CSC488
ASSIGNMENT 2
TESTING

Contents

1. Overview

Suppose R is the reference language and $L(G)$ is the language recognized by our grammar G. The goal of our test cases was to demonstrate with some confidence that

$$x \in R \iff x \in L(G)$$

So we came up with a number of "failing" strings, or $x \notin R$, in the hopes that the string was also not in the language recognized by our parser, or $x \notin L(G)$. We also provided some reasonably complicated "passing" strings, or $x \in R$, in the hopes that those strings would also be in the language recognized by our grammar, or $x \in L(G)$. Below is a listing of the test cases we used to test our grammar.

2. Failing

2.1. **A1e.** A program from A1 that fails due to declarations in a scope without statements

2.2. **assign_no_expr.** Assignment statement with no expression

2.3. **decl_after_stmt.** Declaration of a variable after a statement

2.4. **decl_func_missing_type.** Function declaration with type missing

2.5. **decl_in_if.** Declaration in an if statement outside of a new scope

2.6. **decl_no_stmt.** Declaration of a variable with no statements following it

2.7. **decl_var_missing_bound.** Declaration of a variable with missing bound

2.8. **decl_var_missing_general_bound.** Declaration of a variable with missing general bound

2.9. **decl_var_missing_type.** Declaration of a variable with type missing

2.10. **func_missing_arguments.** Function call with missing arguments

2.11. **get_no_input.** Get statement with missing input

2.12. **if_else_no_stmt.** If then else statement with no statements within the "else" condition

2.13. **if_no_expr.** If statement with the condition expression missing

2.14. **if_no_stmt.** If statement with no statements within the "then" condition

2.15. **no_program.** An empty program, without a scope

2.16. **not_associative_eq.** Attempt to associate equality predicate

2.17. **not_associative_gt.** Attempt to associate greater than predicate

2.18. **not_associative_gteq.** Attempt to associate greater than or equal predicate

2.19. **not_associative_lt.** Attempt to associate less than predicate

2.20. **not_associative_lteq.** Attempt to associate less than or equal predicate

2.21. **not_associative_not_eq.** Attempt to associate not-equality predicate

2.22. **proc_missing_arguments.** Procedure call with arguments missing in argument list

2.23. **put_no_output.** Put statement with missing output

2.24. **result_no_expr.** Result statement without an expression

2.25. **unclosed_scope.** A scope that is not closed

3. Passing

3.1. **A1a.** A program that uses all arithmetic, logical and comparison operators

3.2. **A1b.** A program using arrays including both forms of array declaration, positive and negative bounds

3.3. **A1c.** A program using all forms of loop building and loop exit constructs

3.4. **A1d.** A program using non-recursive functions and procedures with and without parameters

3.5. **A1e.fixed.** Fixed scope issues in a program using recursive functions and procedures with and without parameters include at least one nested procedure and one nested function declaration

3.6. **arithmetic.** A verifiable arithmetic test

3.7. **input_and_output.** Exercises the "input" and "output" statements

3.8. **scope_empty.** Empty scope

3.9. **scope_statement.** Scope with statements but no declarations

3.10. **scope_statement_declaration.** Scope with statements and declarations

3.11. **arith_not_operand. NOT RECOGNIZED**. Although we believe this program to be in the reference language it unfortunately is not recognized by the parser. We filed it in the passing section because we believe it is part of the reference language. This test includes a case of the not operator being used within numeric expressions. The design challenges related to this test are discussed in more detail in "DESIGN.pdf"