CSL253 - Theory of Computation

Tutorial 7

Team Members

1. V.Suneetha - 12342290
2. Harshitha - 12342310
3. Hansika - 12340870

Question 14

A language C is Turing recognizable if and only if there exists a decidable language D such that for all strings x:

$$x \in C \iff \exists y \, \langle x, y \rangle \in D$$

This means C is recognizable exactly when membership can be verified through some witness y where $\langle x, y \rangle$ belongs to a decidable language D.

Key Components

Turing Recognizable (C)

A language for which a TM can accept all members (but may loop on non-members)

Decidable Witness (D)

A helper language where:

- y serves as a verifiable "proof certificate" for $x \in C$
- D must be decidable (we can algorithmically verify proofs)

Proof Structure

Forward Direction ($\Rightarrow$)

1. Given recognizer M for C
2. Define D using accepting computation histories of M
3. Show D is decidable and correctly represents C

Reverse Direction ($\Leftarrow$)

1. Given decidable D
2. Construct recognizer for C by searching for valid y proofs
3. Show it correctly recognizes C

Solution(forward Direction)

Assume C is Turing recognizable.
That means: There is a TM M such that:

- If $x \in C$, M will accept after some finite time.
- If $x \notin C$, M may reject or run forever.

We define a new language D where:
$\langle x, y \rangle \in D$ iff y is a valid accepting computational proof that shows M accepts x.
$\langle x, y \rangle \Rightarrow$ basically a claim & its justification.
What's y?

- It's a string that encodes the sequence of machine configurations showing M accepts x.

Example:
$y = C_0 \# C_1 \# \cdots \# C_m \Rightarrow$ accepting configuration
$\downarrow$
starting configuration of M on input x
$\rightarrow$ Each C_{i+1} follows correctly C_i per M's transition rules.
Turing Machine M_D takes $\langle x, y \rangle$ and checks:

- Does y represent a valid sequence of steps that shows M accepts x?
- If yes, accept. Otherwise, reject.

So D is a decidable language because this verification is straightforward.

Construction

To recognize C, you can:

1. Go through all possible strings of y.
2. For each one, check if $\langle x, y \rangle \in D$.
3. If any such y works, accept x.
4. If none work, run forever (since we're just recognizable).

This proves:

$$x \in C \Leftrightarrow \exists y \text{ such that } \langle x, y \rangle \in D$$

Example(forward Direction)

Let

$$C = \{x \mid x \text{ is a string that contains the same number of 0's and 1's}\}$$

M works like:

- Input: $x =$ "0011"
- Read input x
- Count 0's and 1's
- Accept if they are equal

Valid Computation:

1. $y =$ "[start]#0011#step1#step2#...#accept" (shows equal 0s and 1s)
2. verifier$_D$("0011", y) $\Rightarrow$ returns true
3. recognizer$_C$("0011") $\Rightarrow$ accepts

Solution (Reverse Direction)

Assume D is decidable.
That means: There is a TM M_D such that:

- On input $\langle x, y \rangle$, it checks if y is a valid accepting computation of some TM on x
- Accepts if the computation is valid, rejects otherwise

Now, define the language:

$$C = \{x \mid \exists y \text{ such that } \langle x, y \rangle \in D\}$$

This means:

- $x \in C$ if there's some certificate y (like a proof) such that y shows $x \in C$
- y serves as a justification that x is accepted

How to construct a recognizer for C?
We build a Turing Machine M that:

- On input x, enumerate all possible strings $y \in \Sigma^*$
- For each y, run the decider M_D on $\langle x, y \rangle$
- If M_D accepts for some y, accept x
- If no such y is found, keep searching (never halts on non-members — allowed for recognizers)

Why does this work?

- If $x \in C$, then there exists some valid y such that $\langle x, y \rangle \in D$
- Since D is decidable, M_D will accept that $\langle x, y \rangle$
- So, our machine M will eventually find such a y and accept x

Therefore, C is Turing-recognizable.

Example (Reverse Direction)

Let

$D = \{\langle x, y \rangle \mid y$ is a valid accepting computation history proving that x has the same number of 0's and

Given: D is decidable — we can check whether y is a valid proof that $x \in C$.
Build recognizer M_C for C:

- Input: $x =$ "0011"
- Enumerate all strings $y \in \{0, 1, \#, \text{symbols}\}^*$
- For each y, run the decider for D on $\langle x, y \rangle$
- If D accepts for some y, then accept x

Explanation:

1. $y =$ "[start]#0011#step1#step2#...#accept" (valid history showing equal 0s and 1s)
2. $\text{decider}_D(\text{"0011"}, y) \Rightarrow$ returns true
3. $\text{recognizer}_C(\text{"0011"}) \Rightarrow$ accepts

Thus, C is Turing-recognizable.

Question 20

A **useless state** in a Nondeterministic Finite Automaton (NFA) is a state that is *never entered* on any input string. We want to determine whether a given NFA has any useless states.

Language Formulation

Define the language:

$$L = \{\langle M\rangle \mid M \text{ is an NFA and } M \text{ has at least one useless state}\}$$

Solution:

We will define a language,

$$L_{\text{useless}} = \{\langle M\rangle \mid M \text{ is a NFA and has at least one useless state}\}$$

Proving Decidability

To show that L_{useless} is decidable, we need to construct a Turing machine decider which always stops and determines whether a given NFA M has at least one useless state. Let:

$$M = (Q, \Sigma, \delta, q_0, F)$$

be a DFA, where:

- Q is a finite set of states
- Σ is the input alphabet
- $\delta : Q \times \Sigma \to Q$ is the transition function
- q_0 is the start state
- F is the set of accepting states

To check for useless states...
(P.T.O.)

Steps to Find Useless States in an NFA

1 First we will find Reachable States

- We will start from the start state q_0.
- We can perform either a BFS / DFS on the states through epsilon transitions and other transitions (following the transition function δ), on the NFA.
- We will mark the states in the process.
- Any state that is not marked is never entered on any input and is thus **useless**.

2 Deciding and Output:

- If there is a state in the set Q that is not marked as reachable, then M has at least one **useless state**, else all states are reachable.

$$\Rightarrow \text{ Start BFS from } q_0, \quad R \rightarrow \text{ set of states that are reachable from } q_0$$

$$\text{BFS}(q_0, \text{NFA}) \Rightarrow$$

$$\text{If reachable from } q_0, \text{ then keep in the set } R$$

After the BFS ends, if R = Q then there are no useless states, else there are one (or) more useless states in the NFA.

How can we say L_{useless} is decidable?
Since the BFS (or DFS) runs in polynomial time, it always halts. And since it halts and correctly identifies whether M has useless states,

$$\Rightarrow \text{Time complexity} = O(|Q| \times |\Sigma|)$$

Hence, L_{useless} is decidable.