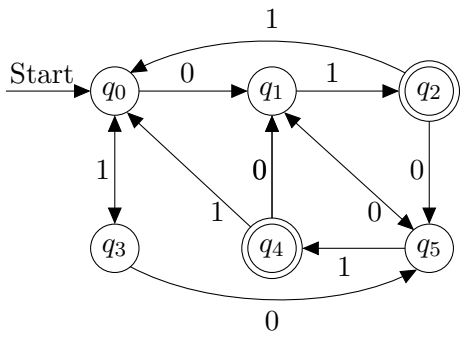
Master Degree in Computer Engineering

Final Exam for Automata, Languages and Computation

February 18th, 2025

1. **[5 points]** Consider the DFA A whose transition function is graphically represented below (arcs with double direction represent two arcs in opposite directions)



   (a) Provide the definition of equivalent pair of states for a DFA.

   (b) Apply to A the tabular algorithm presented in the textbook for detecting pairs of equivalent states, reporting all the **intermediate steps**.

   (c) Specify the minimal DFA equivalent to A and provide a graphical representation of its transition function.

Solution

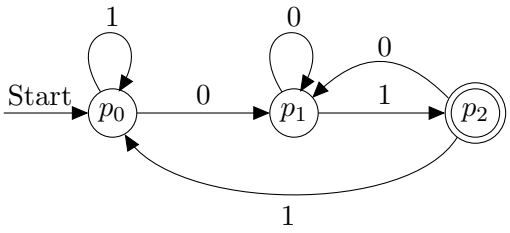
   (a) The required definition can be found in Section 4.4.1 of the textbook.

   (b) The textbook describes an inductive algorithm for detecting distinguishable state pairs. On input A, the algorithm constructs the table reported below.

q_1	Y				
q_2	X	X			
q_3		Y	X		
q_4	X	X		X	
q_5	Y		X	Y	X
	q_0	q_1	q_2	q_3	q_4

We have marked with X the entries in the table corresponding to distinguishable state pairs that are detected in the base case of the algorithm, that is, state pairs that can be distinguished by the string ε. We have then marked with Y distinguishable state pairs detected at the next iteration by some string of length one. At the successive iterations, strings of length larger than one do not provide any new distinguishable state pairs.

(c) From the above table we get the following state equivalences: $q_0 \equiv q_3$, $q_1 \equiv q_5$ and $q_2 \equiv q_4$. This results in three equivalence classes, which becomes the states of the minimal DFA equivalent to A: $p_0 = \{q_0, q_3\}$, $p_1 = \{q_1, q_5\}$, $p_2 = \{q_2, q_4\}$. The minimal DFA has initial state p_0, final state p_2, and the transition function can be graphically represented as



2. **[9 points]** Let $\Sigma = \{a, b, c\}$. For $w \in \Sigma^*$ and $X \in \Sigma$, we write $\#_X(w)$ to denote the number of occurrences of X in w. Consider the following languages

$$\begin{aligned} L_1 &= \{cxcyc \mid x, y \in \{a, b\}^*,\ \#_a(x) = \#_a(y)\} \\ L_2 &= \{cxyc \mid x, y \in \{a, b\}^*,\ \#_a(x) = \#_a(y)\} \end{aligned}$$

State whether each of L_1 and L_2 belongs to REG, to CFL$\setminus$REG, or else whether it is outside of CFL. Provide a mathematical proof for 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 $w = ca^N ca^N c \in L_1$ with $|w| \geq N$.

We now consider all possible factorizations of the form $w = xyz$ satisfying the conditions $|y| \geq 1$ and $|xy| \leq N$ of the pumping lemma. Note that, since $|xy| \leq N$, string y can only span over the substring of w placed to the left of the middle occurrence of c. We need to distinguish two cases in our discussion.

Case 1: y spans the leftmost occurrence of c in w, and possibly more symbols from w. This means that $x = \epsilon$. We then choose $k = 0$ and obtain the string $w_0 = xy^0z = z$ which has fewer than 3 occurrences of symbol c, and therefore $w_0 \notin L_1$.

Case 2: y does not span the leftmost occurrence of c in w. As already observed, y cannot reach the middle occurrence of c, and therefore can only contain occurrences of symbol a. Again, we choose $k = 0$ and obtain a string $w_0 = xy^0z$ which has the form $ca^{N-|y|}ca^N c$. Because of the condition $|y| \geq 1$, we have that $N - |y| < N$, and therefore $w_0 \notin L_1$.

Since we have considered all possible factorizations for string w, and for each one of these we have failed to satisfy the pumping lemma, we must conclude that L_1 cannot be a regular language.

As a second part of the answer, we need to show that L_1 belongs to the class CFL. To do this, we need to provide a CFG or a PDA for L_2. It turns out that the second case is much easier than the first. The transitions of our PDA M can be summarized as follows.

- M starts in its initial state q_0 and with special symbol Z_0 in its stack. If it reads symbol c, M moves to state q_1; if it reads any other symbol from Σ, M moves to a trap state, that is, a non-final state with no further possible transitions.
- In state q_1, M can read any symbol from Σ. If it reads a, then M pushes the special symbol X into the stack; if it reads b, M just moves forward without changing its stack; if it reads c, M switches to state q_2.
- In state q_2, M can read any symbol from Σ. If it reads a with top-most stack symbol X, then M pops X; if it reads a with top-most stack symbol Z_0, then M halts in a trap state; if it reads b, M just moves forward; and if it reads c, M switches to state q_3.
- In state q_3 and with Z_0 at the top of its stack, M moves to final state q_f; in any other case, M moves to the trap state.

It is not too difficult to see that $L(M) = L_2$.

(b) L_2 belongs to the class REG.

