CSE355 Spring 2013 – Homework 2 Sample Solutions

1) These regular expressions were produced by conversion from the DFAs for homework 1 using the procedure in the textbook. Other regular expressions for the same languages are possible.

1a) \((10000* \cup 0(1000* \cup 0(100* \cup 00*(\varepsilon \cup 10*)))) \cup 0(1(1000* \cup 0(100* \cup 00*(\varepsilon \cup 10*)))) \cup 00*(\varepsilon \cup 10*(\varepsilon \cup 10*)) \cup 1(100* \cup 00*(\varepsilon \cup 10*)))\)

1b) \((111*00 \cup 0 \cup 100)* 101 ((11*00 \cup 0)*1)* (\varepsilon \cup 11* \cup 11*0 \cup (11*00 \cup 0)*0*)\)

1c) \(\varepsilon \cup 1 \cup 11 \cup (0 \cup 10 \cup 110)(10 \cup 110)*(\varepsilon \cup 1 \cup 11)\)

1d) \(1*00*1(0 \cup 1)*\)

1e) \((0(11)*0 \cup (1 \cup 0(11)*10)(00 \cup 01(11)*10)*(1 \cup 01(11)*0))*\)

2) These NFAs (beginning on the next page) have been produced from the regular expressions in question 1, with minimal simplification performed for readability. The primary simplification performed is the reduction of \(\varepsilon\)-transitions. Other NFAs are possible based on level of simplification or differences in the regular expression used as a basis.
2a) \( (1000^* \cup 0(100^* \cup 0(100^* \cup 00^*(0 \cup 10^*)))) \cup 0(1(1000^* \cup 0(100^* \cup 00^*(0 \cup 10^*)))) \cup 0(00^*(0 \cup 10^*(0 \cup 10^*)) \cup 1(100^* \cup 00^*(0 \cup 10^*)))) \)

2b) \( ((11^00 + 0 + 100)^* 101((11^00 + 0)^* 0)^* (0 + 11^* + 11^0 + (11^00 + 0)^* 0)^*) \)

Somewhat simplified
2c) \( \epsilon\{\text{UIUUI}\} \{\text{UIOUIIIO}\} \{\text{II0VII0}\} \{\text{II10UII1}\} \)
(Somewhat Simplified)

2d) \( I^*00^*1(0V1)^* \)
(Somewhat Simplified)

2e) \( (0(11)^*0 \cup (1 \cup 0(11)^*10)(00 \cup 01(11)^*10)^* (1+01(11)^*0))^* \)
We can show that our procedure outlined above correctly creates a regular expression for \( L \). We can construct an NFA that mimics the derivations of this grammar.

Let \( G = (V, \Sigma, R, V_0) \). Construct \( N = (Q, \Sigma, \delta, V_0, \{V_R\}) \) with \( Q = V \cup \{V_R\} \). Define \( \delta \) as follows:

- For rules in \( G \) of the form \( V_i \rightarrow a_iV_{i+1} \), \( a_i \in \Sigma \), define \( \delta(V_i, a_i) = V_{i+1} \).
- For rules in \( G \) of the form \( V_i \rightarrow a \), \( a \in \Sigma \), define \( \delta(V_i, a) = V_F \).

What does this look like? Suppose we have rules \( V_0 \rightarrow a_0V_1, V_1 \rightarrow a_1V_2, \ldots, V_n \rightarrow a_n \). Then the derivation produces \( V_0 \rightarrow a_0a_1\ldots a_n \). Similarly, since \( \delta^*(V_0, a_0\ldots a_n) = V_F \), then there is a walk in \( N \) such that \( N \) begins in \( V_0 \), reads \( a_0\ldots a_n \), and ends in \( V_F \). Since there is an NFA for \( L \), the language generated by some regular grammar, \( L \) must be regular.

If a language, \( L' \), is regular, then there is an NFA, \( N' = (Q, \Sigma, \delta, q_0, F) \), that recognizes it. We can construct from \( N' \) a regular grammar \( G' = (V', \Sigma, R', S') \) that generates \( L' \), as follows:

- Let \( S' = q_0 \).
- For \( \delta'(q_i, a) = q_j, a \in \Sigma \), add the rule \( q_i \rightarrow aq_j \) to \( R' \), and add \( q_0, q_j \) to \( V' \).
- For \( q_k \in F \), add the rule \( q_k \rightarrow \varepsilon \) to \( R' \).

