HW Solution

CS/ECE 374: Algorithms & Models of Computation, Spring 2019 Version: **1.0**

Submitted by:

- **≪Alan Lee≫: ≪alanlee2≫**
- **≪Joshua Burke≫: ≪joshuab3≫**
- **≪Gerald Kozel≫: ≪gjkozel2≫**

(100 PTS.) Draw me a giraffe.

For each of the following languages in **1** – **3**, draw an NFA that accepts them. Your automata should have a small number of states. Provide a short explanation of your solution, if needed.

1 (25 PTS.) All strings in $\{0, 1, 2\}^*$ such that at least one of the symbols 0, 1, or 2 occurs at most 4 times. (Example: 1200201220210 is in the language, since 1 occurs 3 times.)

2 (25 PTS.) $\left((01)^*(10)^* + 00\right)^* \cdot (1 + 00 + \varepsilon) \cdot (11)^*$.

3 (25 PTS.) All strings in $\{0, 1\}^*$ such that the last symbol is the same as the third last symbol. (Example: 1100101 is in the language, since the last and the third last symbol are 1.)

4 (25 PTS.) Use the power-set construction (also called subset construction) to convert your NFA from **3** to a DFA. You may omit unreachable states.

7 Solution:

A (page 1) We thought about this problem as the following: the initial state must take in any valid input of 0, 1, or 2. After this input, an epsilon transition will choose which path to to take: 4 0's, 4 1's, or 4 2's. Within each path, the only input that will transition it to the next sequence is if it is the path's designated number. All other inputs will simply keep it at the current state.

B (page 2) We thought about the problem as separate machines first. the first part $\left((01)^*(10)^*+00\right)^*$ and $(1 + 00 + \varepsilon) \cdot (11)^*$. By doing this, we had the first part of our NFA as a start state with a choice of either $\left((01)^*(10)^*, 00\right)^*$ or the empty set which would move it on to the next part of the concatenation. The next part had 3 choices:$(1 + 00 + \varepsilon)$ by choosing one of them, we linked it with an epsilon transition which signified the choice of path. using the previous example, we then built a similar NFA to concatenate onto our existing larger NFA for the last part $(11)^*$.

C (page 3) We thought about this problem as two parts. The first part of the string can be made up of any number of 0's or 1's, then when we are in the last 3 characters. Either it has a 0...0 or a 1...1 ending. Then after we see that, we can move onto the accept state.

D (page 4 and 5) To approach this problem, we took our previous NFA from 3 and went through the table method of all possibilities for all inputs then concatenated the states into superstates. Once we completed the table where all result states led to existing states we found, we knew we completed the table. From this we constructed our new DFA