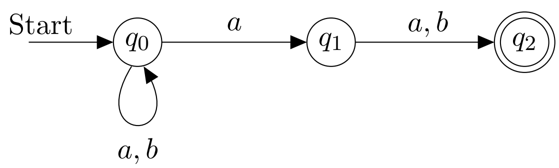
Master Degree in Computer Engineering

Final Exam for
Automata, Languages and Computation

January 21st, 2025

1. **[6 points]** Assume the NFA N whose transition function δ_N is graphically represented below.



Answer the following questions.

(a) The textbook defines the extended transition function $\hat{\delta}_N$ as

   i. base: $\hat{\delta}_N(q, \epsilon) = \{q\}$

   ii. induction: $\hat{\delta}_N(q, xa) = \cup_{p \in \hat{\delta}_N(q,x)} \delta_N(p, a)$

   Assess whether the string $abab$ belongs to the language $L(N)$ by computing the value of $\hat{\delta}_N(q_0, abab)$. Report all of the **intermediate steps**.

(b) Transform N into an equivalent deterministic finite automaton D, with transition function δ_D, by applying the subset construction together with the lazy evaluation. Depict the graphical representation of the function δ_D.

Solution

(a) This amounts to tracing each step of the computation of N on the input string $abab$

- $\hat{\delta}_N(q_0, \epsilon) = \{q_0\}$
- $\hat{\delta}_N(q_0, a) = \cup_{p \in \{q_0\}} \delta_N(p, a) = \delta_N(q_0, a) = \{q_0, q_1\}$
- $\hat{\delta}_N(q_0, ab) = \cup_{p \in \{q_0, q_1\}} \delta_N(p, b) = \delta_N(q_0, b) \cup \delta_N(q_1, b) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
- $\hat{\delta}_N(q_0, aba) = \cup_{p \in \{q_0, q_2\}} \delta_N(p, a) = \delta_N(q_0, a) \cup \delta_N(q_2, a) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\hat{\delta}_N(q_0, abab) = \cup_{p \in \{q_0, q_1\}} \delta_N(p, b) = \delta_N(q_0, b) \cup \delta_N(q_1, b) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

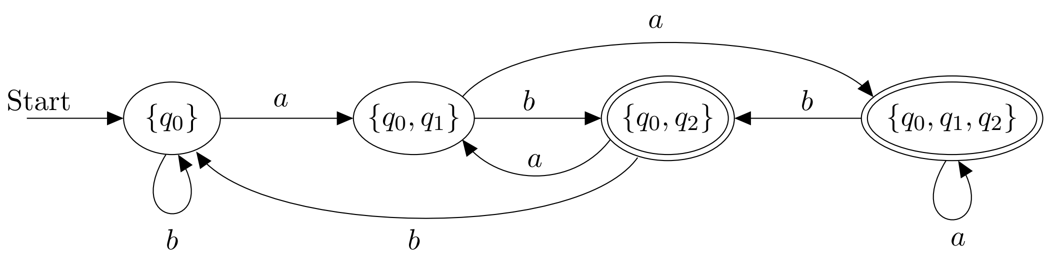
Since q_2 is a final state for N, we conclude that the input string $abab$ is accepted by N.

(b) Recall that the states of D are subsets of the states of N. In addition, the lazy evaluation prescribes that we apply the subset construction only to those states of D that are accessible from the initial state of D.

According to the subset construction, the initial state of D is $\{q_0\}$. Starting from $\{q_0\}$, the transition function δ_D can be obtained as follows

- $\delta_D(\{q_0\}, a) = \{q_0, q_1\}$, $\delta_D(\{q_0\}, b) = \{q_0\}$
- $\delta_D(\{q_0, q_1\}, a) = \{q_0, q_1, q_2\}$, $\delta_D(\{q_0, q_1\}, b) = \{q_0, q_2\}$
- $\delta_D(\{q_0, q_2\}, a) = \{q_0, q_1\}$, $\delta_D(\{q_0, q_2\}, b) = \{q_0\}$
- $\delta_D(\{q_0, q_1, q_2\}, a) = \{q_0, q_1, q_2\}$, $\delta_D(\{q_0, q_1, q_2\}, b) = \{q_0, q_2\}$

The final states of D are $\{q_0, q_2\}$ and $\{q_0, q_1, q_2\}$. The graph representation of the function δ_D is reported below



2. **[8 points]** Let R represent the string reversal operator, which we extend to languages as usual. Consider the following languages, defined over the alphabet $\Sigma = \{a, b\}$:

$$\begin{aligned} L_1 &= \{a^m a^n bba^n \mid m, n \geq 1\} \\ L_2 &= \{ba^m a^n a^n b \mid m, n \geq 1\} \\ L_3 &= L_2 \cdot L_2^R \end{aligned}$$

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

Solution

(a) L_1 belongs to the class CFL$\smallsetminus$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 $w = a^{N+1}bba^N \in L_1$ 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. Since $|xy| \leq N$, string y can only span over the occurrences of a placed at the left of bb, therefore we need to consider only one case in our discussion.

