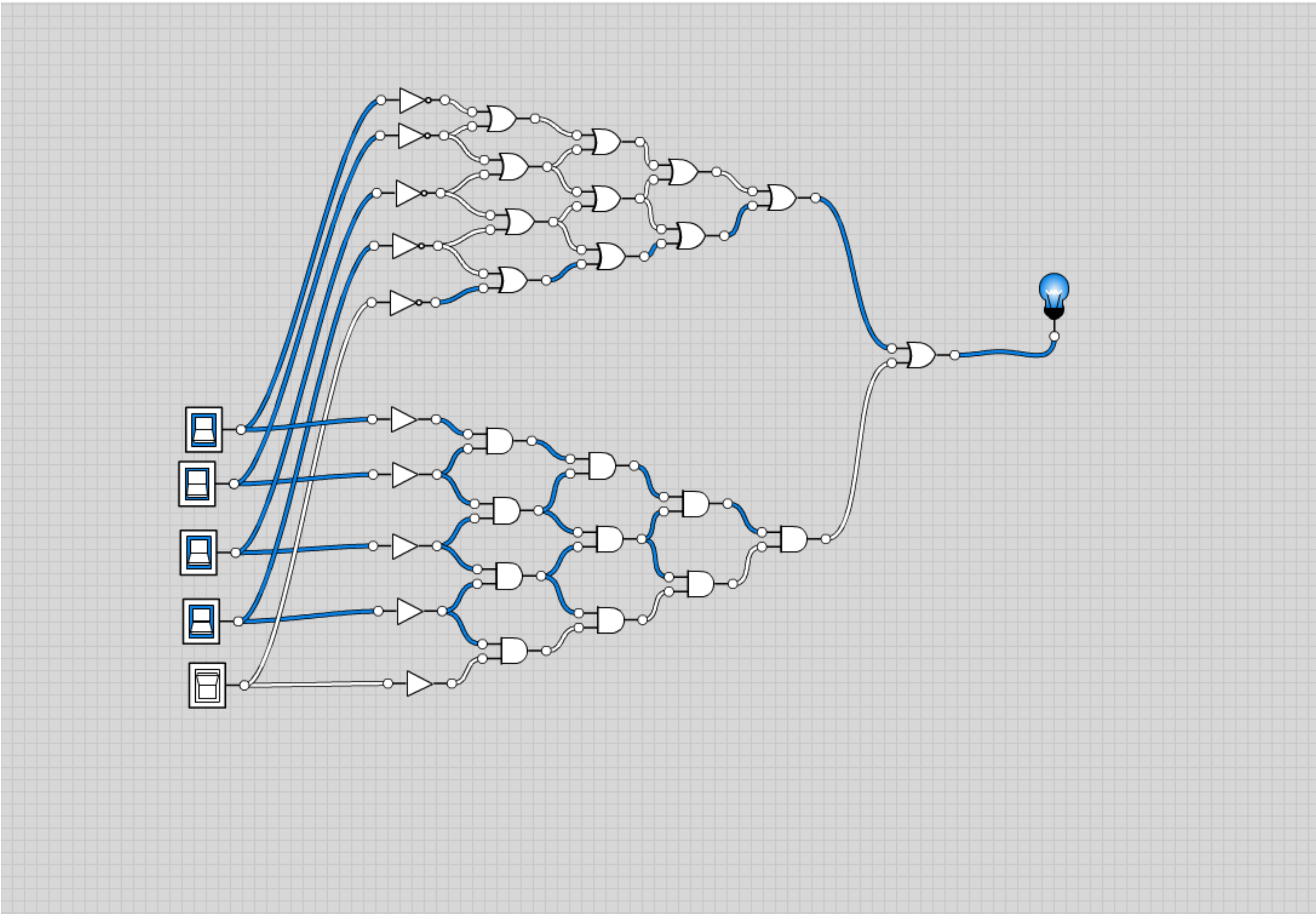
The Y Tree System

Author: Nahom Ketema



1. Introduction

The **Y Tree System** is a computational logic structure designed to optimize Boolean Satisfiability (SAT) problem solving. It systematically expands Boolean expressions using a recursive branching pattern that ensures efficient evaluation and potential polynomial reductions in complexity.

---

2. Formal Definition of the Y Tree

2.1 Tree Structure

A **Y Tree** is defined as a **binary recursive expansion** of logic gates, where each node branches into two sub-nodes, forming a **Y-like pattern**.

Mathematical Definition:

Let *V* be a set of Boolean variables. The Y Tree, denoted as *Y(n)*, is recursively defined as:

$$Y(n) = \begin{cases} V, & \text{if } n = 0 \\ g_i(Y(n-1), Y(n-1)) \quad | \quad g_i \in G, & \text{if } n > 0 \end{cases}$$

where:

- *Y(n)* represents the set of logic gates at depth *n*.
- *G* is the set of Boolean operations *{AND, OR, NOT}*.
- Each new level consists of gates operating on the outputs of the previous level *n-1*.

2.2 Growth Formulas

Total Number of Gates

The number of gates in a Y Tree follows the formula:

$$g = \left(\sum_{i=0}^{n} i \right) \times 2 + 1 - n$$

Simplifying:

$$g = \left(\frac{n(n+1)}{2} \right) \times 2 + 1 - n$$

$$g = n(n+1) + 1 - n$$

$$g = n^2 + n + 1 - n$$

$$g = n^2 + 1$$

Thus, the total number of gates in the Y Tree is:

$$g = n^2 + 1$$

Finding the Number of Variables from Gates

If the total number of gates *g* is known, the number of variables can be determined as:

$$n = \sqrt{g - 1}$$

Number of AND Gates

The number of AND gates, denoted as a, is given by:

$$a = \sum_{i=0}^{n-1} i$$

Simplifying:

$$a = \frac{(n-1)n}{2}$$

Number of OR Gates

The number of OR gates, denoted as r, is one more than the number of AND gates:

$$r = a + 1$$

$$r = \frac{(n-1)n}{2} + 1$$

$$r = \frac{(n-1)n + 2}{2}$$

where:

- g is the total number of logic gates.
- n represents the number of variables.
- a represents the total number of AND gates.
- r represents the total number of OR gates.

These formulas are derived from the recursive pattern of the Y Tree, where each level expands quadratically while ensuring a structured and deterministic evaluation path.

---

3. Properties of the Y Tree

3.1 Satisfiability Theorem

If a Boolean formula can be represented within a fully expanded Y Tree, a satisfying assignment exists if and only if the final node evaluates to true.

3.2 Logical Compression

The Y Tree structure minimizes redundant logical checks by restructuring conjunctive and disjunctive normal forms into an optimized recursive format.

3.3 Computational Complexity

- **Worst-case complexity:** *$O(n^2)$* due to quadratic gate expansion.
- **Optimized cases:** *$O(n \log n)$* when redundant operations are pruned.
- **Path reduction:** The depth of the tree is *$O(\log n)$* when optimized for minimal evaluation paths.

---

4. Applications

- **SAT Solving**: Alternative to conventional SAT solvers.
- **AI Decision Trees**: Used in recursive decision-making models.
- **Cryptographic Boolean Optimization**: Improves logical circuit efficiency.
- **Quantum Computing**: Mapping classical logic to quantum gates.

---

5. Conclusion

The Y Tree presents a structured approach to solving Boolean logic problems through recursive expansion. Its ability to reduce complexity and improve evaluation efficiency makes it a valuable tool in computational logic, AI, and cryptography.
