Individual Report

2IMP20 Domain Specific Language Design

Yang Yang 1697692 Filipe Sobrinho 1890352

1. Description/documentation of the language

Purpose of the language and motivation

2. Abstract and concrete syntax description

Concrete Syntax:

The concrete syntax defines the lexical rules and grammar rules for the QL language. It specifies how the language is written in terms of tokens and syntax rules.

- Lexical Rules:
    - **lexical String**: Matches sequences of characters enclosed in double quotes.
    - **lexical Integer**: Matches integer values.
    - **lexical Boolean**: Matches the keywords **true** or **false**.
    - **lexical Identifier**: Matches valid identifiers.
- Grammar Rules:
    - **syntax Questionnaire**: Represents the entire questionnaire, composed of components.
    - **syntax Component**: Represents a component, which can be a question, computed question, if statement, or if-else statement.
    - **syntax Question**: Represents a simple question that requires user input.
    - **syntax ComputedQuestion**: Represents a question that is computed based on previously defined variables.
    - **syntax IfElseStatement**: Represents an if-else statement.
    - **syntax IfStatement**: Represents an if statement.
    - **syntax LogicalExpression**: Represents logical expressions used in computed questions and conditionals.
    - **syntax Type**: Represents the supported variable types in the questionnaire.

Abstract Syntax:

The abstract syntax defines the structure and data types used to represent the QL language in an abstract syntax tree (AST). It provides a higher-level representation of the language.

- Data Types:
    - **data AQuestionnaire**: Represents a questionnaire in the AST.
    - **data AComponent**: Represents a component in the AST.
    - **data ALogicalExpression**: Represents logical expressions in the AST.
    - **data AIdentifier**: Represents an identifier in the AST.
    - **data AType**: Represents variable types in the AST.

The abstract syntax defines the structure of the AST and the data types used to represent different elements of the QL language.

By separating the concrete syntax from the abstract syntax, it becomes easier to parse the input code and transform it into a structured representation for further analysis and processing.