Master Degree in Computer Engineering

Final Exam for Automata, Languages and Computation

February 21st, 2022

1. **[5 points]** In relation to the notion of regular expression, answer the following questions.

   (a) Provide the recursive definition of regular expression E over an alphabet Σ and the generated language $L(E)$.

   (b) Using structural induction, prove that a regular expression E over Σ without the Kleene operator '$*$' generates a finite language.

   Solution

   (a) The required definition can be found in Chapter 2 of the textbook, Section 3.1.2.

   (b) Base case: $E = \epsilon$, $E = \emptyset$, or $E = \boldsymbol{a}$ for $a \in \Sigma$. These regular expressions do not have any occurrence of the Kleene operator and generate a finite language by definition.

   Induction: Let E be a regular expression with no occurrence of the Kleene operator. According to the definition of regular expression in (a), we distinguish the following cases.

   i. If $E = F + G$, then F and G do not have any occurrence of the Kleene operator. We can apply the inductive hypothesis, deriving that $L(F)$ and $L(G)$ are both finite. Since $L(E) = L(F) \cup L(G)$, we have $|L(E)| \leq |L(F)| + |L(G)|$ and thus $L(E)$ is finite as well.

   ii. If $E = FG$, then F and G do not have any occurrence of the Kleene operator. We can apply the inductive hypothesis, deriving that $L(F)$ and $L(G)$ are both finite. Since $L(E) = L(F)L(G)$, we have $|L(E)| \leq |L(F)| \cdot |L(G)|$ and thus $L(E)$ is finite.

   iii. The case $E = F^*$ is not considered here, because E contains occurrence of the Kleene operator.

   iv. If $E = (F)$, then F does not have any occurrence of the Kleene operator. We can apply the inductive hypothesis, deriving that $L(F)$ is finite. Since $L(E) = L(F)$, $L(E)$ must be finite.

2. **[9 points]** Consider the following languages, defined over the alphabet $\Sigma = \{0, 1\}$

$$
\begin{aligned}
L_1 &= \{w \mid w = uu,\ u \in \Sigma^+\} \ ; \\
L_2 &= \{w \mid w = uxu,\ u \in \Sigma^+,\ x \in \Sigma^*\} \ ; \\
L_3 &= \{w \mid w = xuyuz,\ u \in \Sigma^+,\ x, y, z \in \Sigma^*\} \ .
\end{aligned}
$$

   State whether the above languages 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 use the pumping lemma for regular languages. Let N be the pumping lemma constant. We choose the string $w = 0^N 1 0^N 1 \in L_1$ 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 0 from the left run 0^N in w.

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 $w_0 = 0^{N-m} 1 0^N 1$. Assuming that m is even, we can factorize w_0 into two strings of equal length $w_0 = uu'$. It is easy to see that u ends with an occurrence of 0, since $m \geq 1$, while u' ends with an occurrence of 1. Then $u \neq u'$ and $w_0 \notin L_1$, which is a contraddiction. We thus conclude that L_1 is not a regular language.

(b) L_2 is not a regular language. Let N be the pumping lemma constant. We choose the string $w = 0^N 1 0^N 1 \in L_2$. Again, we have that y can only contain occurrences of symbol 0 from the left run 0^N in w.

According to the pumping lemma, the string $w_k = xy^k z$ should be in L_2 for every $k \geq 0$. Let $|y| = m \geq 1$ and consider $k = 2$. We then have $w_2 = 0^{N+m} 1 0^N 1$. For w_2 to be in L_2, we must find a factorization $w_2 = uxu'$ such that $u = u'$, where $u \in \Sigma^+$ and $x \in \Sigma^*$. We observe that u' must end with an occurrence of 1. Thus we can only choose $u = 0^{N+m}1$. However, since $m \geq 1$, there is no choice for a string u' to the right of u satisfying the desired equivalence $u = u'$, because we only have N occurrences of symbol 0 to the right of u. We thus conclude that L_2 is not a regular language.

(c) L_3 is a regular language. Intuitively, we cannot apply to L_3 the same reasoning in (a) and (b) above, because in a string in L_3 we do not know where the boundaries for the two occurrences of u are placed.

