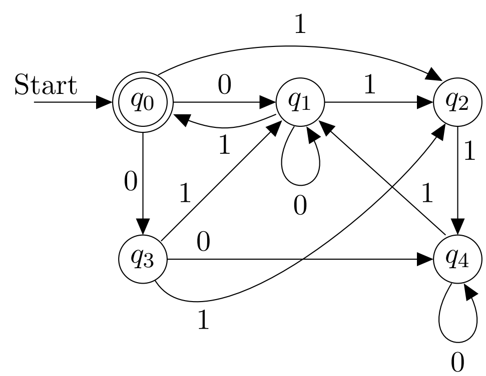
Master Degree in Computer Engineering

Final Exam for
Automata, Languages and Computation

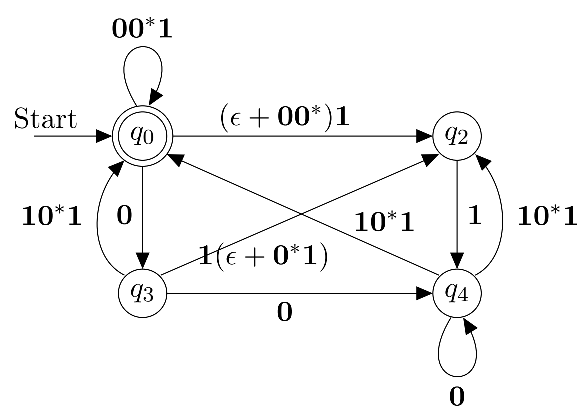
February 12th, 2021

1. **[6 points]** Consider the FA A whose transition function is graphically represented as follows

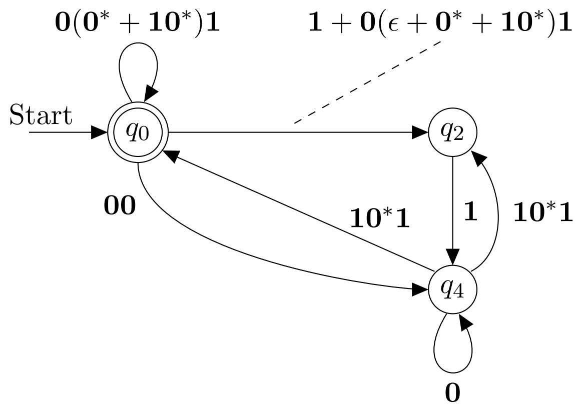


Consider the algorithm for transforming a FA into a regular expression, based on state elimination. Apply the algorithm to eliminate state q_1 from A, and display the resulting automaton A'. Furthermore, eliminate state q_3 from A', and display the resulting automaton A''. If you simplify any of the resulting regular expressions, **add some discussion**.

Solution After the elimination of q_1 from A we obtain the automaton A', graphically represented as



After the elimination of q_3 from A' we obtain the automaton A'', graphically represented as



2. **[8 points]** Consider the following languages, defined over the alphabet $\Sigma = \{a, b\}$

$$\begin{aligned} L_1 &= \{a^p b a^q b a^q b a^r \mid p, q, r \geq 1\} , \\ L_2 &= \{a^p b a^q b a^{q+r} \mid p, q, r \geq 1\} , \\ L_3 &= \{a^{p+q} b a^{q+r} \mid p, q, r \geq 1\} . \end{aligned}$$

State whether L_1, L_2 and L_3 are regular languages, and provide a mathematical proof of your answers.

Solution

(a) L_1 is not a regular language. To prove this statement we apply the pumping lemma for regular languages. Let N be the pumping lemma constant associated with L_1. We choose the string $w = aba^N ba^N ba$, $w \in L_1$, and consider all possible factorizations $w = xyz$ satisfying the conditions $y \neq \varepsilon$ and $|xy| \leq N$. We separately discuss all possible cases in what follows.

- If y contains an occurrence of b, the iterated string $xyyz$ will not belong to L_1, since it contains more than three occurrences of b. Therefore in the next items we assume that y does not contain any occurrence of b.
- If y contains the leftmost occurrence of a in w, we must have $x = \varepsilon$ and $y = a$. Then the iterated string xz will not belong to L_1, since it starts with b.
- If y contains any occurrence of a from the second run of a's in the string $w = aba^N ba^N ba$, we have that the iterated string xz has the form $aba^p ba^N ba$ with $p < N$, which again cannot be in L_1 because the second and the third runs of a's do not have the same length.
- No other case is possible for the factorization $w = xyz$, since from condition $|xy| \leq N$ we have that y cannot contain any occurrence of a from the third run of a's in the string $w = aba^N ba^N ba$.

Since we have falsified the pumping lemma, we must conclude that L_1 is not a regular language.

(b) L_2 is not a regular language. To prove this statement we again apply the pumping lemma for regular languages. Let N be the pumping lemma constant associated with L_2. We choose the string $w = aba^N ba^{N+1}$, $w \in L_2$, and consider all possible factorizations $w = xyz$. We separately discuss all possible cases in what follows; the discussion is very similar to the one in item (a) above.

- If y contains an occurrence of b, the iterated string $xyyz$ will not belong to L_2, since it contains more than two occurrences of b. Therefore in the next items we assume that y does not contain any occurrence of b.
- If y contains the leftmost occurrence of a in w, we must have $x = \varepsilon$ and $y = a$. Then the iterated string xz will not belong to L_2, since it starts with b.
- If y contains any occurrence of a from the second run of a's in the string $aba^N ba^{N+1}$, we have that the iterated string $xyyz$ has the form $aba^{N+p}ba^{N+1}$ with $p > 0$, which cannot be in L_2 because the second run of a's is not shorter in length than the third run of a's.
- No other case is possible for the factorization $w = xyz$, since from condition $|xy| \leq N$ we have that y cannot contain any occurrence of a from the third run of a's in the string w.

Since we have falsified the pumping lemma, we must conclude that L_2 is not a regular language.

(c) L_3 is a regular language. In fact, it is not difficult to see that L_3 is generated by the regular expression $\boldsymbol{aaa^*baaa^*}$.

3. **[6 points]** Considering the intersection operation between two languages, answer the following questions

(a) Show that the class of context-free languages is not closed under intersection.

(b) Specify in detail the construction that takes as input a PDA P and a DFA A and produces a PDA P' that accepts the language $L(P) \cap L(A)$.

Solution

(a) The textbook reports two languages in CFL whose intersection is no longer in CFL.

(b) The specification of the state set and of the transition function of the intersection PDA are reported in the textbook.

4. **[6 points]** Assess whether the following statements are true or false, providing a mathematical proof for all of your answers.

(a) Every language in CFL is also in $\mathcal{P}$, and there exists a language $L \in \mathcal{P}$ such that L is not in CFL.

(b) Let L_1 and L_2 be two languages such that $L_1 \leq_m L_2$ and $\overline{L_1}$ not in REC. In some cases, we might have $\overline{L_2}$ in REC.

(c) Let L_1, L_2 and L_3 be languages such that $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$. Then we have $L_1 \leq_p L_3$ (symbol $\leq_p$ indicates the existence of a polynomial time reduction between two languages).

Solution

(a) True. Here $\mathcal{P}$ is the class of languages that can be recognized in polynomial time by a TM. Let L be a CFL. To decide whether an input string belongs to L, we use the CKY algorithm specified in the textbook, which takes polynomial time on a RAM machine. Since we can simulate a RAM program with a TM with only polynomial time overhead, we can decide whether any input string belongs to L on a TM using polynomial time.

Consider now the language $L = \{a^n b^n c^n \mid n \geq 1\}$. It is easy to show that L is not in CFL, using the pumping lemma. Furthermore, it is not difficult to define a TM recognizing L and running in polynomial time.

(b) False. From $\overline{L_1}$ not in REC we have that L_1 must also be outside of REC, following a theorem from the textbook. From $L_1 \leq_m L_2$ and the definition of reduction, we have that L_2 must be outside of REC as well. But then $\overline{L_2}$ cannot belong to REC.

(c) True. In other words, we need to prove that the polynomial time reduction relation is transitive. Since $L_1 \leq_p L_2$, there exists a TM program P_1 that runs in polynomial time $O(n^{p_1})$ and that transforms instances of the problem associated with L_1 into instances of the problem associated with L_2, with the property that $P_1(x)$ is a positive instance if and only if x is a positive instance. Similarly, from $L_2 \leq_p L_3$ we have that there exists a TM program P_2 that runs in polynomial time $O(n^{p_2})$ and that transforms instances of the problem associated with L_2 into instances of the problem associated with L_3, with the property that $P_2(x)$ is a positive instance if and only if x is a positive instance. If we compose P_1 and P_2 into $P_2 \circ P_1$, we have a TM program that runs in polynomial time $O((n^{p_1})^{p_2}) = O(n^{p_1 \cdot p_2})$. It is easy to see that $P_2 \circ P_1$ transforms instances of the problem associated with L_1 into instances of the problem associated with L_3, with the property that $P_2 \circ P_1(x)$ is a positive instance if and only if x is a positive instance. We thus conclude that $L_1 \leq_p L_3$.

Note: a common mistake for this question is to assert that $P_2 \circ P_1$ runs in polynomial time $O(n^{p_1+p_2})$.

5. **[7 points]** Define the following property of the RE languages defined over the alphabet $\Sigma = \{0,1\}$

$$\mathcal{P} \;\; = \;\; \{L \mid L \in \text{RE},\ L \neq \Sigma^*\} \; .$$

Assess whether the language $L_{\mathcal{P}}$ belongs to the class REC, and provide a mathematical proof of your answer. Consider also the following language

$$L \;\; = \;\; \{\mathsf{enc}(M, M') \mid L(M) \cap L(M') \neq \Sigma^*\}$$

where M, M' are generic TMs accepting languages defined over Σ, and $\mathsf{enc}(M, M')$ is a binary string representing a fixed encoding for M, M'. Assess whether the language L belongs to the class REC, and provide a mathematical proof of your answer.

Solution Recall that $L_{\mathcal{P}} = \{\mathsf{enc}(M) \mid L(M) \in \mathcal{P}\}$. We prove that $L_{\mathcal{P}}$ is not in REC by applying Rice's theorem. We need to show that $\mathcal{P}$ is not a trivial property.

- $\mathcal{P} \neq \emptyset$. Consider a string $w \in \Sigma$ and the finite language $L = \{w\}$. We have $L \neq \Sigma^*$ and thus $L \in \mathcal{P}$ and $\mathcal{P}$ is not empty.
- $\mathcal{P} \neq$ RE. It is immediate to see that the language Σ^* does not belong to $\mathcal{P}$. Since Σ^* is in RE, we have that $\mathcal{P}$ is not RE.

Since $\mathcal{P}$ is not trivial, from Rice's theorem we have that $\mathcal{P}$ is not in REC.

To answer the second point, we show that L is not in REC. Note that we cannot apply Rice's theorem in this case, since a string in L is not the encoding of a single TM, it is instead the encoding of a *pair* of TMs. We then need to produce a reduction. Since we know from the first part of this question that $L_{\mathcal{P}}$ is not in REC, we prove $L_{\mathcal{P}} \leq_m L$.

We need to provide an effective construction, that is, a construction that can be implemented by a TM with output that always halts, that maps strings of the form $\mathsf{enc}(M)$ into strings of the form

$\mathsf{enc}(M', M'')$ such that $\mathsf{enc}(M) \in L_{\mathcal{P}}$ if and only if $\mathsf{enc}(M', M'') \in L$. To do this, we simply set $M' = M$ and $M'' = M$. In this way, we have the following chain of logical equivalences, which proves that the construction is a valid reduction

$$\begin{array}{llll} \mathsf{enc}(M) \in L_{\mathcal{P}} & \text{iff} & L(M) \neq \Sigma^* & \text{(definition of } L_{\mathcal{P}}) \\ & \text{iff} & L(M) \cap L(M) \neq \Sigma^* & \text{(definition of } \cap) \\ & \text{iff} & L(M') \cap L(M'') \neq \Sigma^* & \text{(construction of } M', M'') \\ & \text{iff} & \mathsf{enc}(M', M'') \in L & \text{(definition of } L). \end{array}$$