Master Degree in Computer Engineering

Final Exam for
Automata, Languages and Computation

February 13th, 2023

1. **[5 points]** Let E be a regular expression and let R be the reversal operator. Specify the construction presented in the textbook for converting E into a regular expression E^R such that $L(E^R) = (L(E))^R$, and prove the equivalence relation using structural induction.

   Solution

   The required construction along with the proof of the relation $L(E^R) = (L(E))^R$ is reported in Theorem 4.11 from Chapter 4 of the textbook.

2. **[7 points]** Let $\Sigma = \{a, b\}$. For $w \in \Sigma^*$ and $X \in \Sigma$, we write $\#_X(w)$ to denote the number of occurrences of X in w. Consider now the following two languages

$$\begin{aligned} L_1 &= \{w \mid 0 \leq \#_a(w) \leq \#_b(w)\} ; \\ L_2 &= \{w \mid 0 \leq \#_a(w) \leq \#_b(w) \leq 17\} . \end{aligned}$$

   (a) Prove that L_1 is not REG.

   (b) Show that L_1 is in CFL.

   (c) Argue that L_2 is in REG.

   Solution

   (a) L_1 is not in REG. To show this, we apply the pumping lemma for the class REG.

   Let N be the pumping lemma constant for L_1. We choose the string $w = a^N b^N \in L_1$ with $|w| \geq N$, and we 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 $k = 2$. We then have $w_2 = a^{N+m} b^{2N}$. From $m \geq 1$, it is immediate to see that $w_2 \notin L_1$, since in w_2 the number of occurrences of symbol a exceeds the number of occurrences of symbol b. We thus conclude that L_1 is not a regular language.

   (b) L_1 is in CFL. To show this, we informally describe a PDA M such that $L(M) = L_1$. M has state set $Q = \{q_0, q_1\}$, with q_0 the inital state and q_1 the only final state. The stack symbol set of M is $\Gamma = \{A, B, Z_0\}$, with Z_0 the initial stack symbol.

   Computations of M are informally described in what follows.

   - In state q_0 and with Z_0 at the top of the stack, if M reads a it pushes A into the stack, if M reads b it pushes B into the stack; M stays in state q_0.

- In state q_0 and with A at the top of the stack, if M reads a it pushes A into the stack, if M reads b it pops A from the stack; M stays in state q_0.
- In state q_0 and with B at the top of the stack, if M reads b it pushes B into the stack, if M reads a it pops B from the stack; M stays in state q_0.
- In state q_0 and with B or Z_0 at the top of the stack, M can nondeterministically take an ε-transition and move to final state q_1, from which no further move is possible.

We observe that at any time in the computation, symbol Z_0 is always at the bottom of the stack, and symbols A and B cannot be both present in the stack. If the stack contains some A, then the number of a's that have been processed exceeds the number of b's. Symmetrically, if the stack contains some B, then the number of b's that have been processed exceeds the number of a's. Finally, if the stack contains only Z_0, then an equal number of a's and b's have been processed. From the above, it is not difficult to see that $L(M) = L_1$.

(c) L_2 is in REG. To see this, we observe that a string in L_2 can have at most 17 occurrences of a and at most 17 occurrences of b. This means that strings in L_2 have length bounded by 34, and thus L_2 is a finite language. Since finite languages are all in REG, we have completed the proof.

3. **[5 points]** With reference to the membership problem for context-free languages, answer the following two questions.

(a) Specify the dynamic programming algorithm developed in class for the solution of this problem.

(b) Consider the CFG G defined by the following rules:

$$\begin{aligned} S &\to BC \\ B &\to BB \mid b \\ C &\to BC \mid c \end{aligned}$$

Assuming as input the CFG G and the string $w = bbbbc$, trace the application of the above algorithm.

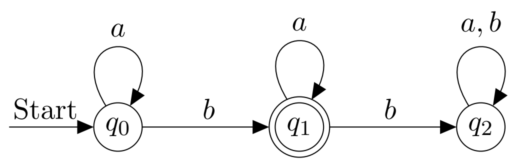
Solution

(a) The required dynamic programming algorithm is reported in Section 7.4.4 from Chapter 7 of the textbook.

(b) The algorithm constructs a table filling its rows one by one, in a bottom-up way. Each entry in the table is filled with a set of variables of the grammar. On input w and G, the algorithm constructs the table reported below.

$\{S, C\}$				
$\{B\}$	$\{S, C\}$			
$\{B\}$	$\{B\}$	$\{S, C\}$		
$\{B\}$	$\{B\}$	$\{B\}$	$\{S, C\}$	
$\{B\}$	$\{B\}$	$\{B\}$	$\{B\}$	$\{C\}$
b	b	b	b	c

4. **[6 points]** Consider the alphabet $\Sigma = \{a, b\}$ and the DFA A over Σ whose transition function is graphically represented as



(a) Describe in words the language $L(A)$.

(b) For each state q of A, provide a definition for properties $\mathcal{P}_q$ in such a way that, for any string $x \in \{a, b\}^*$, we have

$$\mathcal{P}_q(x) \Leftrightarrow \hat{\delta}(q_0, x) = q \ .$$

(c) Using mutual induction, prove $\hat{\delta}(q_0, x) = q_1 \Rightarrow \mathcal{P}_{q_1}(x)$.

