HOMEWORK TWO SOLUTION– CSE 355

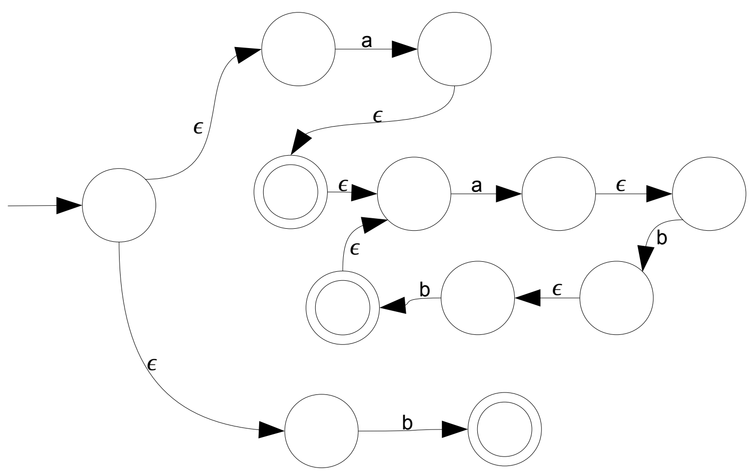
DUE: 22 FEBRUARY 2011

Please note that there is more than one way to answer most of these questions. The following only represents a sample solution.

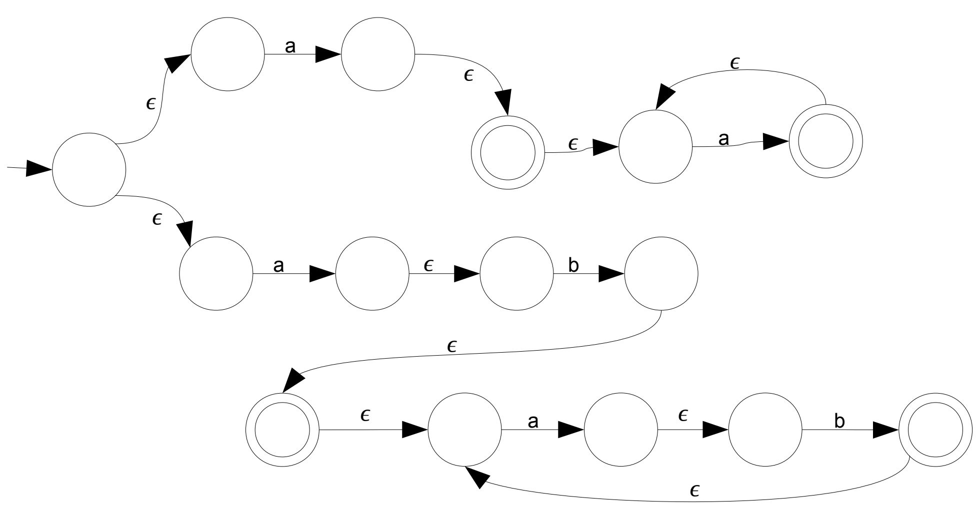
(1) **1.28: Regular Expressions to NFAs**

Solution:

