Problem Set 3

Spring 10

Due: Thursday Feb 18 in class before the lecture.

Please follow the homework format guidelines posted on the class web page:

`http://www.cs.uiuc.edu/class/sp10/cs373/`

1. [**Category**: Comprehension, **Points**: 20]

Give a regular expression for the following languages:

(a) $\Sigma = \{a, b\}$: The set of all strings where the second letter from the start and the end is an a.

(b) $\Sigma = \{a, b\}$: The set of all strings that have both aa and bb as a substring.

(c) $\Sigma = \{a, b, c\}$: The set of all strings, such that between any a and c there's at least one b.

Describe the language of each of the following regular expressions in your own words. Please be specific and try to minimize the amount of mathematical notation you use.

(a) $\Sigma = \{a, b\}$. $(ab + ba)*$

(b) $\Sigma = \{a, b\}$. $((a^*)b(a^*)b(a^*))^*b$

(c) $\Sigma = \{a, b, c\}$. $((\epsilon + a + aa + aaa)(b + c))^*(\epsilon + a + aa + aaa)$

Solution:

Give a regular expression for the following languages:

(a) $\Sigma = \{a, b\}$: The set of all strings where the second letter from the start and the end is an a.
Answer: $(a + b)a(a + b)^*a(a + b)$

(b) $\Sigma = \{a, b\}$: The set of all strings that have both aa and bb as a substring.
Answer: $((a + b)^*aa(a + b)^*bb(a + b)^*) + ((a + b)^*bb(a + b)^*aa(a + b)^*)$

(c) $\Sigma = \{a, b, c\}$: The set of all strings, such that between any a and c there's at least one b.
Answer: $((\epsilon + aa^* + cc^*)b)^*(\epsilon + aa^* + cc^*)$

Describe the language of each of the following regular expressions in your own words. Please be specific and try to minimize the amount of mathematical notation you use.

(a) $\Sigma = \{a, b\}$. $(ab + ba)*$
Answer: The set of all strings of as and bs that have an equal amount of as and bs.

(b) $\Sigma = \{a, b\}$. $((a^*)b(a^*)b(a^*))^*b$

Answer: The set of all strings of as and bs that have an odd amount of bs and end with b.

(c) $\Sigma = \{a, b, c\}$. $((\epsilon + a + aa + aaa)(b + c))^*(\epsilon + a + aa + aaa)$

Answer: The set of all strings of as, bs and cs that contain no more than 3 consecutive as.

2. Intersect 'em [**Category**: Construction, **Points**: 20]

You are given two NFAs $A_1 = (P, \Sigma, \delta_1, p_0, F_1)$ and $A_2 = (Q, \Sigma, \delta_2, q_0, F_2)$.

Construct an NFA that will accept the language $L(A_1) \cap L(A_2)$ with no more than $|P| * |Q|$ states. Also, prove that it indeed accepts the language of the intersection as stated above.

Solution:

The language $L(A_1) \cap L(A_2)$ is accepted by the NFA $A = (R, \Sigma, \delta, r_0, F)$, where

$R = P \times Q$;

δ is a transition function $R \times \Sigma_\epsilon \to 2^R$. For any state $(p, q) \in R$, where $p \in P$, $q \in Q$, and for any input character $x \in \Sigma_\epsilon$, $\delta((p, q), x) = \{(p', q') \mid p' \in \delta_1(p, x) \wedge q' \in \delta_2(q, x)\}$. Moreover, $\delta((p, q), \epsilon) = \{(p', q') \mid p' \in \delta_1(p, x) \cup \{p\} \wedge q' \in \delta_1(q, x) \cup \{q\}\}$;

$r_0 = (p_0, q_0)$;

$F = \{(p', q') \mid p' \in F_1 \wedge q' \in F_2\}$.

Note that the number of states in A is $|P| * |Q|$. To prove that A indeed accepts the language of the intersection, we need to show that for any string w in Σ^*, A accepts w if and only if both A_1 and A_2 accepts w:

($\Rightarrow$) If A accepts w, without loss of generality, it suffices to show that A_1 accepts w. By the definition of acceptance, there is a sequence of states $s_0, s_1, \ldots, s_n$ in R and a sequence of inputs $x_1, x_2, \ldots, x_n$ in Σ_ϵ, such that $w = x_1 x_2 \ldots x_n$, $s_0 = r_0$, $s_n \in F$, and $s_{i+1} \in \delta(s_i, x_{i+1})$ for every $0 \leq i \leq n - 1$. Since the states of A is the product of P and Q, let $s_i = (u_i, v_i)$ for each $0 \leq i \leq n$, where $u_i \in P$, $v_i \in Q$. Now consider the sequence of states $u_0, u_1, \ldots, u_n$. Removing those u_{i+1} and x_{i+i} such that $u_i = u_{i+1}$ and $x_{i+1} = \epsilon$ yields a sequence $u_0, u_1, \ldots, u_m$ and a sequence $x_1 x_2 \ldots x_m$ By the definitions of r_0, F and δ, it is easy to see that $u_0 = p_0$, $u_m \in F_1$, and $u_{i+1} \in \delta(u_i, x_{i+1})$ for every $0 \leq i \leq m - 1$. Hence A_1 accepts the sequence $x_1 x_2 \ldots x_m$, i.e., accepts w.

