CSL253 - Theory of Computation

Tutorial 7

Team Members

1. Arpit Kumar - 12340350
2. Saurav Gupta - 12341940
3. Keshav Mishra - 12341140

Question 12

Let $A = \{< R > |\ R$ is a regular expression describing a language containing at least one string w that has 111 as a substring (i.e., $w = x111y$ for some x and $y)\}$.*Show* that A is decidable.

Question 22

Answer the following:

1. Let $\text{BALN}_{\text{DFA}} = \{\langle M \rangle \mid M$ is a DFA that accepts some string with an unequal number of 0s and 1s$\}$. Is BAL_{DFA} decidable? Justify your answer.
2. Comment on the closure properties of Turing Machines in the context of the above problem.

Solution Question 12:

Intuition :

- We know that the set of all strings containing "111" as a substring is a regular language.
- The language of regular expression R is also regular.
- If their intersection is nonempty, it means R describes some string containing the substring "111" in it.
- Checking the emptiness of a regular language is a standard, decidable procedure.

The decidability of the language A is proved as follows

- We define a language S such that

$$S = \{w \in \Sigma^* \mid w \text{ consists 111 as a substring}\}.$$

- The regular expression (RE) for the language S is

$$(0 \cup 1)^*111(0 \cup 1)^*.$$

  Therefore, S is a regular language.

- The DFA D_S for the language S is shown below:

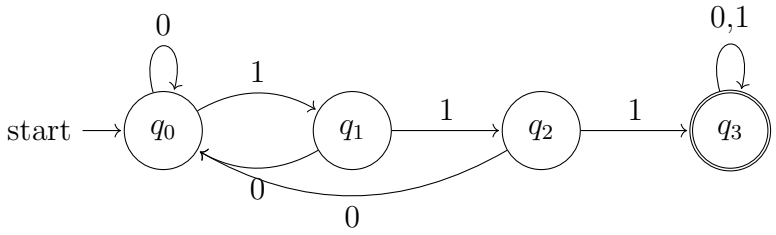


Figure 1: DFA for S: strings containing "111".

Decidability Argument:

- Let R be a regular expression on input alphabet Σ, and $L(R)$ be the language described by R.
- If $S \cap L(R) \neq \emptyset$, then R produces a string containing 111 as a substring. Thus, $\langle R \rangle \in A$.
- Similarly, if $S \cap L(R) = \emptyset$ then R does not produce any string that contains 111. Thus, $\langle R \rangle$ does not belong to A.

- Therefore:
$$\begin{cases} \text{If } L(R) \cap S \neq \emptyset, & \text{then } \langle R \rangle \in A, \\ \text{If } L(R) \cap S = \emptyset, & \text{then } \langle R \rangle \notin A. \end{cases}$$

- We observe the following characterization of membership in A:
$$\langle R \rangle \in A \iff L(R) \cap S \neq \emptyset,$$
where $S = \{w \in \Sigma^* \mid 111 \text{ is a substring of } w\}$.

- Since $L(R)$ is described by regular language, $L(R)$ is a regular language. Both S and $L(R)$ are regular languages.

- Now, $S \cap L(R)$ is regular because regular languages are closed under intersection. Thus, $S \cap L(R)$ has some DFA $D_{S \cap L(R)}$.

- Deciding whether the DFA $D_{S \cap L(R)}$ accepts any string is the same as deciding whether its language is empty or not. This is exactly the problem of deciding E_{DFA}.

- Now we know that,
$$E_{\text{DFA}} = \{\langle K \rangle \mid K \text{ is a DFA with } L(K) = \emptyset\}$$
is decidable(Theorem 4.4). Thus, there exists a Turing Machine TM which determines E_{DFA}.

