Proposal & Design Description for Activity Diagram

Tyler Dickson

Topic: Propositional Logic Calculator

Introduction

The Propositional Logic Calculator is an innovative tool designed to compute and graphically present truth tables derived from user-specified logical propositions. This application provides an intuitive interface for exploring the interplay of logical connectives in propositional logic.

Background

Logical connectives are pivotal operators that amalgamate or modify logical statements to construct intricate propositions, enabling the articulation of relationships among these statements. These are indispensable in building logical expressions and in analytical reasoning. Truth tables are essential tools in visualizing and resolving logical dilemmas, engaging in Boolean algebra, assessing logical equivalence, and facilitating decision-making processes. Boolean operations underpin many everyday functionalities, including search engine queries.

Program Description

Users input a sequence of logical connectives, and the program renders these inputs either as truth tables or as logic circuit diagrams. A suite of algorithms will parse and iterate over characters, variables, and constants to produce the desired outputs.

Primary Programmatic Approach

The code leverages C and C++ libraries for backend operations. Truth table results are displayed in binary format, while logic circuit visualizations are created using a specialized software, Graph.

Secondary Programmatic Approach

Utilizing Android Studio, the application accommodates the integration of C and C++ code within Android projects by housing this code in a designated cpp directory within a project module. Upon build, this code compiles into a native library packaged with the application by Gradle. Java or Kotlin codes can invoke functions from this native library via the Java Native Interface (JNI), facilitating an interaction between the native and Java layers.

High-Level Programmatic Approach and Design

1. **User Input Management:**
    - The application accepts logical expressions comprising variables labeled as a, b, c, d, and e.
    - Connectives are denoted using symbols like "||" for " ∨ " (or), "&&" for " ∧ " (and), and "!" for "¬" (not).
    - The system robustly handles and corrects erroneous inputs.
2. **Functionality Selection:**
    - Users can choose between generating a logic circuit diagram or a truth table.
3. **Logic Gate Parsing:**

- The application processes user-provided logical expressions to produce the selected graphical representation.

4. **Truth Table Generation:**
   - Leveraging a modified version of the shunting algorithm, the program parses logical operators into tokens and characters.
   - It evaluates each row sequentially, storing results in a map and ultimately displaying them in a binary format as illustrated in subsequent diagrams.

Truth Table Example:
For the input $(p \lor \neg q) \rightarrow (p \land q)$, the program outputs:

```
A B | OUTCOME
---------
1 1 | 1
1 0 | 0
0 1 | 1
0 0 | 0
```

