编译原理

第六章 LR语法分析技术（4）

方徽星

扬州大学 信息工程学院(505)

fanghuixing@yzu.edu.cn

2018年春季学期

本章主要内容

一. 自下向上语法分析

二. LR分析

- SLR
- LR(1)
- LALR

三. 使用二义性文法

四. Yacc

3.1 二义文法的应用

- 二义文法的特点
  - 不是LR文法
  - 直观、简洁
- 例：考虑表达式的二义文法：

二义的	非二义的
$E \to E + E$ $\mid E * E$ $\mid (E)$ $\mid \mathbf{id}$	$E \to E + T$ $\mid T$ $T \to T * F$ $\mid F$ $F \to (E)$ $\mid \mathbf{id}$

3.1 二义文法的应用

- 二义文法的特点
  - 不是LR文法
  - 比非二义性文法更加直观、简洁
  - 可以使用优先级和结合性解决冲突

需要预先设定优先级和结合性，在此前提之下构造遵循优先级和结合性约定的LR语法分析表

3.2 “悬空-else”的二义性

- 例：考虑二义文法

$$stmt \rightarrow \textbf{if } expr \textbf{ then } stmt$$
$$\mid \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt$$
$$\mid \textbf{other}$$

文法抽象化：

$$\boldsymbol{S' \rightarrow S}$$
$$\boldsymbol{S \rightarrow i\ S\ e\ S \mid i\ S \mid a}$$

$\boldsymbol{i}$: **if** $expr$ **then**
$\boldsymbol{e}$: **else**
$\boldsymbol{a}$: **other**

3.2 “悬空-else”的二义性

- **抽象文法：**
  - $S' \to S$
  - $S \to iSeS \mid iS \mid a$

I_0
$S' \to \bullet S$
$S \to \bullet iSeS$
$S \to \bullet iS$
$S \to \bullet a$

I_1
$S' \to S\bullet$

I_2
$S \to i\bullet SeS$
$S \to i\bullet S$
$S \to \bullet iSeS$
$S \to \bullet iS$
$S \to \bullet a$

I_3
$S \to a\bullet$

I_4
$S \to iS\bullet eS$
$S \to iS\bullet$

I_5
$S \to iSe\bullet S$
$S \to \bullet iSeS$
$S \to \bullet iS$
$S \to \bullet a$

I_6
$S \to iSeS\bullet$

3.2 “悬空-else”的二义性

- **抽象文法：**
  - $S' \rightarrow S$
  - $S \rightarrow iSeS \mid iS \mid a$

I_4

$S \rightarrow iS \bullet eS$

$S \rightarrow iS \bullet$

$Follow(S) = \{e, \$\}$

- $[S \rightarrow iS \bullet eS]$要求将$e$移进
- $[S \rightarrow iS \bullet]$要求归约

移进/归约冲突！

3.2 “悬空-else”的二义性

- **抽象文法：**
  - $S' \rightarrow S$
  - $S \rightarrow iSeS \mid iS \mid a$

I_4

$S \rightarrow iS \bullet eS$

$S \rightarrow iS \bullet$

假设栈中内容为：

if $expr$ **then** $stmt$

且**else**为第一个输入符号，

- 将**else**(对应e)移进栈中？
- 还是将**if** $expr$ **then** $stmt$归约？

3.2 “悬空-else”的二义性

- **抽象文法：**
  - $S' \rightarrow S$
  - $S \rightarrow iSeS$
    $\mid iS$
    $\mid a$

因为代表else的e只能作为以iS开头的产生式体的一部分

I_4

$S \rightarrow iS \bullet eS$

$S \rightarrow iS \bullet$

应该将else(对应e)移进栈中！

3.2 “悬空-else”的二义性

- **抽象文法：**
  1. $S \rightarrow iSeS$
  2. $S \rightarrow iS$
  3. $S \rightarrow a$

