Lab3

概要

我的程序中所遇到的 shift reduce conflict 有以下三个：

```
State 1 conflicts: 1 shift/reduce
State 28 conflicts: 1 shift/reduce
State 31 conflicts: 1 shift/reduce
```

下面我们逐一分析其出现的原因和无害的理由。

State 1 conflicts

State 1 对应的 y.output 如下所示：

```
State 1

   15 exp: ID . LPAREN args RPAREN
   28    | ID . LBRACE recorditems RBRACE
   29    | ID . LBRACK exp RBRACK OF exp
   62 lvalue: ID .
   63       | ID . LBRACK exp RBRACK

    LPAREN  shift, and go to state 15
    LBRACK  shift, and go to state 16
    LBRACE  shift, and go to state 17

    LBRACK    [reduce using rule 62 (lvalue)]
    $default  reduce using rule 62 (lvalue)
```

对于 State 1 来说，在遇到左方括号 LBRACK（即 “[” ） 时，

可以根据 lvalue -> ID · 进行 reduce 操作，

也可以根据 lvalue -> ID · LBRACK exp RBRACK 进行 shift 操作，

这里我们进行了 reduce 操作。

在 tiger.y 中，我们对句法定义如下：

```
lvalue: ID                          {$$=A_SimpleVar(EM_tokPos, S_Symbol($1));}
    | ID  LBRACK exp RBRACK         {$$=A_SubscriptVar(EM_tokPos, A_SimpleVar(EM_tokPos, S_Symbol($1)), $3);}
    | lvalue  DOT ID                {$$=A_FieldVar(EM_tokPos, $1, S_Symbol($3));}
    | lvalue  LBRACK exp RBRACK     {$$=A_SubscriptVar(EM_tokPos, $1, $3);}
```

此时进行 reduce 操作，我们将 ID 变成了 lvalue，

但之后仍可以根据 lvalue -> lvalue LBRACK exp RBRACK 进行 Parse，从而 reduce 到 lvalue；

这个结果和直接进行几次 shift 操作，根据 lvalue -> ID LBRACK exp RBRACK，reduce 到 lvalue 的结果是相同的。

所以 State 1 的 shift reduce conflict 是无害的。

State 28 conflicts

State 28 对应的 y.output 如下所示：

```
State 28

   49 tydecs: tydec .
   50       | tydec . tydecs

    TYPE  shift, and go to state 27

    TYPE      [reduce using rule 49 (tydecs)]
    $default  reduce using rule 49 (tydecs)

    tydec   go to state 28
    tydecs  go to state 65
```

对于 State 28 来说，在遇到类型声明 TYPE 时，

可以根据 tydecs -> tydec · 进行 reduce 操作，

也可以根据 tydecs -> tydec · tydecs 进行 shift 操作，

这里我们进行了 reduce 操作。

在 tiger.y 中，我们对句法定义如下：

```
tydecs: tydec                   {$$=A_NametyList($1, NULL);}
    | tydec tydecs              {$$=A_NametyList($1, $2);}
```

此时进行 reduce 操作，我们将 tydec 变成了 tydecs，

这个结果和直接进行 shift 操作，根据 tydecs -> tydec tydecs，reduce 到 tydecs 的结果是相同的。

所以 State 28 的 shift reduce conflict 是无害的。

State 31 conflicts

State 31 对应的 y.output 如下所示：

```
State 31

   55 fundecs: fundec .
   56        | fundec . fundecs

    FUNCTION  shift, and go to state 25

    FUNCTION  [reduce using rule 55 (fundecs)]
    $default  reduce using rule 55 (fundecs)

    fundec   go to state 31
    fundecs  go to state 66
```

对于 State 31 来说，在遇到函数声明 FUNCTION 时，

可以根据 fundecs -> fundec · 进行 reduce 操作，

也可以根据 fundecs -> fundec · fundecs 进行 shift 操作，

在 tiger.y 中，我们对句法定义如下：

```
fundecs: fundec                 {$$=A_FundecList($1, NULL);}
    | fundec fundecs            {$$=A_FundecList($1, $2);}
```

此时进行 reduce 操作，我们将 fundec 变成了 fundecs，

这个结果和直接进行 shift 操作，根据 fundecs -> fundec fundecs，reduce 到 fundecs 的结果是相同的。

所以 State 31 的 shift reduce conflict 是无害的。