编译原理PA2实验报告

提前上学2018 李嘉图

2018 年 11 月 23 日

1 主要任务

PA2的主要任务是，在PA1-A的基础上，通过遍历AST建立符号表，并进行静态类型检查。

2 具体流程

为了对新增语言特性进行语义分析，编译器对AST进行了两趟遍历，第一次在AST上建立符号表，这部分代码主要在`typecheck/BuildSym.java`；第二次进行静态类型检查，这部分代码主要在`typecheck/TypeCheck.java`。而类型被定义在：`symbol/Symbol|Function|Variable|Class.java`中。

对于每种需要支持的语句，应当在`typecheck/BuildSym.java`中建立相关的符号表，进行变量重名等的检查，并给类、函数和变量在符号表中分配位置；在`typecheck/TypeCheck.java`中进行类型检查，当出现错误时，尽可能推导表达式“应有的”类型，以减少重复报错，发现更多的错误。另外，由于自动类型推导和`foreach`语句比较特殊，要进行特殊的分析和处理。

3 实验操作

3.1 类的浅复制

由于浅复制不涉及建立符号表，对于`typecheck/BuildSym.java`不需要进行修改。在`typecheck/TypeCheck.java`中新增函数`visitScopy`，依次

访问其Scopy.left/source。由于在PA1-A中将Scopy.left建立为了Expr/LValue，不需要再对此标识符是否合法进行检查。

如果Scopy.left的类型为错误，则不报错以防错误扩散；如果类型不是错误，但并不是ClassType，要报左边不是合法类的错误，再对Scopy.source做同样的检查；如果左边是合法的类，但左右类型不同，要报复制两边类型不同的错误。

3.2 sealed修饰词

要支持sealed，首先要对symbol/Class.java进行修改，在类Symbol/Class中加入private boolean sealed，修改构造函数并提供接口Class.isSealed()、Class.setSealed(boolean)以便对属性进行查询和修改；在typecheck/BuildSym.java中的函数BuildSym.visitTopLevel中新建类对应的语句，在构造函数将sealed传入。

为了实现类型检查时的报错，应当在typecheck/TypeCheck.java中的TypeCheck.visitClassDef中新增：在符号表中查询父类的属性，如果父类存在且是sealed，那么报sealed类被继承的错误。

3.3 串行条件卫士语句

首先在typecheck/BuildSym.java中加入函数visitGuardedStmt、visitIfSubStmt以建立符号表，在typecheck/TypeCheck.java中加入函数visitGuardedStmt、visitIfSubStmt进行类型检查，这些函数都只需要依次访问所有儿子节点即可。唯一需要修改的是visitIfSubStmt，需要调用函数TypeCheck.checkTestExpr对条件卫士语句的判断表达式进行检查。

3.4 简单的类型推导

对于类型推导的支持分为四步：

1. 在type/BaseType.java中新增一种名为UNKNOWN的类型，表示这个变量的类型还没有确定。修改此类的函数BaseType.compatible(Type)，如果传入的参数类型是UNKNOWN，则返回true。

2. 在Tree/Var中加入变量Variable symbol，表示这个语句创建出的变量；在typecheck/BuildSym.java中加入函数visitVar，内容类似新定义变量的visitVarDef，即检查变量名是否已经被其他变量使用，如果没

有，则在符号表中加入新的变量，并将该语句的symbol设置为新的变量。将此语句的type和新变量的type设置为UNKNOWN。

3. 在typecheck/TypeCheck.java中加入函数visitVar，如果var.type是null，要将其设为UNKNOWN（这是因为当其出现在表达式中是符合语法而不符合语义的，这样可以检测出更多的错误）。并将绑定变量的类型设置为当前语句的类型（这是使用继承属性，利用visitAssign或visitForeachStmt中的推导结果）。

4. 在typecheck/TypeCheck.java中的函数visitAssign加入类型推导，即当赋值语句左边的类型是UNKNOWN时，要将其类型设置为右边的类型并访问之。

3.5 数组初始化常量表达式

由于数组初始化常量表达式是一种二元运算符，要在typecheck/TypeCheck.java的函数TypeCheck.checkBinaryOp中增加一种case，并做类型检查：

1. 判断数组初始化次数表达式是否是BaseType.INT，不是则报重复次数不是整数的错误；

2. 左边是否不是BaseType.VOID或者BaseType.UNKNOWN，否则报数组元素必须是已知、非VOID元素的错误。

3.6 数组下标动态访问表达式

为了便于检查，首先给Type/ArrayType新增一个返回元素类型的接口ArrayType.getElementType()，同时在父类新建Type.getElement()并返回错误类型。

在typecheck/TypeCheck.java中新增函数visitArrayDefault并依次访问儿子，对其做类型检查：

1. 如果数组不是合法的数组，给出对非数组使用了数组操作的错误；

2. 如果下标不是整数，给出数组下标不是整数的错误；

3. 如果ArrayDefault.name的元素类型和ArrayDefault.def不是同一种类型，给出默认元素和数组元素不同的错误；

如果数组是合法的，优先使用数组元素类型作为结果类型，否则使用默认类型作为结果类型。

3.7 数组迭代语句

支持数组迭代语句需要三步：

1. 首先在tree/Tree.java的类Tree/ForeachStmt中增加一个变量LocalScope associatedScope，用于管理语句的作用域；增加一个变量Block scopeBlock，表示设定用于管理作用域的Block；增加变量Tree.VarDef varDef和Tree.Var var用于变量处理的方便。

2. 在typecheck/BuildSym.java中增加函数visitForeachStmt，当其包含的代码段foreachStmt.stmt是Tree.Block时，直接令scopeBlock = stmt；否则新开一个Tree.Block，将stmt加入到其中。利用scopeBlock新开符号表。根据循环变量是否需要类型推导，访问var或者varDef，然后访问条件语句和语句块。关闭符号表。

3. 在typecheck/TypeCheck.java中新增函数visitForeachStmt，利用scopeBlock新开符号表。如果迭代数组不是数组，给出对非数组使用了数组操作的错误；如果迭代变量类型错误，报迭代变量类型错误的错误；如果使用自动类型推导，将var的类型设置为需要的类型，并访问之。最后访问条件和Block，并关闭表。