Stage-3 实验报告

计03 王文琦 2020010915

实验内容

Step 7

`Step 7` 整体比较简单，主要是引入了局部作用域和块语句，主要修改两处。

- 首先是需要在 `backend/dataflow/cfg.py` 的 `CFG` 模块中添加 `unreachable` 函数。具体的做法是根据当前的 id 进行 DFS，方向是当前访问节点的 `prev` 节点。在访问的过程中使用 `stack` 来模拟回溯的过程，使用 `set` 来对已经访问的节点去重。递归 DFS 的递归基是已经访问的 `set` 中包含了程序的入口语句块 `0`。
- 然后我们就需要在 `frontend/typecheck/namer.py` 的 `visitBlock` 函数中，在按顺序检查 `Block` 的所有子语句之前，`open` 一个局部作用域并压栈，在所有检查结束之后 `close` 这个局部作用域并且出栈。

Step 8

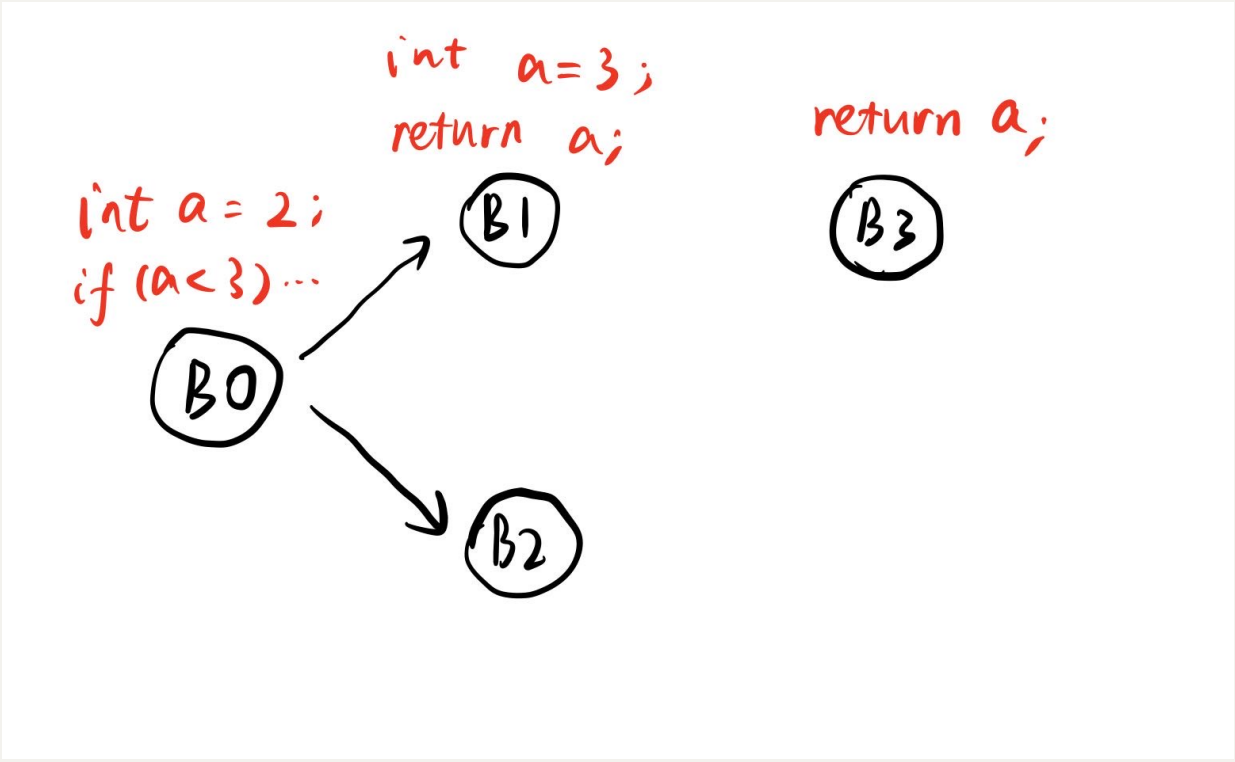
`Step 8` 加入了循环语句。内容有些繁琐，但主要分为以下几个部分：

- 在 `frontend/ast/tree.py` 中添加 `For/DoWhile/Continue` 的抽象语法树节点。
- 在 `frontend/ast/visitor.py` 中添加相应的 visit 函数。
- 在 `frontend/lexer/lex.py` 中添加保留字。
- 在 `frontend/parser/ply_parser.py` 中定义相应的语法规则（构造相应的语法树节点）。
- 在 `frontend/typecheck/namer.py` 中添加相应的语法检查规则，这里需要注意 for 语句需要打开两个新的作用域，并且最终都需要出栈。
- 在 `frontend/tacgen/tacgen.py` 中生成相应的三地址码。

思考题

1. **Step 7:** 请画出下面 MiniDecaf 代码的控制流图。

```
int main(){
 int a = 2;
 if (a < 3) {
     {
         int a = 3;
         return a;
     }
     return a;
 }
}
```



2. **Step 5:** 将循环语句翻译成 IR 有许多可行的翻译方法，例如 while 循环可以有以下两种翻译方式，从执行的指令的条数这个角度（`label` 指令不计算在内，假设循环体至少执行了一次），请评价这两种翻译方式哪一种更好?

```
第一种:
1. label BEGINLOOP_LABEL: 开始下一轮迭代
2. cond 的 IR
3. beqz BREAK_LABEL: 条件不满足就终止循环
4. body 的 IR
```

```
5. label CONTINUE_LABEL：continue 跳到这
6. br BEGINLOOP_LABEL：本轮迭代完成
7. label BREAK_LABEL：条件不满足，或者 break 语句都会跳到这儿

第二种：
1. cond 的 IR
2. beqz BREAK_LABEL：条件不满足就终止循环
3. label BEGINLOOP_LABEL：开始下一轮迭代
4. body 的 IR
5. label CONTINUE_LABEL：continue 跳到这
6. cond 的 IR
7. bnez BEGINLOOP_LABEL：本轮迭代完成，条件满足时进行下一次迭代
8. label BREAK_LABEL：条件不满足，或者 break 语句都会跳到这儿
```

循环执行 N 次时：第一种需要 4N+2 条指令跳出，第二种需要 3N+2 条指令跳出。**故第二种翻译方式更好**。