Consider the language $L_3' = \{w \mid w = xayaz,\ a \in \Sigma,\ x, y, z \in \Sigma^*\}$. L_3' is a regular language and it can be generated by the following regular expression

$$(\mathbf{0}+\mathbf{1})^*\mathbf{0}(\mathbf{0}+\mathbf{1})^*\mathbf{0}(\mathbf{0}+\mathbf{1})^* + (\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})^* \,.$$

We show that $L_3 = L_3'$.

i. $L_3' \subseteq L_3$. Any string $w \in L_3'$ can be factorized as $w = xayaz$ with $a \in \Sigma$ and $x, y, z \in \Sigma^*$. By letting $a = u \in \Sigma^+$, we immediately have $w \in L_3$.

ii. $L_3 \subseteq L_3'$. Any string $w \in L_3$ can be factorized as $w = xuyuz$, with $u \in \Sigma^+$ and $x, y, z \in \Sigma^*$. We can write $u = au'$ with $a \in \Sigma$ and $u' \in \Sigma^*$. This provides $w = xau'yau'z$, which implies $w \in L_3'$.

3. **[6 points]** With reference to the class of context-free languages, answer the following questions.

(a) Define the notion of substitution over some alphabet Σ, and extend the definition to strings and languages.

(b) Prove that if L is a CFL defined over Σ and s is a substitution on Σ such that, for each $a \in \Sigma$, $s(a)$ is a CFL, then $s(L)$ is a CFL.

Solution

The required definition and proof can be found in Chapter 7 of the textbook, Section 7.3.1.

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

   (a) For strings $w_1, w_2 \in \Sigma^*$, we say that w_2 is a *proper prefix* of w_1 if $w_1 = w_2 u$ for some $u \in \Sigma^+$. There exists an infinite regular language L such that, for any two strings $w_1, w_2 \in L$, w_1 is not a proper prefix of w_2.

   (b) There exist languages L_1, L_3 in CFL$\smallsetminus$REG and L_2 in REG, all defined over the same alphabet Σ, such that $L_1 \subseteq L_2 \subseteq L_3$.

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

Solution

(a) True. Let $\Sigma = \{a, b\}$ and consider the infinite regular language $L = \{w \mid w = a^n b,\ n \geq 0\}$, which can be generated by the regular expression $\boldsymbol{a^* b}$. Let $w_1 = a^p b \in L$. String w_1 is a proper prefix of some string w_2 if and only if $w_2 = a^p b a^q$ with $q \geq 1$. But then w_2 cannot be in L.

(b) True. Let $\Sigma = \{a, b\}$ and consider the languages

$$\begin{aligned} L_1 &= \{w \mid w = a^n b^n,\ n \geq 0\}\ , \\ L_2 &= \{w \mid w = a^n b^m,\ n, m \geq 0\}\ , \\ L_3 &= \{w \mid w = b^n a^n,\ n \geq 0\} \cup L_2\ . \end{aligned}$$

It is easy to show that L_2 is in REG and L_1, L_3 are CFL. Furthermore, using the pumping lemma, we can show that L_1, L_3 are not in REG. The containments $L_1 \subseteq L_2$ and $L_2 \subseteq L_3$ directly follow from the language definitions.

(c) True. Let L_1, L_2 be two arbitrary languages in $\mathcal{P}$. By definition of $\mathcal{P}$, there exist TMs M_1, M_2, both working in polynomial time, such that $L(M_1) = L_1$ and $L(M_2) = L_2$. We can now construct the desired TM M such that $L(M) = L_1 L_2$. Let w be the input string to M. For each i with $0 \leq i \leq |w|$, M performs the following steps:

- split w into substrings u, v such that $w = uv$ and $|u| = i$;
- simulate M_1 on u and simulate M_2 on v
- if both simulations accept, then halt and accept;
- if $i < |w|$ continue the for loop, otherwise halt and reject.

