Tiger-Compiler Report

词法分析

我们使用 flex 进行词法分析。

首先我们定义了两种状态， 与 ，当解析到 "/*" 与 "\""时分别转换状态。当在 状态时，解析到 "*/" 就减少当前注释嵌套层数，直到层数变为 0 就跳出 状态。当在 状态时，解析到 "\"" 就跳出 状态，并且将字符串存在当前的 BUFFER 里返回。这里的 BUFFER 的初始大小是 10 ，如果新增字符串的长度加上现有长度大于当前的 capacity ，我们就重新分配一个长度为 (2*current capacity +length) 的空间，并把原来的字符串链接上新的并复制过去。当在初始状态时，parse 所有的关键字，ID, INT 与运算符。

由于大多数规则都是以关键字或者单个表达式的情形给出的，我们这里不需赘言，关键的正则表达式只有：

ID	**[a-zA-Z][a-zA-Z0-9_]***
INT	[0-9]+

数据结构跟原理如课本所述，应当是基于 *NFA* 与 *convert NFA to DFA* 两者。

语法分析

我们使用 yacc 进行文法分析。

文法描述详见 `tiger.y` 。

yacc 在分析整个文档的同时，建立好了一个其数据结构定义在 `absyn.h` 中的抽象语法树，其根节点为 `absyn_root` 。 `absyn.h` 包括 `A_SimpleVar` , `A_FieldVar` , `A_CallExp` 等以 `A_` 开头的函数，用来构造 `A_var_` , `A_exp_` , `A_dec_` , `A_ty_` 等数据结构，作为树的节点。

中间代码

`absyn_root` 随后被 `semant.h` 模块利用，完成对抽象语法树到中间表示树的翻译。 `semant.h` 有四个接口函数，分别为： `transVar` , `transExp` , `transDec` , `transTy` ，分别负责对变量，表达式，声明，类型的翻译。但 `semant.h` 不直接对 IR 树进行操作，而是通过 `translate.h` 进行局部的翻译。在翻译的过程中， `translate.h` 还会调用 `frame.h` `symbol.h` `temp.h` 中的成员，用来维护栈帧、分配内存与寄存器，维护符号表，生成临时标号和变量。

`translate.h` 生成的片段列表存放在 `fraglist` 中，当中间树翻译完毕，可以遍历这个 List ，并调用 `printtree.h` 中的 `printStmList` 来进行中间树的可视化。下面是几个样例：

Tiger

```
let
    var arr1:=1
in
    arr1
end
```

IR Tree

```
 EXP(
  ESEQ(
   EXP(
    ESEQ(
     MOVE(
      TEMP t101,
      CONST 1),
     CONST 0)),
   TEMP t101))
```

Tiger

```
/* correct if */
let
function main(): int=
  if (10 > 20) then 30 else 40
in
end
```

IR Tree

```
 LABEL L1
EXP(
 ESEQ(
  CJUMP(EQ,
   ESEQ(
    MOVE(
     TEMP t102,
     CONST 1),
    ESEQ(
     CJUMP(GT,
      CONST 10,
      CONST 20,
      L2,L3),
     ESEQ(
      LABEL L3,
      ESEQ(
       MOVE(
        TEMP t102,
        CONST 0),
       ESEQ(
        LABEL L2,
        TEMP t102))))),
   CONST 0,
   L5,L4),
  ESEQ(
   LABEL L4,
   ESEQ(
    MOVE(
     MEM(
      TEMP t103),
     CONST 30),
    ESEQ(
     JUMP(
      NAME L6),
     ESEQ(
      LABEL L5,
      ESEQ(
       MOVE(
        MEM(
         TEMP t103),
        CONST 40),
       ESEQ(
        JUMP(
         NAME L6),
        ESEQ(
         LABEL L6,
         TEMP t103)))))))))
EXP(
 ESEQ(
  EXP(
   CONST 0),
  CONST 0))
```

Tiger

```
/* define valid mutually recursive procedures */
let

function do_nothing1(a: int, b: string)=
                do_nothing2(a+1)

function do_nothing2(d: int) =
                do_nothing1(d, "str")

in
        do_nothing1(0, "str2")
end
```