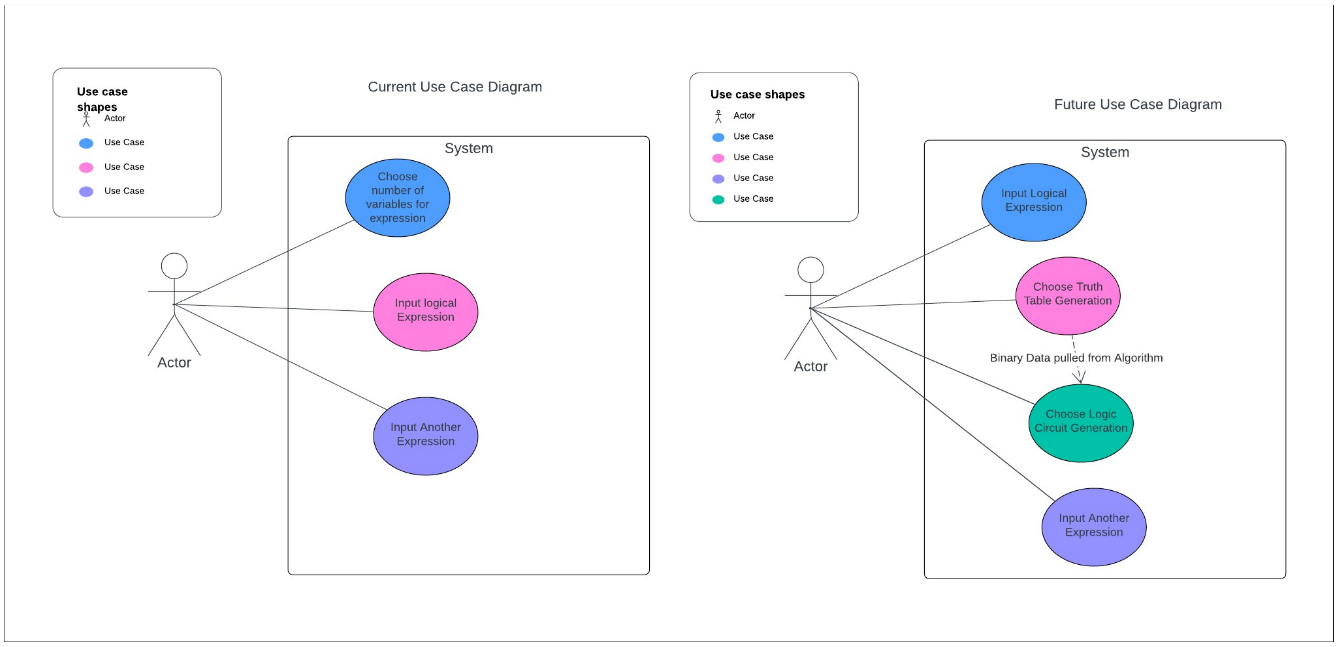

Figure 2 – Use Case Diagram

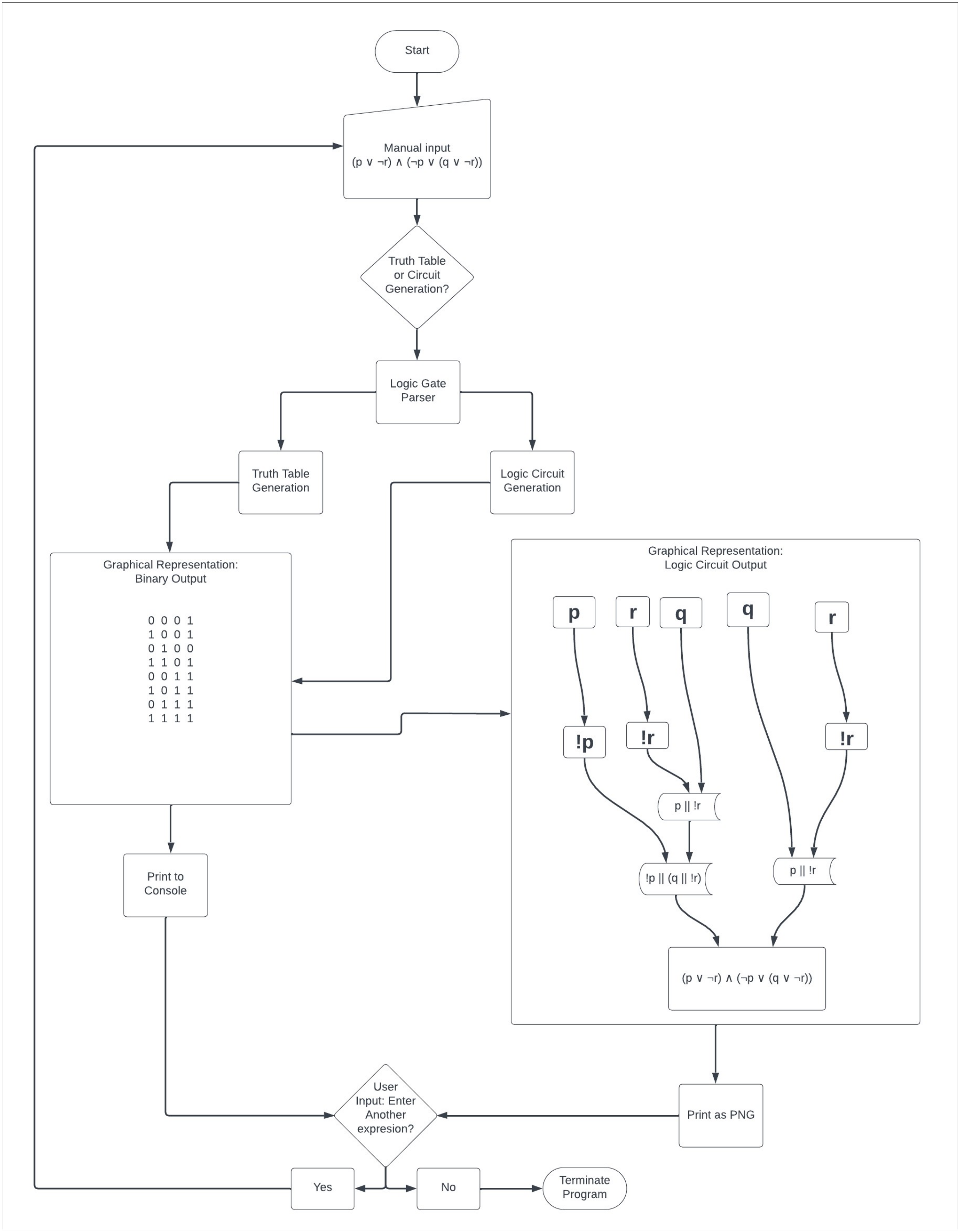

Figure 2 – Activity Diagram