- Therefore, We can relate a TM to $D_{S \cap L(R)}$ to determine if
$$L(R) \cap S \neq \emptyset.$$

Steps involved

To decide whether a regular expression R generates any string that contains the substring `111`, the following steps are performed:

1. **Convert the regular expression R into a DFA D_R.** We can first convert any regular expression into an equivalent NFA. Then, we can convert that NFA into a DFA using the subset construction method. Thus, we obtain a DFA D_R that accepts the same language as the original regular expression R.

2. **Construct a DFA for the intersection language $S \cap L(R)$.** We already have a DFA D_S for the language $S = \{w \in \Sigma^* \mid 111 \text{ is a substring of } w\}$. Now, using D_S and D_R, we build a new DFA $D_{S \cap L(R)}$ that accepts only those strings that are in both S and $L(R)$. We can construct this as DFAs are closed under intersection.

3. **Run a Turing machine T that decides the problem $E_{\mathbf{DFA}}$.** This Turing machine checks whether the language of $D_{S\cap L(R)}$ is empty or not. The input to T is the encoding $\langle D_{S\cap L(R)}\rangle$.

4. **Make the final decision based on the output of T.**

   - If T accepts (i.e., the language is empty), then reject — this means R does not generate any string containing `111`.
   - If T rejects (i.e., the language is not empty), then accept — this means R does generate at least one string containing `111`.

Summarization of the above discussion contributes the subsequent Turing machine T to decide A:

$T =$ "On input $\langle R\rangle$, where R is a regular expression:

- Transform R into a DFA D_R.
- Build a DFA $D_{S\cap L(R)}$ for the language $S \cap L(R)$ from the DFAs D_S and D_R.
- Run TM T that decides E_{DFA} on input $\langle D_{S\cap L(R)}\rangle$.
- If T accepts, reject. If T rejects, accept.

The Turing machine T decides A. Therefore, the language A is decidable.

Solution Question 22:

1 Problem Statement

Let BALN = $\{\langle M \rangle \mid M$ is a DFA that accepts some string containing an unequal number of 0s and 1s$\}$. Prove that BALNDFA is decidable.

2 Solution

2.1 Understanding the Problem

We need to determine whether a given DFA M accepts at least one string that has an unequal number of 0s and 1s. This requires:

1. Characterizing strings with unequal numbers of 0s and 1s
2. Determining if the intersection of this language with $L(M)$ is non-empty

2.2 Step 1: Characterize the Language of Unbalanced Strings

Let $L_{\text{unbal}} = \{w \in \{0,1\}^* \mid \text{number of 0s} \neq \text{number of 1s}\}$.

This language is context-free, and can be generated by the grammar:

$$
\begin{aligned}
S &\to P \mid Q \\
P &\to XAX \mid PP \\
Q &\to XBX \mid QQ \\
X &\to 0X1 \mid 1X0 \mid XX \mid \varepsilon \\
A &\to 0A \mid 0 \\
B &\to 1B \mid 1
\end{aligned}
$$

Where:

- S generates all unbalanced strings
- P generates strings with more 0s than 1s
- Q generates strings with more 1s than 0s
- X generates balanced strings (equal number of 0s and 1s)
- A generates sequences of only 0s
- B generates sequences of only 1s

This grammar can be used to construct a pushdown automaton (PDA) P that recognizes L_{unbal}.

2.3 Step 2: Construct a Decider for BALNDFA

We build a Turing machine M_{BALNDFA} to decide BALNDFA:

1. Let P be the PDA for L_{unbal}, which recognizes all strings with an unequal number of 0s and 1s. This PDA can be constructed from the context-free grammar for L_{unbal}.
2. On input $\langle \text{B} \rangle$, where B is a DFA, use B and P to construct a new PDA R that recognizes the intersection of the languages of B and P, $L(B) \cap L(P)$. This construction is possible because the intersection of a regular language (recognized by B) and a context-free language (recognized by P) is context-free.
3. Check if $L(R) = \emptyset$ using the algorithm for testing emptiness of CFLs. This can be done by converting the PDA to a CFG and checking if the start symbol can derive any terminal string.
4. If $L(R) = \emptyset$, reject. This means the DFA B does not accept any strings with an unequal number of 0s and 1s.
5. If $L(R) \neq \emptyset$, accept. This means the DFA B accepts at least one string with an unequal number of 0s and 1s

