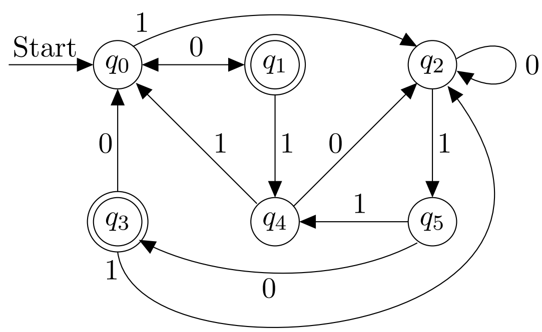
Master Degree in Computer Engineering

Final Exam for Automata, Languages and Computation

February 22nd, 2024

1. **[5 points]** Consider the DFA A whose transition function is graphically depicted 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 from the textbook for detecting pairs of equivalent states, reporting **all the intermediate steps**.

   (c) Specify the minimal DFA equivalent to A.

Solution

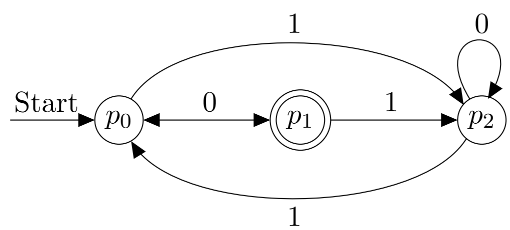
   (a) The required definition can be found in Section 4.4.1 of the textbook.

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

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

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: $q_0 \equiv q_5$, $q_1 \equiv q_3$ and $q_2 \equiv q_4$. This results in three equivalence classes, which becomes the states of the minimal DFA equivalent to A: $p_0 = \{q_0, q_5\}$, $p_1 = \{q_1, q_3\}$, $p_2 = \{q_2, q_4\}$. The minimal DFA is then



2. **[7 points]** Consider the following languages, defined over the alphabet $\Sigma = \{a, b\}$:

$$L_1 = \{xby \mid x, y \in \Sigma^*,\ |x| = |y| = n,\ n \geq 0\}$$
$$L_2 = \{xby \mid x, y \in \Sigma^*,\ |x| + |y| = 2n,\ n \geq 0\}$$

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

Solution

(a) A careful analysis of the definition of L_1 reveals that this language contains only strings of odd length greater equal than one, under te condition that the central symbol is an occurrence of b. 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 $w = a^N b a^N \in L_1$ 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 from the part of w to the left of the central b.

According to the pumping lemma, the string $w_k = xy^kz$ should be in L_1 for every $k \geq 0$. Let $|y| = m \geq 1$ and consider $k = 2$. We have $w_2 = a^{N+m} b a^N$. From $m \geq 1$, we have $N + m > N$ and the only occurrence of b in w_2 is no longer placed in the central position of the string, assuming that w_2 has odd length, which is not even guaranteed. We thus conclude that $w_2 \notin L_1$, against the statement of the pumping lemma, and L_1 is not a regular language.

We now show that L_1 belongs to the class CFL. Consider the CFG G with productions:

$$S \to TST \mid b$$
$$T \to a \mid b$$

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

(b) Similarly to L_1, the strings in L_2 have all odd length greater equal than one, with at east one occurrence of symbol b. However, this occurrence needs not be placed in the central position of the string, since $|x|$ and $|y|$ need no longer be equal. For this reason, L_2 is a regular language. It is not difficult to see that L_2 can be generated by the following regular expression

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

An alternative solution to this question, perhaps simpler than the previous solution, can be obtained as follows. Consider the language L_2' containing all and only the strings over Σ of odd length greater equal than one. Furthermore, consider the language L_2'' containing all and only the strings over Σ with at least one occurrence of b. It is not difficult to see that L_2' and L_2'' are both regular languages, and that $L_2 = L_2' \cap L_2''$. Since we know that the intersection of two regular languages is still a regular language, we can conclude that L_2 is a regular language as well.

3. **[5 points]** Provide the construction we have seen in the context-free lectures that converts a regular expression E into a CFG G such that $L(E) = L(G)$. The construction uses structural induction on E.

Solution The required construction is reported in the slide number 62 of the lecture on context-free grammars (file `05_context_free_grammars` in the moodle page of the course).

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

(a) If the complement of a language L is finite, then L is in REG (the class of regular languages).

