COMP30026 Models of Computation Assignment 2

Emmanuel Macario - 831659

October, 2018

Challenge 4

a. Let D be a DFA $(Q, \Sigma, \delta, q_0, F)$ that recognises regular language, R. We can transform D into an NFA N that recognises $drop(\mathsf{a}, R)$ by replacing every transition involving the symbol a with an epsilon transition.

   More formally, we can define N to be the five-tuple $N = (Q, \Sigma_\epsilon \setminus \{\mathsf{a}\}, \delta', q_0, F)$ with transition function,

$$\delta'(q, x) = \begin{cases} \{\delta(q, x)\} & \text{if } q \in Q \text{ and } x \in \Sigma \setminus \{\mathsf{a}\} \\ \{\delta(q, \mathsf{a})\} & \text{if } q \in Q \text{ and } x = \epsilon \end{cases}$$

   Hence, since $drop(\mathsf{a}, R)$ can be recognised by NFA N, then $drop(\mathsf{a}, R)$ is also a regular language.

b. $L = \{a^m b^n c^n \mid m, n \geq 0\}$

Challenge 5

Let A be an arbitrary regular language. Since A is regular, there must exist some DFA D that acts as a recogniser for A.

We can systematically translate DFA $D = (Q, \Sigma, \delta, q_0, F)$ into a PDA $P = (Q', \Sigma', \Gamma, \delta', q_0', F')$ with only three states, that recognises A.

The general premise of this translation is to let the PDA use its stack to keep track of the current state in the DFA. Additionally, we will only allow the PDA to accept an input string if the state on top of the stack is an original accept state in the DFA.

Hence, we can formally define P to be the six-tuple $P = (\{q_I, q_R, q_A\}, \Sigma, Q, \delta', q_I, \{q_A\}\}$ with transition function,

$$\delta'(q, x, s) = \begin{cases} \{(q_R, q_0\$)\} & \text{if } q = q_A,\ x = \epsilon \text{ and } s = \epsilon \\ \{(q_R, q_n)\} & \text{if } q = q_R,\ x \in \Sigma \text{ and } \delta(s, x) = q_n \\ \{(q_A, \epsilon)\} & \text{if } q = q_R,\ x = \epsilon \text{ and } s \in F \end{cases}$$

INSERT IMAGE HERE

Challenge 6

We will show that $L(G) = A$ in two parts. Firstly, we will show $L(G) \subseteq A$ by using structural induction on the form of G. Then, we will show $A \subseteq L(G)$ by induction on the length of strings in A.

Proof: $L(G) \subseteq A$

Let $w \in L(G)$ such that $S \Rightarrow^* w$. That is, w an arbitrary string in $L(G)$ that is derived from start symbol, S. We will use structural induction to show that $w \in A$.

Consider base case $w = \epsilon$. Clearly, w does not contain the substring 001.

For the first recursive case, let $w = w'0$ where $w' \in L(G)$. By the inductive hypothesis, w' does not contain the substring 001. Hence, in this case w cannot contain substring 001. This is because any new substrings generated will have 0 as the end symbol, whereas 001 has 1 as its end symbol.

For the second recursive case, let $w = 1w'$ where $w' \in L(G)$. By the inductive hypothesis, w' does not contain the substring 001. Hence, in this case w cannot contain the substring 001. This is because any new substrings generated will have 1 as the start symbol, whereas 001 has 0 as its start symbol.

For the final recursive case, let $w = 01w'$ where $w' \in L(G)$. By the inductive hypothesis, w' does not contain the substring 001. Hence, in this case w cannot contain the substring 001. This is because any newly generated substrings of length three will be of the form $01x$ or $1xx$, where $x \in \Sigma_\epsilon$. Clearly, none of these substrings are equal to 001.

Hence, in no case does w contain substring 001. Therefore, $L(G) \subseteq A$.