Proof by Minecraft

The greatest proof simulator in recorded history

UQCS Hackathon
Pajorn, Josiah, Matt, Gus, and Zwe
August 16, 2025

Abstract

Minecraft is widely known as a sandbox for creativity, but we extend its potential to the domain of formal reasoning: First Order Logic (FOL) and Boolean Algebra. Our project, Proof by Minecraft, transforms logical expressions into interactive visual proofs. Using Python, we parse and simplify expressions via trees and graph theory, then generate corresponding Minecraft circuits. A Tkinter-based interface allows users to input one or two expressions; the system produces a truth table, evaluates equivalence, and constructs a Minecraft representation of the logic. This approach makes formal logic both intuitive and engaging, offering a novel way to explore proofs—because sometimes, the best argument really is Proof by Minecraft.

Keywords: First Order Logic/Boolean Algebra (FOL), Graph Theory, Trees

Rationale - Key Ideas, Requirements, and Goals.

The main idea behind 'Proof by Minecraft' is to make a formal logic system which is not only rigorous but visually and interactively intuitive. It both provides a 'graphical' representation of logical expressions, and also allows users to interact with the circuit, using levers to change the value of the inputs, and to evaluate the expression. The system also provides definitive feedback like truth tables, and equivalences which are essential for rigorously evaluating logical expressions. By embedding First Order Logic (FOL) expressions within Minecraft, we provide users with a tangible way to explore equivalence, simplification, and proofs. Our rationale stems from three guiding goals: accessibility, interactivity, and clarity.

Key Ideas

- Each FOL expression is represented as a Minecraft world where levers act as input variables and redstone lamps display output values.
- When two expressions are entered, they are rendered side by side with shared inputs, enabling direct comparison.
- To evaluate equivalence, the program systematically generates all input cases (2^n possibilities) and maps them into Minecraft.

Requirements

- **Parsing and Representation:** Logical expressions must be parsed into a graph/tree structure using graph theory and algorithmic simplification.
- **User Interface:** A Python GUI (Tkinter) must allow users to input expressions, following Nielsen's 10 heuristics for usability.
- **Minecraft Integration:** The parsed structure must be translated into a Minecraft world file, ensuring correct placement of redstone, lamps, and levers.

Goals

- Provide an engaging, gamified method to explore logical proofs.
- Demonstrate how graph theory and algorithms can be applied creatively.
- Produce both a truth table and equivalence check alongside the Minecraft visualization.

Section Name

- First bullet point item
- Second bullet point item
- Third bullet point item

1. First numbered list item
2. Second numbered list item

Conclusion

References

Figure 1: An example fish.