What does this do? If a string, \( w = a_0\ldots a_k \), is in \( L' \), then there is a walk from \( q_0 \ldots q_k \), \( q_k \in F \), through \( N' \) by reading \( w \). Then \( G' \) has a derivation of \( w \) that mimics this walk, such as \( q_0 \Rightarrow a_0q_1 \Rightarrow a_0a_1q_2 \Rightarrow \ldots \Rightarrow a_0a_1\ldots a_k \). If a string \( x \) is not in \( L' \), then the walk through \( N' \) does not end in a final state, and there will be no way to eliminate the last variable in the sentential form of \( x \) to result in a string of terminals, so \( G' \) will not have a derivation for \( x \). Since there is an equivalent regular grammar for every NFA, if a language is regular, it is generated by some regular grammar.

If \( L \) is regular, then there is a regular expression, \( R \), for \( L \). We can use \( R \) to construct a regular expression, \( R^R \), for \( L^R \), by observing that:

\[
\begin{align*}
    a^R &= a, \text{ for any } a \in \Sigma \\
    \varepsilon^R &= \varepsilon \\
    \emptyset^R &= \emptyset \\
    (r_1 \cup r_2)^R &= r_1^R \cup r_2^R \\
    (r_1 \cdot r_2)^R &= r_1^R \cdot r_2^R \\
    (r_1)^* &= (r_1)^R \\
\end{align*}
\]

For example, if \( R = r_1 \cdot r_2 \), we apply \( R^R = (r_1 \cdot r_2)^R = r_2^R \cdot r_1^R \). Now if \( r_1 = r_3 \cup r_4 \), we apply \( r_1^R = (r_3 \cup r_4)^R = r_4^R \cdot r_3^R \), etc. We apply the reversal to \( R \) and then to each component resulting from the reversal until we perform the reversal on one of the primitive regular expressions (\( a, \varepsilon, \emptyset \)) which are unchanged after reversal.

We can show that our procedure outlined above correctly creates a regular expression for \( L^R \), that is, \( (L(R))^R = L(R^R) \), by induction on the length of \( R \):

- Base case: if \( |R| = 1 \), then \( R \) is either of the form \( a, \varepsilon, \emptyset \). If \( R = a \), then \( a^R = a \); if \( R = \varepsilon \), \( \varepsilon^R = \varepsilon \), and if \( R = \emptyset \), then \( \emptyset^R = \emptyset \).
- Inductive hypothesis: assume that for all \( |R| < n \), \( (L(R))^R = L(R^R) \).
- Inductive step: let \( |R| = n + 1 \). Then \( R \) must be one of the following where \( r_1 \) and \( r_2 \) are regular expressions, \( |r_1|, |r_2| < n 

\[
\begin{align*}
    &\circ R = r_1 \cup r_2, \ \text{w} \in L(R) \iff w \in r_1 \cup r_2. \ \text{Then} \ W^R \in (L(R))^R \iff W^R \in (r_1 \cup r_2)^R = r_1^R \cup r_2^R = L(R^R). \\
    &\circ R = r_1 \cdot r_2, \ \text{w} \in L(R) \iff w \in r_1 \cdot r_2. \ \text{Then} \ W^R \in (L(R))^R \iff W^R \in (r_1 \cdot r_2)^R = r_2^R \cdot r_1^R = L(R^R). \\
    &\circ R = r_1^*, \ \text{w} \in L(R) \iff w \in r_1^*. \ \text{Then} \ W^R \in (L(R))^R \iff W^R \in (r_1^*)^R = (r_1^R)^* = L(R^R). \\
\end{align*}
\]
Note that in each case, the reversal is applied recursively to regular expressions with shorter lengths until the base case is reached, so the reversal procedure is guaranteed to terminate. Then, \( R^R \) is the reversal of \( R \), and \( R^R \) is the regular expression for \( L^R \). Since there is a regular expression for \( L^R \), \( L^R \) must be regular.

5) No. Let \( L = 0^*1^* \) which we know to be regular. Let \( L' = \{ w : w \in 0^n1^n, n > 0 \} \). \( L' \subseteq L \), specifically, \( L' \) is the set of all strings in \( L \) that have the same number of 0s and 1s, but \( L' \) is not regular. This counterexample shows that regular languages are not closed under subset.