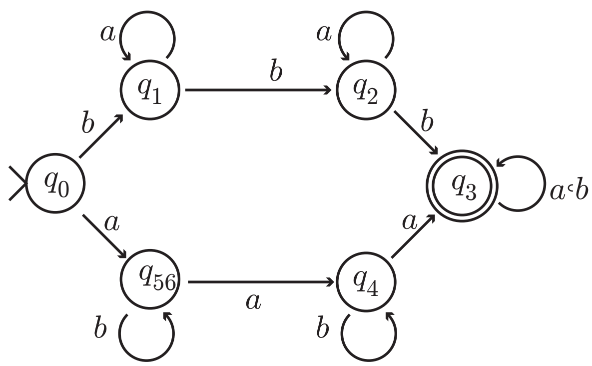
---

3. Find the minimum Deterministic Finite Automaton which is equivalent to



and show it has no indistinguishable states.

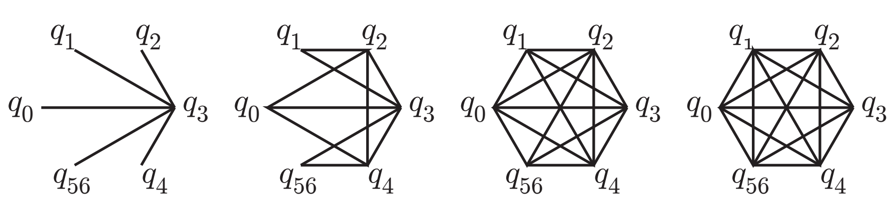
First, without applying the algorithm, states q_5 and q_6 are indistinguishable since the a and b transitions map to the exact same states. So the automaton is equivalent to:



Now we apply to the algorithm to this machine to see if these six states are all distinguishable. Initially, all states are distinguishable from the final state q_3. Next we note that b transforms only q_2 to a final state, so all non-final states are distinguishable from q_2, and the only non-final state which transforms by a to a final state is q_4, this we had all edges.

Now, applying b to q_0 and q_1 shows those states are distinguishable and applying b to q_0 and q_{56} does the same, so q_0 is distinguishable from all states.

Lastly we check q_1 and q_{56} and both a and b distinguish them. So our six state machine is the minimal machine.



---

4. Let G be the grammar given by

$$\begin{aligned} G: S &\to A \mid B \\ A &\to abA \mid \lambda \\ B &\to aBb \mid \lambda. \end{aligned}$$

a) Give a set theoretic description of $L(G)$.
$\{(ab)^j \mid j \geq 0\} \cup \{a^j b^j \mid j \geq 0\}$

---

b) Show the grammar is ambiguous.
ab has two rightmost derrivations:

$$S \Rightarrow A \Rightarrow abA \Rightarrow ab$$

and

$$S \Rightarrow B \Rightarrow aAb \Rightarrow ab$$

---

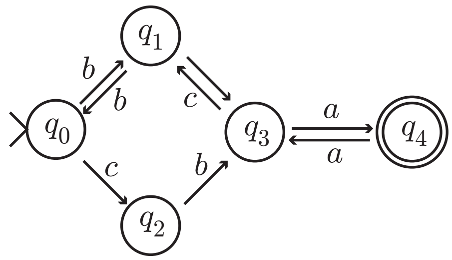
c) Show that there is no regular expression for $L(G)$.
All strings in L have $n_a = n_b$.

Let K be the number of states in an automaton M with $L_M = L$. Consider the string $a^K b^K$. The pumping lemma says that $a^K b^k$ can be factored as uvw with length(uv) $\leq K$, and the subword v a pumpable string. But uv consists only of a's, and pumping v gives a string with more a's than b's, and such a string is not in L.

So L is not regular by the Pumping Lemma.

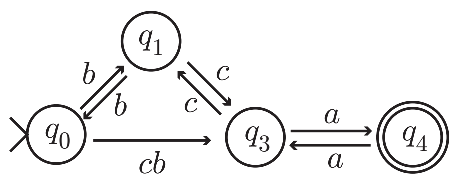
---

5. Use expression graphs to find a regular expression for the Language of the following incompletely deterministic Finite Automaton.

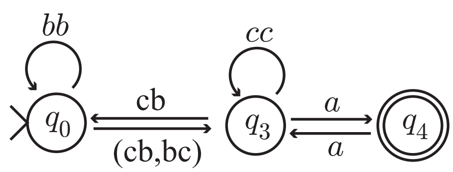


To algorithm requires us to start with an automaton with a single final state, which we have.

To delete a state, which must and labeled edges for every pair consisting of an edge into that state, and an edge out of it. The easiest one to delete is q_2.

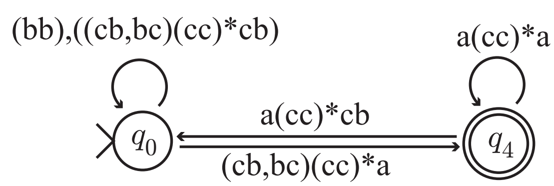


State q_1 has two incoming and two outgoing edges, so there are four edges to be added, two of which are loops.



(because of font problems, the unions are are indicated by commas in the figures.)

State q_3 also has two incoming, and two outgoing edges, but it also has a loop labeled cc, so $(cc)^*$ must be in the middle of every new edge label.



From this we can now read off the regular expression

$$[(b^2 \cup (cb \cup bc)(bb)^* cb)(cb \cup bc)(cc)^* a][(a(cc)^* a) \cup (a(cc)^* cb(b^2 \cup (cb \cup bc)(bb)^* cb)^*(cb \cup bc)(cc)^* a]^*$$

---

6. Let G be the grammar given by

$$\begin{aligned} G: S &\to AB \mid BCS \\ A &\to aA \mid C \\ B &\to bB \mid b \\ C &\to cC \mid \lambda \end{aligned}$$

a) Convert G to an equivalent essentially non-contracting grammar.

