CSL253 - Theory of Computation

Tutorial 7

Team Members

1. Arpit Bhomia - 12340340
2. Ashish Ranjan - 12340370
3. Kethu Shanmukha Reddy - 12341160

Question 13

Prove that $\mathrm{EQ}_{\mathrm{DFA}_L}$ is decidable by testing the two DFAs on all strings up to a certain size (called limit DFA).

Solution

1. Problem Understanding

We are given the language:

$$\text{EQ}_{\text{DFA}_L} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$$

Our goal is to prove that this language is decidable by testing A and B on all strings up to a certain length — this method is known as the **Limit DFA Method**.

2. Key Idea: The Limit Theorem

Let A and B be DFAs with n_1 and n_2 states respectively. Then: - If $L(A) \neq L(B)$, there exists a string w such that $w \in L(A) \triangle L(B)$ (i.e., w is accepted by one but not the other). - Such a distinguishing string w exists with $|w| < n_1 \cdot n_2$.

Therefore, to check whether $L(A) = L(B)$, it is enough to test all strings w with length less than $n_1 \cdot n_2$.

3. Step-by-Step Decidability Algorithm (Limit DFA Method)

Step 1: Get Number of States

Let:

$$|Q_A| = n_1, \quad |Q_B| = n_2$$

These are the number of states in DFA A and B, respectively.

Step 2: Generate All Strings Up to Length $(n_1 \cdot n_2 - 1)$

Enumerate all strings $w \in \Sigma^*$ such that:

$$|w| < n_1 \cdot n_2$$

Step 3: Compare Outputs of A and B

For each string w, simulate both A and B on input w:

- If A and B both accept or both reject w, continue.
- If one accepts and the other rejects, then $L(A) \neq L(B)$. **Reject**.

Step 4: Accept if No Differences Found

If for all strings w with $|w| < n_1 \cdot n_2$, the outputs of A and B are identical, then:

$$L(A) = L(B)$$

Accept.

4. Final Algorithm

Given DFAs A and B:

1. Let $n_1 = |Q_A|$, $n_2 = |Q_B|$.
2. Let $k = n_1 \cdot n_2$.
3. For all strings $w \in \Sigma^*$ where $|w| < k$:
   - Simulate A and B on w.
   - If outputs differ, reject.
4. If all outputs match, accept.

5. Conclusion

The language $\mathrm{EQ}_{\mathrm{DFA}_L}$ is decidable because:

1. We only need to compare A and B on finitely many strings — those of length less than $n_1 \cdot n_2$.
2. Simulation of DFAs is efficient and decidable.
3. If no distinguishing string is found, the DFAs must accept the same language.

Hence, there exists a Turing machine that decides whether two DFAs accept the same language using the Limit DFA Method.

Question 23

Let $\text{PAL}_{\text{DFA}} = \{\langle M \rangle \mid M \text{ is a DFA that accepts some palindrome}\}$. Show that PAL_{DFA} is decidable.

Solution

1. Problem Understanding

We are given the language:

$$\text{PAL}_{\text{DFA}} = \{\langle M \rangle \mid M \text{ is a DFA and } L(M) \cap \text{PAL} \neq \emptyset\}$$

Where **PAL** is the set of all palindromes over the alphabet Σ. Our goal is to show that PAL_{DFA} is decidable, i.e., there exists a Turing machine that decides whether a given DFA M accepts at least one palindrome.

2. Key Idea

The key idea is to use the fact that: - The set of palindromes, PAL, is a context-free language (CFL). - The intersection of a regular language (DFA) and a context-free language is a context-free language. - Emptiness for CFLs (given by a CFG or PDA) is decidable.

We can construct a context-free language $L = L(M) \cap \text{PAL}$ and test whether L is empty. If it is not empty, then M accepts some palindrome.

3. Step-by-Step Decidability Algorithm

Step 1: Represent the Set of Palindromes

Let PAL be the language:

$$\text{PAL} = \{w \in \Sigma^* \mid w = w^R\}$$

This language is context-free. There exists a CFG G_{PAL} that generates all palindromes.

Step 2: Construct CFG for the Intersection

Let M be a DFA. We know that for any DFA M and any CFG G, there is a construction to obtain a new CFG G' such that:

$$L(G') = L(M) \cap L(G)$$

So, construct a CFG G' such that:

$$L(G') = L(M) \cap \text{PAL}$$

Step 3: Check Emptiness of CFG

Use the decidable procedure for the emptiness problem of context-free grammars:

$$\text{Is } L(G') = \emptyset?$$

This problem is decidable. So: - If $L(G') = \emptyset$, reject. - If $L(G') \neq \emptyset$, accept.

4. Final Algorithm

Given a DFA M, do the following:

1. Construct a CFG G_{PAL} that generates all palindromes.
2. Construct a CFG G' for the intersection $L(M) \cap L(G_{PAL})$.
3. Test if $L(G') = \emptyset$:
   - If yes, reject (i.e., M accepts no palindrome).
   - If no, accept (i.e., M accepts at least one palindrome).

5. Conclusion

The language PAL_{DFA} is decidable because:

1. We can construct a CFG for palindromes.
2. We can compute the intersection of a DFA and a CFG.
3. We can decide if the resulting CFG is empty.

Hence, there exists a Turing machine that decides whether a DFA accepts some palindrome.