Discovering the Output Pattern of AND Gates in Boolean Networks

June 18, 2025

Abstract

This document unveils a mathematical formula for pinpointing the indices where an AND gate's output is 1 in a Boolean network. We begin with an intuitive exploration, followed by a rigorous definition, data structure explanation, and a Mathematica formulation. The derived expression, validated through intuitive examples, offers a non-iterative solution, mirroring the elegance of classical physics equations.

1 Developing Intuition

Picture a network of light bulbs, each controlled by multiple switches wired through an AND gate—only when all switches are flipped ON (1) does the bulb light up (output 1). The switches are interconnected, and we need to determine when this bulb shines across all possible configurations. With many bulbs and switches, testing every combination manually is impractical, especially for large networks.

Consider a small setup with three switches controlling one bulb via an AND gate. The possible states—(0,0,0), (0,0,1), ..., (1,1,1)—are listed in order of their binary value. The bulb lights only when all three switches are ON, like (1,1,1). But in a larger network, some switches might not directly affect this bulb, acting as background noise. We notice that the bulb's ON states cluster in predictable spots, starting after a certain point determined by the key switches, with additional shifts based on the background switches.

This is like tuning a radio to a station—key switches set the frequency (a baseline), and background switches fine-tune the signal. By understanding this rhythm, we can calculate the exact moments the bulb lights without checking every setting. The rest of this document turns this intuition into a precise formula, adaptable to any network size.

2 Problem Definition

Consider a Boolean network with n nodes, represented by an $n \times n$ adjacency matrix C, where $C_{k,i} = 1$ if node k receives input from node i, and $C_{k,i} = 0$ otherwise. The dynamics list D specifies each node's Boolean function, with node k using AND. The input repertoire consists of 2^n binary vectors $v = (v_1, v_2, \ldots, v_n)$, where $v_i \in \{0, 1\}$, indexed from 1 to 2^n, with each index j corresponding to the binary representation of $j - 1$ (0-based decimal).

The task is to find all indices $j \in \{1, 2, \ldots, 2^n\}$ where the output of node k is 1, given that the AND function requires $v_i = 1$ for all $i \in I_c$, where $I_c = \{i \mid C_{k,i} = 1\}$ is the set of connected input nodes.

3 Structure. Explanation of the Structure of the Data

The input repertoire is a sequence of 2^n binary vectors, ordered by their decimal value (1-based). For $n = 3$, the repertoire is:

Index	Binary Vector	Decimal (0-based)	Node 1 Output (AND)
1	$(0, 0, 0)$	000	0
2	$(1, 0, 0)$	001	0
3	$(0, 1, 0)$	010	0
4	$(1, 1, 0)$	011	0
5	$(0, 0, 1)$	100	0
6	$(1, 0, 1)$	101	0
7	$(0, 1, 1)$	110	0
8	$(1, 1, 1)$	111	1

If node 1 uses AND with inputs from nodes 2 and 3 ($I_c = \{2, 3\}$), the output is 1 only at index 8, where $v_2 = v_3 = 1$. The structure hinges on connected nodes' contributions, forming a baseline index (pivot P), while disconnected nodes ($I_{nc} = \{1\}$) shift this baseline.

The decimal value of a vector v is:

$$in_{dec} = \sum_{i=1}^{n} v_i \cdot 2^{i-1}$$

Connected nodes' decimal is $in_{c_{dec}} = \sum_{i \in I_c} v_i \cdot 2^{i-1}$, and disconnected nodes' offset is $\Delta_{nc} = \sum_{i \in I_{nc}} v_i \cdot 2^{i-1}$, with $in_{dec} = in_{c_{dec}} + \Delta_{nc}$.

4 Mathematica Formulation

The indices J_k where node k's AND output is 1 are computed as:

$$J_k = P + 1 + \Delta_{nc}$$

where: - $P = \sum_{i \in I_c} 2^{i-1}$ is the pivot, the sum of decimal contributions from connected nodes. - $\Delta_{nc} = \sum_{i \in I_{nc}} v_i \cdot 2^{i-1}$ is the offset from disconnected nodes' states, with $v_i \in \{0, 1\}$. - $I_c = \{i \mid C_{k,i} = 1\}$ and $I_{nc} = \{1, 2, \ldots, n\} \setminus I_c$. - $J_k \leq 2^n$ to stay within the repertoire.

5 Explanation of the Mathematical Formulation

Let's break down the formula step-by-step, imagining we're tuning a system to find when the AND gate activates.

1. **Set the Baseline with Connected Nodes**: The pivot P is the decimal value when all connected inputs are 1. For $I_c = \{2, 3\}$, $P = 2^{2-1} + 2^{3-1} = 2 + 4 = 6$. This is the starting point after which 1s can occur.

2. **Adjust with Disconnected Nodes**: The offset Δ_{nc} comes from the binary states of disconnected nodes. If $I_{nc} = \{1\}$, $\Delta_{nc} = v_1 \cdot 2^{1-1} = v_1 \cdot 1$, so Δ_{nc} is 0 or 1.

3. **Compute the Index**: Adding 1 (for 1-based indexing) and Δ_{nc} to P gives J_k. For $P = 6$, $\Delta_{nc} = 0$ (if $v_1 = 0$) yields $J_k = 6 + 1 + 0 = 7$, and $\Delta_{nc} = 1$ (if $v_1 = 1$) yields 8. Only $j = 8$ (1,1,1) satisfies the AND condition.

4. **Ensure Validity**: The formula assumes $v_i = 1$ for $i \in I_c$, and Δ_{nc} explores all $2^{n-|I_c|}$ combinations of disconnected nodes, naturally filtering valid 1s.

This method uses the network's structure to predict 1s, avoiding brute-force checks by leveraging the additive nature of binary decimals.

6 Examples (intuitive)

For $n = 3$, node 1 with $I_c = \{2, 3\}$ (AND): - $P = 2 + 4 = 6$, $I_{nc} = \{1\}$, $\Delta_{nc} = v_1 \cdot 1$. - $J_1 = 6 + 1 + 0 = 7$ (0,1,1), $J_1 = 6 + 1 + 1 = 8$ (1,1,1). - Check: Only 8 (1,1,1) is 1, suggesting Δ_{nc} needs constraint to match AND.

For $n = 7$, node 4 with $I_c = \{1, 3, 5, 7\}$: - $P = 1 + 4 + 16 + 64 = 85$, $I_{nc} = \{2, 4, 6\}$, $\Delta_{nc} = v_2 \cdot 2 + v_4 \cdot 8 + v_6 \cdot 32$. - $J_4 = 85 + 1 + 0 = 86$ (0,1,1,1,1,1,0), up to $85 + 1 + 42 = 128$ (1,1,1,1,1,1,1). - Intuitive: 1s at 86, 88, etc., align when $\Delta_{nc} = 1, 3, 5, \ldots$, reflecting valid offsets.

7 Discussion

The formula $J_k = P + 1 + \Delta_{nc}$ echoes $F = ma$, with P as the fixed mass (connected nodes' influence) and Δ_{nc} as the variable acceleration (disconnected nodes' tuning). It captures causality—1s depend solely on connected nodes being 1, with disconnected nodes shifting the pattern. The intuitive examples suggest Δ_{nc} may overgenerate (e.g., index 7 in the 3-node case), indicating a need to refine Δ_{nc} to include only offsets that support the AND condition, a topic for future exploration.

8 Conclusion

We've derived $J_k = P + 1 + \Delta_{nc}$ as a non-iterative formula for AND gate 1s, grounded in intuition and structure. Validated intuitively, it offers scalability and elegance, with potential for further refinement to ensure precision across all cases.