LR分析表

状态	ACTION				GOTO
	i	e	a	$	S
0	s2		s3		1
1				acc	
2	s2		s3		4
3		r3		r3	
4		s5		r2	
5	s2		s3		6
6		r1		r1	

3.2 “悬空-else”的二义性

- **抽象文法：**
  1. $S \rightarrow iSeS$
  2. $S \rightarrow iS$
  3. $S \rightarrow a$

栈	符号	输入	动作
0		$iiaea\$$	移进
0 2	i	$iaea\$$	移进
…	…	…	…
0 2 2 4	iiS	$ea\$$	移进
…	…	…	…

3.3 LR语法分析中的错误恢复

- 消除包含语法错误的短语：
  1. 语法分析器确定一个从A推导出的串中**包含错误**
  2. 该串的**一部分已经被处理**，并形成了栈中的一个状态序列
  3. 通过从栈中删除状态，**跳过一部分输入**，并将**GOTO(s,A)压入栈**
  4. 语法分析器**假装找到了A的一个实例**，并继续进行正常的语法分析

3.3 LR语法分析中的错误恢复

- 短语层次错误恢复方法：
  1. 检查LR语法分析表中的每个报错条目
  2. 根据语言的使用方法来决定程序员所犯的何种错误最有可能引起该语法错误
  3. 构造出适当的恢复过程
     - 通常是根据各个报错条目来确定适当的修改方法
     - 修改栈顶状态
     - 修改第一个输入符号

3.3 LR语法分析中的错误恢复

例：再次考虑表达式文法：

$$E \rightarrow E+E \mid E*E \mid (E) \mid \mathbf{id}$$

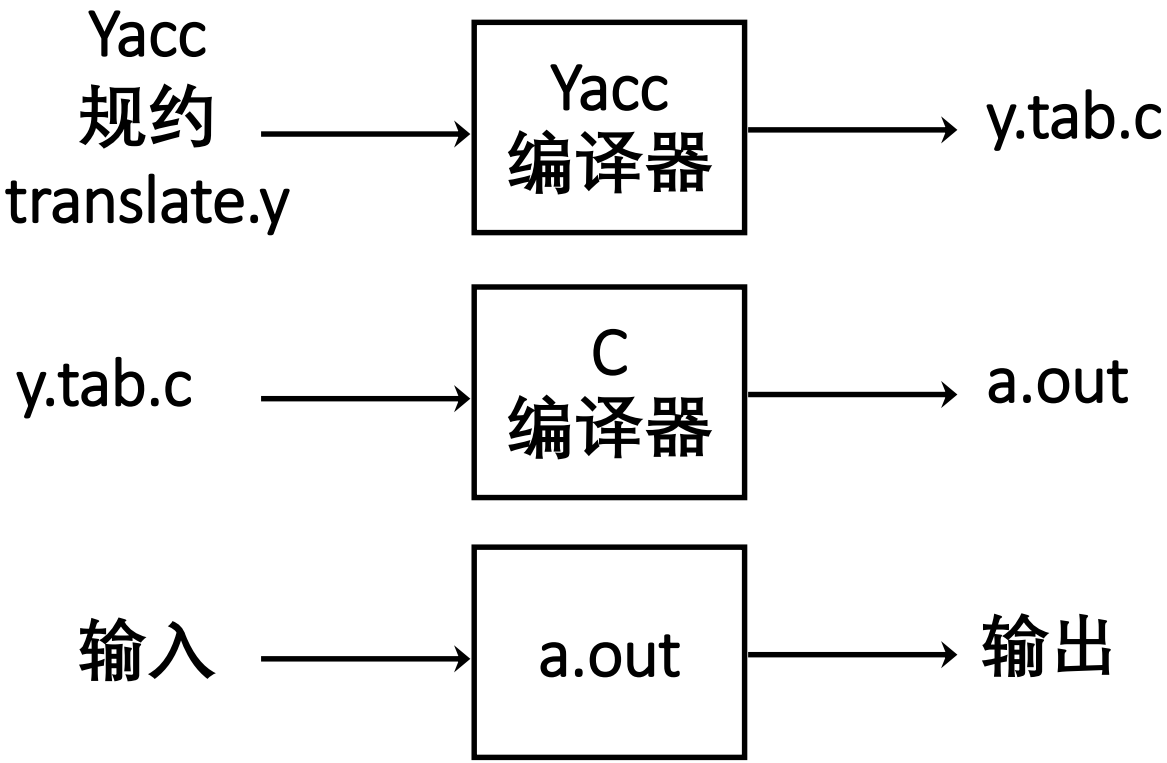
状态	ACTION						GOTO
	id	+	*	(	)	$	*E*
0	s3	e1	e1	s2	e2	e1	1
2	s3	e1	e1	s2	e2	e1	6
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8

