Assignment #4

Gustavo Estrela de Matos
CSCE 433: Formal Languages and Automata

March 28, 2016

Question 1.

Question 2.

(a)

Describing each state with a regular expression with strings that take us to an acceptance state we can build the system:

$$\begin{cases} q_0 = \epsilon q_1 + b q_2 \\ q_1 = a q_1 + \epsilon q_2 + \epsilon \\ q_2 = \epsilon \end{cases}$$

- Substitute q_2 into the second equation

$$\begin{aligned} q_1 &= a q_1 + \epsilon + \epsilon \\ &= a^* \text{ (by Arden's Lemma)} \end{aligned}$$

- Substitute q_1 into the first equation

$$\begin{aligned} q_0 &= \epsilon a^* + b\epsilon \\ &= a^* + b \end{aligned}$$

Since q_0 is the start state of the NFA, we have that the regular expression that describes this NFA is $a^* + b$.

(b)

Describing each state with a regular expression with strings that take us to an acceptance state we can build the system:

$$\begin{cases} q_0 = 0q_1 + 1q_2 + \epsilon \\ q_1 = 0q_0 + 1q_3 \\ q_2 = 0q_3 + 1q_0 \\ q_3 = 0q_2 + 1q_1 \end{cases}$$

- Substitute q_1 and q_2 into the last equation

$$\begin{aligned} q_3 &= 0(0q_3 + 1q_0) + 1(0q_0 + 1q_3) \\ &= 00q_3 + 01q_0 + 10q_0 + 11q_3 \\ &= (00 + 11)q_3 + (01 + 10)q_0 \\ &= (00 + 11)^*(01 + 10)q_0 \text{ (by Arden's Lemma)} \end{aligned}$$

- Substitute q_3 into the second and third equation

$$\begin{aligned} q_1 &= 0q_0 + 1((00 + 11)^*(01 + 10)q_0) \\ &= 0q_0 + 1(00 + 11)^*(01 + 10)q_0 \\ &= (1(00 + 11)^*(01 + 10) + 0)q_0 \end{aligned}$$

$$\begin{aligned} q_2 &= 1q_0 + 0((00 + 11)^*(01 + 10)q_0) \\ &= 1q_0 + 0(00 + 11)^*(01 + 10)q_0 \\ &= (0(00 + 11)^*(01 + 10) + 1)q_0 \end{aligned}$$

- Substitute q_1 and q_2 into the first equation

$$\begin{aligned} q_0 &= 0((1(00 + 11)^*(01 + 10) + 0)q_0) + 1((0(00 + 11)^*(01 + 10) + 1)q_0) \\ &= (01(00 + 11)^*(01 + 10) + 00)q_0 + (10(00 + 11)^*(01 + 10) + 11)q_0 \\ &= (01(00 + 11)^*(01 + 10) + 00 + 10(00 + 11)^*(01 + 10) + 11)q_0 \\ &= ((01 + 10)(00 + 11)^*(01 + 10) + 00 + 11)q_0 \\ &= ((01 + 10)(00 + 11)^*(01 + 10) + 00 + 11)^* \text{ (by Arden's Lemma)} \end{aligned}$$

Since q_0 is the start state of the NFA, we have that the regular expression that describes this NFA is $((01 + 10)(00 + 11)^*(01 + 10) + 00 + 11)^*$.

Question 3.

(a) $G = (V, \Sigma, R, S)$ where:

- $V = \{S\}$

- $\Sigma = \{a, b\}$

Note that $\epsilon \in L_1$, therefore our first rule is $S \to \epsilon$. To build other strings we have to add 1's into the end and twice this number of 0's at the beggining of the string, then the second rule is $S \to 00S1$.

- $R = \{S \to \epsilon,\ S \to 00S1\}$

(b) $G = (V, \Sigma, R, S)$ where:

- $V = \{S\}$
- $\Sigma = \{a, b\}$

Note that $\epsilon \in L_1$, therefore our first rule is $S \to \epsilon$. To build other strings of this language we have to guarantee that $m \geq n$ and keep m and n both odd or even. Which means that:

1 $S \to aSb$; whenever adding an a you have to add a new b to keep $m \geq n$ and $m - n$ even.

2 $S \to Sbb$; whenever adding a b we need to add two of them to keep $m - n$ even.

Therefore the rules for this language are:

- $R = \{S \to \epsilon,\ S \to Sbb,\ S \to aSb\}$

(c) $G = (V, \Sigma, R, S)$ where:

- $V = \{S, S_1, R_1, T_1, S_2, R_2, T_2\}$
- $\Sigma = \{a, b, c, d\}$

Note that the equation $m + n = p + q$ means that for every a or b added we need to add a new c or d. To do that we are going to build the string from the outside to inside adding for every a or b a new d or a new c. You should also note that it's hard have the pair of rules "for every a add a new c" and "for every b add a new d" because they intersect each other; to deal with this we are going to create rules for two different situations: $m \geq q$ and $m \leq q$.

In the case where $m \geq q$ we are looking for the language $L_1 = \{a^m b^n c^p d^q \mid m + n = p + q, m \geq q\}$ generate the strings with the following rules:

1 $S_1 \to aS_1d | R_1$

2 $R_1 \to aR_1c | T_1$

3 $T_1 \to bT_1c | \epsilon$

We do not need a rule that adds a new d when adding a b because we assumed that $m \geq q$ therefore every d of a string would already have been added by rule 1.

In the case where $m \leq q$ we are looking for the language $L_2 = \{a^m b^n c^p d^q \mid m + n = p + q, m \leq q\}$ generate the strings with the following rules:

4 $S_2 \to aS_2d | R_2$

5 $R_2 \to bR_2d | T_2$

6 $T_2 \to bT_2c | \epsilon$

Simmilarly we do not need a rule that adds a new c when adding a a because we assumed that $m \leq q$ therefore every a of a string would already have been added by rule 4. Now the last rule we need is the one that makes the union of L_1 and L_2:

7 $S \rightarrow S_1|S_2$

- $R = \{S \rightarrow S_1|S_2,\ S_1 \rightarrow aS_1d|R_1,\ R_1 \rightarrow aR_1c|T_1,\ T_1 \rightarrow bT_1c|\epsilon,\ S_2 \rightarrow aS_2d|R_2,\ R_2 \rightarrow bR_2d|T_2,\ T_2 \rightarrow bT_2c|\epsilon\}$

Question 4.

$G = (V, \Sigma, R, S)$ where:

- $V = \{S, R\}$
- $\Sigma = \{a, b, *, (,), +, \varepsilon\}$; to avoid ambiguity we are going to use the symbol ε to represent the empty string in the regular expressions we want to build.

To create all the regular expressions we are going to use two different variables: R and S. The variable S represents any non-empty regular expression while R can be either S or the empty string ϵ.

1 $S \rightarrow aR$
2 $S \rightarrow bR$
3 $S \rightarrow \varepsilon R$
4 $S \rightarrow S^*R$
5 $S \rightarrow (S)R$
6 $S \rightarrow S + S$
7 $R \rightarrow S$
8 $R \rightarrow \epsilon$

- $R = \{S \rightarrow aR, S \rightarrow bR, S \rightarrow \varepsilon R, S \rightarrow S^*R, S \rightarrow (S)R, S \rightarrow S + S, R \rightarrow S, R \rightarrow \epsilon\}$