CSL 705: Theory of Computation

II semester 2011-12

Mon 06 Feb 2012 | 14:30-15:30 | I Minor | WS-213 | Max Marks 40

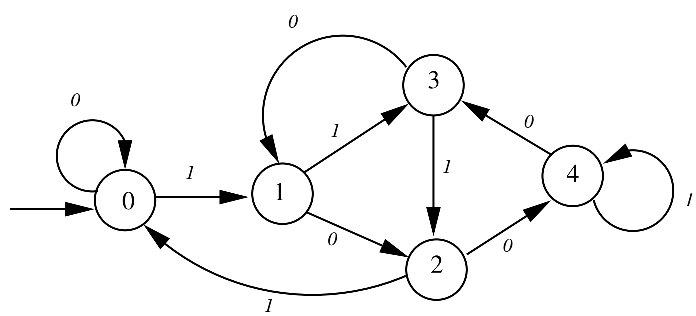
1. *Please answer in the space provided on the question paper. The other sheets are only for rough work and will not be collected.*
2. *You may use any paper-based material including your class notes and any other text books.*
3. *You are not allowed to share reference material or rough pages during the exam.*
4. *You are not allowed to bring into the exam hall any electronic gadgets such as computers, mobile phones or calculators.*
5. *Please keep your identity card with you. You may be asked for it at any time for verification.*

---

1. **(10 marks)** Construct regular expressions $\mathbf{r}_0, \cdots, \mathbf{r}_4$ over the alphabet $\mathbb{2} = \{0,1\}$ representing each of the following languages. You don't need to prove anything, but it must be intuitively clear that your answer correctly represents the required language.

$$\begin{array}{rcl}\mathcal{L}(\mathbf{r}_0) & = & \{x \in \mathbb{2}^* \mid (x)_2 \ \mathbf{mod}\ 5 = 0\} \\ \mathcal{L}(\mathbf{r}_1) & = & \{x \in \mathbb{2}^* \mid (x)_2 \ \mathbf{mod}\ 5 = 1\} \\ \mathcal{L}(\mathbf{r}_2) & = & \{x \in \mathbb{2}^* \mid (x)_2 \ \mathbf{mod}\ 5 = 2\} \\ \mathcal{L}(\mathbf{r}_3) & = & \{x \in \mathbb{2}^* \mid (x)_2 \ \mathbf{mod}\ 5 = 3\} \\ \mathcal{L}(\mathbf{r}_4) & = & \{x \in \mathbb{2}^* \mid (x)_2 \ \mathbf{mod}\ 5 = 4\}\end{array}$$

Solution. The laziest and easisest thing to say is perhaps that the five languages may be described as the solution to the set of simultaneous equations of regular expressions in the variables $\mathbf{r_0}$, $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$ and $\mathbf{r_4}$ as derived from the following diagram.



$$\begin{array}{rcl}\mathbf{r_0} & = & \mathbf{0^* + r_2.1} \\ \mathbf{r_1} & = & \mathbf{r_0.1 + r_3.0} \\ \mathbf{r_2} & = & \mathbf{r_1.0 + r_3.1} \\ \mathbf{r_3} & = & \mathbf{r_1.1 + r_4.0} \\ \mathbf{r_4} & = & \mathbf{1 * + r_2.0}\end{array}$$

An alternative method is to find one regular expression (say $\mathbf{r_0}$) by appealing to the proof of the theorem that every regular language may be be represented by a regular expression. After that we may use the above equations to express every other expression in terms of $\mathbf{r_0}$ by substitution.

The most tedious and mind-bending method would be to use the proof of the theorem individually for each one of the expressions $\mathbf{r_0}$, $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_3}$ and $\mathbf{r_4}$.

2. **(10 marks)** Let $L, M \subseteq \Sigma^*$ be languages on a nonempty finite alphabet Σ. For any $x, y \in \Sigma^*$, the zip of x and y (denoted $x \bowtie y$) is defined by induction as follows:

$$\begin{array}{lcll} \varepsilon \bowtie y & = & \{y\} & \\ x \bowtie \varepsilon & = & \{x\} & \\ ax' \bowtie by' & = & ab.(x' \bowtie y') & \text{if } x = ax' \text{ and } y = by' \end{array}$$

It is extended to languages L and M as follows:

$$L \bowtie M = \{x \bowtie y \mid x \in L, y \in M\}$$

Prove that $L \bowtie M$ is a regular language if L and M are regular.