(b) If the language $L_1 \cup L_2$ is in REG, then L_1, L_2 are both in REG.

(c) If a CFG G has only one variable, then $L(G)$ is in REG.

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

Solution

(a) True. Let $\overline{L}$ be the complement of L. Since $\overline{L}$ is finite, $\overline{L}$ is also a regular language. We know that regular languages are closed under complementation, therefore $\overline{\overline{L}}$ must be a regular language as well. We now observe that $\overline{\overline{L}} = L$.

(b) False. Consider as a counterexample the language $L_1 = \{a^n b^n \mid n \geq 0\}$ and the language $L_2 = \overline{L_1}$, where the complement is defined with respect to the alphabet $\Sigma = \{a, b\}$. We have $L_1 \cup L_2 = \Sigma^*$, which is a regular language. But we know that L_1 is not in REG.

(c) False. As a counterexample consider the CFG with a single variable S, consisting of the rules $S \rightarrow aSb$ and $S \rightarrow \epsilon$, generating the language $\{a^n b^n \mid n \geq 0\}$ which is not in REG.

(d) True. Consider two arbitrary languages L_1 and L_2 in $\mathcal{P}$. By the definition of the class $\mathcal{P}$, there exist TMs M_1 and M_2 that run in polynomial time and such that $L(M_1) = L_1$ and $L(M_2) = L_2$. We then construct a TM M_c that works as follows.

- Given an input string w, M runs a loop for $i = 0, ..., |w|$ executing the following actions.

  - M factorizes w into $w = uv$ with $|u| = i$ and $|v| = |w| - i$.
  - M simulates M_1 on u and M_2 on v
  - If both $u \in L(M_1)$ and $v \in L(M_2)$, then M stops in a final state.
- If M completes the loop without any break, then it stops in a non-final state.

It is not difficult to see that $L(M) = L_1 L_2$ and that M works in polynomial time. We therefore conclude that $\mathcal{P}$ is closed under concatenation.

5. **[5 points]** Introduce the definition of multi-tape TM. Highlight the basic idea behind the proof of equivalence between multi-tape TM and standard TM, discussing also the total computation time of the construction.

Solution

The required definition and proof, along with the discussion of computation time, can be found in Section 8.4.1 of the textbook.

6. **[7 points]** With reference to TMs, answer to the following questions.

(a) Consider the following languages

$$L_1 = \{\mathsf{enc}(M) \mid L(M) \text{ contains exactly 5 strings}\}$$
$$L_2 = \{\mathsf{enc}(M) \mid M \text{ contains exactly 5 tape symbols}\}$$

State whether the above languages are in the class REC, and provide a mathematical proof for your answers.

(b) Consider the language

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

where $\mathsf{enc}(M_1, M_2)$ is some encoding of TMs M_1 and M_2. Using an appropriate reduction, show that L_3 cannot be in RE.

Solution

(a) Language L_1 is not in REC. To show this, consider the following property of the RE languages

$$\mathcal{P} \;=\; \{L \mid L \in \text{RE}, \ |L| = 5\}$$

and observe that $L_1 = L_{\mathcal{P}} = \{\mathsf{enc}(M) \mid L(M) \in \mathcal{P}\}$. We can now apply Rice's theorem and show that property $\mathcal{P}$ is not trivial.

- $\mathcal{P} \neq \emptyset$. Consider the language $L = \{\epsilon, 0, 1, 01, 10\}$ which is in RE, and observe that $L \in \mathcal{P}$.
- $\mathcal{P} \neq \text{RE}$. Consider the language $\{0,1\}^*$ which is in RE, and observe that $L \notin \mathcal{P}$.

In contrast, language L_2 is in REC. In fact, it is not difficult to devise a TM for L_2 that takes as input a string $\mathsf{enc}(M)$, verifies that it properly encodes a TM M, and checks that the tape alphabet of M contains exactly 5 symbols.

(b) To prove that L_3 is not in RE, we establish a reduction $L_e \leq_m L_3$. The reduction takes as input a string $\mathsf{enc}(M)$ and produces as output a string $\mathsf{enc}(M_1, M_2)$, where $M_1 = M$ and M_2 is a TM such that $L(M_2) = \Sigma^*$.

We now show that the proposed mapping represents a valid reduction, that is, $\mathsf{enc}(M) \in L_e$ if and only if $\mathsf{enc}(M_1, M_2) \in L_3$.

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