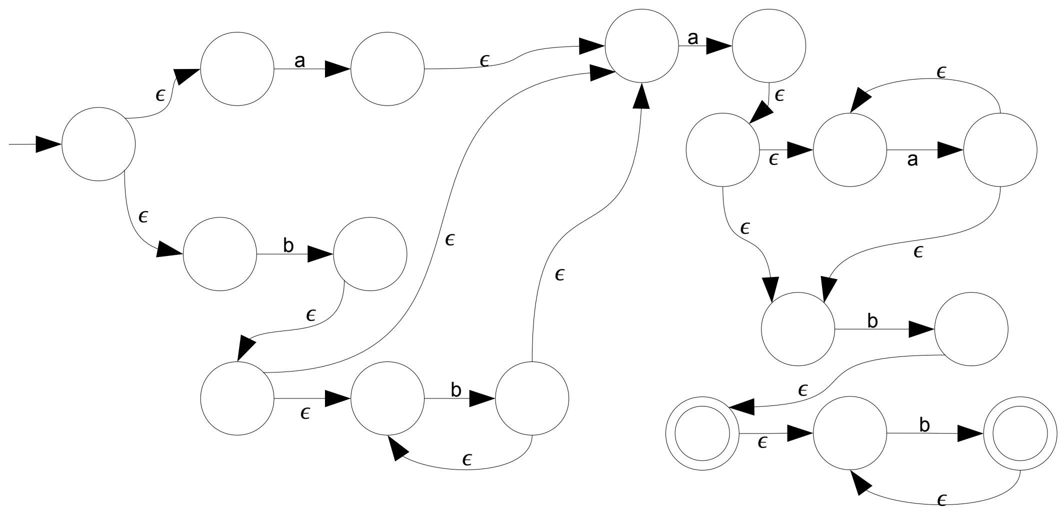
(a) $a(abb)^* \cup b$



(b) $a^+ \cup (ab)^+ = aa^* \cup ab(ab)^*$



(c) $(a \cup b^{+})a^{+}b^{+} = (a \cup bb^{*})aa^{*}bb^{*}$



(2) **1.31: If a language A is regular, then so is $A^{\mathcal{R}}$, its reverse**

Solution:

We will provide two solutions to this problem. One using automatas (DFA and NFA) and the other using regular expressions.

First we will use automatas. Since A is regular, there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that M recognizes A. We will construct an NFA $N = (Q', \Sigma, \delta', q_0', F')$ that will recognize $A^{\mathcal{R}}$. The informal idea behind N is to change M by adding a new start state that ϵ-transitions to M's accepting states and then reversing all the transition arrows of M and accepting only in M's start state. More formally, we will define N as follows:

- $Q' = Q \cup \{q_0'\}$, we add a new start state,
- $F' = \{q_0\}$, we only accept if our computation ends in the start state of the original machine.
- To define δ' we will use an idea similar to the pre-image. Define $\delta^{-1}(q, a) = \{q' | \delta(q', a) = q\}$. That is $\delta^{-1}(q, a)$ is the set of all states that have a transition to q on input a. We are now ready to define δ'.

$$\delta'(q, a) = \begin{cases} F & \text{if } q = q_0' \text{ and } a = \epsilon \\ \delta^{-1}(q, a) & \text{if } q \neq q_0' \text{ and } a \neq \epsilon \\ \emptyset & \text{otherwise} \end{cases}$$

  The first part of δ' uses ϵ-transitions from our new start state to F. The second part of δ' nondeterministically runs backward on M, trying all the possible ways an input string could have reached that state with M's transition function on input a. In essence, it reverses the direction of all transition arrows in M. The last part of δ' catches all other possibilities as leading to a sink state.

By how we defined δ', $q_0, q_1, \ldots, q_n$ is an accepting computation of M on input $w_1 w_2 \ldots w_n$ if and only if $q_0', q_n, \ldots, q_1, q_0$ is an accepting computation of N on input $w_n \ldots w_2 w_1$ (since

$\delta(q_i, w_{i+1}) = q_{i+1}$ iff $q_i \in \delta'(q_{i+1}, w_{i+1})$ for $0 \leq i < n$ and $q_n \in \delta'(q_0', \epsilon)$). Thus, N recognizes $A^{\mathcal{R}}$. Since for any regular language A there exists an NFA that recognizes $A^{\mathcal{R}}$, we conclude that the class of regular languages is closed under reverse.

Now we will use regular expressions. Because A is a regular language, A is the language of a regular expression R. We produce a regular expression $R^{\mathcal{R}}$ for $A^{\mathcal{R}}$ as follows:

$R^{\mathcal{R}} = reverse(R)$, where

$reverse := proc(R)$
  If $R = \emptyset$ then return R
  Else if $R = \epsilon$ then return R
  Else if $R = a$ then return R /* for each $a \in \Sigma$ /*
  Else if $R = (R_1 \cup R_2)$ then return $(reverse(R_1) \cup reverse(R_2))$
  Else if $R = R_1R_2$ then return $reverse(R_2)reverse(R_1)$
  Else if $R = (R_1)^*$ then return $(reverse(R_1))^*$
End

Informally, since R is finite and $reverse(R)$ either terminates (in the cases $R = \emptyset$, $R = \epsilon$, or $R = a$ for some $a \in \Sigma$) or recursively calls itself with a shorter regular expression (R_1 and R_2 must be shorter than R in the other cases since they are proper sub-components of R), we see that for any R, $reverse(R)$ will eventually terminate in a regular expression. The way $reverse(R)$ was defined guarantees that it will indeed produce a regular expression whose language is the reverse of the original language. Thus, if A is a regular language, there is a regular expression that describes $A^{\mathcal{R}}$. Therefore by Theorem 1.54, $A^{\mathcal{R}}$ is regular.

