CMSI-585-Homework-2 Type Checker

Shuyang Liu

February 2024

1 Introduction

In this homework, you are going to implement a type checker for a subset of Java, MiniJava. This document provides the syntax of MiniJava, and a specification of the MiniJava type system. The parser for MiniJava will be generated using JTB and JavaCC.

2 System Requirement

It is recommended to use a Unix-based system for this homework.

- Java 8. There is no restriction on which JDK you should use as long as it uses Java 8.
- Gradle $\geq$ 6.5.1
- `timeout` or `gtimeout` if your system does not have one. Most Linux systems comes with it. On MacOS, you can install this utility program by `brew install coreutils`.

3 The MiniJava Syntax

The syntax of MiniJava in BNF form:

⟨*Goal*⟩ g ::= $mc\ d_1 \ldots d_n$

⟨*MainClass*⟩ mc ::= `class` id { `public static void main (String []` id^S`){`
$t_1\ id_1;\ \ldots\ ; t_r\ id_r; s_1 \ldots s_q\}\}$

⟨*TypeDeclaration*⟩ d ::= `class` $id\ \{t_1\ id_1;\ \ldots\ ; t_f\ id_f;\ m_1 \ldots m_k\}$

⟨*MethodDeclaration*⟩ m ::= `public` $t\ \ id^M\ \ (t_1^F id_1^F, \ldots, t_n^F id_n^F)${`}`
$t_1\ id_1;\ \ldots\ ; t_r\ id_r; s_1 \ldots s_q$ `return` e`;` `}`

⟨*Type*⟩ t ::= `int[]` | `bool` | `int` | id

⟨*Statement*⟩ s ::= `{`$s_1 \ldots s_q$`}` | id `=` e`;` | id`[`e_1`]` `=` e_2`;`
| `if (`e`)` s_1 `else` s_2 | `while (`e`)` s | `System.out.println(`e`);`

⟨*Expression*⟩ e ::= p_1 `&&` p_2 | p_1 `<` p_2 | p_1 `+` p_2 | p_1 `-` p_2 | p_1 `*` p_2 | p_1`[`p_2`]`
| p`.length` | p`.`id`(`$e_1, \ldots, e_n$`)` | p

⟨*PrimaryExpression*⟩ p ::= c | `true` | `false` | id | `this` | `new int[`e`]` | `new` id`()` | `!`e | `(`e`)`

⟨*IntegerLiteral*⟩ c ::= INTEGER_LITERAL

⟨*Identifier*⟩ id ::= IDENTIFIER

4 Visitors

There are four kinds of visitor interfaces.

- `Visitor`: the most basic visitor interface that has no arguments and no return value for each `visit()` method.
- `GJVoidVisitor<A>`: a generic visitor interface that allows a single argument to be passed among the `visit()` methods but no return value. The type of the argument is parameterized by `A`.
- `GJNoArguVisitor<R>`: a generic visitor interface that allows a return value but no argument passed among the `visit()` methods. The type of the return value is parameterized by `R`.
- `GJVisitor<R,A>`: a generic visitor interface that allows both arguments and return values. The type of the argument is parameterized by `A`. The type of the return value is parameterized by `R`.

5 The MiniJava Type System

5.1 Type Environment

A type environment is a finite map from identifiers (variable names) to types. Let $id_1, \ldots, id_r$ be distinct identifiers, then we use the notation $[id_1 : t_1, \ldots, id_r : t_r]$ to denote a type environment that maps each id_i to type t_i, for $i = 1, \ldots, r$. Let A_1, A_2 be two type environments. We define $A_1 \cdot A_2$ as:

$$(A_1 \cdot A_2)(id) = \begin{cases} A_2(id) & \text{if } id \in dom(A_2) \\ A_1(id) & \text{otherwise} \end{cases}$$

where $dom(A_2)$ is the domain of A_2.