e1:在状态0、2、4和5上被调用

- 期望读入一个运算分量的第一个符号，这个符号可能是**id或左括号，**
- **但实际读入的却是**+、 * 或输入结束标记

将状态3(状态0、2、4和5在输入id上的GOTO目标)压入栈中
发出诊断信息“缺少运算分量”

4.1 语法分析器生成工具Yacc



4.2 用Yacc处理二义文法

- 例：简单计算器
  - 输入一个表达式并回车，显示计算结果
  - 也可以输入一个空白行

4.2用Yacc处理二义文法

```
%{
# include  <ctype .h>
# include  <stdio.h >
# define YYSTYPE double
%}

%token  NUMBER
%left  '+'  '–'
%left  '*'  '/'
%right  UMINUS
%%
```

包含断言isdigit

将栈定义为double类型

声明NUMBER是一个词法单元

声明+和-具有相同的优先级，且都是左结合的

声明UMINUS为右结合

第一个%%出现之前

声明部分

4.2用Yacc处理二义文法

```
lines   : lines expr '\n'   {printf ( "%g \n", $2 ) }
        | lines '\n'
        |  /* empty */
        ;
expr    : expr '+' expr      {$$ = $1 + $3; }
        | expr '–' expr      {$$ = $1 – $3; }
        | expr '*' expr      {$$ = $1 * $3; }
        | expr '/ ' expr     {$$ = $1 / $3; }
        | '(' expr ')'       {$$ = $2; }
        | '–' expr  %prec UMINUS      {$$ = –$2; }
        | NUMBER
        ;
%%
```

表示空串

$$代表产生式头的属性值

$1代表产生式体第一个文法符号的属性值

指示单目减运算符优先级和结合性与UMINUS相同

第二个%%之前翻译规则

4.2 用Yacc处理二义文法

词法分析器
词法单元的属性值存在yylval中

```
yylex ( ) {
        int c;
        while ( ( c = getchar ( ) ) == ‘  ’ );
        if ( ( c == ‘.’ ) | | (isdigit (c) ) ) {
                ungetc (c, stdin);
                scanf ( “% lf ”,  &yylval);
                return  NUMBER;
        }
        return c;
}
```

辅助性C语言例程部分

getchar
逐个读入字符

ungetc
将一个字符退回输入流

为了C编译器能准确报告yylex函数中错误的位置，需要在生成的程序y.tab.c中使用编译命令#line

4.3 Yacc的错误恢复

- 用户
  - 决定哪些“主要的”非终结符将具有相关的错误恢复动作
  - 为各主要非终结符A加入形式为$A \rightarrow$ **error** α的错误产生式，其中
    - α是一个可能为空的文法符号串
    - **error为Yacc的一个保留字**
    - Yacc把这样的错误产生式当作普通产生式处理

本章小结

- 重要方法：
  - 移进-归约分析
- 重要概念：
  - 可行前缀
  - 有效项
- LR语法分析器：
  - SLR、
  - 规范LR、
  - LALR
- 二义性文法和Yacc