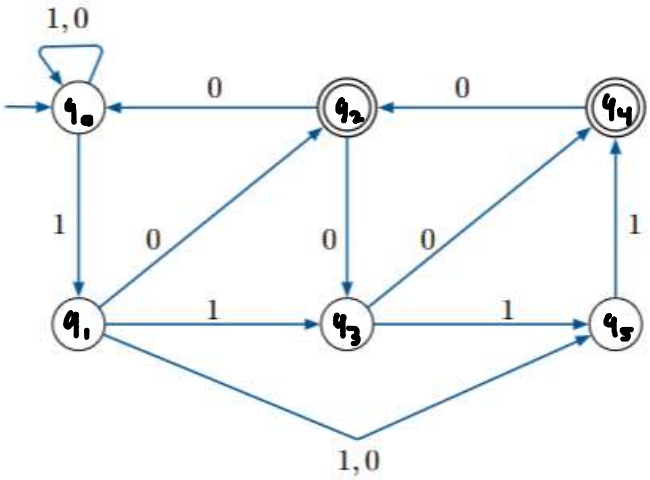
Question 1

Thursday, October 10, 2024 2:06 PM

Q1 Devise a regular expression that represents the language accepted by the following NFA. Present your steps for full credit. **[4 points]**



Let's break our NFA down into smaller paths to create our regex:

Cost of the path q_0 -> q_1 is representable via the following regex (taking into account the initial loop at q_0); path 1:
((1+0)*●1)

Now we need to consider several costs:
Cost of the path q_1 -> q_2 is 0
Cost of the path q_1 -> q_4 is 10 (for path q_1 -> q_3 -> q_4) or 111 (for path q_1 -> q_3 -> q_5 -> q_4) or (1+0)1 (for path q_1 -> q_5 -> q_4)
Cost of the path q_2 -> q_4 is 00 (for path q_2 -> q_3 -> q_4) or 011 (for path q_2 -> q_3 -> q_5 -> q_4)
Cost of the path q_4 -> q_2 is 0

Now we have to combine these costs into larger paths:
Path 2:
q_1 -> q_2●((q_2 -> q_4●q_4 -> q_2)* + q_2 -> q_4)
Represented by regex:
0●(((00 + 011)●0)* + (00+011))

Path 3:
q_1 -> q_4●((q_4 -> q_2 ● q_2 -> q_4)* + q_4 -> q_2)
Represented by regex:
(10 + 111 + ((1 + 0)●1))●((0●(00 + 011)*) + 0)

Thus, the regular express representing the language accepted by the NFA is:
Path 1 ● (Path 2 + Path 3)
= ((1+0)*●1)
●
(
0●(((00 + 011)●0)* + (00+011))

+
(10 + 111 + ((1 + 0)●1))●((0●(00 + 011)*) + 0)
)

Question 2

Monday, October 14, 2024 11:34 PM

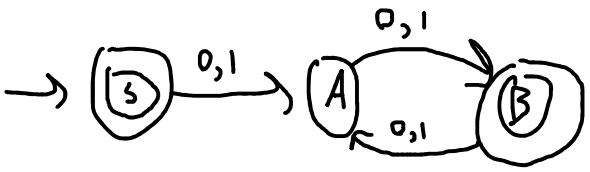
Q2 Devise a left-linear grammar for **the complement of the language** represented by the following regex. Present your reasoning/steps supporting your answer for full credit. **[4 points]**

$$\Big(\big(\big((10+\big((01)*\big)\big)*\big)\cdot(11+\lambda)\big)*+00\Big)*$$

First, let's attempt to reduce an otherwise complicated regex into a simpler form:
((((10 + ((01)*))*)●(11 + λ))* + 00)*
Notice that ((01)*) = 01*
Now we have,
((((10 + 01*)*)●(11 + λ))* + 00)*
Notice that 10 + 01* = 10 + 01
Now we have,
((10 + 01)*●(11 + λ))* + 00)*
Notice that since the union 10 + 01 cover 00, we lack the need to have an additional union with 0.
This results in a more reduced form:
((10 + 01)*●(11 + λ))*

