York University
CSE 2001 Fall 2017 – Assignment 2 of 4
Instructor: Jeff Edmonds

1. **Program to DFA:**
Note in binary if $x = 101_2 = 5$ and $y = 1011_2 = 11$ then $y = 2 \cdot x + 1$.
Remember $x \ mod \ 3 = 2$ is the remainder when you divide x by 3.
Consider the following program:

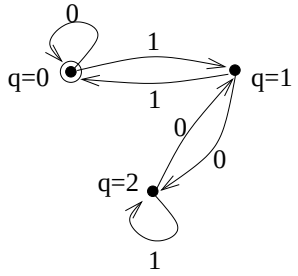
```
q = 0
loop until no more characters
    get(c)                    % c ∈ {0,1}
    q = (2 · q + c) mod 3     % q ∈ {0,1,2}
end loop
if q = 0 then
    return("accept")
else
    return("reject")
end if
```

   (a) Describe this language in one easy to understand English sentence.
   Hint: Look at examples in slides.

   - Answer: $L = \{\alpha \in \{0,1\}^* \mid x_\alpha$ is divisible by three, where x_α is the integer obtained when thinking of α in binary. $\}$.

   (b) Convert the program into a DFA.

   - Answer:



   (c) Convert the DFA into a regular expression.

   - Answer: Add an ϵ transition from a new start state to $q=0$ and another from the accept state $q=0$ to the new accept state.
   Ripping out state $q=2$ gives a self loop on state $q=1$ labeled 01^*0.
   Ripping out state $q=1$ gives a self loop on state $q=0$ labeled $0 \cup 1(01^*0)^*1$.
   Ripping out state $q=0$ gives the final answer $[0 \cup 1(01^*0)^*1]^*$.

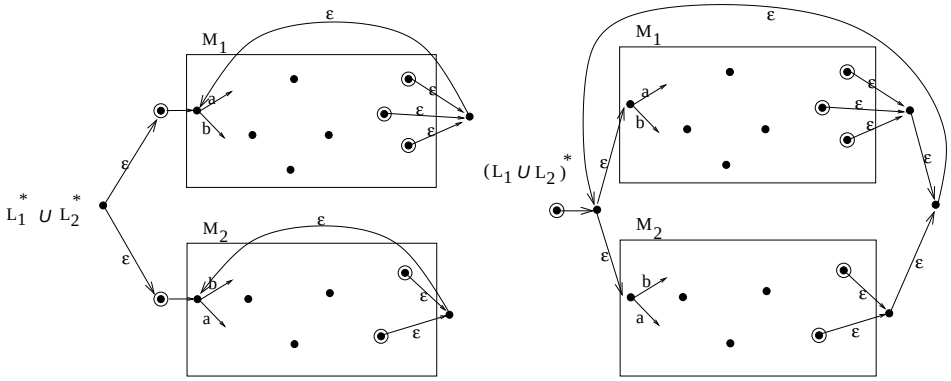
2. **NFA:**
Let L_1 and L_2 be arbitrary languages and let M_1 and M_2 below be diagrams representing the NFA for them.

   (a) Explain (as if to a 1030 student) the key differences between the languages $L_1^* \cup L_2^*$ and $(L_1 \cup L_2)^*$. Give an example of a string that is in one but not in the other and vice versa.

   - Answer in Notes: Let $L_1 = \{a\}$ and $L_1 = \{b\}$. $(L_1^* \cup L_2^*)$ contains strings that either only contain a's or only contain b's. On the other hand, $(L_1 \cup L_2)^*$ contains strings that contains only a's and b's. Let $\omega = ab$. It is in the second and but not the first. Everything in the first is in the second.

(b) Draw an NFA for the language $L_1^* \cup L_2^*$ for this generic L_1 and L_2.
(Do not do it for simply $L_1 = \{a\}$ and $L_1 = \{b\}$.)
(c) Draw an NFA for the language $(L_1 \cup L_2)^*$.

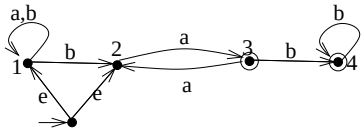
- Answer:



(d) Explain (as if to a 1030 student) the key differences between the structures of your $L_1^* \cup L_2^*$ and $(L_1 \cup L_2)^*$ NFAs and why these differences cause the difference in the languages accepted. Hint: Describe how clones can travel through the machines. Use the word "commit".

- Answer: In both machines there is a state that has one epsilon edge to the start of one M_1 and another to that of M_2. Any clone on this state must choose which machine to enter. However, in the machine for $L_1^* \cup L_2^*$ this is a bigger commitment than it is in the machine for $(L_1 \cup L_2)^*$. In the second, the clone can later switch to the other machine; however, in the first, the clone must stick forever with the machine first chosen.
  In both machines there are epsilon edges from all the accept states of M_1 and of M_2 back the beginning so that the machine can be traversed again. (In my figure, to make the figure easier, I had these edges collect together before going back. This is not necessary.) In the machine for $L_1^* \cup L_2^*$, these go back to the start state of the same M_1 or M_2 so that the same one must be traversed again. In contrast, in the machine for $(L_1 \cup L_2)^*$, these edges go back before the choice between M_1 and M_2 so that the clone again has a choice of whether to enter M_1 or M_2.
  In the L^* NFA construction, the start state must be an accept state so that the NFA accepts the empty string. However, we don't want to make the first state of M_1 an accept state because that might change the workings of this machine. To accomplish this, the construction adds a new start state.