We choose $k = 0$ in the pumping lemma, and obtain the new string $w_{k=0} = xy^0z = xz$, which has the form $a^{N+1-|y|}bba^N$. Since $|y| \geq 1$, the number of occurrences of symbol a to the left of bb is smaller or equal to the number of occurrences of symbol a to the right of bb, thus violating the definition of L_1. We conclude that L_1 does not satisfy the pumping lemma, and therefore cannot be a regular language.

As a second part of the answer, we need to show that L_1 belongs to the class CFL. Consider the CFG G_1 with productions:

$$\begin{aligned} S &\rightarrow aS \mid aB \\ B &\rightarrow aBa \mid abba \end{aligned}$$

It is not too difficult to see that $L(G_1) = L_1$.

(b) L_2 belongs to the class REG.

To see this, we observe that we can rewrite the definition of this language as $L_2 = \{ba^m a^{2n} b \mid m, n \geq 1\}$. Then the regular expression $R = \boldsymbol{baa^*aa(aa)^*b}$ generates L_2.

(c) L_3 belongs to the class REG.

We have already proved that L_2 is in REG. We know from the textbook that the class REG is closed under the reversal operator R. Therefore L_2^R must be in REG. Finally, we know from the textbook that the class REG is closed under concatenation. Therefore $L_2 L_2^R = L_3$ must be in REG as well.

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

Solution

The required construction is reported in Theorem 6.9 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_1 be the complement of a finite language and let L_2 be a language in CFL. Then the language $L_1 \cap L_2$ is always in CFL.

(b) Let R represent the string reversal operator, which we extend to languages as usual. There exists languages L_1 and L_2 both in REG such that $L_1 L_2^R$ is in CFL$\smallsetminus$REG.

(c) There exists languages L_1 and L_2 both in CFL$\smallsetminus$REG such that $L_1 \smallsetminus L_2$ is in REG.

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

Solution

(a) True. Every finite language is also in REG, and the class REG is closed under complementation. Therefore L_1 must be in REG. The statement follows from the fact that CFL is close under intersection with REG.

(b) False. We know from the textbook that the class REG is closed under the reversal operator as well as under the concatenation operator. Then the language $L_1 L_2^R$ is always in REG.

(c) True. Let $L_1 = \{a^n b^n \mid n \geq 0\}$ and let $L_2 = \{a^n b^n \mid n \geq 1\}$. We know from the textbook that both L_1 and L_2 are in CFL$\smallsetminus$REG. We now have $L_1 \smallsetminus L_2 = \{\epsilon\}$, which is a regular language.

(d) True. Let L_1 and L_2 be two arbitrary languages in $\mathcal{P}$. From the definition of $\mathcal{P}$, there exist TMs M_1 and M_2, both running in polynomial time, such that $L(M_1) = L_1$, and $L(M_2) = L_2$.

Consider the Turing machine M_3 defined as follows.

- Given as input a string w, M_3 simulates M_1 and M_2 on w;

- If M_1 accepts and M_2 rejects, then M_3 accepts. Otherwise, M_3 rejects.

It is immediate to see that $L(M_3) = L_1 \smallsetminus L_2$. Furthermore, since both M_1 and M_2 run in polynomial time and are simulated only once, M_3 also runs in overall polynomial time.

5. [**8 points**] In relation to the theory of Turing machines (TMs), answer the following questions. All the TMs introduced below are defined over the input alphabet $\Sigma = \{0, 1\}$.