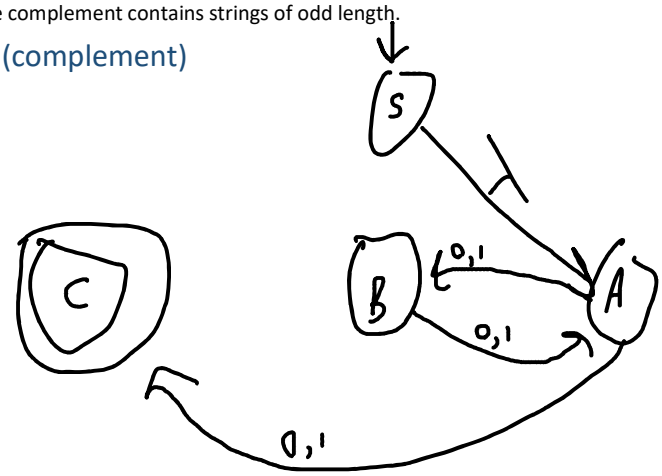
From here, we can better represent our NFA:

L



The complement contains strings of odd length.

L^R (complement)



Steps:
1. Reverse the arrows
2. Swap the roles of the final and start state

Thus, the left-linear grammar of the complement of the language is:
S -> A
A -> 0B | 1B | 0C | 1C
B -> 0A | 1A
C -> λ

Question 3

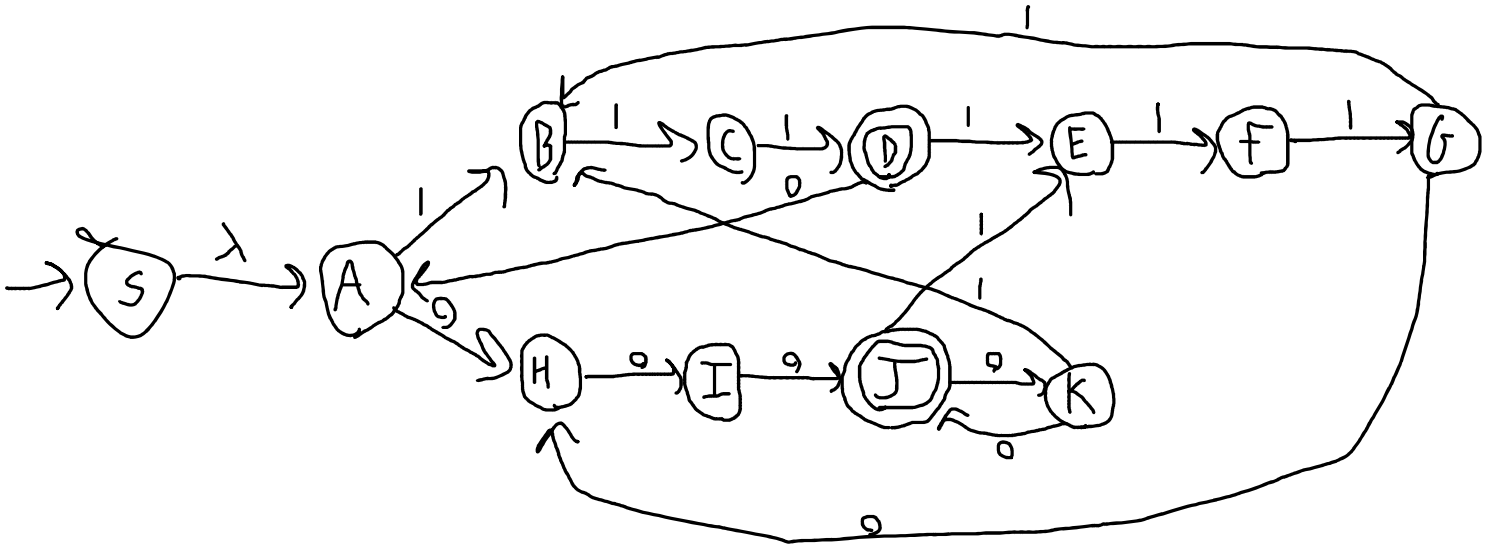
Monday, October 14, 2024 11:34 PM

Q3 Devise a right-linear grammar for $L_1 \cap L_2$, where

$$L_1 = \left\{ s \in \{0,1\}^* : \begin{array}{l} \text{Every zero run in } s \text{ is of length at least 3 and} \\ \text{every one run in } s \text{ has a length that is a multiple of 3} \end{array} \right\}$$

$$L_2 = \{00, 01, 10, 11\}^* \circ \{0,1\}$$

[4 points]



Thus, given the NFA above representing $L_1 \cap L_2$, we can devise the right-linear grammar:

S -> A
A -> 1B | 0H
B -> 1C
C -> 1D
D -> 0A | 1E | λ
E -> 1F
F -> 1G
G -> 0H | 1B
H -> 0I
I -> 0J
J -> 0K | 1E | λ
K -> 0J | 1B

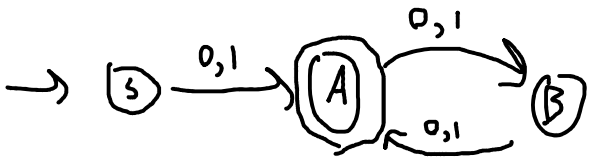
Question 4

Monday, October 14, 2024 11:35 PM

Q4 Provide a right-linear grammar for **the complement of the language** represented by the regex in Q2. Clearly state what are the variables, terminals, start variable and production rules in your grammar. [**4 points**]

Our NFA from Q2:

L



The right-linear grammar for the language:
S -> 0A | 1A
A -> 0B | 1B | λ
B -> 0A | 1A

Question 5

Monday, October 14, 2024 11:35 PM

Q5 Define a bitstring-to-bitstring function Fun via the following rules:

- $\mathsf{Fun}(s) = s$ for $s = \lambda, 0, 1$.
- $\mathsf{Fun}(00s) = (10) \circ \mathsf{Fun}(s)$ for any $s \in \{0,1\}^*$.
- $\mathsf{Fun}(10s) = (00) \circ \mathsf{Fun}(s)$ for any $s \in \{0,1\}^*$.
- $\mathsf{Fun}(01s) = (11) \circ \mathsf{Fun}(s)$ for any $s \in \{0,1\}^*$.
- $\mathsf{Fun}(11s) = (01) \circ \mathsf{Fun}(s)$ for any $s \in \{0,1\}^*$.

Establish that if L is any regular language, then $\{\mathsf{Fun}(s) : s \in L\}$ is also a regular language. **[2 points]**

The pattern is that Fun reads two bits at a time where the first bit is flipped and the second bit remains the same. Further, due to the nature of Fun, every odd position bit will be flipped for string s.

Proof (taken from the Closure Properties of Regular Languages lecture):
Since L is regular, it can be represented as an FA. Create two copies of L, an upper and a lower copy of L. Let the upper copy of L represent the transitions of the original FA of L (containing the start state) and the lower copy of L handle modified transitions such that for every odd position bit, we flip the corresponding bit and move to the lower copy and for every even position bit, we maintain transition in the upper copy of L. We should note that for the final state where the length of the string is odd, the last bit remains unchanged.

Since this upper L and lower L still behave like an FA (regular), then {Fun(s) : s ∈ L} is also a regular language.

Question 6

Monday, October 14, 2024 11:36 PM

Q6 Define a bitstring-to-bitstring function Fun via the following recursive rules:

- $\mathsf{Fun}(\lambda) = \lambda$.
- $\mathsf{Fun}(s0) = \mathsf{Fun}(s) \circ 0$ for any $s \in \{0,1\}^*$.
- $\mathsf{Fun}(s1) = 1 \circ \mathsf{Fun}(s)$ for any $s \in \{0,1\}^*$.

Does Fun preserve regularity? i.e., is $\mathsf{Fun}(L)$ regular for every regular language L?
If yes, provide your reasoning, if not, provide an L for which $\mathsf{Fun}(L)$ is not regular. **[2 points]**

If the regular language is L = { 01 }*
And the resulting language is Fun(L) = $\{1^n0^n : n \geq 0\}$

Proof by contradiction with pumping lemma, proving Fun(L) is not regular:

1. Assume Fun(L) is a regular
2. ∃ a DFA, D, (minimal) that accepts L
3. Let n = |Q| (the number of states of D)
4. $\mathbf{1^n0^n} \in \mathbf{L}$ (n is the number of states in the DFA that accepts L)
5. w = xyz = 1^n0^n, the two sub-points below are proven by pumping lemma:
   5.1 $|xy| \leq n$
   5.2 $|y| > 0$
   What this means: n number of 1's (xy) followed by n number of 0's (z)
6. Therefore y = 1^j for $j \geq 1$
7. xz ∈ L (xz is in the language) by pumping lemma
8. (The number of 1's) - (the number of 0's) is equal to j (all in xz)
9. Therefore, Fun(L) does not preserve regularity by contradiction, QED