To show that $L_1 L_2 \subseteq L(M)$, consider strings $u \in L_1$ and $v \in L_2$. When given as input string uv, M will halt and accept for $i = |u|$. To show that $L(M) \subseteq L_1 L_2$, assume that on input w M halts and accepts at the i-th execution of the for loop. Let $w = uv$ with $|u| = i$. From the specification of M, we have $u \in L_1$ and $v \in L_2$, and therefore $w = uv \in L_1 L_2$. We thus conclude that $L(M) = L_1 L_2$. Finally, M runs the body of its for loop $|w| + 1$ times, and the body of the for loop can be executed in polynomial time in $|w|$. We thus conclude that M runs in polynomial time.

5. **[7 points]** In relation to the notion of Turing machine (TM), answer the following questions.

(a) Let M be a TM defined over the input alphabet $\Sigma = \{0, 1\}$, and let $\mathsf{enc}(M)$ be some binary encoding of M. Consider the languages

$$\begin{aligned} L_1 &= \{\mathsf{enc}(M) \mid \text{there is some input such that } M \text{ accepts in exactly 5 steps}\} , \\ L_2 &= \{\mathsf{enc}(M) \mid \text{there is some input of length 5 that is accepted by } M\} . \end{aligned}$$

Assess whether L_1 and L_2 belong to the class REC.

(b) Let M_1 and M_2 be TMs defined over the input alphabet $\Sigma = \{0, 1\}$, and let $\mathsf{enc}(M_1, M_2)$ be some binary encoding of M_1 and M_2. Consider the language

$$L_3 = \{\mathsf{enc}(M_1, M_2) \mid \overline{L(M_1)} = L(M_2)\} ,$$

where $\overline{L}$ is the complement of language L with respect to Σ^*. Assess whether L_3 belongs to the classes RE or not.

Solution

(a) Language L_1 belongs to REC. To see this, we observe that in 5 steps a TM M can only read the 5 leftmost symbols of its input tape. We can then simulate M on all of the finitely many possible configurations of the input tape having only the leftmost 5 cells filled in by some input alphabet symbol. We accept if any simulation leads to acceptance, and reject otherwise. It is immediate to see that the procedure specified above always halts.

Language L_2 does not belong to REC. To see this, we define a property of the RE languages $\mathcal{P} = \{L \mid L \in \text{RE}, \text{ there exists some } w \in L \text{ such that } |w| = 5\}$. We now define $L_{\mathcal{P}} = \{\mathsf{enc}(M) \mid L(M) \in \mathcal{P}\}$, and observe that $L_{\mathcal{P}} = L_2$. We can then apply Rice's theorem and show that $\mathcal{P}$ is not trivial. First, Σ^* is in RE and contains some string w such that $|w| = 5$. Therefore we have $\Sigma^* \in \mathcal{P}$ and $\mathcal{P}$ is not empty. Second, the empty language $\emptyset$ is in RE and does not contain any string w such that $|w| = 5$. Therefore we have $\emptyset \notin \mathcal{P}$, and thus $\mathcal{P}$ does not contain every RE language. Since $\mathcal{P}$ is not trivial, we can conclude that L_2 is not in REC, according to Rice's theorem.

(b) L_3 is not in RE. To show this, we consider the language L_e, that is, the language of the encodings of all TMs that accept the empty language, and show a reduction $L_e \leq_m L_3$. Since L_e is not in RE, the reduction proves the desired claim.

We need to map instances $\mathsf{enc}(M)$ of L_e into instances $\mathsf{enc}(M_1, M_2)$ of L_3. Let M_{Σ^*} be a TM such that $L(M_{\Sigma^*}) = \Sigma^*$. We set $M_1 = M$ and $M_2 = M_{\Sigma^*}$. The following chain of logical equivalences shows that the construction represents a valid reduction:

$$\begin{array}{llll} \mathsf{enc}(M) \in L_e & \text{iff} & L(M) = \emptyset & \text{(definition of } L_e\text{)} \\ & \text{iff} & \overline{L(M)} = \Sigma^* & \text{(definition of complementation)} \\ & \text{iff} & \overline{L(M_1)} = L(M_2) & \text{(definition of our reduction)} \\ & \text{iff} & \mathsf{enc}(M_1, M_2) \in L_3 & \text{(definition of } L_3\text{) .} \end{array}$$