Solution

(a) DFA A accepts the language L defined as the set of all strings over $\{a, b\}$ that contain exactly one occurrence of symbol b.

(b) For $x \in \Sigma^*$ and $X \in \Sigma$, we write $\#_X(x)$ to denote the number of occurrences of X in x. We can define the required properties as follows. For every $x \in \{a, b\}^*$:

- $\mathcal{P}_{q_0}(x)$ holds if and only if $\#_b(x) = 0$;
- $\mathcal{P}_{q_1}(x)$ holds if and only if $\#_b(x) = 1$, which amounts to $x \in L$;
- $\mathcal{P}_{q_2}(x)$ holds if and only if $\#_b(x) > 1$.

(c) Proof of $\hat{\delta}(q_0, x) = q_1 \Rightarrow \mathcal{P}_{q_1}(x)$. The proof is by mutual induction on the length of x.

Base. We have $|x| = 0$, that is, $x = \varepsilon$. Since $\hat{\delta}(q_0, x) = q_1$ is false, the implication is true.

Induction. Let $|x| = n > 0$. We can then write $x = yY$, where $Y \in \{a, b\}$, $y \in \{a, b\}^*$, and $|y| = n - 1$. We need to distinguish two cases.

- Case 1: $Y = a$. An inspection of the graphical representation of the transition function of A shows that the DFA was already in state q_1 after reading the prefix string y, that is, $\hat{\delta}(q_0, y) = q_1$. Since $|y| = n - 1$, we can apply induction, that is, we can use the implication $\hat{\delta}(q_0, y) = q_1 \Rightarrow \mathcal{P}_{q_1}(y)$. From the definition of $\mathcal{P}_{q_1}(y)$ we have that $\#_b(y) = 1$. Since $Y = a$, we conclude that $\#_b(x) = 1$ as well, and therefore $\mathcal{P}_{q_1}(x)$ holds true, as desired.
- Case 2: $Y = b$. From the transition function of the automaton, we derive that A can only have reached state q_1 coming from state q_0, that is, $\hat{\delta}(q_0, y) = q_0$. Since $|y| = n - 1$, we apply mutual induction and use the implication $\hat{\delta}(q_0, y) = q_0 \Rightarrow \mathcal{P}_{q_0}(y)$. We then conclude that $\mathcal{P}_{q_0}(y)$ holds true. This means that $\#_b(y) = 0$ and, since $Y = b$, we derive $\#_b(x) = 1$, that is, $\mathcal{P}_{q_1}(x)$ holds true, as desired.

5. **[3 points]** Define the Post correspondence problem (PCP) and discuss a simple example.

   Solution

   The required definition along with some examples can be found in Section 9.4.1 from Chapter 9 of the textbook.

6. **[7 points]** Consider the following property of the RE languages defined over the alphabet $\Sigma = \{0, 1\}$

$$\mathcal{P} \quad = \quad \{L \mid L \in \text{RE}, \text{ for every pair } u, v \in L \text{ we have } u \cdot v \notin L\}$$

and define $L_\mathcal{P} = \{\mathsf{enc}(M) \mid L(M) \in \mathcal{P}\}$. Assess whether the language $L_\mathcal{P}$ belongs to the classes REC, RE$\smallsetminus$REC, or else does not belong to RE.

Solution Language $L_\mathcal{P}$ is not in REC. To prove this, we use Rice's theorem and show that property $\mathcal{P}$ is not trivial. First, consider the finite language $L_1 = \{01, 10, 11, 00\}$, which is in RE. Since every string in L_1 has length 2, it follows that the concatenation of every pair of strings from L_1 provides a string of length 4, which is not in L_1. Therefor L_1 has the property $\mathcal{P}$, and thus $\mathcal{P}$ is not empty. Second, consider the finite language $L_2 = \{0, 00, 000\}$, which is in RE. For the pair of strings $0, 00 \in L_2$ we have $0 \cdot 00 = 000 \in L_2$. Therefor L_2 does not have the property $\mathcal{P}$, and thus $\mathcal{P}$ is not the whole class RE. We then conclude that property $\mathcal{P}$ is not trivial.

Consider now the complement of set $\mathcal{P}$ with respect to RE

$$\overline{\mathcal{P}} \quad = \quad \{L \mid L \in \text{RE}, \text{ for some pair } u, v \in L \text{ we have } u \cdot v \in L\}$$

and define $L_{\overline{\mathcal{P}}} = \{\mathsf{enc}(M) \mid L(M) \in \overline{\mathcal{P}}\}$. It is easy to see that $L_{\overline{\mathcal{P}}} = \overline{L_\mathcal{P}}$.

Since $L_\mathcal{P}$ is not in REC, from a theorem of Chapter 9 in the textbook we have that $\overline{L_\mathcal{P}}$ cannot be in REC.

We now argue that $\overline{L_\mathcal{P}}$ belongs to RE. To see this, we can specify 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 three strings $w_1, w_2, w_3 \in \Sigma^*$ and checks that each w_i is in $L(M)$ by simulating the universal TM on input $\mathsf{enc}(M, w_i)$.

- If the previous step terminates and is successful, N tests the equality $w_1 w_2 = w_3$, and answers accordingly. In all other cases, N answers no or runs for ever.

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 Chapter 9 of 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.