Solution. Actually $L \bowtie M$ is regular if both L and M are regular. Let $B = \langle P, \Sigma, \beta, p_0, E\rangle$ and $C = \langle Q, \Sigma, \gamma, q_0^1, F\rangle$ be DFAs with $\mathcal{L}(B) = L$ and $\mathcal{L}(C) = M$. The definition of $\bowtie$ especially for strings of unequal length complicates the matter a little bit. Hence given that $x = y \bowtie z$ it is not in general, possible to determine which symbol in x came from y and which from z. So we mark the symbols coming from each language as follows. Let b and c be two new symbols and let $B^b = \langle P, \{b\} \times \Sigma, \beta^b, p_0, E\rangle$ and $C^c = \langle Q, \{c\} \times \Sigma, \gamma^c, q_0, F\rangle$ be DFAs such that $\beta^b(p, (b, a)) = p'$ iff $\beta(p, a) = p'$ and $\gamma^c(q, (c, a)) = q'$ iff $\gamma(q, a) = q'$ It is then clear that $a_1 \ldots a_n \in \mathcal{L}(B) = L$ iff $(b, a_1) \ldots (b, a_n) \in \mathcal{L}(B^b) = L^b$ and $a_1 \ldots a_n \in \mathcal{L}(C)$ iff $(c, a_1) \ldots (c, a_n) \in \mathcal{L}(C^c) = M^c$. For any $y = a_0 \ldots a_m \in L$ and $z = a'_0 \ldots a'_n \in M$, we design a composite automaton D^{bc} over the alphabet $\{b\} \times \Sigma \cup \{c\} \times \Sigma$ which accepts the string $x^{bc} = y^b \bowtie z^c$ where $y^b = (b, a_0) \ldots (b, a_m) \in L^b$ and $z^c = (c, a'_0) \ldots (c, a'_n) \in M^c$ so that

$$x^{bc} = \begin{cases} (b, a_0)(c, a'_0) \ldots (b, a_m)(c, a'_m)(c, a'_{m+1}) \ldots (c, a'_n) & \text{if } m < n \\ (b, a_0)(c, a'_0) \ldots (b, a_m)(c, a'_m)\text{if } m = n & \\ (b, a_0)(c, a'_0) \ldots (b, a_n)(c, a'_n)(b, a_{n+1}) \ldots (b, a_m) & \text{if } m > n \end{cases}$$

To exercise the right amount of control we incorporate two pieces of control information in the state space of the composite automaton. We define the state-space Q^{bc} of the composite automaton D^{bc} as $Q^{bc} = (T \times \{b, c\} \times P \times Q) \cup \{q_{err}\}$, where $T = \{\emptyset, b, c\}$ is "termination" information, the second component represents "turn" information and q_{err} is an error state. Hence for example a state $(\emptyset, c, p, q) \in Q^{bc}$ denotes the fact that for a given string input $x^{bc} = y^b \bowtie z^c$, neither y^b nor z^c is known to be empty and the next input symbol must be of the form (c, a). On the other hand if it is known that $y^b = \epsilon$ then the state $(\{b\}, c, p, q) \in Q^{bc}$ denotes the fact that no more symbols from $\{b\} \times \Sigma = \Sigma^b$ are expected and the rest of the input is from $\{c\} \times \Sigma = \Sigma^c$. The initial state is $q_0^{bc} = (\emptyset, b, p_0, q_0)$. The set of accepting states is $F^{bc} = T \times \{b, c\} \times E \times F$. The transition function δ^{bc} is defined as follows

$$\begin{array}{lcll} \delta^{bc}((\emptyset, b, p, q), (b, a)) & = & (\emptyset, c, p', q) & \text{if} \quad \beta^b(p, a) = p' \\ \delta^{bc}((\emptyset, c, p, q), (c, a)) & = & (\emptyset, c, p, q') & \text{if} \quad \gamma^c(q, a) = q' \\ \delta^{bc}((\emptyset, b, p, q), (c, a)) & = & (\{b\}, c, p, q') & \text{if} \quad p \in E \wedge \gamma^c(q, a) = q' \\ \delta^{bc}((\emptyset, c, p, q), (b, a)) & = & (\{c\}, b, p', q) & \text{if} \quad q \in F \wedge \beta^b(p, a) = p' \\ \delta^{bc}((\{b\}, c, p, q), (b, a)) & = & q_{err} & \\ \delta^{bc}((\{c\}, b, p, q), (c, a)) & = & q_{err} & \\ \delta^{bc}(q_{err}, (b, a)) & = & q_{err} & \\ \delta^{bc}(q_{err}, (c, a)) & = & q_{err} & \\ \delta^{bc}((\{b\}, c, p, q), (c, a)) & = & (\{b\}, c, p, q') & \text{if} \quad p \in E \wedge \gamma^c(q, a) = q' \\ \delta^{bc}((\{c\}, b, p, q), (b, a) & = & (\{c\}, b, p', q) & \text{if} \quad q \in F \wedge \beta^b(p, a) = p' \end{array}$$

Once D^{bc} has been designed to take care of all the control aspects, we may remove the labels b and c from the alphabet of the automaton to produce the (non-deterministic) automaton $N_{\bowtie}$ which is identical to D^{bc} except for the fact that the alphabet is restored to Σ. The notion of acceptance is such that $x \in \Sigma^*$ may be accepted provided there exist $y \in L$ and $z \in M$ such that $x = y \bowtie z$.