5.2 Helper Functions

- The function $\mathsf{className}(.)$ returns the name of a class.
  - $\mathsf{className}(\texttt{class}\ id\ \{\ \texttt{public static void main (String [] }\ id^S\texttt{)\{...\}}\}) = id$
  - $\mathsf{className}(\texttt{class}\ id\ \{\ldots\}) = id$
- The function $\mathsf{methodName}(.)$ returns the name of a method.

$$\mathsf{methodName}(\texttt{public}\ t\ id^M\ \texttt{(...)\{...\}}) = id^M$$

- The function $\mathsf{distinct}(.)$ checks that the identifiers in a list are pairwise distinct.

$$\frac{\forall i \in 1 \ldots n, \forall j \in 1 \ldots n, id_i = id_j \Rightarrow i = j}{\mathsf{distinct}(id_1, \ldots, id_n)}$$

- The function fields returns the type environment constructed from the fields.

$$\frac{\texttt{class}\ id\ \{t_1\ id_1;\ \ldots\ ;t_f\ id_f;\ m_1 \ldots m_k\} \text{ is in the program}}{\mathsf{fields}(id) = [id_1 : t_1, \ldots, id_f : t_f]}$$

- We use the notation $\mathsf{methodType}(id,\ id^M)$ to denote the list of argument types of the method with name id^M in class id together with the return type (or $\perp$ if no such method exists). The result of $\mathsf{methodType}$ is of the form $(t_1, \ldots, t_n) \to t$, or $\perp$.

$$\frac{\begin{array}{c}\texttt{class}\ id\ \{\ldots\ m_1 \ldots m_k\} \text{ is in the program} \\ \text{for some } j \in 1 \ldots k,\ \mathsf{methodName}(m_j) = id^M \\ m_j = \texttt{public}\ t\ id^M\ \texttt{(}t_1^F id_1^F, \ldots, t_n^F id_n^F\texttt{)\{...\}}\end{array}}{\mathsf{methodType}(id, id^M) = (id_1^F : t_1^F, \ldots, id_n^F : t_n^F) \to t}$$

$$\frac{\texttt{class}\ id\ \texttt{\{}\ldots\ m_1 \ldots m_k\texttt{\}} \text{ is in the program} \quad \text{for all } j \in 1 \ldots k,\ \mathsf{methodName}(m_j) \neq id^M}{\mathsf{methodType}(id, id^M) = \bot}$$

5.3 Type Rules

We use the following forms of type judgement.

- $\cdot \vdash g$ means "the goal g type checks".
- $\cdot \vdash mc$ means "the main class mc type checks".
- $\cdot \vdash d$ means "the class declaration d type checks".
- $C \vdash m$ means "if defined in class C, the method declaration m type checks".
- $A, C \vdash s$ means "in a type environment A, if defined in class C, the statement s type checks".
- $A, C \vdash e : t$ means "in a type environment A, if defined in class C, the expression e has type t".
- $A, C \vdash p : t$ means "in a type environment A, if defined in class C, the primary expression p has type t".

5.3.1 Goal

$$\frac{\begin{array}{c}\mathsf{distinct}(\mathsf{className}(mc), \mathsf{className}(d_1), \ldots, \mathsf{className}(d_n)) \\ \vdash mc \\ \vdash d_i \qquad i \in 1 \ldots n\end{array}}{\vdash mc\ d_1, \ldots, d_n} \quad \text{(Goal)}$$

5.3.2 Main Class

$$\frac{\begin{array}{c}\mathsf{distinct}(id_1, \ldots, id_r) \\ A = [id_1 : t_1, \ldots, id_r : t_r] \\ A, \bot \vdash s_i \qquad i \in 1, \ldots q\end{array}}{\begin{array}{l}\vdash \texttt{class}\ id\ \texttt{\{ public static void main (String [] } id^S\texttt{)\{} \\ \quad t_1\ id_1\texttt{;}\ \ldots\texttt{;}\ t_r\ id_r\texttt{;}\ s_1 \ldots s_q\ \texttt{\}\}}\end{array}} \quad \text{(Main)}$$

