编译原理 PA2

计 71 张程远 2017011429

1 工作简介

1.1 前期准备

首先把 PA1-A 原始框架升级为 PA2 的框架，然后手动把 PA1-A 的修改合并到 PA2 的框架中。如果 PA1-A 使用了足够鲁棒的改法，应能使合并完的框架编译通过。此时的测试点进度应该为 2/40。

1.2 abstract

首先做些简单的修改：模仿 static 添加 abstract 的输出；在 visitTopLevel 中定位到查找 main 函数的逻辑，判断是否抽象；在 typer.visitNew 里判断 new 的对象是否是 abstract。然后在 ClassSymbol 里放一个维护函数名的列表，记录抽象类中未被实现的抽象方法名。在 visitClassDef 的时候，把父类的列表复制过来，自己类里有抽象方法就加入，实现了父类的就删除，如果父类列表没有子类又抽象了报错。最后如果子类不是抽象类且列表长度大于 0 就报错。再在 abstract-OK 里修补一下，达到 7/40。

1.3 var

如果 namer 里发现了 var，那么 typer 里直接把右面的类型复制给 stmt.symbol。8/40。

1.4 函数类型

在 typeLitVisited 里加入 visitTLambda，它本身是 Funtype。对于 Tlambda 存储的 argType 和 returnType 一一访问，返回一个新的 FuncType。再补补 basic 测例里的东西，这时候 20/40。

1.5 Lambda

首先定义 LambdaScope 和 LambdaSymbol，参照 FormalScope 和 MethodSymbol。需要注意的是，body 会被自动加入 LambdaScope 下属的 LocalScope 里，但 expr 不会。所以要手动开个 LocalScope(LambdaScope)。另外，如果 LambdaScope 在 LocalScope 里，还要让 LocalScope 能够继续向下访问到 LambdaScope，所以要 LocalScope.setNested(LambdaScope)。

Typer 里同样先做一些访问。此时 typer 要定出 LambdaSymbol 的类型，需要 returnType 和 ArgType。这里首先收集所有 return。我们需要判断 return 都在哪些 Lambda 域里面，因此需要知道当前 Lambda 的嵌套深度，故在 stack 里加了 num 来处理。如果当前 num > 0，

就在 Stack 里找到距离自己最近的 Lambda 域，把 return 加到这个域维护的 List 里。
接下来按照实验指导书算法推导类型的上下界，拿到 FuncType。算法不再赘述。写到这里 Lambda 已经有了基本的样子。这只能通过 1 个点，然后对着错误加一些修改，例如 visit 一些本来不会被 visit 到的部分，维护 lambda 表达式的 leftname 以便判断 UndeclVarError 等，能够完成 28/40。

1.6 Call

检查 Call 的测例后发现对 Call 的处理需要重新写。首先判断 expr 的类型，如果是 void 则看做 error，如果不是函数类型就直接返回，如果是 VarSel 需要加入 LambdaBadCountError 的错误，如果是 Lambda 或者连环调用需要检查参数，诸如此类。同时将一部分功能转移到 VarSel 中实现。此时已经能够到达 34/40，余下 6 个再一个一个对着错误调整代码，最终过到 40 个。

2 思考题

Q1. 实现查找符号：每个作用域里有一个 map，键值对为 name 和 symbol，通过 declare 语句维护它的正确性。作用域通过 find 寻找 symbol，stack 里提供了 findwhile 来遍历每一个栈中的 scope 并访问 symbol，并以此为基础提供了 lookup 等含有一些判断条件的函数。
有何不同：符号定义时使用 findconflict 函数，这一函数有一定的判断条件，即如果当前作用域是 Formal、Lambda 或者 Local，那么符号就不能与 formal、Lambda、local 或全局中的符号冲突；如果当前是其他 Scope 如 Class 和 Global，就不能与任何定义冲突。
引用的时候则是调用 LookupBefore 找之前的定义。

Q2：

第一问：第一次声明了作用域与符号，把这些符号与作用域关联起来，并报与符号定义时有关的错误；第二次根据第一次构建的符号与作用域表实现引用符号、类型检查、类型推导的功能。

第二问：第一次确定的节点类型包括 classdef、methoddef、vardef、typelit 的各个派生类和不包含 var 的 Localvardef，包括自己构建的 Tlambda；第二次确定了 Expr 的派生类（在第一次的时候这些都需要 visit），包括 typelit/unary/binary/newarray/newclass/this/varsel/indexsel/call/classtest/classcast/readint/readline 以及定义 var 的 localclassdef。

Q3：

使用访问者模式，每个节点里都提供了访问者进行访问的函数，由被访问者自己调用这些函数，每个访问者可以重写访问方法。