

HCMC University of Technology
Faculty of Computer Science & Engineering



Assignment 3

Static Checker

Author

Dr. Nguyen Hua Phung

October 3, 2021

Contents

1	Specification	2
2	Static Checker	2
2.1	Redeclared Variable/Constant/Attribute/Class/Method/Parameter:	3
2.2	Undeclared Identifier/Attribute/Method/Class:	3
2.3	Cannot Assign To Constant:	3
2.4	Type Mismatch In Statement:	3
2.5	Type Mismatch In Expression:	4
2.6	Type Mismatch In Constant:	4
2.7	Break/Continue not in loop:	5
2.8	Illegal Constant Expression:	5
2.9	IllegalArrayLiteral:	5
2.10	IllegalMemberAccess:	5
3	Submissions	5
4	Plagiarism	5
5	Change Log	6

Assignment 3

version 1.0

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) Python program to implement that.

1 Specification

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

- Read carefully the specification of BKOOL language
- Download and unzip file **assignment3.zip**
- If you are confident on your Assignment 2, copy your BKOOL.g4 into src/main/bkool/parser and your ASTGeneration.py into src/main/bkool/astgen and you can test your Assignment 3 using BKOOL input like the first two tests (300-301).
- Otherwise (if you did not complete Assignment 2 or you are not confident on your Assignment 2), don't worry, just input AST as your input of your test (like test 302-303).
- Modify **StaticCheck.py** in src/main/bkool/checker to implement the static checker and modify **CheckSuite.py** in src/test to implement 100 testcases for testing your code.

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 BKOOL language. The **input** of the checker is in the **AST** of a BKOOL 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

! ↑
Lis + ???
..

For each semantics error, students must 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/Constant/Attribute/Class/Method/Parameter:

An identifier must be declared before used. However, the declaration must be unique in its scope. Otherwise, the exception Redeclared(<kind>,<identifier>) is released, where <kind> is the kind (Variable/Constant/Attribute/Class/Method/Parameter) of the identifier in the second declaration. The scope of an identifier (variable, constant, attribute, class, method, parameter) is informally described as in Section 7 of BKOOL specification.

Note that an attribute/method of a class is redeclared if there is another previous member of the same class with the same name.

2.2 Undeclared Identifier/Attribute/Method/Class:

The exception UndeclaredIdentifier(<identifier>) is released when there is an identifier is used but its declaration cannot be found. The identifier can be a variable, constant or parameter. Exception UndeclaredClass(<class name>) is released in similar situation for a class usage. Exception UndeclaredAttribute(<attribute name>) is released when there is an access to an attribute of a class but there is no declaration of the attribute of the class. Exception UndeclaredMethod(<method name>) is released in similar situation for a method invocation.

Note that BKOOL is an OO language so all members of a class may be inherited by its subclasses.

2.3 Cannot Assign To Constant:

The left-hand side (LHS) of an assignment statement operator cannot be declared as a constant (immutable field/variable), otherwise, the exception CannotAssignToConstant(<statement>) is released.

An assignment operator may appear in an assignment or a for statement. If the error happens in an assignment statement, the assignment statement is passed in the error message. In the case of a for statement, just the assignment part in this statement is printed out in the error message.

2.4 Type Mismatch In Statement:

A statement must conform the corresponding type rules for statements, otherwise the exception TypeMismatchInStatement(<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 a **scalar variable**, **expression 1** and **expression 2** in a **for** statement must be **integer**.
- For an assignment statement, the **left-hand** side can be in **any type except void 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 BKOOL, just the **integer** can coerce to the **float** or a **subtype** can coerce to its **super type**. When LHS is in an array type, RHS must be in array type whose **size is the same** and whose **element type** can be either the same or able to **coerce** to the element type of LHS.
- For a **call** statement **E.<method name>(<args>)**, E must be in class type; the callee must have **void as return type**. In addition, for parameter passing, the rule for an **assignment** is applied to **parameter** passing where a parameter is considered as the LHS and the corresponding argument is the RHS.
- For a **return** statement, the return expression can be considered as RHS of an implicit **assignment** whose LHS is the return type.

2.5 Type Mismatch In Expression:

An expression must conform the type rules for expressions, otherwise the exception **TypeMismatchInExpression(<expression>)** is released.

The type rules for expression are as follows:

- For an array subscripting **E1[E2]**, E1 must be in array type and E2 must be integer.
- **For a binary and unary expression, the type rules are described in the BKOOL specification.**
- For a method call **E.<method name>(<args>)**, E must be in class type; the callee **<method name>** must have **non-void** as return type. The type rules for arguments and parameters are the same as those mentioned in a procedure call.
- For an attribute access **E.id**, E must be in class type.

2.6 Type Mismatch In Constant:

The types of the left and right hand sides of a constant declaration must conform the type rule for an assignment described above. Otherwise, the exception **TypeMismatchInConstant(<ConstDecl>)** must be released. For example,
 final int a = 1.2;

2.7 Break/Continue not in loop:

A break/continue statement must be inside directly or indirectly a loop otherwise the exception `MustInLoop(<statement>)` must be thrown.

2.8 Illegal Constant Expression:

The expression to initialize a constant must be not None and evaluated statically so its operands must be a literal or an immutable attribute and they are combined together by operators only. If this constraint is violated, the exception `IllegalConstantExpression(<the violating subexpression>)"` is released. For example,

```
int a;  
final int x; IllegalConstantExpression(None) must be thrown  
final int y = 1 + a; IllegalConstantExpression(<ast of expression 1+a>) must be thrown.
```

Handwritten notes: A red checkmark is next to the first line. A red box is drawn around the expression `1 + a` in the third line, with an arrow pointing to it from the text "num const + -" written in red.

2.9 IllegalArrayLiteral.

All literals in an array literal must be in the same type otherwise the exception `IllegalArrayLiteral(<array literal>)` must be thrown. For example, `{1, 2.0}` is an example of this error.

2.10 IllegalMemberAccess:

A static or instance attribute/method must be accessed as concerned in Section 5.6 BKOOL Specification, otherwise the exception `IllegalMemberAccess(<field access or method invocation>)` must be thrown. For example, `E.a` where `a` is an instance field of class `E` or `x.b` where `x` is in `E` type and `b` is a static field of class `E` is a sample of this error.

3 Submissions

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

The deadline is announced in course website and that is also the place where you MUST submit your code.

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

R:

~~g/ah~~