IR Tree

```
 LABEL L1
 EXP(
  CALL(
   NAME L2,
    BINOP(PLUS,
     TEMP t103,
     CONST 1)))
 LABEL L2
 EXP(
  CALL(
   NAME L1,
    NAME L3))
 EXP(
  ESEQ(
   EXP(
    CONST 0),
   CALL(
    NAME L1,
     NAME L4)))
```

Tiger

```
let

 type any = {any : int}
 var buffer := getchar()

function readint(any: any) : int =
 let var i := 0
     function isdigit(s : string) : int =
                 ord(buffer)>=ord("0") & ord(buffer)<=ord("9")
     function skipto() =
       while buffer=" " | buffer="\n"
         do buffer := getchar()
  in skipto();
     any.any := isdigit(buffer);
     while isdigit(buffer)
       do (i := i*10+ord(buffer)-ord("0"); buffer := getchar());
     i
 end

 type list = {first: int, rest: list}

 function readlist() : list =
    let var any := any{any=0}
        var i := readint(any)
     in if any.any
         then list{first=i,rest=readlist()}
         else nil
    end

 function merge(a: list, b: list) : list =
   if a=nil then b
   else if b=nil then a
   else if a.first < b.first
      then list{first=a.first,rest=merge(a.rest,b)}
      else list{first=b.first,rest=merge(a,b.rest)}

 function printint(i: int) =
  let function f(i:int) = if i>0
             then (f(i/10); print(chr(i-i/10*10+ord("0"))))
   in if i<0 then (print("-"); f(-i))
      else if i>0 then f(i)
      else print("0")
 end

 function printlist(l: list) =
   if l=nil then print("\n")
   else (printint(l.first); print(" "); printlist(l.rest))

   var list1 := readlist()
   var list2 := (buffer:=getchar(); readlist())


  /* BODY OF MAIN PROGRAM */
```

```
 in printlist(merge(list1,list2))
end
```