3. Consider the following NFA.
   Here e is the empty string.



(a) Explain in words what language is accepted by this NFA.
Hint: Break the string α into non-empty blocks of a's and b's. What is the requirement on the lengths of these blocks?

- Answer: The last block could be of a's or of b's. But to be accepted, the last non-empty block of a's (either the very last or the second last) must have odd length.

(b) Formally prove that this NFA computes your stated language as follows:
We want to prove that $\forall \alpha \in \{a, b\}^*,\ M(\alpha) = L(\alpha)$.

Proof: Let α be an arbitrary string in $\{a, b\}^*$.

$$\begin{array}{lll} L(\alpha) = yes & \text{iff} & \alpha \in L \\ & \text{iff} & \alpha \text{ has stated property} \\ & \text{iff} & \alpha \text{ has decomposition property} \\ & \Rightarrow_1 \Leftarrow_2 & \exists \text{ constructed path labeled } \alpha \\ & \Rightarrow_3 \Leftarrow_4 & \exists \text{ accepted path labeled } \alpha \\ & \text{iff} & M(\alpha) = yes \end{array}$$

i. Given that the string $\alpha \in L$ has the property stated in the definition of L, what *decomposition property* does α have?
Hint: Decompose the string into three substrings.
What properties does each of these substrings have?

- Answer: By the def^n of L, the stated property of $\alpha \in L$ is that "The last block of a's in α has odd length"
By def^n of "last block of a's", this ensures that α has the deconstruction property that $\alpha = \alpha_0 \cdot \alpha_a \cdot \alpha_1$ where
α_a is the last block of a's, α_0 is the part before this and α_1 is the part after.
- If α_a begins α, then $\alpha_0 = \epsilon$. Else, α_0 can be any string as long as it ends in a b, i.e. it cant end an a or it would be included in α_a.
- α_a is a block of a's of odd length.
- If α_a ends α, then $\alpha_1 = \epsilon$. Else, α_b is a string of at least one b.

ii. Define what *constructed paths* through NFA M look like.
Hint: Deconstruct the path into three stages.
What are the key land marks between these stages?

- Answer: A constructed path can be decomposed into $path = path_0 \cdot path_a \cdot path_1$, where
- $path_0$ starts at state q_0, possibly cycles multiply through state q_1, and ends at state q_2.
- $path_a$ cycles back and forth between state q_2 and q_3 some number of time. It starts in q_2 and ends in q_3.
- $path_1$ starts in q_3 and either halts there or goes on to q_4 and then cycles there until the end.

iii. Prove each of these four implications $\Rightarrow_1$, $\Leftarrow_2$. $\Rightarrow_3$, and $\Leftarrow_4$.

- Answer:

  $\Rightarrow_1$**:** Given α with the decomposition property, we can construct such a path.
  Given any substring α_0 with the stated property, it should be clear how to find a subpath $path_0$ as described that it follows. Same for α_a along $path_a$ and α_1 along $path_1$.

  $\Leftarrow_2$**:** String labeling constructed paths have the decomposition property.
  Given any subpath $path_0$ with the stated property, it should be clear that the substring α_0 labeling it has the stated properties. Same for α_a along $path_a$ and α_1 along $path_1$.

  $\Rightarrow_3$**:** Constructed path are accepting.
  By definition they start in the start state, end in state q_3 or q_4 both of which are accepting, and is connected into a path.

  $\Leftarrow_4$**:** All accepting paths are constructed.
  Looking at edges of the NFA M, it breaks into three pieces, $M = M_0 q_2 M_a q_3 M_1$ separated by the states q_2 and q_3.
  Note our path starts in M_0.
  When it leaves M_0, it must go into M_a and when it does so it has *committed* in that it can never go back to M_0.
  Similarly, when it leaves M_a, it must go into M_1 and when it does so it has *committed* in that it can never go back to M_a.

Hence, our path must have three parts $path = path_0 \cdot path_a \cdot path_1$, each going through the corresponding part of M as described in our definition of a constructed path.

(c) Do not do any long conversion. Write an extended regular expression that expresses the same language.

- Answer: $(\{a,b\}^* b \cup \epsilon)\ a(aa)^*\ b^*$

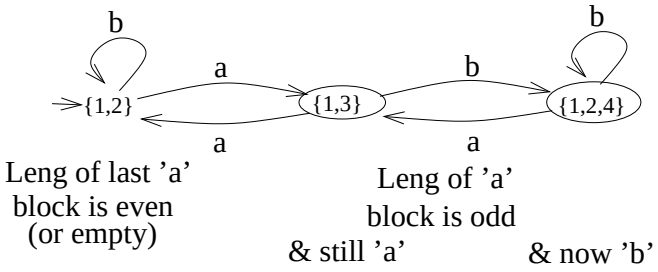
(d) Without doing the conversion, design a DFA for this language. Label the states with meaningful names.
Hint: The loop invariant states that what is remembered about the prefix read so far is:
- whether we are working on a block of a's or a block of b's.
- whether the last block of a's has even or odd length.
This implies there are four states.
You don't need to, but my DFA collapses two of these states into one.

- Answer: See answer below.

(e) Do the steps with the table to convert this NFA into a DFA.

- Answer:

	a	b
1	{1}	{1,2}
2	{3}	{}
3	{2}	{4}
4	{}	{4}



4. Do one step of converting this NFA into a regular expression by ripping out state 2.

Answer: