Master Degree in Computer Engineering

Final Exam for Automata, Languages and Computation

January 25th, 2023

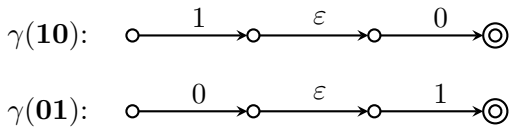
1. **[4 points]** Consider the regular expression $R = (\mathbf{10})^*\emptyset(\epsilon + \mathbf{01})$. Convert R into an equivalent ϵ-NFA using the construction provided in the textbook, and report all the intermediate steps.
Important: do not use any other construction different from the one presented in class, and do not simplify the regular expression R before applying the construction.

Solution

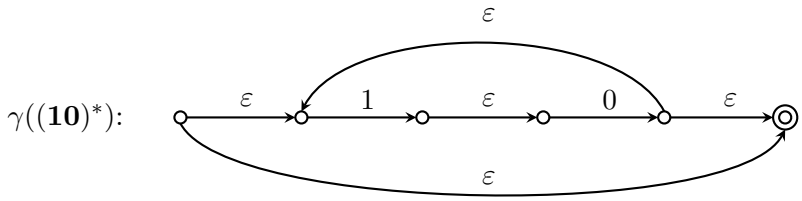
The construction to convert a regular expression into an equivalent ϵ-NFA is presented in Theorem 3.7 from Chapter 3 of the textbook. The construction must be applied using structural induction, that is, it must be applied to all the subexpressions of the input regular expression. For a subexpressions s of r, we write $\gamma(s)$ to represent its conversion into an equivalent ϵ-NFA.

In the base case, we convert the regular expressions $\mathbf{0}$, $\mathbf{1}$, ϵ and $\emptyset$, resulting in the following automata:

Next, we use $\gamma(\mathbf{0})$ and $\gamma(\mathbf{1})$ to convert the regular expressions $\mathbf{10}$ and $\mathbf{01}$, resulting in the following automata:

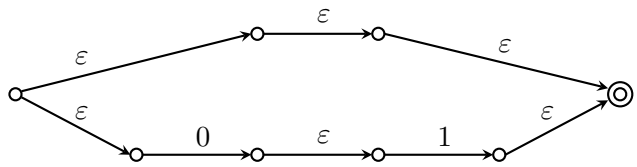


We can now use $\gamma(\mathbf{10})$ to convert the regular expression $(\mathbf{10})^*$, resulting in the following automaton:



To convert the regular expression $\epsilon + \mathbf{01}$ we use the automata $\gamma(\epsilon)$ and $\gamma(\mathbf{01})$ obtained at some previous steps, resulting in the following automaton:

$\gamma(\epsilon + \mathbf{01})$:



We need to perform two more steps. First we convert the regular expression $(\mathbf{10})^*\emptyset$ by concatenating $\gamma(\mathbf{10})$ with $\gamma(\emptyset)$. This is done by adding an ε-arc from the final state of $\gamma(\mathbf{10})$ to the initial state of $\gamma(\emptyset)$. In the resulting automaton $\gamma((\mathbf{10})^*\emptyset)$, the initial state is the initial state of $\gamma(\mathbf{10})$ and the final state is the final state of $\gamma(\emptyset)$. Observe that in $\gamma((\mathbf{10})^*\emptyset)$ the final state cannot be reached from the initial state, as expected.

Similarly, we convert the regular expression $(\mathbf{10})^*\emptyset(\epsilon+\mathbf{01})$ by concatenating $\gamma((\mathbf{10})^*\emptyset)$ with $\gamma(\epsilon+\mathbf{01})$. The resulting automaton is the desired ϵ-NFA.

2. **[9 points]** Consider the following languages, defined over the alphabet $\Sigma = \{a, b\}$

$$\begin{aligned} L_1 &= \{a^n b^{2n} \mid n \geq 1\} ; \\ L_2 &= \{a^n v \mid n \geq 1,\ v \in \Sigma^*,\ |v| = 2n\} ; \\ L_3 &= \{uv \mid n \geq 1,\ u, v \in \Sigma^*,\ |u| = n,\ |v| = 2n\} . \end{aligned}$$

For each of the above languages, state whether it belongs to REG or else CFL$\setminus$REG, and provide a mathematical proof for all of your answers.

Solution

(a) L_1 belongs to the class CFL$\setminus$REG. We first show that L_1 is not a regular language, by applying the pumping lemma for this class.

Let N be the pumping lemma constant for L_1. We choose the string $w = a^N b^{2N} \in L_1$ with $|w| \geq N$, and consider all possible factorizations $w = xyz$ satisfying the conditions $|y| \geq 1$ and $|xy| \leq N$. Because of the latter condition, we have that y can only contain occurrences of symbol a.

According to the pumping lemma, the string $w_k = xy^k z$ should be in L_1 for every $k \geq 0$. Let $|y| = m \geq 1$ and consider $k = 0$. We then have $w_0 = a^{N-m} b^{2N}$. From $m \geq 1$, it is immediate to see that $w_0 \notin L_1$, against the statement of the pumping lemma for regular languages. We thus conclude that L_1 is not a regular language.

Finally, we need to show that L_1 belongs to the class CFL. Consider the CFG G_1 with productions:

$$S \to aSbb \mid abb$$

It is very easy to see that $L(G_1) = L_1$.

(b) L_2 belongs to the class CFL$\setminus$REG. Again, we first show that L_2 is not a regular language by applying the pumping lemma.

Let N be the pumping lemma constant for L_2. We choose the string $w = a^N b^{2N} \in L_2$ with $|w| \geq N$, and consider all possible factorizations $w = xyz$ satisfying the conditions $|y| \geq 1$

and $|xy| \leq N$. Because of the latter condition, we have that y can only contain occurrences of symbol a.

According to the pumping lemma, the string $w_k = xy^kz$ should be in L_2 for every $k \geq 0$. Let $|y| = m \geq 1$ and consider $k = 0$. It is easy to see that $w_0 = a^{N-m}b^{2N}$. We now observe that, for w_0 to be in L_2, we must be able to chop w_0 into three parts of equal length, and the first part must be entirely composed of occurrences of symbol a. The length of each part is $(3N - m)/3 = N - \frac{1}{3}m$, and the second and third parts together have $2N - \frac{2}{3}m$ symbols. Since there are $2N > 2N - \frac{2}{3}m$ occurrences of b in w_0, we see that some occurrence of b must appear in the first part, and therefore w_0 cannot belong to L_2. Again, we conclude that L_2 is not a regular language.

Consider now the following CFG G_2:

$$S \to aSC \mid aC$$
$$C \to aa \mid ab \mid ba \mid bb$$

It is not difficult to see that $L(G_2) = L_2$. We thus conclude that L_2 belongs to the class CFL.

(c) L_3 is in REG. To show this, we observe that every string $w \in L_3$ has length $n + 2n = 3n$ for some $n \geq 1$, and there is no restrictions on its symbols. Therefore L_3 can be generated by the following regular expression

$$(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})((\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b}))^*.$$

3. **[5 points]** With reference to push-down automata (PDA), answer the following questions.

   (a) Provide the definition of language accepted by final state and language accepted by empty stack.

   (b) Prove that if P_F is a PDA accepting by final state, then there exists a PDA P_N accepting by empty stack such that $L(P_F) = N(P_N)$.

Solution

The required construction is reported in Theorem 6.11 from Chapter 6 of the textbook.

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

   (a) Let L be a language in REG and let R be the reversal operator. Then the language $L' = \{w \mid w \in L \text{ or } w \in L^R\}$ is also in REG.

   (b) The class RE is closed under the operation of complementation with respect to $\Sigma = \{0, 1\}$ (recall that every language in RE is defined over the alphabet Σ).

   (c) The class $\mathcal{P}$ of languages that can be recognized in polynomial time by a TM is closed under intersection with the class REG.

   (d) The class REG is closed under intersection with the class $\mathcal{P}$ of languages that can be recognized in polynomial time by a TM.

Solution

(a) True. We observe that $L' = L \cup L^R$. Since the class REG is closed under the reversal operator, the language L^R must be in REG. Since the class REG is closed under the union operator, the language $L \cup L^R$ must be in REG.

(b) False. We know from Chapter 9 of the textbook that the language L_e is the complement of language L_{ne}, and that L_{ne} is in RE but L_e is not in RE.

(c) True. Let L_1 be a language in $\mathcal{P}$ and let L_2 be a language in REG. Then there exists a TM M working in polynomial time such $L(M) = L_1$ and there exists a DFA A such that $L(A) = L_2$. We know from Chapter 4 of the textbook that we can decide in linear time whether a string is accepted by a DFA. We can then construct a TM M_i accepting the language $L_1 \cap L_2$ and working in polynomial time. On input string w, M_i simulates A on input w in linear time in $|w|$. If A rejects then also M_i rejects. If A accepts, then M_i simulates M on input w in polynomial time in $|w|$. If M rejects then M_i rejects, otherwise M_i accepts. It is easy to see that the cascade of the two simulations above can be run in polynomial time in $|w|$.

(d) False. Let $L_1 = \{a, b\}^*$ and let $L_2 = \{a^n b^n \mid n \geq 1\}$. It is well known that L_1 is in REG. We now show that L_2 is in $\mathcal{P}$ by specifying a TM M that recognizes L_2 in polynomial time.

M uses a tape with two tracks: the first track contains the input string, and the second track is used to mark the symbols that have already been matched. The machine first checks that the input has at least one occurrence of symbol a and at least one occurrence of symbol b, with all of the b's placed to the right of all of the a's. Then M reads the leftmost occurrence of a, marks it with a special symbol, and matches it to the leftmost occurrence of b by marking the latter symbol as well. It then proceeds with the next matches, if any, moving back and forth on the tape and adding two new markers at each pass, each to the right of the existing markers. M accepts if the count of the a's equals the count of the b's. All of the above can be easily implemented in polynomial time in the size of the input.