To see this, we observe that the definition of L_2 requires that all its string have the form czc with $z \in \{a, b\}^*$, and that z can be factorized as $z = xy$ such that $\#_a(x) = \#_a(y)$. This in turn is equivalent to say that the number of occurrences of symbol a in z is an even number. More formally, we can then write

$$L_2 \quad = \quad \{czc \mid z \in \{a, b\}^*,\ \#_a(z) = 2n,\ n \geq 0\}\ .$$

Then the regular expression $R = \boldsymbol{c}(\boldsymbol{b}^*\boldsymbol{a}\boldsymbol{b}^*\boldsymbol{a}\boldsymbol{b}^*)^*\boldsymbol{c}$ generates L_2.

3. **[6 points]** With reference to the membership problem for context-free languages, answer the following two questions.

(a) Specify the recursive relation underlying the dynamic programming algorithm reported in the textbook for the solution of this problem.

(b) Consider the CFG G in Chomsky normal form defined by the following rules:

$$\begin{aligned} S &\to CD \\ C &\to AC \mid AD \\ D &\to AC \mid AD \mid BD \mid BB \\ A &\to a \\ B &\to b \end{aligned}$$

Assuming as input the CFG G and the string $w = aaabbbb$, trace the application of the algorithm in (a) and assess whether $w \in L(G)$.

Solution

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

(b) On input w and G, the algorithm constructs the table reported below.

$\{S, C, D\}$						
$\{C, D\}$	$\{S, C, D\}$					
$\{C, D\}$	$\{C, D\}$	$\{S, C, D\}$				
	$\{C, D\}$	$\{C, D\}$	$\{D\}$			
		$\{C, D\}$	$\{D\}$	$\{D\}$		
			$\{D\}$	$\{D\}$	$\{D\}$	
$\{A\}$	$\{A\}$	$\{A\}$	$\{B\}$	$\{B\}$	$\{B\}$	$\{B\}$
a	a	a	b	b	b	b

Since the start symbol S belongs to the topmost cell of the triangular table, we can conclude that $w \in L(G)$.

4. **[9 points]** Assess whether the following statements are true or false, providing motivations for all of your answers.

(a) There exist languages L_1 and L_2 in CFL$\setminus$REG, with $L_1 \neq L_2$, such that $L_1 \setminus L_2$ is in REG.

(b) There exist a language L_1 in REG and some language L_2 not in REG such that $L_1 \setminus L_2$ is in CFL$\setminus$REG.

(c) There exist a language L_1 in CFL and a string $u \in L_1$ such that the language $L_2 = \{w \mid w \in L_1,\ w \neq u\}$ is outside of CFL.

(d) Let $\mathcal{P}$ be the class of languages that can be recognized in polynomial time by a TM. There exist a language $L \in \mathcal{P}$ such that L is defined over $\Sigma = \{a, b\}$ and L is not in CFL.

Solution

(a) True. Let $L_1 = \{a^n b^n \mid n \geq 1\}$ and let $L_2 = \{a^n b^n \mid n \geq 2\}$. We know from the textbook that both L_1 and L_2 are in CFL$\setminus$REG. We now have $L_1 \setminus L_2 = \{ab\}$, which is a finite language and therefore a regular language.

(b) True. Let $\Sigma = \{a, b\}$ and let $L = \{a^n b^n \mid n \geq 0\}$. We can choose $L_1 = \Sigma^*$, which is in REG, and $L_2 = \overline{L}$ (the complement of L with respect to Σ^*). The fact that L_2 is not in REG, as required by the hypotheses, follows from the closure property of the class REG under complementation: if L_2 is in REG, then also L would be in REG, but we know from the textbook that is is not the case. We then have

$$L_1 \setminus L_2 = \Sigma^* \setminus \overline{L} = \overline{\overline{L}} = L$$

and we know from the textbook that L is in CFL$\setminus$REG.

(c) False. Let L_1 be an arbitrary language in CFL, and let u be an arbitrary string in L_1. Consider the language $L_3 = \{u\}$. We can then rewrite L_2 as

$$L_2 = L_1 \setminus L_3 = L_1 \cap \overline{L_3} \; .$$

Since L_3 is a finite language, it is also in REG and its complement $\overline{L_3}$ is in REG as well. Since L_1 is in CFL and the class CFL is closed under intersection with REG, we can conclude that L_2 is in CFL as well. Note that we have chosen L_1 and $u \in L_1$ arbitrarily. We can then conclude that the statement is false.

(d) True. Consider the language $L = \{a^n b a^n b a^n \mid n \geq 0\}$, defined over Σ. We know from the textbook that L is not in CFL, since its recognition requires to check that three sequences of symbol a have the same length.

We now show that the language L is in $\mathcal{P}$, by specifying a TM M that works in polynomial time such that $L(M) = L$. M uses a tape with two tracks, and performs the following steps.

- M checks that w has the form $\boldsymbol{a}^*\boldsymbol{b}\boldsymbol{a}^*\boldsymbol{b}\boldsymbol{a}^*$, and rejects if this is not the case.
- M uses the second track to mark occurrences of symbol a in w. More precisely, think of w as containing three blocks of the form $\boldsymbol{a}^*$, separated by the two occurrences of symbol b. M marks the first occurrences of a in each of the three blocks, then it marks the second occurrences of a in each of the three blocks, and so on. If all of the three blocks get completely marked at the same step, then M accepts w; otherwise M rejects w.

It is not difficult to see that M works in polynomial time in the length of w.

5. **[4 points]** Prove the following statements, using the same proofs reported in the textbook.

(a) If language L is in REC, then also the language $\overline{L}$ is in REC.

(b) If both languages L and $\overline{L}$ are in RE, then then L is in REC.

Solution The required proofs are reported in Section 9.2.2 of the textbook.