5.3.3 Class Declarations

$$\frac{\begin{array}{c}\mathsf{distinct}(id_1, \ldots, id_f) \\ \mathsf{distinct}(\mathsf{methodName}(m_1), \ldots, \mathsf{methodName}(m_k)) \\ id \vdash m_i \qquad i \in 1, \ldots k\end{array}}{\vdash \texttt{class}\ id\ \texttt{\{}t_1\ id_1\texttt{;}\ \ldots\texttt{;}\ t_f\ id_f\texttt{;}\ m_1 \ldots m_k\texttt{\}}} \quad \text{(Class)}$$

5.3.4 Method Declarations

$$\frac{\begin{array}{c}\mathsf{distinct}(id_1^F, \ldots, id_n^F, id_1, \ldots, id_r) \\ A = \mathsf{fields}(C) \cdot [id_1^F : t_1^F, \ldots, id_n^F : t_n^F, id_1 : t_1, \ldots, id_r : t_r] \\ A, C \vdash s_i \qquad i \in \{1, \ldots q\} \qquad A, C \vdash e : t\end{array}}{\begin{array}{l}C \vdash \texttt{public}\ t\ id^M\ \texttt{(}t_1^F\ id_1^F, \ldots, t_n^F\ id_n^F\texttt{) \{} \\ \quad t_1\ id_1\texttt{;} \ldots\texttt{;}\ t_r\ id_r\texttt{;}\ s_1 \ldots s_q\ \texttt{return}\ e\texttt{;}\ \texttt{\}}\end{array}} \quad \text{(Method)}$$

5.3.5 Statements

$$\frac{A, C \vdash s_i \qquad i \in \{1, \ldots q\}}{A, C \vdash \{s_1, \ldots, s_q\}} \quad \text{(Block)}$$

$$\frac{A, C \vdash e : \texttt{int}}{A, C \vdash \texttt{System.out.println(}e\texttt{)}} \quad \text{(Print)}$$

$$\frac{A, C \vdash e : \texttt{boolean} \qquad A, C \vdash s}{A, C \vdash \texttt{while (}e\texttt{)}\ s} \quad \text{(While)}$$

$$\frac{A, C \vdash e : \texttt{boolean} \qquad A, C \vdash s_1 \qquad A, C \vdash s_2}{A, C \vdash \texttt{if (}e\texttt{)}\ s_1\ \texttt{else}\ s_2} \quad \text{(If-Else)}$$

$$\frac{A(id) = t \qquad A, C \vdash e : t}{A, C \vdash id\ \texttt{=}\ e\texttt{;}} \quad \text{(Assignment)}$$

$$\frac{A(id) = \texttt{int[]} \qquad A, C \vdash e_1 : \texttt{int} \qquad A, C \vdash e_2 : \texttt{int}}{A, C \vdash id\texttt{[}e_1\texttt{]}\ \texttt{=}\ e_2} \quad \text{(Array Assignment)}$$

5.3.6 Expressions and Primary Expressions

$$\frac{\begin{array}{c} A, C \vdash p : D \\ \mathsf{methodType}(D, id) = (t_1, \ldots, t_n) \to t \\ A, C \vdash e_i : t_i \ \text{ for } i \in \{1, \ldots n\} \end{array}}{A, C \vdash p.id\texttt{(}e_1, \ldots, e_n\texttt{)} : t} \quad \text{(Method Call)}$$

$$\frac{A, C \vdash p_1 : \texttt{boolean} \qquad A, C \vdash p_2 : \texttt{boolean}}{A, C \vdash p_1\ \texttt{\&\&}\ p_2 : \texttt{boolean}} \quad \text{(And)}$$

$$\frac{A, C \vdash p_1 : \texttt{int} \qquad A, C \vdash p_2 : \texttt{int}}{A, C \vdash p_1\ \texttt{<}\ p_2 : \texttt{boolean}} \quad \text{(Less Than)}$$

$$\frac{A, C \vdash p_1 : \texttt{int} \qquad A, C \vdash p_2 : \texttt{int}}{A, C \vdash p_1\ \texttt{+}\ p_2 : \texttt{int}} \quad \text{(Plus)}$$

$$\frac{A, C \vdash p_1 : \texttt{int} \qquad A, C \vdash p_2 : \texttt{int}}{A, C \vdash p_1\ \texttt{-}\ p_2 : \texttt{int}} \quad \text{(Minus)}$$

