SuperCoding

Boogle Project
Software Requirements Specifications

Version 1.1

Revision History

Date	Version	Description	Author
02/29/2024	1.0	Initial revision	SuperCoding
03/04/2024	1.1	Full revision	SuperCoding

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

To provide an intuitive boolean simulator that can provide accurate outputs for both simple boolean operations and truth tables.

1.2 Scope

The software must be capable of running C++ and running the resulting executable file.

1.3 Definitions, Acronyms, and Abbreviations

- SRS - Software Requirements Specs
- Boolean - The process of using only 0s and 1s
- Parsing - Analyzing string symbols in order to determine output data

1.4 References

- EECS 348: Software Engineering I LEC Project Files (https://canvas.ku.edu/courses/119254/files/folder/Project?)
- EECS 348 announcements

1.5 Overview

- The Software-Requirements-Spec document is organized into 5 sections, with possible subsections included. The sections include:
    - Introduction - Purpose, Scope, Definitions, References, Overview
    - Overall Description - Interfaces, Operations, Functionality, Constraints
    - Specific Requirements
    - Classification of Functional Requirements
    - Appendices

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

2.1.2 User Interfaces

- Current project scope projects terminal input based user interface but can be expanded to GUI if project timeline allows.

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

- If GUI is implemented, the application will need to interface with it; otherwise, it will only need to work with the OS and terminal.

2.1.5 Communication Interfaces

2.1.6 Memory Constraints

- Memory constraints require 64-bit based operating systems as of now.

2.1.7 Operations

2.2 Product functions

- Boolean operation simulations
- Truth table simulations

2.3 User characteristics

- EECS students/faculty/field
- Math students/faculty/field
- Science students/faculty/field
- Clients/users with understanding of boolean logic

2.4 Constraints

- Written in C/C++
- Compiled for Linux
- Case insensitivity input into terminal
- Output display is limited to “t” and “f”
- Deadline

2.5 Assumptions and dependencies

- Compiled using the C Standard Library
- Terminal I/O (e.g. Bash) for Linux
- If GUI implemented, will require a 3rd-party GUI library.

2.6 Requirements subsets

3. Specific Requirements

3.1 Functionality

3.1.1 Boolean Algebra parsing/evaluation with symbolic operators

The software will be able to read, parse, and evaluate boolean algebra expressions, including parentheses and symbolic operators:

- *A & B [“AND”]*
- *A | B [“OR”]*
- *A $ B [“XOR”]*
- *A @ B [“NAND”]*
- *!A [“NOT”]*
- *e.g. (!A) & (C | B)*

3.1.2 Literal statements within boolean expressions

The software will support literal (case-insensitive) booleans within expressions:

- *“true”*
- *“false”*
- *e.g. (!A | false)*

3.1.3 Variable assignment

The software will allow the user to assign boolean values to variables, except for certain reserved (case-insensitive) names:

- *true, false*
- *AND, OR, XOR, NAND, NOT (if word-type operators are implemented)*

3.1.4 Truth Table output

The software will, optionally to the user, output a full truth table for every value of variables;

- “true”
- “false”

3.1.5 Support for word-type operators

- “AND”
- “OR”
- “XOR”
- “NAND”
- “NOT”

3.1.6 Graphical User Interface

- User entry field
- Boolean logic buttons
- Output display

3.1.7 Error Handler

- Boolean expression is invalid
- Missing parentheses
- Overloading
- Invalid input

3.1.8 History

- Output history viewable through output display

3.2 Use-Case Specifications

Use Cases

- 1. Evaluate Boolean Expression
  - Actor: User
  - Description: The user inputs a boolean expression using symbolic operators and/or word-type operators. The system parses and evaluates the expression, returning the result.
  - Preconditions:
    - System is operational and awaiting input.
  - Basic Flow:
    - User enters a boolean expression.
    - System parses and evaluates the expression.
    - System displays the result.
  - Alternate Flows:
    - If the expression is invalid, the system notifies the user of the error and suggests correction.
    - If the expression includes undefined variables, the system prompts the user for values.
  - Postconditions: User has received the evaluation result or an error message.
- 2. Generate Truth Table
  - Actor: User
  - Description: The user requests a truth table for a given boolean expression. The system outputs a complete truth table, showing all possible inputs and their corresponding outputs.
  - Preconditions:
    - A valid boolean expression is provided.
  - Basic Flow:
    - User inputs a boolean expression and requests a truth table.
    - System generates and displays the truth table.
  - Alternate Flows:
    - If the expression is invalid, an error is displayed.
  - Postconditions: User has received a truth table or an error message.
- 3. Assign Variable Values
  - Actor: User
  - Description: Allows the user to assign values to variables within expressions to simplify or specify expressions further.
  - Preconditions:
    - System is ready to accept input.
  - Basic Flow:
    - User specifies variable assignments (e.g., A=true).
    - System accepts and registers the assignments.
  - Alternate Flows:
    - If the variable name is reserved, an error message is displayed.

- Postconditions: Variables are assigned values for future expressions.
- 4. Interact with GUI (If Implemented)
  - Actor: User
  - Description: The user interacts with the graphical user interface to input expressions, request evaluations or truth tables, and view results.
  - Preconditions:
    - GUI is implemented and operational.
  - Basic Flow:
    - User navigates the GUI to input boolean expressions or requests.
    - System processes the requests and displays the results within the GUI.
  - Alternate Flows: N/A
  - Postconditions: User has interacted with the system via the GUI.

3.3 Supplementary Requirements

- Case Sensitivity: The system treats variable names and operators insensitively, meaning "A" is the same as "a", and "AND" is the same as "and".
- Error Handling: The system provides meaningful error messages for invalid inputs, including suggestions for correction.
- Memory Constraints: Requires a 64-bit operating system to handle memory requirements efficiently.

4. Classification of Functional Requirements

Functionalitys	Type
Boolean Algebra parsing/evaluation with symbolic operators (!|&@$)	Essential
Literal statements within boolean expressions (e.g. ‘TRUE’)	Essential
Variable assignment	Essential
Truth Table output	Desirable
Support for ‘word’-type operators (e.g. “AND”, “OR”)	Desirable
Graphical User Interface	Optional
History (viewing, editing, rollback)	Optional
Truth table with box-drawing characters	Desirable

5. Appendices

- Online.Visual-Paradigms (Not Required)
- A non-functional Mock-Up (Not strictly required; functional ideal)
- The original project definition by Dr. Saiedian (Required)