We now observe that $L_1 \cap L_2 = L_2$, and we know that L_2 is not in REG (by application of the pumping lemma for REG). We then conclude that the class REG is not closed under intersection with the class $\mathcal{P}$.

5. **[9 points]** Consider the following property of the RE languages defined over the alphabet $\Sigma = \{0, 1\}$:

$$\mathcal{P} \quad = \quad \{L \mid L \in \text{RE}, \ \text{ every string in } L \text{ has prefix } 111\}$$

and define $L_{\mathcal{P}} = \{\mathsf{enc}(M) \mid L(M) \in \mathcal{P}\}$.

(a) State whether $\emptyset \in \mathcal{P}$ is true or false, and motivate your answer.

(b) Use Rice's theorem to show that $L_{\mathcal{P}}$ is not in REC.

(c) Prove that $L_{\mathcal{P}}$ is not in RE.

(d) Consider the language

$$L_{\subseteq} \quad = \quad \{\mathsf{enc}(M_1, M_2) \mid L(M_1) \subseteq L(M_2)\} \ .$$

Specify a reduction $L_{\mathcal{P}} \leq_m L_{\subseteq}$ to show that $L_{\subseteq}$ is not in RE.

Solution

(a) True. Since there is no string in $\emptyset$, the condition $\mathcal{P}$ is trivially satisfied.

(b) We apply Rice's theorem and show that $\mathcal{P}$ is not trivial. First, the language $\emptyset$ is in RE and we have already shown that $\emptyset \in \mathcal{P}$. Therefore we have $\mathcal{P} \neq \emptyset$. Second, the language $\{101\}$ is in RE and does not belong to $\mathcal{P}$. Therefore we have $\mathcal{P} \neq$ RE. We then conclude that $L_{\mathcal{P}}$ is not in REC.

(c) To prove that $L_{\mathcal{P}}$ is not in RE, it is convenient to consider the complement language $\overline{L_{\mathcal{P}}}$, whose strings are all the encodings of TMs M such that $L(M)$ contains at least one string that does not start with 111. We show that $\overline{L_{\mathcal{P}}}$ belongs to RE by specifying a nondeterministic TM N such that $L(N) = \overline{L_{\mathcal{P}}}$. Since every nondeterministic TM can be converted into a standard TM, we conclude that $\overline{L_{\mathcal{P}}}$ is in RE.

Our nondeterministic TM N takes as input the encoding of a TM M and performs the following steps.

- N nondeterministically guesses a string $w \in \Sigma^*$, $\Sigma = \{0, 1\}$, such that w does not have prefix 111.
- N simulates the universal TM U on input $\mathsf{enc}(M, w)$. If U halts in an accepting state, then N accepts. If U halts in a non-accepting state, then N rejects. If U does not halt, then N does not halt as well.

It is not difficult to see that $L(N) = \overline{L_{\mathcal{P}}}$.

Since $\overline{L_{\mathcal{P}}}$ is in RE, if its complement language $L_{\mathcal{P}}$ were in RE as well, then we would conclude that both languages are in REC, from a theorem in the textbook. But we have already shown that $L_{\mathcal{P}}$ is not in REC. We must therefore conclude that $L_{\mathcal{P}}$ is not in RE.

(d) To specify a reduction $L_{\mathcal{P}} \leq_m L_{\subseteq}$ we need to describe a mapping from strings $\mathsf{enc}(M)$ to strings $\mathsf{enc}(M_1, M_2)$ with the property that $\mathsf{enc}(M) \in L_{\mathcal{P}}$ if and only if the associated string $\mathsf{enc}(M_1, M_2) \in L_{\subseteq}$.

Consider the regular language $\{111\} \cdot \Sigma^*$. It is not difficult to construct a TM M_{111} accepting such language. Our mapping then sets $M_1 = M$ and $M_2 = M_{111}$. The following chain of logical equivalences shows that the proposed mapping represents a valid reduction:

$$\begin{array}{lll} \mathsf{enc}(M) \in L_{\mathcal{P}} & \text{iff} \quad L(M) \in \mathcal{P} & \text{(definition of } L_{\mathcal{P}}\text{)} \\ & \text{iff} \quad L(M) \subseteq \{111\} \cdot \Sigma^* & \text{(definition of } \mathcal{P}\text{)} \\ & \text{iff} \quad L(M_1) \subseteq L(M_2) & \text{(definition of our mapping)} \\ & \text{iff} \quad \mathsf{enc}(M_1, M_2) \in L_{\subseteq} & \text{(definition of } L_{\subseteq}\text{)} \end{array}$$

Since we have already shown that $L_{\mathcal{P}}$ is not in RE, from the above reduction we conclude that $L_{\subseteq}$ is not in RE as well.