$$\frac{A, C \vdash p_1 : \texttt{int} \qquad A, C \vdash p_2 : \texttt{int}}{A, C \vdash p_1\ \texttt{*}\ p_2 : \texttt{int}} \quad \text{(Multiplication)}$$

$$\frac{A, C \vdash p_1 : \texttt{int[]} \qquad A, C \vdash p_2 : \texttt{int}}{A, C \vdash p_1\texttt{[}p_2\texttt{]} : \texttt{int}} \quad \text{(Array Element)}$$

$$\frac{A, C \vdash p : \texttt{int[]}}{A, C \vdash p\texttt{.length} : \texttt{int}} \quad \text{(Array Length)}$$

$$A, C \vdash c : \texttt{int} \quad \text{(Integer Literals)}$$

$$A, C \vdash \texttt{true} : \texttt{boolean} \quad \text{(True)}$$

$$A, C \vdash \texttt{false} : \texttt{boolean} \quad \text{(False)}$$

$$\frac{id \in dom(A)}{A, C \vdash id : A(id)} \quad \text{(Variable)}$$

$$\frac{C \neq \bot}{A, C \vdash \texttt{this} : C} \quad \text{(This)}$$

$$\frac{A, C \vdash e : \texttt{int}}{A, C \vdash \texttt{new int[}e\texttt{]} : \texttt{int[]}} \quad \text{(New Array)}$$

$$A, C \vdash \texttt{new}\ id\texttt{()} : id \qquad \text{(New Object)}$$

$$\frac{A, C \vdash e : \texttt{boolean}}{A, C \vdash \texttt{!}e : \texttt{boolean}} \qquad \text{(Negation)}$$

$$\frac{A, C \vdash e : t}{A, C \vdash \texttt{(}e\texttt{)} : t} \qquad \text{(Parenthesis)}$$

6 Specification

You will be given a package called `CMSI-585-hw2.tar` which consists of building scripts and code templates for this homework. The package is downloadable from BrightSpace under "Content" → "Homework 2".

To decompress the package:

```
tar -xvf CMSI-585-hw2.tar
```

Start the homework by setting variables. In `settings.gradle`, set the variable `rootProject.name` to your student ID. For example,

```
rootProject.name = 123456789;
```

for student ID 123456789. In addition, in `gradle.properties`, set the student ID and the homework number. For this homework, it should be `hw1`.

```
myUID=123456789
homework=hw2
```

Your source code should all be placed in `src/main/java/` with `Typecheck.java` as the class containing the `main` method. There is a list of reserved file names that you should avoid. You can find the list in `misc/illegalfiles`.

The program should take inputs from the standard input channel `System.in`, parse the input, and print out exactly one of:

- `Program type checked successfully`
- `Type error`

depending on whether the input is a valid program according to the given type rules in Section 5.3. In addition, there is a 5s timeout set by the grading script. All test cases should be checked under 5s.

To build the project,

```
gradle build
```

There are 10 public test cases included in `testcases/hw2/`. To automatically use the test cases and compare your results with the expected results:

```
gradle pregrade
```

You are also encouraged to write more custom test cases and add them to the `testcases/hw2/` directory. Each test case is paired with its expected output with `*.out` extension. Please see the files `testcases/hw2/` directory for examples.

7 Submission

You are required to submit only your source code in a single `tar` file. To generate such a `tar` file,

```
gradle sourceTar
```

The resulting `tar` file can be found at `build/distributions/<UID>.tar`.
Please only submit this tar file, not the whole `CMSI-585-hw2.tar` package.
Please do not submit any file from `src/parse/java/` directory.

8 Grading

This homework will be graded based on the number of test cases passed by your program. You are given 16 public test cases and there will be similar hidden test cases on the testing server.
You are **not** allowed to use any third-party library. If there is any such use in the code, you will receive a 0%.

Since there are only two kinds of output, it's tempting to just print out one of the required output without actually implementing anything to get some scores. Although I do not plan to look into the details in everyone's code, I will roughly skim through several random submissions to check whether they are indeed trying to implement homework 2. If not, a 0% will be given.