PA2 report

大致思路

ABSTRACT

这个比较简单，对于每个类维护一个未被重载的抽象类列表，判断一个类是否必须是抽象类的时候直接看这个列表是否为空。其他的部分都是简单的判断。

VARTYPE

定义变量的时候把类型设为 `var`，在类型推导时把类型设为推导出的类型。

FUNTYPE

（吐槽一下，为什么有些地方写的是 `funtype`，有些地方写的是 `functype`。）

`FunType` 都不用自己写了，只需要写一个 `VisitTLambda`。

CALL

就在 `visitVarSel` 的时候判断一下结果是什么类型的，如果是变量就调用原来的 `visitVarSel` 中的代码，如果是方法就调用原来 `typeCall` 的代码。

`visitCall` 比较简单。

LambdaScope

加了一个 `LambdaScope` 和 `LambdaSymbol`。然后 `Namer` 中要把所有包含 `Expr` 的部分 visit 一下。

LambdaReturnType

对于每个 `Stmt` 类型的节点都记下子树内所有的 `ReturnStmt` 和 `ReturnType`，在 `visitLambdaDef` 里面统一处理。另外还实现了 `getUpperTypeBound` 和 `getLowerTypeBound`。

遇到的问题

有一些细节文档里面没有，要自己看测例去拟合。

还有一些细节文档里没说测例里面也没有。

其他问题修完锅就忘了。

要求回答的问题

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?

在作用域栈中查。每个作用域栈中维护所有符号的 `Set`，查询时从栈顶往下依次在每个作用域栈中找，找到就推出。

在定义时，如果不是成员变量，就在当前作用域到上一个 `FormalScope` / `LambdaScope` 以及 `GlobalScope` 中查找。否则在所有作用域中查找。另外，在 `visitLocalVarDef` 即变量定义节点的时候要先把左侧正在定义的变量从作用域栈中删掉再 visit initVal，这样就不需要手动 ban 掉一些变量了。

在引用时，在是 `LocalScope` 且位置在当前位置之前部分以及其他包含该作用域的作用域中查找。

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？

第一遍：建立符号表，判断每个新定义的变量是否和已经定义的变量冲突等，确定了输出中明确给出了类型的节点的类型，例如大部分 `LocalVarDef`、`MethodDef` 等。

第二遍：类型推导，判断每个函数的返回值和实际 `Ruturn` 语句的返回值是否对应，判断引用的变量是否存在等，确定了 `Expr`、`LambdaDef` 等需要推导类型的节点的类型（就是第一遍还没确定类型的节点的类型）。

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。

使用了访问者模式。

对每一个 AST 节点类型都定义了一个成员方法 `accept(Visitor<C> v, C ctx)`，其中 `v` 是我们处理使用的实例，`ctx` 是一些其他信息例如作用域栈。然后在该方法中调用实际处理该节点的方法 `v.visitLambdaDef(this, ctx);`，这样当我们需要对一个节点进行处理的时候可以直接调用 `a.accept(this, ctx)`，这样就会依据 `a` 的实际类型调用不同的类的 `accept` 方法，进而调用不同的处理的函数进行处理。

Reference

参考了实验指导中给出的 维护每个类未被重载的 abstract method 的方法、计算类型上界的方法、新建 `Lambda Scope` 和 `Lambda Symbol` 。