CS 536 Fall 2008 Homework 1

Priyananda Shenoy (shenoy@cs.wisc.edu)

September 28, 2008

1.1 DFSM to accept strings ending with 10.

1.2 DFSM to accept strings with penultimate character as 1.

1.3 DFSM to accept strings in which there are at least two 1's and at least one 0.

2.1 NDFSM to accept strings ending with 10.

2.2 NDFSM to accept strings with penultimate character as 1.

3.1 RE for strings whose length is a multiple of 3.
Expression: $((0|1)(0|1)(0|1))^+$
Explanation: A string in the language can always be broken up into groups of three characters, which each character being 0 or 1. It is assumed here that the null string is not in the language. If the null string needs to be part of the language then we need to change the outer $+$ to $*$.

3.2 RE for strings that contain 000 as a substring.
Expression: $(0|1)^*000(0|1)^*$
Explanation: A string in the language has to have 000 somewhere in the string, preceded and succeeded by any number of any characters.

3.3 RE for strings that contain exactly three 0's
Expression: $1^*01^*01^*01^*$
Explanation: A valid string in the language will have three zeros with any number of 1s between, before and after the 0s.

4. The given RE is $(0^*)|(0^*10^*)|(0^*1(00^+|1)^*10^*)$.

4.1 The string **00100** is **accepted** by the RE. The second part of the RE is (0^*10^*), which accepts zero or more 0s followed by a single 1 followed by zero or more 0s, which matches our given string.

4.2 The string **001100** is **accepted** by the RE. The third part of the RE is $(0^*1(00^+|1)^*10^*)$. Taking the middle part of this: $(00^+|1)^*$ to be empty, we get (0^*110^*), which accepts two 1s surrounded by any number of 0s, which matches our string.

4.3 The string **01010** is **not accepted** by the RE. The first part of the RE: (0^*) obviously doesn't match the string. The second part matches strings with one a single 1, so that doesn't match this string as well. The third part: $(0^*1(00^+|1)^*10^*)$ matches two 1s only if they are contiguous or else they are separated by two or more 0s. In our string the 1s are separated by one 0, so the RE doesn't match our string.

5.

1. *The assertion $RS = SR$ is* ***false*** *for arbitrary regular expressions R and S.*

Proof. We will disprove this assertion by counter-example. Let $\Sigma = \{a, b\}$ and $R = a, S = b$ be trivial regular expressions, then $L(R) = \{a\}$ and $L(S) = \{b\}$. Consider,

$$\begin{aligned} L(RS) &= \{x \cdot y | x \in L(R), y \in L(S)\} \\ &= \{ab\} \\ L(SR) &= \{x \cdot y | x \in L(S), y \in L(R)\} \\ &= \{ba\} \\ L(RS) &\neq L(SR) \end{aligned}$$

□

2. *The assertion $(S|R)^* = R^*|S^*$ is **false** for arbitrary regular expressions R and S.*

Proof. Let R, S be the same regular expressions as in the previous proof. Consider the string $s = aabbaa$. Clearly $s \in L((S|R)^*)$, since each character in the string is either in R or in S. But $s \notin L((R^*|S^*))$, since the RE cannot captures strings which end with strings in R. Since there is atleast one string which is there in one language and not in the other, we can say that the REs are not equal. □

3. *The assertion $(R^*S^*)^* = (R|S)^*$ is **true** for arbitrary regular expressions R and S.*

Proof. Intuition: Any string in $L((R^*S^*)^*)$ can be broken up into zero or more parts where each part is matched by (R^*S^*). Each part can be further split into two where each part is matched by R^* and S^* respectively. In general, each part can be matched by $(R|S)^*$. Hence very string in $L((R^*S^*)^*)$ is also in $L((R|S)^*)$. Similiarly we can show that every string in $L((R|S)^*)$ is also in $L((R^*S^*)^*)$, hence they are identical.

Let R, S be arbitrary REs over some alphabet Σ.We will show that $L((R^*S^*)^*) \subseteq L((S|R)^*)$ and $L((S|R)^*) \subseteq L((R^*S^*)^*)$, and hence the languages are equal.

Let $x \in L((R^*S^*)^*)$. Then by definition,

$$\exists k \geq 0 \text{ such that } x = x_1x_2 \cdots x_k \text{ where each } x_i \in L(R^*S^*).$$

Furthermore, we can break each x_i into $r_{i1} \cdots r_{ip}s_{i1} \cdots s_{iq}$ such that each $r_{ij} \in L(R)$ and $s_{ij} \in L(S)$. Now every $r_{ij}, s_{ij} \in (L(R) \cup L(S)) = L(R|S)$, therefore we can write x as,

$$\exists k' \geq 0 \text{ such that } x = x'_1x'_2 \cdots x'_{k'} \text{ where each } x'_i \in L(R|S).$$

Therefore by definition, $x \in L((R|S)^*)$. Since '|' is commutative, $x \in L((S|R)^*)$. Since this is true for every string in $L((R^*S^*)^*)$, $L((R^*S^*)^*) \subseteq L((S|R)^*)$.

Let $x \in L((S|R)^*)$. Then by definition,

$$\exists k \geq 0 \text{ such that } x = x_1x_2 \cdots x_k \text{ where each } x_i \in L((S|R)).$$

We repeatedly apply the following procedure to the RHS of the expression, without changing the string.

1. If there is a sequence of strings $x_p \cdots x_q$ such that $\forall i \in [p, q], x_i \in L(R)$, replace it with a string $r_p \in L(R^*)$.
2. Similiary group all adjacent strings of $L(S)$.
3. If the first group is not in $L(R^*)$, append a null string at the beginning. Similarly if the string doesn't end with a string in $L(S^*)$, append a null string at the end.
4. Replace each pair of adjacent groups $(r_i s_i)$ by y_i where $y_i \in L(R^* S^*)$.

This procedure clearly yields a string which is in $L((R^* S^*)^*)$. Since this procedure doesn't change the string and will terminate in finite steps, $x \in L((R^* S^*)^*)$. Hence $L((S|R)^*) \subseteq L((R^* S^*)^*)$. $\square$