HCMC University of Technology
Faculty of Computer Science & Engineering

Assignment 3

Static Checker

Author Dr. Nguyen Hua Phung

October 27, 2017

Contents

Assignment 3

version 1.1

After completing this assignment, you will be able to

- explain the principles how a compiler can check some semantic constraints such as type compatibility, scope constraints,... and
- write a medium (300-500LOC) Scala program to implement that.

1 Specification

In this assignment, you are required to write a static checker for a program written in MC. To complete this assignment, you need to:

- Read carefully the specification of MC language
- Download and unzip file **assignment3.zip**
- Starting from the code of Assignment 2. If you did not complete Assignment 2, don't worry, just follow these steps:
  - download initial.zip, and unzip it, and change the folder initial into assignment3
  - remove folder src/main/mc/codegen and files AstUtils.scala, TestCodeGen.scala and CodeGenSuite.scala in folder src/test/scala/
  - modify src/main/mc/Main.scala by commenting out 2 lines "import jasmin.Main" and "import mc.codegen._"
  - download upload.zip from Assignment 2, unzip it, and copy files into the corresponding folders as described in Assignment 2.
- Make folder src/main/mc/checker (if you deleted it) and copy StaticCheck.scala and StaticError.scala into this folder
- Copy TestChecker.scala and CheckerSuite.scala into src/test/scala
- Modify StaticCheck.scala to implement the static checker. **Please fill in your id in the headers of this file**.

2 Static Checker

A static checker plays an important role in modern compilers. It checks in the compiling time if a program conforms to the semantic constraints according to the language specification. In this assignment, you are required to implement a static checker for MC language. The input of the checker is in the AST of a MC program, i.e. the output of the assignment 2. The output of the checker is nothing if the checked input is correct, otherwise, an error message is released and the static checker will stop immediately

For each semantics error, students should throw corresponding exception given in StaticError.scala to make sure that it will be printed out the same as expected. Every test-case has at most one kind of error. The semantics constraints required to check in this assignment are as follows.

2.1 Redeclared Variable/Function/Parameter:

An identifier must be declared before used. However, the declaration must be unique in its scope. Otherwise, the error messages "Redeclared <kind>: "+<identifier> is released, where <kind> is the kind of the identifier in the second declaration. The scope of an identifier (variable, function, parameter) is informally described as in Section 10 of MC specification.

2.2 Undeclared Identifier/Function:

The error message "Undeclared Identifier: "+<identifier> is released when there is an identifier is used but its declaration cannot be found. The identifier can be a variable or parameter. "Undeclared Function: "+<function name> is released in similar situation for a function invocation.

2.3 Type Mismatch In Statement:

A statement must conform the corresponding type rules for statements, otherwise the error message "Type Mismatch In Statement: "+<statement> is released.

The type rules for statements are as follows:

- The type of a conditional expression in an **if** statement must be boolean.
- The type of expression 1 and expression 3 in a **for** statement must be integer while the type of expression 2 is boolean.
- The type of condition expression in **do while** statement must be boolean.
- For a **return** statement, if the return type of the enclosed function is void, the expression in the return statement must be empty. Otherwise, the type of the return

expression must be equal to or be coerced to the return type of the function. An exception is an array pointer or array can be returned to an array pointer, i.e., the type of return expression can be in array pointer or array type while the return type of the enclosed function is in array pointer type with the same element type.

2.4 Type Mismatch In Expression:

An expression must conform the type rules for expressions, otherwise the error message "Type Mismatch In Expression: "+<expression> is released.

The type rules for expression are as follows:

- For an array subcripting E1[E2], E1 must be in array type or array pointer type and E2 must be integer.
- For a binary and unary expression, the type rules are described in the MC specification.
- For an assignment expression, the left-hand side can be in any type except void, array pointer type and array type. The right- hand side (RHS) is either in the same type as that of the LHS or in the type that can coerce to the LHS type. In MC, just the integer can coerce to the float.
- For a function call <function name>(<args>), the number of the actual parameters must be the same as that of the formal parameters of the corresponding function. The rule for an assignment is applied to parameter passing where a formal parameter is considered as the LHS and the corresponding actual parameter is the RHS. An exception is an array pointer or array can be passed to an array pointer, i.e., the actual parameter can be in array pointer or array type while the corresponding formal parameter is in array pointer type with the same element type.

The three following errors are required just for the students in gifted class (KSTN):

2.5 Function not return:

A function that does not return **void** must return something in every its execution paths. If there exists one path where there is nothing returned, the error message "Function Not Return: <function name>" will be released.

2.6 Break/Continue not in loop:

A break/continue statement must be inside directly or indirectly a loop otherwise the error message "Break/Continue Not In Loop" will be released.

2.7 Unreachable statement:

An unreachable statement is a statement in a function which the control flow cannot reach. For example, the statement after a return statement is an unreachable statement. The value of an expression is ignored when determining an unreachable statement. For instance, the **stmt** in **if (false) stmt** is reachable as the value of the condition expression is ignored. The error message "Unreachable statement: "+<statement> will be released when the <**statement**> is an unreachable statement.

3 Submissions

This assignment requires you submit 2 files: **StaticCheck.scala** containing class **StaticChecker** with the entry method **check**, and **CheckerSuite.scala** containing 100 testcases.

The deadline is **Friday, November 10th, 2017 at 16:30**.

4 Plagiarism

- You must complete the assignment by yourself and do not let your work seen by someone else.

If you violate any requirement, you will be punished by the university rule for plagiarism.

5 Change Log

- Add array pointer type and array type in the return statement (2.3).
- Add array pointer type in array subscripting and assignment (2.4).
- Change function call in 2.4.