($\Leftarrow$) If both A_1 and A_2 accepts w, by definition there is a sequence of states $u_0, u_1, \ldots, u_m$ in P and a sequence of inputs $x_1, x_2, \ldots, x_m$ in Σ_ϵ, such that $w = x_1x_2\ldots x_m$, $u_0 = p_0$, $u_m \in F_1$, and $u_{i+1} \in \delta(u_i, x_{i+1})$ for every $0 \leq i \leq m-1$. Similarly, there is a sequence of states $v_0, v_1, \ldots, v_k$ in Q and a sequence of inputs $y_1, y_2, \ldots, y_k$ in Σ_ϵ, such that $w = y_1y_2\ldots y_k$, $v_0 = q_0$, $v_k \in F_2$, and $v_{i+1} \in \delta(v_i, y_{i+1})$ for every $0 \leq i \leq k-1$. Then we can unify $x_1x_2\ldots x_m$ and $y_1y_2\ldots y_k$ to a sequence $z_1z_2\ldots z_n$ by inserting some ϵ properly. Both sequences of states are extended to $u_0u_1\ldots u_n$ and $v_0v_1\ldots v_n$ We claim the sequence $(u_0, v_0), (u_1, v_1), \ldots, (u_n, v_n)$ accepts the sequence $z_1z_2\ldots z_n$. The start and final states are easy to verify. For any $0 \leq i \leq n-1$, if $z_{i+1} = \epsilon$, then either u_i or v_i makes a missing transition to u_{i+1}(or v_{i+1}). Otherwise both u_i/v_i make a normal transition to u_{i+1}/v_{i+1}, respectively. In both cases $(u_{i+1}, v_{i+1}) \in \delta((u_i, v_i), z_{i+1})$. Thus A also accepts w.

3. Reverse determinism [**Category**: Construction, **Points**: 20]

Recall the formal definition of an NFA (Sipser p. 53). Let's generalize the definition by substituting the unique start state q_0 by a set of start state S, so that the computation of an NFA is allowed to start from any state in S. A *Reverse Deterministic Automaton*(RDA) is an generalized NFA $A = (Q, \Sigma, \delta, S, F)$ where

(a) for each state $q \in Q$, $\delta(q, \epsilon) = \varnothing$;

(b) for each state $q \in Q$ and each character $x \in \Sigma$, there is a unique $p \in Q$ such that $q \in \delta(p, x)$;

(c) $|F| = 1$.

Graphically, an RDA does not allow two distinct states to merge into one state via two transitions reading the same input. Moreover, an RDA has multiple start states, a unique accept state, and no ϵ-transition.

Given an RDA $A = (Q, \Sigma, \delta, S, q_f)$, construct an RDA $\bar{A}$ with no more than $|Q|$ states that will accept the complement language $\overline{L(A)}$. Proof that $\bar{A}$ is indeed an RDA and complements A.

Solution:

Let the language recognized by A be L, then the complement language $\bar{L}$ is simply accepted by $\bar{A} = \{Q, \Sigma, \delta, Q - S, q_f\}$. Here is a proof.

Let the reverse language of L be $L^R = \{w^R \mid w \in L\}$. It is easy to prove that $\bar{L} = (\overline{L^R})^R$. Starting from A, we are going to built automata for L^R, $\overline{L^R}$ and $(\overline{L^R})^R$, respectively.

First, the language L^R is recognized by a DFA $A^R = \{Q, \Sigma, \delta^R, q_f, S\}$ where δ^R is defined so that for any $q \in Q$ and $x \in \Sigma$, $\delta^R(q, x) = p$ where $p \in Q$ is the unique state such that $q \in \delta(p, x)$. Note that by the definition of an RDA, we can always find p. A^R is indeed an DFA since the start state q_f is unique, the transition function δ^R is deterministic. Then we claim that $L(A^R) = L^R$. By the definition of acceptance, for any string $w = x_1 x_2 \ldots x_n$, w is accepted by A^R if and only if there exists a sequence of states $r_0 r_1 \ldots r_n$ that fulfills the acceptance conditions of A^R. This is equivalent to the existence of a reverse sequence $r_n \ldots r_2 r_1$ for accepting $w^R = x_n \ldots x_2 x_1$ by A:

- A^R starts out in the start state q_f and ends up in an accept state in S if and only if A starts in a state in S and ends up in q_f;
- According to δ^R, A^R goes from q to p by reading x if and only if q is one of the allowable next states when A is in state p and reading x.

Second, thank to the nice closure property of DFAs under complement, the language $\overline{L^R}$ is recognized by a DFA $\overline{A^R} = \{Q, \Sigma, \delta^R, q_f, Q - S\}$, which simply flips the accept/reject states of A^R.

Finally, by swapping back the start/accept states, an RDA $\bar{A} = \{Q, \Sigma, \delta, Q - S, q_f\}$ recognizes the reverse language of $\overline{L^R}$, i.e., $\overline{L^R}^R$. $\bar{A}$ is indeed an RDA because simply flipping the starting/non-starting states of A affects none of the three conditions for an RDA. The proof is similar to that in the first step. Since $\bar{L} = (\overline{L^R})^R$, $\bar{A}$ recognizes $\bar{L}$. Note that the number of states in A and $\bar{A}$ are the same.