MAU22C00: ASSIGNMENT 3 SOLUTIONS

1) (40 points) Let L be the language over the alphabet $A = \{0, 1\}$ consisting of all words containing an even number of zeroes.

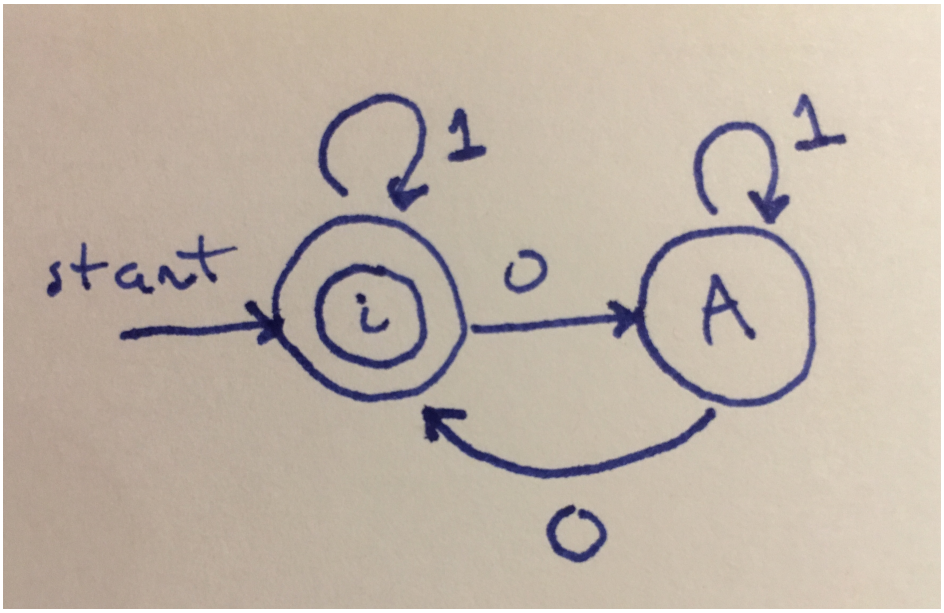
(a) Draw a finite state acceptor that accepts the language L. Carefully label all the states including the starting state and the finishing states as well as all the transitions. Make sure you justify it accepts all strings in the language L and no others.

(b) Devise a regular grammar in normal form that generates the language L. Be sure to specify the start symbol, the non-terminals, and all the production rules.

(c) Write down a regular expression that gives the language L and justify your answer.

(d) Prove from the definition of a regular language that the language L is regular.

Solution: (a) Here is the picture of our finite state acceptor:



Zero is an even number because $0 = 2 \cdot 0$. Therefore, the initial state corresponding to zero zeroes has to be an accepting state. Whichever number of 1's we receive as input does not modify the number of zeroes

in the string, so we stay in the initial state. When we receive the input 0, we move to a new state. Let us call it A. This is a non accepting state as it corresponds to 1 zero, which is an odd number of zeroes. Whichever number of 1's we receive as input does not modify the number of zeroes in the string, so we stay in this state A. If we receive the input 0, then we can return to the initial state, which is an accepting state. We will have had 2 zeroes, an even number, and however many 1's, whose number is of no concern. It turns out this is all we need to do. All strings with an even number of zeroes will take us to the initial state, which is an accepting state, and an odd number of zeroes will take us to state A, which is a non-accepting state (convince yourself by trying out strings with 3 zeroes, 4 zeroes, etc.) Please note that other correct solutions with more than two states exist, so your own solution may look different.

Marking rubric: 10 marks total, 6 marks for drawing the finite state acceptor and 4 marks for the justification.

(b) The finite state acceptor has two states $\{i, A\}$, where i is the initial state. Correspondingly, we use two non-terminals in our regular grammar: the start symbol $\langle S\rangle$ corresponding to the initial state i and $\langle A\rangle$ corresponding to state A. We first write the production rules corresponding to the transitions out of the initial state i :

(1) $\langle S\rangle \to 1\langle S\rangle$.
(2) $\langle S\rangle \to 0\langle A\rangle$.

Next, we write the production rules corresponding to the transitions out of state A :

(3) $\langle A\rangle \to 1\langle A\rangle$.
(4) $\langle A\rangle \to 0\langle S\rangle$.

Rules (1)-(4) are of type (i). For each accepting state, we will write down a rule of type (iii). Since there is only one accepting state, i, we have only one such rule:

(7) $\langle S\rangle \to \epsilon$.

Since you are not being asked for a justification, you can simply write down the production rules without any other explanation. Note that if your finite state acceptor in part (a) is more complicated, and you are translating its transitions into production rules of a regular grammar in normal form as we did in lecture, you will end up with more production rules than what is above. As long as your production rules are of a regular grammar in normal form generating the language L, you will receive full marks.

Marking rubric: 10 marks total. 2 marks for each production rule.

(c) $1^* \cup (1^* \circ 0 \circ 1^* \circ 0 \circ 1^*)^*$ is the regular expression that produces L. The 1^* part gives us all strings with no zeroes since as we remarked in part (a), zero is an even number so a string consisting only of 1's has an even number of zeroes. The other part $(1^* \circ 0 \circ 1^* \circ 0 \circ 1^*)^*$ yields all strings with an even number of zeroes. The Kleene star on the outside guarantees that whatever is inside gets repeated m times for $m \in \mathbb{N}$, where m could also be zero. Therefore, what we have inside the parentheses should be the general expression for a string with exactly two zeroes. Indeed, it is. Before the two zeroes we could have a string of 1's hence the first factor of 1^*. We note here that 1^* also yields the empty word, ϵ, corresponding to no 1's before the first zero. In between the two zeroes we could have a string of 1's, hence the second factor of 1^*. After the two zeroes we could have a string of 1's, hence the third factor of 1^*.