There is only one λ-rule, $Null(G) = \{A, C\}$. So

$$\begin{aligned} G'': S &\to AB \mid BCS \mid B \mid BS \\ A &\to aA \mid C \mid a \\ B &\to bB \mid b \\ C &\to cC \mid c \end{aligned}$$

And since S is not nullable, we do not add $S \to \lambda$.

---

b) If there are any chain rules in your answer to part a), covert to an equivalent essentially non-contracting grammar without chain rules.

$Chain(S) = \{S, B\}$ $Chain(A) = \{A, C\}$ $Chain(B) = \{B\}$ $Chain(C) = \{C\}$ *So*

$$\begin{aligned} G'': S &\to AB \mid BCS \mid Bb \mid b \mid BS \\ A &\to aA \mid cC \mid c \mid a \\ B &\to bB \mid b \\ C &\to cC \mid c \end{aligned}$$

---

c) Convert to Chomsky normal form.

In G'' there is only one right hand side of length greater then 2, so

$$\begin{aligned} G'': S &\to AB \mid BD \mid BJ_b \mid b \mid BS \\ A &\to J_aA \mid J_cC \mid c \mid a \\ B &\to J_bB \mid b \\ C &\to J_cC \mid c \\ D &\to CS \\ J_a &\to a \\ J_b &\to b \\ J_c &\to c \end{aligned}$$

---

7. Rewrite this rule in the form of equivalent rules with no left recursion.

$$A \to ABC \mid ACD \mid AAA \mid BC \mid DAACA$$

The right hands sides $\{ABC, ACD, AAA\}$ *are left recursive, and* $\{BC, BAACA\}$ *are "escapes".*

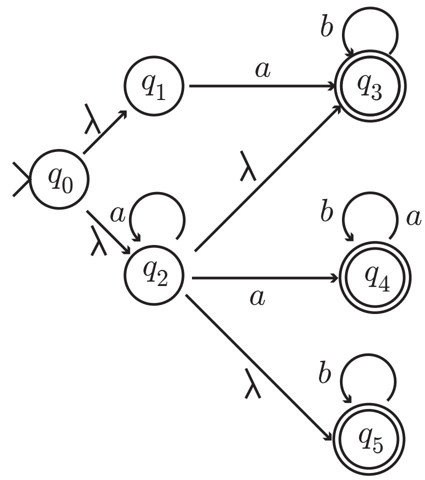
So

$$\begin{array}{rcl} A & \to & BC \mid DAACA \mid BCA' \mid DAACAA' \\ A' & \to & BC \mid CD \mid AA \mid BCA' \mid CDA' \mid AAA' \end{array}$$

has no left recursion.

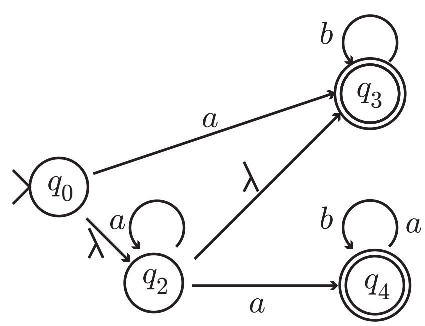
For my taste, this was the easiest problem on the exam!

8. Find an equivalent Deterministic Finite Automaton.



First note that q_1 *can be eliminated and replaced with an* a *arrow from* q_0 *to* q_3.

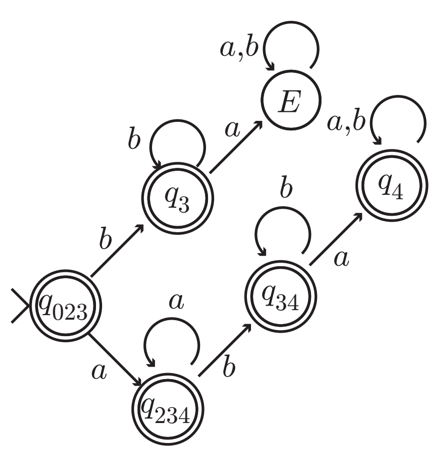
Second, note that states q_5 *and* q_3 *are indistinguishable, so the automaton is equivalent to the four state machine*



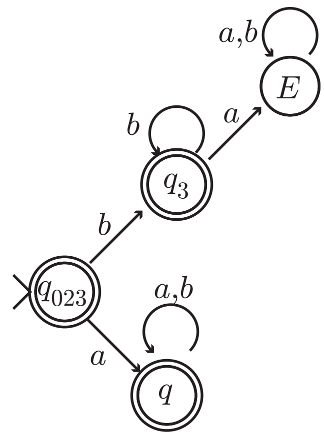
We can now more quickly compute t.

	$\lambda-$ *closure*	a	b
q_0	$\{q_0, q_2, q_3\}$	$\{q_2, q_3\}$	q_3
q_2	$\{q_2, q_3\}$	$\{q_2, q_3, q_4\}$	q_3
q_3	q_3	$\emptyset$	q_3
q_4	q_4	q_4	q_4

And we check that the last two columns are unions of λ*-closures, which they are. With* t *we can compute a deterministic machine:*



Although it was not asked, this deterministic machine is not the minimum state machine since q_{234}, q_{34} and q_4 are all indistinguishable, (any string from them lands on an accepting state.) So the machine reduces to



and we see that the language is the set of strings which are either empty, start with a, or consist only of b's.