PA2 实验报告

计 76 沈诣博 2017011427

本人在这一次而实验中完成了以下任务:

1. 类的浅复制的实现:
根据新的题意, 修改了/frontend/parser.y, 将 scopy(String,Expr) 的文法改为了 scopy(LValue,Expr).
修改了/tree/tree.java 文件, 在 ASL 树中增加 Tree.Scopy 类, 存储 Scopy 语句中的 ident 和 expr, 以及它们的相应位置.
修改了/typecheck/BuildSym.java 文件, 增加了 visitScopy 的函数重载, 在第一轮遍历的时候访问 Expr 语句, 以确认该语句内是否有对象的创建.
修改了/typecheck/TypeCheck.java 文件, 重载了 visitScopy 函数, 实现了如下功能:

1. 遍历访问其中的 LValue 语句和 Expr 语句

2. 检查 LValue 语句是否归约为 class 变量, 否则报告 BadScopyArgError 错误

3. 在上一条没有错的情况下检查 LValue 语句是否和 Expr 语句类型相同, 否则报告 BadScopySrcError 错误

4. 检查 Expr 语句是否归约为 class 变量, 否则报告 BadScopyArgError 错误

2.sealed 语句的实现:
修改了/frontend/parser.y, 新加入 sealed 文法
修改了/tree/Tree.java 文件, 在 ClassDef 类中增加一个布尔变量判断是否为 sealed 类
修改了/typecheck/BuildSym.java 文件, 增加了一个 set<String> 用来存储 sealed 的类名, 在重载的 visitTopTree 函数中两次遍历所有设计到的类, 第一次填充这个 set, 第二次判断是否继承 sealed 类

3. 串行条件卫士的实现:
修改了/frontend/parser.y 和/tree/Tree.java 文件, 具体修改方式如 PA1, 增加了文法和相应的 AVL 树节点.
修改了/typecheck/TypeCheck.java 文件, 重载了 visitGuardStmt(新增串行

条件卫士) 和 visitSubStmt(每一个条件-操作关系) 函数, 遍历访问其中的条件句, 判断是否合法.

4. 简单类型推导的实现:
修改了/frontend/parser.y 文件, 在构造 LValue 的时候增加了类型推导文法
修改了/tree/Tree.java 文件, 在 AVL 树中增加了 VarValue 类型继承自 LValue, 新增 String 类型存储推导类型的名字
修改了/type/BaseType.java 文件, 在 BaseType 类中增加了 UNKNOWN 静态对象, 用来临时标明类型推导变量.
修改了/typecheck/BuildSym.java 文件, 重载了 visitVarValue 函数和 visitAssign 函数.

对于后者, 遍历其等号左的语句, 从而遍历到所有合法的 VarValue 变量.

对于前者, 将所涉及的变量 type 类型设置为 BaseType.UNKNOWN, 并判断有误定义冲突从而存储进符号表或者报错.
修改了/typecheck/TypeCheck.java, 重载了 visitVarValue 函数, 修改了 visitAssign 函数.

对于前者, 将所涉及的变量的类型设置为 BaseType.UNKNOWN, 从而方便报错

对于后者, 新增了简单类型推导的判定, 若通过类型判定为推导型, 则根据等号右侧式子的类型, 或者修改符号表相应域中的变量定义, 或者报错.

5. 数组相关语句的实现:
修改了/frontend/parser.y 文件, 增加了"Expr %% Expr" 语句处理初始化语句,"Expr[Expr] default Expr" 的 default 语句和 foreach 语句.
对于 foreach 语句, 如果文法中最后的一个 Stmt 非 StmtBlock 区块, 则新建一个 StmtBlock, 存储该语句并加入 AVL 树.
修改了/tree/Tree.h 文件, 增加了 TimeArrayConst 结点处理初始化语句,DefaultConst 结点处理 default 语句,ForeachLoop 结点处理 foreach 语句
对于 ForeachLoop 结点, 为了应对一般声明变量和类型推导变量两种推导方法, 内部封装有 VarValue 和 VarDef 两个指针, 供调用时选择
修改了/type/BuildSym.java 文件, 在其中重载了 visitForEachLoop 函数, 按照顺序完成了以下工作:

1. 在符号表 table 表中加入新的临时域

2. 检查 foreach 语句, 并且将定义的变量或者推导型变量加入符号表

3. 检查 block 区块中语句

修改了/typecheck/TypeCheck.java 文件, 依次重载了 visitTimeArrayConst,visitDefaultConst 和 visitForeachLoop 函数

对于重载的 visitTimeArrayConst 函数, 其递归遍历了左右两个表达式, 并且判定左侧表达式是否合法和右侧表达式是否是 int 类型, 从而决定报错或者通过

对于重载的 visitDefaultConst 函数, 其递归遍历了左表达式, 取数的秩, 以及右表达式.

, 并依次进行如下操作:

1. 判断左表达式是否合法的数组类型, 否则报错, 并且将 default 语句的类型置为有表达式的类型, 若右表达式为空类型则置为错误.

2. 判断秩是否为 int 类型, 否则报错.

3. 若左表达式合法, 则判断右表达式类型是否等于左表达式, 否则报错.

对于重载的 visitForeachloop 函数, 我在遍历表达式的同时跟随遍历中的表达式进行了如下操作:

1. 判断遍历变量是否是类型推导变量

2. 检查被遍历的是否是合法数组, 非则报错, 是则检查和遍历变量的兼容情况. 如果遍历变量是类型推导变量, 则类型置为数组基类

3. 检查条件语句是否布尔类型

4. 遍历域中语句块

此外, 为了兼容修改的操作, 我在/typecheck/中间的 lexer.c 和 Semvalue.java 中间进行了修改. 对于后者, 我为了兼容串行卫士, 我加入了一个 SubStmt(子语句) 类型的变量和列表.