Marking rubric: 10 marks total. 1 mark for recognising the need for the 1^* part that gives strings with no zeroes, 5 marks for the block that represents all strings with exactly two zeroes, 1 mark for realising that one needs a Kleene star after that block, and 3 marks for the justification.

(d) Let the alphabet $A = \{0, 1\}$. Recall that the definition of a regular language allows for finite subsets of A^*, the Kleene star, concatenations, and unions. We need to define a finite sequence of languages L_j for $j = 1, \ldots, n$ such that $L_n = L$ and each L_j is a finite subset of A^*, the Kleene star of a previous L_j, a concatenation of two previous L_j's or a union of two previous L_j's. We let $L_1 = \{0\}$ and $L_2 = \{1\}$. Both are finite subsets of A^*. We let $L_3 = L_2^*$. $L_4 = L_3 \circ L_1$. $L_5 = L_4 \circ L_3$. $L_6 = L_5 \circ L_1$. $L_7 = L_6 \circ L_3$. $L_8 = L_7^*$. $L_9 = L_3 \cup L_8 = L$, and we are done. $n = 9$ in this case. Note that this process is constructing step by step the regular expression from part (c). Note also that you may have a different regular expression in part (c) that is equivalent and hence a different number of L_j's and differences in how your L_j's are defined compared to the solution here.

Marking rubric: 10 marks total. 1 mark for demonstrating understanding of the definition of a regular language and 1 mark for each L_j.

2) (20 points) Let M be the language over the alphabet $\{a, r, c\}$ given by $M = \{a^i r^j c^k \mid i, j, k \geq 0 \quad i = 2j - k\}$.

(a) Use the Pumping Lemma to show this language is not regular.
(b) Write down the production rules of a context-free grammar that generates exactly M and justify your answer.

Solution: (a) Note that $i = 2j - k \iff 2j = i + k$. Assume that this language is regular and hence it has a pumping length p. We seek to obtain a contradiction. We can choose any word w we wish in order to obtain the contradiction as long as w has length at least p. We choose $w = a^p r^p c^p$, which is in the language because $2p = p + p$. According to the Pumping Lemma, since $w \in L$ and $|w| \geq p$, there exists a decomposition of w as $w = xuy$ such that $|xu| \leq p$, $|u| > 0$, and $xu^m y \in L$ for every $m \in \mathbb{N}$. Since $w = a^p r^p c^p = xuy$ and $|xu| \leq p$, $x = a^i$ and $u = a^j$ with $j > 0$ and $i + j \leq p$. In other words, both x and u can only be composed of a's, so there is only this one case to consider. We see immediately that $xu^2y = xuuy = a^{p+j} r^p c^p$, and $2p < p + j + p$ because $j > 0$. We conclude that $xu^2y \notin L$, which is the contradiction we were seeking. Therefore, L is not a regular language.

Marking rubric: 10 marks total. Depending upon which word w you chose, there might be one case as above or several. No marks will be awarded if you worked with specific numeric values for i, j, and k – a very common mistake.

(b) We note that the case $i = j = k = 0$ corresponding to $a^0 r^0 c^0 = \epsilon \in L$. Proceeding from here, in order for $i = 2j - k \iff 2j = i + k$ to hold, if we add one r, we must add two characters that are either a or c. This gives three possibilities in total:

(i) We add 2 a's and zero c's.

(ii) We add zero a's and 2 c's.

(iii) We add one a and one c.

With these observations in mind, here is a context-free grammar that generates L :

(1) $\langle S \rangle \to \langle A \rangle \langle B \rangle$.
(2) $\langle A \rangle \to \epsilon$.
(3) $\langle B \rangle \to \epsilon$.
(4) $\langle A \rangle \to aa\langle A \rangle r$.
(5) $\langle B \rangle \to r\langle B \rangle cc$.
(6) $\langle A \rangle \to a\langle C \rangle r$.
(7) $\langle B \rangle \to \langle D \rangle c$.
(8) $\langle C \rangle \to \langle A \rangle$.
(9) $\langle D \rangle \to \langle B \rangle$.

Rules (1), (2), and (3) together generate the empty word, which is in L as explained above. (4) takes care of case (i), (5) takes care of case (ii), and rules (6), (7), (8), and (9) together take care of case (iii). Note that without using the two extra non-terminals $\langle C \rangle$ and $\langle D \rangle$, we cannot ensure that rules (6) and (7) are applied as a pair, which is

exactly what needs to be done. Rules (8) and (9) return us to the original non-terminals $\langle A \rangle$ and $\langle B \rangle$ since the next addition of one r may need to happen according to case (i) or (ii).

Marking rubric: 10 marks total. 1 mark for the correct rules that yield ϵ, 3 marks for the correct rules that address cases (i) and (ii), 3 marks for the correct rules that address case (iii) (which is harder), and 3 marks for the justification. Please note that your own solution may differ quite a bit from the one above and still be correct.