For a string $w \in \Sigma^*$ and a symbol $X \in \Sigma$, we write $\#_X(w)$ to represent the number of occurrences of X in w. We define $L_c = \{w \mid w \in \Sigma^*,\ \#_0(w) = \#_1(w)\}$. Consider the following property of the RE languages

$$\mathcal{P} \quad = \quad \{L \mid L \in \text{RE},\ L \subseteq L_c\}$$

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

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

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

(c) For TMs M_1, M_2 and M_3, we write $\mathsf{enc}(M_1, M_2, M_3)$ to represent some fixed binary encoding of these machines. Consider the language

$$L \quad = \quad \{\mathsf{enc}(M_1, M_2, M_3) \mid L(M_1) \subseteq L(M_2) \subseteq L(M_3)\}\ .$$

Show that L is not in RE by establishing a reduction $L_{\mathcal{P}} \leq_m L$.

Solution

(a) We need to show that the property $\mathcal{P}$ is not trivial, that is, $\mathcal{P}$ is neither empty nor equal to RE. First, we observe that the language L_c is context-free, since a PDA can easily recognize it. Therefore L_c is also in RE. It is immediate to see that $L_c \in \mathcal{P}$; therefore $\mathcal{P}$ is not empty. Second, consider the string $w = 010$, $w \notin L_c$. The language $\{w\}$ is finite and therefore also in RE. It is immediate to see that $\{w\} \notin \mathcal{P}$; therefore $\mathcal{P}$ is not equal to RE. We can now apply Rice's theorem and conclude that, since $\mathcal{P}$ is not trivial, $L_{\mathcal{P}}$ is not in REC.

(b) We now show that $L_{\mathcal{P}}$ is not in RE. The most convenient way to do this is to consider the complement language $\overline{L_{\mathcal{P}}} = L_{\overline{\mathcal{P}}}$, where $\overline{\mathcal{P}}$ is the complement of the class $\mathcal{P}$ with respect to RE and can be specified as

$$\overline{\mathcal{P}} \quad = \quad \{L \mid L \in \text{RE},\ \text{there exists a string } w \in L \text{ such that } w \notin L_c\}\ .$$

We specify a nondeterministic TM N such that $L(N) = L_{\overline{\mathcal{P}}}$. Since every nondeterministic TM can be converted into a standard TM, this shows that $L_{\overline{\mathcal{P}}}$ is in RE. Our nondeterministic TM N takes as input the encoding $\mathsf{enc}(M)$ of a TM M and performs the following steps.

- N nondeterministically guesses a string $w \in \Sigma^*$ and checks that $w \in L_c$ by counting the occurrences of 0 and the occurrences of 1 in w.
- N simulates M on w. If this computation terminates with a positive answer, then N accepts and halts. If the computation terminates with a negative answer, then N does not accept and halts. Finally, if the simulation of M on w does not halt, then N runs for ever and therefore does not accept its input.

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

Since $L_{\overline{\mathcal{P}}}$ 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 in (a) that $L_{\mathcal{P}}$ is not in REC. We must therefore conclude that $L_{\mathcal{P}}$ is not in RE.

(c) Recall from the theory of TM that, in order to provide a reduction $L_{\mathcal{P}} \leq_m L$, we need to establish a mapping m from input instances of $L_{\mathcal{P}}$ to output instances of L such that positive instances are mapped to positive instances and negative instances are mapped to negative instances. From a known theorem about reductions, since $L_{\mathcal{P}}$ is not in RE then L cannot be in RE as well.

We need to map strings of the form $\mathsf{enc}(M)$ into strings of the form $\mathsf{enc}(M_1, M_2, M_3)$. As already observed, L_c is in CFL and therefore in RE. Then there must be some TM M_c such that $L(M_c) = L_c$. Let also $M_\emptyset$ be some TM such that $L(M_\emptyset) = \emptyset$. We then set $M_1 = M_\emptyset$, $M_2 = M$, and $M_3 = M_c$.

To conclude the proof, we now show the desired relation between the mapped instances, by means of the following chain of logical equivalences:

$$
\begin{array}{lll}
\mathsf{enc}(M) \in L_{\mathcal{P}} & \text{iff} \quad L(M) \in \mathcal{P} & \text{(definition of } L_{\mathcal{P}}) \\
& \text{iff} \quad L(M) \subseteq L_c & \text{(definition of } \mathcal{P}) \\
& \text{iff} \quad \emptyset \subseteq L(M) \subseteq L_c & \text{(from set theory)} \\
& \text{iff} \quad L(M_1) \subseteq L(M_2) \subseteq L(M_3) & \text{(definition of our reduction)} \\
& \text{iff} \quad \mathsf{enc}(M_1, M_2, M_3) \in L & \text{(definition of } L) \,.
\end{array}
$$