PA2实验报告

计73 林俊峰

2017011303

实现思路

新特性1：抽象类

错误0

> 不允许抽象方法重写基类中的非抽象方法

在 `visitMethodDef` 时多加入一条判断即可

错误1

> 若一个类中含有抽象成员，或者它继承了一个抽象类但没有重写所有的抽象方法，那么该类必须声明为抽象类。

根据文档提示，每个类使用“未被重载的抽象方法”列表来实现抽象类的判断。每处理一个类时，先处理其父类，将自己的该列表初始化为父类的列表，每遇到一个抽象方法，就将其加入列表(若列表中已存在该方法，则不做处理)；每遇到一个非抽象方法，判断其是否在列表中，若存在则删去(即完成重写)，最后若列表非空，则该类应为抽象类。

错误2

> 抽象类不能使用 `new` 进行实例化。

在第二轮遍历时， `visitNewClass` 处进行特判即可

新特性2：局部类型推导

在第一次遍历时无法确定类型，但是要创建symbol, symbol的type暂时为null; 在第二次遍历的时候通过先处理 `initVal` 推导出变量类型，再修改对应的symbol的type，在这过程中要判断类型是否为 `Void`

新特性3： First-class Functions

函数类型

在 `TypeLitVisited` 接口中重写 `visitTLambda` 方法

Lambda 表达式作用域

使用一个栈来记录lambda作用域，每次进入lambda作用域时压栈，离开时出栈，由此可以判断当前是否处于lambda作用域以及获取它。同时使用另一个栈来记录正在赋值lambda表达式的符号。在 `visitLocalVarDef` 方法中判断正在定义的符号以及使用的符号是否为捕获的外层的非类作用域的符号(通过借助栈在当前lambda作用域能否查找到该符号来判断)。

Lambda 表达式返回类型

在这里我给 `AST` 的 `Stmt` 节点增加了一个成员 `returnTypes` 来记录一个语句拥有的所有 `return` 语句的返回类型，在最上层的 `block` 收集了一个lambda表达式中的所有 `return` 语句的返回类型，求出上界赋给Lambda的返回类型即可(要特判没有 `return` 以及类型为 `null` 的情况)

函数变量

在 `Typer` 修改 `VarSel` 的处理节点即可

函数调用

由于 `函数变量` 中已经对 `ValSel` 节点增加了对函数的处理，故原先的 `Call` 节点的大部分处理可以弃用，只需保留对参数列表的处理即可。同时需要增加对lambda表达式的特判，因为是新的错误类型。

符号表格式化打印

在这里需要修改symbol和scope的结构。我新增了 `LambdaScope` 来表示lambda表达式的参数域，将 `LocalScope` 的 `nested` 成员变量修改为同时支持 `LocalScope` 和 `LambdaScope` ，最后在 `PrettyScope` 的 `pretty` 方法中增加 `LambdaScope` 的处理

问题回答

Q1：实验框架中是如何实现根据符号名在作用域中查找该符号的？在符号定义和符号引用时的查找有何不同?

A1：作用域基类Scope维护了一个符号表symbols，类型为map<String, Symbol>，符号声明时向表中插入符号，查找分为两种，一种是get, 以符号名为键值直接在map里查找，用来判断某符号是否在某作用域声明; 还有一种是lookup, 用于不断地向上层scope查找。符号定义时用的是第一种查找，用来判断是否重复定义，因为一个scope内符号只能定义一次; 符号引用时用第二种查找，因为引用的符号可能在上层scope定义。

Q2: 对 AST 的两趟遍历分别做了什么事？分别确定了哪些节点的类型?

A2: 第一趟遍历由Namer完成，生成符号信息，建立了符号表，确定了类、类方法和非 `var` 类型变量的类型 ;第二趟遍历由Typer完成，进行类型检查，确定了lambda表达式、 `var` 类型变量、函数调用的类型。

Q3: 在遍历 AST 时，是如何实现对不同类型的 AST 节点分发相应的处理函数的？请简要分析。

A3：通过访问者模式，在两趟遍历时访问者分别为Namer和Typer。只需要在Name和Typer中实现不同节点的visit方法，在相应节点调用accept方法时就会执行对应的visitor的visit方法进行处理。