CS352s2014 Project 4: Type Checking

March 14, 2014

1 Objective

The objective of this project is to write a type checker for MiniJava SSA, as represented in the classes `edu.purdue.cs354.minijava.ssa.*`.

2 Code

The project's code environment is available on the course web page. We are providing Java class files for a complete compiler, including the parser, but are not providing accompanying Java source. Through the course of these projects, it will be your job to rewrite the Java source for each component, creating a complete compiler of your own design. You are advised to keep a copy of the Java class files so that you can use them to generate correct output and test against your own.

You are free to use the Java class files provided in lieu of your own implementations of the lexer, parser and SSA compiler, or to use your own if you prefer.

2.1 Testing

The compiler provides a number of tools for testing each stage. They are available in the `bin` directory. Two are of value for testing the translator.

- `bin/mjcompile-ssa`: Compiles MiniJava code into SSA form, optionally type checks, and writes it to standard output. To enable type-checking, use the `-t` flag.
- `bin/mjcompile-mips`: Compiles MiniJava code into MIPS assembler, using type-checked SSA as an intermediary.

The behavior of your type checker should be identicial to the behavior of the version provided. If you use the same SSA compiler (whether yours or the version provided), then your type checker should give identical results to the provided type checker.

3 Type Checking

MiniJava's type checking is performed over SSA statements, rather than the AST. However, since the SSA was generated without type information, examination of the AST may be required at times.

Type checking will proceed from the `SSAProgram`, through each `SSAClass` and `SSAMethod`, and ultimately through each `SSAStatement`. It may or may not be performed in a single pass. Each `SSAField` and `SSAMethod` must be assigned types, and each `SSAStatement` must be both type-checked and assigned an *expression type*. `SSAField` provides `getType` and `setType` methods to get/set the field's type, and `SSAMethod` provides `getRetType`, `setRetType`, `getParamTypes` and `setParamTypes` methods to get/set the return/parameter type(s). `SSAStatement` provides `getType` and `setType` methods for getting/setting the expression type of the operation. Note that checking of `SSAStatements` will require the type information in `SSAField`s and `SSAMethod`s.

Types are representing as `edu.purdue.cs352.minijava.types.StaticType` and its children, `PrimitiveType.IntType`, `PrimitiveType.BooleanType`, `VoidType` and `ObjectType`. Your type checker should create a single instance of each of `PrimitiveType.IntType`, `PrimitiveType.BooleanType` and `VoidType`, and one instance of `ObjectType` for each `SSAClass` in the program. Importantly, any two `SSAStatement`s with the same type should refer to the same `StaticType` in their `type` field.

MiniJava's type rules are documented in `doc/minijava-types.txt`.

4 Goals

For this assignment, you must implement one Java class: `edu.purdue.cs352.minijava.TypeChecker`. `TypeChecker` is constructed with the `SSAProgram` to type check as a constructor argument, and the entry point for type checking is the `typeCheck` method. `typeCheck` takes no arguments and returns nothing, but will throw an exception (of type `Error`) if a type error is found in the program. The particular text of error messages is not important, only that errors are detected and thrown as exceptions.

A template will be provided in `edu/purdue/cs352/minijava/TypeChecker.java-template`.

Please note that type checking is a destructive process: Programs may be rejected, but no new programs accepted, by type checking. However, the type checker *is* additionally expected to determine and set the types within the program.

A number of examples are provided in the `examples` directory. These examples will help in testing, but are *not* the examples that will be used to grade your submission. In particular, none of these examples are incorrect MiniJava. The frontend used to test both SSA compilation and type checking is `bin/mjcompile-ssa`, which in turn uses the `edu.purdue.cs352.minijava.SSACompilerFrontend` class. You are not expected to reimplement `SSACompilerFrontend`, only `TypeChecker`.

4.1 Example

Given the following file in `examples/test.java`:

```
class Simple {
```

```
    public static void main(String[] a) {
        System.out.println(42);
    }
}
```

`SSACompilerFrontend` should output something similar to:

```
program:
main:
method main:
0:  Int *42:  int
1:  Print 0:  void
```

Please note that the string output is not necessarily enough to judge the correctness of your program: The actual types and values of the `type` field must be correct, but only their string conversion will be printed.

5 Submission

You will submit your project code via the `turnin` command. Please include the entire project code directory, including all the class files provided by us, the `bin` and `examples` directories, etc.

To turn in a project directory: `turnin -c cs352 -p proj4 <project directory>`

This project is due by Friday, March 28, 2014 at 12:30PM Purdue time. i.e., before class, *not* before midnight. Late submissions will not be accepted.

6 Grading

We will test your code on machines similar to those in the Linux labs used by this course's PSOs. Your grade will be based on correctness in the following criteria:

- Correct assignment of types to fields/methods.
- Generation of a single, unique StaticType for each type in the program. (No redundancies)
- Rejection of type-incorrect programs.
- Correct assignment of expression types to SSA statements.