PA1-A Report

计71 钟闰鑫 2017010306

实现内容

抽象类

抽象类的设计与`static`类似

- 在`syntax/ast.rs`中的函数`FuncDef`和类`classDef`中添加一个`bool`类型的`abstract_`，修改`print`里面`classDef`和`FieldDef`对`print`的定义，原来框架用宏来定义`classDef`的`print`，这里将其直接展开，在内部添加判断`abstract_`的逻辑，对于`FieldDef`内部的`match`，也是直接在内不添加判断逻辑即可
- 由于抽象函数的函数体为空，因此将函数`FuncDef`中原来的`body`定义由`Block<'a>`改成`Option<Block<'a>>`，修改`print`里面的抽象树输出，使其适应新`body`的定义(`Option`的输出已在宏中定义，因此直接调用`Option`的`print`即可)
- 在`syntax/parser.rs`中添加两个新语法规则

```
FuncDef  -> Abstract Type Id LPar VarDefListOrEmpty RPar Semi
ClassDef -> Abstract Class Id MaybeExtends LBrc FieldList RBrc
```

  没有函数体则用`None`来替代原来的`body`

局部类型推断

将`syntax/ast.rs`中`VarDef`里的`syn_ty`由原来的`SynTy<'a>`改成`Option<SynTy<'a>>`，当为`None`时表示为推断类型(即`Var`)

First-class Functions

- 新增函数类型，在`syntax/src/ty.rs`中增加`SynTyKind`新变种`Lambda(Vec<SynTy<'a>>)`，内部`Vec`[0]为返回值类型，[1..]为参数类型

  并在`print/ast.rs`对于`SynTy`的`print`定义中增加对于`SynTyKind::Lambda`的逻辑
- 表达式新增`Lambda`表达式，因此在`syntax/src/ast.rs`中添加新结构

```rust
pub mod Lambda<'a> {
    pub param: Vec<&'a VarDef<'a>>,
    pub kind: LambdaKind<'a>,
}
pub enum LambdaKind<'a> {
    Expr(Box<Expr<'a>>),
    Block(Block<'a>),
}
```

  其中`LambdaKind`区分block lambda和expression lambda

在`ExprKind`这个`enum`中添加`Lambda(Lambda<'a>)`作为新的表达式类型，并在`print/ast.rs`中添加其打印逻辑

- 在`syntax/parser.rs`添加新的符号`=>`，设定其优先级为最低

  添加函数类型`SynTy`产生式

  ```
  Type -> Type LPar TypeListOrEmpty RPar
  TypeListOrEmpty -> TypeList
  TypeListOrEmpty ->
  TypeList -> TypeList Comma Type
  TypeList -> Type
  ```

  为`lambda`函数表达式添加2个产生式

  ```
  Expr -> Fun LPar VarDefListOrEmpty RPar RightArrow Expr
  Expr -> Fun LPar VarDefListOrEmpty RPar Block
  ```

  并把原来的`Expr -> VarSel LPar ExprListOrEmpty RPar`改为

  ```
  Expr ->  Expr LPar ExprListOrEmpty RPar
  ```

  - 注意此时会出现对于`LPar`移进/规约冲突

    即程序当看到下一个符号为`LPar`时，不确定当前是要对栈顶进行表达式的规约还是移进`LPar`

    比如`a + b(1)`，当栈中为`a+b`时，先规约`a+b`为一个表达式还是先移进`(`会产生不同的结果，框架在这种情况下采用了移进，这也正是正确的操作

    为了消除warning，要对`LPar`进行优先级的规定，从`a+b(1)`的例子可以看出，`LPar`的优先级至少要比这些算术操作符等要高，因此最后选择将`LPar`放到`RPar`同一优先级，由此解决了所有warning

问题回答

Q1

> 有一部分AST节点是枚举类型。若节点B是枚举类型，节点A是他的一个variant，那么语法上会不会A和B有什么关系?限用100字符内一句话说明。

A在语法上是B的一个子类，A可以认为是一种B

Q2

> 原有框架是如何解决空悬else(dangling-else)问题的?限用100字符内说明。

`else`的优先级为最大(比`empty`高)，因此遇到`else`时会优先选择移入，最终会使得`else`会匹配最近的`if`从而解决空悬问题

Q3

> 输入程序lex完得到一个终结符序列，然后构建出具体语法树，最后从具体语法数构建抽象语法树。这个概念模型与框架的实现有什么区别？我们的具体语法树在哪里？限用120字符内说明

在框架的实际实现中没有具体语法树，而是由终结符序列直接构造出了抽象语法树