Construction of PDA R for $L(R) = L(B) \cap L(P)$

Proposition *If L is a CFL and R is a regular language then $L \cap R$ is a CFL.*

Proof. Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

- The stack of P' is the stack of P
- The state of P' at any time is the pair (state of P, state of M)
- These determine the transition function of P'
- The final states of P' are those in which both the state of P and state of M are accepting.

More formally, let $M = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be a DFA such that $\mathcal{L}(M) = R$, and $P = (Q_2, \Sigma, \Gamma, \delta_2, q_2, F_2)$ be a PDA such that $\mathcal{L}(P) = L$. Then consider $P' = (Q, \Sigma, \Gamma, \delta, q_0, F)$ such that

- $Q = Q_1 \times Q_2$
- $q_0 = (q_1, q_2)$
- $F = F_1 \times F_2$

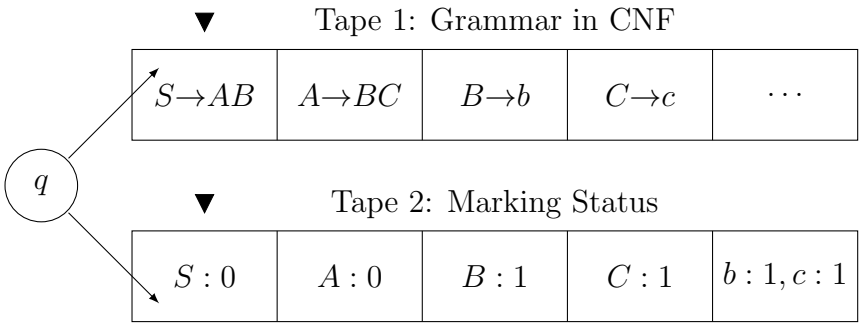
$$\delta((p,q),x,a) = \begin{cases} \{((p,q'),b) \mid (q',b) \in \delta_2(q,x,a)\} & \text{when } x = \epsilon \\ \{((p',q'),b) \mid p' = \delta_1(p,x) \text{ and } (q',b) \in \delta_2(q,x,a)\} & \text{when } x \neq \epsilon \end{cases}$$

2.4 Emptiness of Context Free Language

Proof. To determine if $L(G) = \emptyset$ for a CFG G, we implement a two-tape Turing Machine that:

1. Converts the grammar G to Chomsky Normal Form on tape 1.
2. On tape 2, maintains marking status of terminals and variables:
   (a) Mark all terminal symbols in the transformed grammar.
   (b) Iteratively mark any variable A that has a production $A \to \alpha$ where all symbols in α are already marked.
   (c) Repeat until no more variables can be marked.
3. Accept if the start variable is marked, otherwise reject.

Two-Tape TM for Testing CFG Emptiness



□

2.5 Step 4: Closure Properties

Several important closure properties and decidability results are used:

Lemma 1. *The intersection of a regular language and a context-free language is context-free.*

Lemma 2. *Given a DFA B and a PDA P, we can effectively construct a PDA R such that $L(R) = L(B) \cap L(P)$.*

This construction works by simulating both the DFA and PDA simultaneously, accepting only if both would accept.

Lemma 3. *Testing whether a context-free language is empty is decidable.*

This can be done by checking if the start symbol can derive any string, using algorithms for eliminating useless symbols in a context-free grammar.

3 Conclusion

BALNDFA is decidable because:

1. We can construct a PDA for the language of strings with an unequal number of 0s and 1s
2. We can effectively compute the intersection of a DFA and a PDA
3. We can decide emptiness for context-free languages

The algorithm terminates on all inputs and correctly determines whether a given DFA accepts at least one string with an unequal number of 0s and 1s.