IR Tree

```
 LABEL L2
EXP(
 ESEQ(
  CJUMP(GE,
   CALL(
    NAME ord,
     TEMP t101),
   CALL(
    NAME ord,
     NAME L4),
   L6,L7),
  ESEQ(
   LABEL L6,
   ESEQ(
    MOVE(
     MEM(
      TEMP t109),
     ESEQ(
      MOVE(
       TEMP t110,
       CONST 1),
      ESEQ(
       CJUMP(LE,
        CALL(
         NAME ord,
          TEMP t101),
        CALL(
         NAME ord,
          NAME L5),
        L9,L10),
       ESEQ(
        LABEL L10,
        ESEQ(
         MOVE(
          TEMP t110,
          CONST 0),
         ESEQ(
          LABEL L9,
          TEMP t110)))))),
    ESEQ(
     JUMP(
      NAME L8),
     ESEQ(
      LABEL L7,
      ESEQ(
       MOVE(
        MEM(
         TEMP t109),
        CONST 0),
       ESEQ(
        JUMP(
         NAME L8),
        ESEQ(
```

```
         LABEL L8,
         TEMP t109)))))))))
LABEL L3
EXP(
 ESEQ(
  JUMP(
   NAME L17),
  ESEQ(
   LABEL L18,
   ESEQ(
    MOVE(
     TEMP t101,
     CALL(
      NAME getchar)),
    ESEQ(
     LABEL L17,
     ESEQ(
      CJUMP(EQ,
       ESEQ(
        CJUMP(EQ,
         CALL(
          NAME stringCompare,
           TEMP t101,
           NAME L12),
         CONST 0,
         L15,L14),
        ESEQ(
         LABEL L14,
         ESEQ(
          MOVE(
           MEM(
            TEMP t111),
           CONST 1),
          ESEQ(
           JUMP(
            NAME L16),
           ESEQ(
            LABEL L15,
            ESEQ(
             MOVE(
              MEM(
               TEMP t111),
              CALL(
               NAME stringCompare,
                TEMP t101,
                NAME L13)),
             ESEQ(
              JUMP(
               NAME L16),
              ESEQ(
               LABEL L16,
               TEMP t111)))))))),
       CONST 0,
```

```
       L11,L18),
      ESEQ(
       LABEL L11,
       CONST 0)))))))
LABEL L1
EXP(
 ESEQ(
  EXP(
   CONST 0),
  TEMP t105))
LABEL L23
EXP(
 ESEQ(
  EXP(
   ESEQ(
    MOVE(
     TEMP t122,
     CALL(
      NAME L1,
       TEMP t120)),
    CONST 0)),
  ESEQ(
   CJUMP(EQ,
    MEM(
     BINOP(PLUS,
      TEMP t120,
      CONST 0)),
    CONST 0,
    L28,L27),
   ESEQ(
    LABEL L27,
    ESEQ(
     MOVE(
      MEM(
       TEMP t124),
      ESEQ(
       SEQ(
        MOVE(
         TEMP t123,
         CALL(
          NAME initRecord,
           CONST 16)),
        SEQ(
         MOVE(
          MEM(
           BINOP(PLUS,
            TEMP t123,
            CONST 0)),
          CALL(
           NAME L23)),
         SEQ(
          MOVE(
           MEM(
```

```
            BINOP(PLUS,
             TEMP t123,
             CONST 8)),
           TEMP t122),
          EXP(
           CONST 0)))),
       TEMP t123)),
     ESEQ(
      JUMP(
       NAME L29),
      ESEQ(
       LABEL L28,
       ESEQ(
        MOVE(
         MEM(
          TEMP t124),
         CONST 0),
        ESEQ(
         JUMP(
          NAME L29),
         ESEQ(
          LABEL L29,
          TEMP t124)))))))))
LABEL L24
EXP(
 ESEQ(
  CJUMP(EQ,
   TEMP t115,
   CONST 0,
   L36,L37),
  ESEQ(
   LABEL L36,
   ESEQ(
    MOVE(
     MEM(
      TEMP t129),
     TEMP t114),
    ESEQ(
     JUMP(
      NAME L38),
     ESEQ(
      LABEL L37,
      ESEQ(
       MOVE(
        MEM(
         TEMP t129),
        ESEQ(
         CJUMP(EQ,
          TEMP t114,
          CONST 0,
          L33,L34),
         ESEQ(
          LABEL L33,
```

```
ESEQ(
 MOVE(
  MEM(
   TEMP t128),
  TEMP t115),
 ESEQ(
  JUMP(
   NAME L35),
  ESEQ(
   LABEL L34,
   ESEQ(
    MOVE(
     MEM(
      TEMP t128),
     ESEQ(
      CJUMP(LT,
       MEM(
        BINOP(PLUS,
         TEMP t115,
         CONST 0)),
       MEM(
        BINOP(PLUS,
         TEMP t114,
         CONST 0)),
       L30,L31),
      ESEQ(
       LABEL L30,
       ESEQ(
        MOVE(
         MEM(
          TEMP t127),
         ESEQ(
          SEQ(
           MOVE(
            TEMP t125,
            CALL(
             NAME initRecord,
              CONST 16)),
           SEQ(
            MOVE(
             MEM(
              BINOP(PLUS,
               TEMP t125,
               CONST 0)),
             CALL(
              NAME L24,
               TEMP t114)),
            SEQ(
             MOVE(
              MEM(
               BINOP(PLUS,
                TEMP t125,
                CONST 8)),
```

```
                    MEM(
                     BINOP(PLUS,
                      TEMP t115,
                      CONST 0))),
                    EXP(
                     CONST 0)))),
                 TEMP t125)),
               ESEQ(
                JUMP(
                 NAME L32),