3. **(10 marks)** Let $\Sigma = \{a, b\}$ and let $L = \bigcup\limits_{m>0} a^m ⪤ b^m$. Prove that L is not a regular language.

Solution. Firstly we note the following easy to show facts:

(a) $L \subseteq \{x \in \mathbb{2}^* \mid \#_a(x) = \#_b(x)\}$[1].

(b) For any $n \geq 0$, $a^n b^n \in L$.

(c) For any $m, n \geq 0$, $m \neq n \Rightarrow a^m b^n \notin L$.

Armed with these facts we outline the challenger's winning strategy. For any $m > 0$ chosen by the defender, the challenger chooses $x = a^m b^m$. The defender is then forced (by the constraints $|uv| \leq m$ and $v \neq \epsilon$) to choose $u = a^i$ and $v = a^j$ such that $i \geq 0$, $j > 0$ and $i + j \leq m$. This implies that $w = a^{m-i-j} b^m$. For any such decomposition of x, for each $k \geq 0$ we have $x_k = a^i (a^j)^k a^{m-i-j} b^m = a^{m+j(k-1)} b^m$. Clearly, $x_k \notin L$ for any value of k except $k = 1$. The challenger may then choose $k = 0$.

Remark. This problem also illustrates that whereas finite unions of regular languages are regular, infinite unions of regular languages need not be regular. For each $m \geq 0$ we know that $a^m ⪤ b^m$ must be regular because for each m, $\{a^m\}$ and $\{b^m\}$ are both regular. However L which is an infinite union of regular langauages, is not regular.

[1] In fact we may show equality

4. **(10 marks)** Design a DFA which accepts exactly the following language over the alphabet $\mathbb{2} = \{0, 1\}$.

$$L = \{x \in \mathbb{2}^* \mid (x)_2 \ \mathbf{mod}\ 84 = 47\}$$

Solution. We know that for any $k > 1$, a DFA with k-states may be constructed where each of the k-states numbered from 0 to $k-1$ denotes the value of n **mod** k for any binary string whose value is n.

We have that $84 = 3 \times 4 \times 7$ and since $3, 4, 7$ are relatively prime to each other we may obtain the modulo-84 DFA by taking the product of modulo-3, modulo-4 and modulo-7 automata. Since the automaton accepts only binary strings of value $84k + 47$, we need to determine the accepting state(s) of the 84-state automaton. We have $(84m + 47)$ **mod** $3 = 2$, $(84m + 47)$ **mod** $4 = 3$ and $(84m + 47)$ **mod** $7 = 5$.

Given the three automata $D_{3i+2} = \langle P, \mathbb{2}, \delta_3, p_0, \{p_2\}\rangle$, $D_{4j+3} = \langle Q, \mathbb{2}, \delta_4, q_0, \{q_3\}\rangle$ and $D_{7k+5} = \langle R, \mathbb{2}, \delta_7, r_0, \{r_5\}\rangle$ the following table gives the transition functions of the three automata respectively.

δ_3	0	1	δ_7	0	1
p_0	p_0	p_1	r_0	r_0	r_1
p_1	p_2	p_0	r_1	r_2	r_3
p_2	p_1	p_2	r_2	r_4	r_5
δ_4			r_3	r_6	r_0
q_0	q_0	q_1	r_4	r_1	r_2
q_1	q_2	q_3	r_5	r_3	r_4
q_2	q_0	q_1	r_6	r_5	r_6
q_3	q_2	q_3			

The required automaton is the product of the above three automata and is defined by $D_{84m+47} = \langle P \times Q \times R, \mathbb{2}, \delta_{84}, (p_0, q_0, r_0), \{(p_2, q_3, r_5)\}\rangle$ and $\delta_{84}((p_i, q_j, r_k), b) = (p_{i'}, q_{j'}, r_{k'})$ iff $\delta_3(p_i, b) = p_{i'}$, $\delta_4(q_j, b) = q_{j'}$ and $\delta_7(r_k, b) = r_{k'}$ for each $b \in \mathbb{2}$.