To formally see (and not required for the homework) $reverse(R)$ properly produces a regular expression whose language recognizes the reverse of the language of the original regular expression, we will proceed by strong induction on the length of R.

Inductive Hypothesis: Assume for any regular expression R with $|R| < n$, that $(L(R))^{\mathcal{R}} = L(reverse(R))$.

Inductive Step: Assume $|R| = n$. Then R must have one of six forms:

If $R = \emptyset$, $R = \epsilon$, or $R = a$ for some $a \in \Sigma$, then it is clear in any of these possibilities $(L(R))^{\mathcal{R}} = L(R) = L(reverse(R))$.

If $R = (R_1 \cup R_2)$ for some regular expressions R_1, R_2, then $x \in (L(R))^{\mathcal{R}} = (L(R_1 \cup R_2))^{\mathcal{R}}$ iff $x^{\mathcal{R}} \in L(R_1 \cup R_2)$ iff $x^{\mathcal{R}} \in L(R_1)$ or $x^{\mathcal{R}} \in L(R_2)$ iff $x \in (L(R_1))^{\mathcal{R}}$ or $x \in (L(R_2))^{\mathcal{R}}$ iff $x \in L(reverse(R_1))$ or $x \in L(reverse(R_2))$ iff $x \in L(reverse(R_1) \cup reverse(R_2)) = L(reverse(R))$, where we used that by the induction hypothesis and the facts that $|R_i| < n$ for $i = 1, 2$ (since they are proper sub-components of R and $|R| = n$) we have $(L(R_i))^{\mathcal{R}} = L(reverse(R_i))$ for $i = 1, 2$.

If $R = R_1R_2$ for some regular expressions R_1, R_2, then $x \in (L(R))^{\mathcal{R}} = (L(R_1R_2))^{\mathcal{R}}$ iff $x^{\mathcal{R}} \in L(R_1R_2)$ iff $x^{\mathcal{R}} = y^{\mathcal{R}}z^{\mathcal{R}}$ for some $y^{\mathcal{R}} \in L(R_1)$ and some $z^{\mathcal{R}} \in L(R_2)$ iff $x = zy$ for some $z \in (L(R_2))^{\mathcal{R}}$ and $y \in (L(R_1))^{\mathcal{R}}$) iff $x = zy$ for some $z \in L(reverse(R_2))$ and $y \in L(reverse(R_1))$ iff $x \in L(reverse(R_2)reverse(R_1)) = L(reverse(R))$, where we used that by the induction hypothesis and the facts that $|R_i| < n$ for $i = 1, 2$ (since they are proper sub-components of R and $|R| = n$) we have $(L(R_i))^{\mathcal{R}} = L(reverse(R_i))$ for $i = 1, 2$.

Lastly, if $R = (R_1)^*$ for some regular expression R_1, then $x \in (L(R))^{\mathcal{R}} = (L((R_1)^*))^{\mathcal{R}}$ iff $x^{\mathcal{R}} \in L((R_1)^*)$ iff $x^{\mathcal{R}} = x_n^{\mathcal{R}} \dots x_2^{\mathcal{R}} x_1^{\mathcal{R}}$ where each $x_i^{\mathcal{R}} \in L(R_1)$ iff $x = x_1 x_2 \dots x_n$ where each $x_i \in (L(R_1))^{\mathcal{R}}$ iff $x = x_1 x_2 \dots x_n$ where each $x_i \in L(reverse(R_1))$ iff $x \in L((reverse(R_1))^*) = L(reverse(R))$, where we used that by the induction hypothesis and the facts that $|R_1| < n$ (since it is a proper sub-component of R and $|R| = n$) we have $(L(R_1))^{\mathcal{R}} = L(reverse(R_1))$.

(3) **1.46: Nonregular languages by pumping lemma and closure properties**

Solution:

(a) $A = \{0^n 1^m 0^n | m, n \geq 0\}$

We will use the pumpling lemma to show that A is nonregular. For a contradiction, assume that A is regular. Then A has some pumping length p. Let $s = 0^p 1 0^p$. Then $s \in A$ and $|s| > p$. From the pumping lemma, then there must exist some x, y, z such that $s = xyz$ with (1) $|y| > 0$, (2) $|xy| \leq p$, and (3) for each $i \geq 0$, $xy^i z \in A$. From (1) and (2) it follows that $y = 0^k$ for some $0 < k \leq p$. However, then $xy^2 z = 0^{p+k} 1 0^p \notin A$, since there are more zeros at the start of the string (since $k \neq 0$) than at the end. This is a contradiction of (3). Thus, we conclude that A is not regular.

