MTech CSE – 1st Semester
Student Name: SIDDHARTH KUMAR
Student ID: A125021

Question 12

Given a Boolean circuit whose output evaluates to TRUE, explain how the correctness of this result can be verified in polynomial time using Depth First Search (DFS)

Answer:

Boolean Circuits: Structure and Interpretation

A Boolean circuit is a finite directed acyclic graph (DAG) consisting of:

- logic gates such as AND, OR, and NOT as internal nodes,
- input literals or constants as leaf nodes,
- a unique output gate that represents the circuit's final value.

For a given input assignment, the circuit is said to be a YES-instance if the output gate evaluates to TRUE. The goal is not to recompute the output, but to verify that the claimed result is correct.

Core Verification Principle

Verification relies on checking that:

- each gate's output value is consistent with its logical definition,
- the values of its input gates support the claimed output,

- the verification proceeds systematically from the output toward the inputs.

Depth First Search provides a natural mechanism to perform this recursive validation efficiently.

Modeling the Circuit as a Graph

The Boolean circuit is viewed as a directed graph where:

- vertices correspond to gates or input literals,
- edges represent signal dependencies between gates.

Because circuits do not contain cycles, the resulting graph is acyclic, ensuring that DFS terminates without redundant revisits.

DFS-Based Verification Method

Traversal Strategy

The verification algorithm initiates a DFS from the output gate. During traversal, the algorithm recursively verifies the correctness of each gate by examining its input gates first.

At every gate encountered:

- the algorithm evaluates all prerequisite gates,
- checks whether their values justify the current gate's value,
- propagates the result upward toward the output.

Logical Validation Rules

Let a gate claim a particular Boolean value.

- **AND gate:** The output is TRUE only if all input gates evaluate to TRUE.
- **OR gate:** The output is TRUE if at least one input gate evaluates to TRUE.
- **NOT gate:** The output must be the logical negation of its single input.

- **Input literal:** Its value is directly checked against the provided input assignment.

If any condition fails, verification terminates with failure.

High-Level DFS Verification Algorithm

```
Verify(g):
    if g is an input literal:
        return its assigned truth value

    if g is AND:
        for each input c of g:
            if Verify(c) == false:
                return false
        return true

    if g is OR:
        for each input c of g:
            if Verify(c) == true:
                return true
        return false

    if g is NOT:
        return NOT Verify(single input of g)
```

Verification succeeds if `Verify(output_gate)` returns TRUE.

Correctness of the Approach

DFS ensures correctness because:

- every gate influencing the output is examined,
- each gate is verified only after its dependencies,
- the acyclic structure prevents infinite recursion.

Thus, the output value is justified through a bottom-up logical validation.

Time Complexity Analysis

Let:

- n denote the number of gates,
- m denote the number of wires.

DFS visits each gate and wire at most once. Therefore, the verification runs in:

$$O(n + m),$$

which is polynomial in the size of the circuit.

Connection to NP Verification

The Boolean Circuit Value Problem plays a central role in complexity theory. Given an input assignment as a certificate, the DFS-based method verifies that the circuit evaluates to TRUE in polynomial time.

This directly satisfies the defining property of the class **NP**: YES-instances admit efficiently verifiable certificates.

Relation to Circuit-SAT

In the Circuit-SAT problem, the task is to determine whether some input assignment causes the circuit to output TRUE. When such an assignment is provided, DFS serves as a deterministic polynomial-time verifier of correctness.

Hence, this verification procedure underpins the NP-completeness of Circuit-SAT.

Why DFS Is Preferred

Although other traversal methods exist, DFS is particularly suitable because:

- it naturally reflects recursive gate evaluation,
- it supports dependency-first validation,
- it requires minimal auxiliary memory.

Conclusion

The correctness of a Boolean circuit's TRUE output can be verified in polynomial time using Depth First Search.

By recursively validating each gate's logical consistency starting from the output gate, DFS guarantees both correctness and efficiency.

Intuition

The process resembles checking a proof:

- DFS traces the reasoning backward,
- confirms each intermediate step,
- and validates the final conclusion.