                ESEQ(
                 LABEL L31,
                 ESEQ(
                  MOVE(
                   MEM(
                    TEMP t127),
                   ESEQ(
                    SEQ(
                     MOVE(
                      TEMP t126,
                      CALL(
                       NAME initRecord,
                        CONST 16)),
                     SEQ(
                      MOVE(
                       MEM(
                        BINOP(PLUS,
                         TEMP t126,
                         CONST 0)),
                       CALL(
                        NAME L24,
                         MEM(
                          BINOP(PLUS,
                           TEMP t114,
                           CONST 8)))),
                      SEQ(
                       MOVE(
                        MEM(
                         BINOP(PLUS,
                          TEMP t126,
                          CONST 8)),
                        MEM(
                         BINOP(PLUS,
                          TEMP t114,
                          CONST 0))),
                       EXP(
                        CONST 0)))),
                    TEMP t126)),
                  ESEQ(
                   JUMP(
                    NAME L32),
                   ESEQ(
                    LABEL L32,
```

```
                      TEMP t127))))))))),
              ESEQ(
               JUMP(
                NAME L35),
               ESEQ(
                LABEL L35,
                TEMP t128))))))))),
       ESEQ(
        JUMP(
         NAME L38),
        ESEQ(
         LABEL L38,
         TEMP t129))))))))))
LABEL L39
EXP(
 ESEQ(
  CJUMP(GT,
   TEMP t131,
   CONST 0,
   L41,L42),
  ESEQ(
   LABEL L41,
   ESEQ(
    MOVE(
     MEM(
      TEMP t132),
     CALL(
      NAME print,
       CALL(
        NAME chr,
         BINOP(PLUS,
          BINOP(MINUS,
           TEMP t131,
           BINOP(TIMES,
            BINOP(DIVIDE,
             TEMP t131,
             CONST 10),
            CONST 10)),
          CALL(
           NAME ord,
            NAME L40))))),
    ESEQ(
     JUMP(
      NAME L43),
     ESEQ(
      LABEL L42,
      ESEQ(
       MOVE(
        MEM(
         TEMP t132),
        CONST 0),
       ESEQ(
        JUMP(
```

```
         NAME L43),
        ESEQ(
         LABEL L43,
         TEMP t132)))))))))
LABEL L25
EXP(
 ESEQ(
  EXP(
   CONST 0),
  ESEQ(
   CJUMP(LT,
    TEMP t117,
    CONST 0,
    L49,L50),
   ESEQ(
    LABEL L49,
    ESEQ(
     MOVE(
      MEM(
       TEMP t134),
      CALL(
       NAME L39,
        BINOP(MINUS,
         CONST 0,
         TEMP t117))),
     ESEQ(
      JUMP(
       NAME L51),
      ESEQ(
       LABEL L50,
       ESEQ(
        MOVE(
         MEM(
          TEMP t134),
         ESEQ(
          CJUMP(GT,
           TEMP t117,
           CONST 0,
           L46,L47),
          ESEQ(
           LABEL L46,
           ESEQ(
            MOVE(
             MEM(
              TEMP t133),
             CALL(
              NAME L39,
               TEMP t117)),
            ESEQ(
             JUMP(
              NAME L48),
             ESEQ(
              LABEL L47,
```

```
              ESEQ(
               MOVE(
                MEM(
                 TEMP t133),
                CALL(
                 NAME print,
                  NAME L45)),
               ESEQ(
                JUMP(
                 NAME L48),
                ESEQ(
                 LABEL L48,
                 TEMP t133)))))))))),
        ESEQ(
         JUMP(
          NAME L51),
         ESEQ(
          LABEL L51,
          TEMP t134)))))))))))
LABEL L26
EXP(
 ESEQ(
  CJUMP(EQ,
   TEMP t119,
   CONST 0,
   L54,L55),
  ESEQ(
   LABEL L54,
   ESEQ(
    MOVE(
     MEM(
      TEMP t135),
     CALL(
      NAME print,
       NAME L52)),
    ESEQ(
     JUMP(
      NAME L56),
     ESEQ(
      LABEL L55,
      ESEQ(
       MOVE(
        MEM(
         TEMP t135),
        CALL(
         NAME L26,
          MEM(
           BINOP(PLUS,
            TEMP t119,
            CONST 8)))),
       ESEQ(
        JUMP(
         NAME L56),
```

```
         ESEQ(
          LABEL L56,
          TEMP t135)))))))))
 EXP(
  ESEQ(
   EXP(
    ESEQ(
     MOVE(
      TEMP t137,
      CALL(
       NAME L23)),
     CONST 0)),
   CALL(
    NAME L26,
     CALL(
      NAME L24,
       TEMP t137))))
```

Tiger

```
/* error: mutually recursive types thet do not pass through record or array */
let

type a=c
type b=a
type c=d
type d=a

in
 ""
end
```

Error

```
test6.tig:4.8: Undefined nameTy c
test6.tig:6.8: Undefined nameTy d
```