(c) $C = \{w | w \in \{0,1\}^*$ is not a palindrome$\}$

To show C is not regular, we will use closure properties of regular languages and the pumping lemma. For a contradiction, assume that C is regular. Then $\overline{C} = \{w | w \in \{0,1\}^*$ is a palindrome$\}$, C's complement, is regular. We will now use the pumping lemma to show that $\overline{C}$ is not regular. This is done similarly to how it was done in **(a)**. Then $\overline{C}$ has some pumping length p. Let $s = 0^p 1 0^p$. Then $s \in \overline{C}$ and $|s| > p$. From the pumping lemma, then there must exist some x, y, z such that $s = xyz$ with (1) $|y| > 0$, (2) $|xy| \leq p$, and (3) for each $i \geq 0$, $xy^i z \in \overline{C}$. From (1) and (2) it follows that $y = 0^k$ for some $0 < k \leq p$. However, then $xy^2 z = 0^{p+k} 1 0^p \notin \overline{C}$, since there are more zeros at the start of the string (since $k \neq 0$) than at the end and so it is not a palindrome. This is a contradiction of (3). Thus, we conclude that C is not regular since assuming it was led to the contradiction. (This also shows that the language of palindromes over $\{0, 1\}$ is not a regular language).

(d) $D = \{wtw | w, t \in \{0,1\}^*\}$

To show D is not regular, we will use the pumping lemma. For a contradiction, assume that D is regular. Then D has some pumping length p. Let $s = 0^p110^p1$. Then $s \in D$, with for example $w = 0^p1$ and $t = 1$, and $|s| > p$. (Note that the choice of s that worked for parts **(a)** and **(c)** won't work here!). From the pumping lemma, then there must exist some x, y, z such that $s = xyz$ with (1) $|y| > 0$, (2) $|xy| \leq p$, and (3) for each $i \geq 0$, $xy^iz \in A$. From (1) and (2) it follows that $y = 0^k$ for some $0 < k \leq p$. However, then $xy^2z = 0^{p+k}110^p1$. For a contradiction, assume that $0^{p+k}110^p1$ is in D, that is can be written as wtw. Then w must end with a single 1 from the last part of the string. Therefore, from the first part of the string we must have that $w = 0^{p+k}1$. However, there are only p 0s at the end of the string before the 1, so w cannot be equal to $0^{p+k}1$. This is a contradiction. Thus, $xy^2z = 0^{p+k}110^p1 \notin D$. This then is a contradiction of (3). Thus, we conclude that D is not regular.

(4) **1.51: $x \equiv_L y$ is an equivalence relation**

Solution:

To show that being indistiguishable by some language L is an equivalence relation we must show that the relation is reflexive, symmetric, and transitive. Recall, by definition for strings x and y to be indistinguishable by L ($x \equiv_L y$) then for every string z, $xz \in L$ iff $yz \in L$.

Reflexive: Clearly for any z, $xz \in L$ iff $xz \in L$. Thus, $x \equiv_L x$.
Symmetric: Assume $x \equiv_L y$, then for every string z, $xz \in L$ iff $yz \in L$. But then, $yz \in L$ iff $xz \in L$ and we have $y \equiv_L x$.
Transitive: Assume $x \equiv_L y$ and $y \equiv_L w$. Then for any z, $xz \in L$ iff $yz \in L$ and $yz \in L$ iff $wz \in L$. Thus, $xz \in L$ iff $wz \in L$. Whence, $x \equiv_L w$.

(5) **1.53: $ADD = \{x = y + z | x, y, z$ are binary integers, and x is the sum of y and $z\}$ is not regular**

Solution:

We will use the pumping lemma to show that ADD is not regular. For a contradiction, assume that ADD is regular. Then ADD has some pumping length p. Let $s = 1^p = 1^{p-1}0 + 1$. Then $s \in ADD$ and $|s| > p$. From the pumping lemma, then there must exist some x, y, z such that $s = xyz$ with (1) $|y| > 0$, (2) $|xy| \leq p$, and (3) for each $i \geq 0$, $xy^iz \in ADD$. From (1) and (2) it follows that $y = 1^k$ for some $0 < k \leq p$. However, then $xy^2z = 1^{p+k} = 1^{p-1}0 + 1 \notin ADD$, since $1^{p+k} \neq 1^{p-1}0 + 1$ with binary addition. This is a contradiction of (3). Thus, we conclude that ADD is not regular.