



中山大學
SUN YAT-SEN UNIVERSITY



国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

DCS290

Compilation Principle 编译原理

第四章 语法分析 (3)

郑馥丹

zhengfd5@mail.sysu.edu.cn

CONTENTS

目录

01

自顶向下分析
Top-Down Parsing

02

LL(1)分析
LL(1) Parsing

03

自底向上分析
Bottom-Up Parsing

04

LR分析
LR Parsing

1. 前提条件

- LL(1)分析

- **表驱动**，依赖预先计算的预测分析表
- 使用统一的算法（栈+表驱动），通过查表决定下一步动作
- **必须严格满足LL(1)文法规则**

- 一个上下文无关文法是LL(1)文法的充分必要条件是，若存在产生式 $A \rightarrow \alpha | \beta$ ，则：

- $\text{FIRST}(\alpha) \cap \text{FIRST}(\beta) = \Phi$
- $\epsilon \in \text{FIRST}(\beta) \Rightarrow \text{FIRST}(\alpha) \cap \text{FOLLOW}(A) = \Phi$

两个条件同时满足！

2. 构造预测分析表

- 预测分析表
 - 二维表
 - 元素 $M[A,a]$ 的内容是当非终结符 A 面临输入符号 a (终结符或结束符 $\$$) 时应选取的产生式；当无产生式时，元素内容为转向出错处理。

例：文法 $G[E]$:

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \epsilon$

$F \rightarrow (E) \mid n$

Non-terminal	lookahead					
	n	+	*	(	)	\$
E						
E'						
T						
T'						
F						

2. 构造预测分析表

- 预测分析表构造算法：
 - 对于文法G的每个产生式 $A \rightarrow \alpha$:
 - ✓ 对于 $FIRST(\alpha)$ 中的每个终结符 a , 将 $A \rightarrow \alpha$ 填入表 $M[A, a]$ 中;
 - ✓ 如果 $\epsilon \in FIRST(\alpha)$, 则对于 $FOLLOW(A)$ 中的每个终结符 a , 将 $A \rightarrow \alpha$ 填入表 $M[A, a]$ 中; 如果 $\$ \in FOLLOW(A)$, 也将 $A \rightarrow \alpha$ 填入表 $M[A, \$]$ 中。

2. 构造预测分析表

例：文法G[E]:
 $E \rightarrow TE'$
 $E' \rightarrow +TE' | \epsilon$
 $T \rightarrow FT'$
 $T' \rightarrow *FT' | \epsilon$
 $F \rightarrow (E) | n$

$FIRST(E) = \{ (, n \}$ $FOLLOW(E) = \{), \$ \}$
 $FIRST(E') = \{ +, \epsilon \}$ $FOLLOW(E') = \{), \$ \}$
 $FIRST(T) = \{ (, n \}$ $FOLLOW(T) = \{ +,), \$ \}$
 $FIRST(T') = \{ *, \epsilon \}$ $FOLLOW(T') = \{ +,), \$ \}$
 $FIRST(F) = \{ (, n \}$ $FOLLOW(F) = \{ *, +,), \$ \}$

- 对于FIRST(α)中的每个终结符a, 将 $A \rightarrow \alpha$ 填入表M[A,a]中;
- 如果 $\epsilon \in FIRST(\alpha)$, 则对于FOLLOW(A)中的每个终结符a, 将 $A \rightarrow \alpha$ 填入表M[A,a]中; 如果 $\$ \in FOLLOW(A)$, 也将 $A \rightarrow \alpha$ 填入表M[A,\$]中。

Non-terminal	lookahead					
	n	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow n$			$F \rightarrow (E)$		

- 填写以下文法的预测分析表

- $G[S]: S \rightarrow \text{if } E \text{ then } S \ S' | \text{other}, \ S' \rightarrow \epsilon | \text{else } S, \ E \rightarrow \text{bool}$

Non-terminal	lookahead					
	if	then	else	other	bool	\$
S						
S'						
E						

• 填写以下文法的预测分析表

– $G[S]: S \rightarrow \text{if } E \text{ then } S \ S' | \text{other}, \ S' \rightarrow \epsilon | \text{else } S, \ E \rightarrow \text{bool}$

Non-terminal	lookahead					
	if	then	else	other	bool	\$
S	$S \rightarrow \text{if } E \text{ then } S \ S'$			$S \rightarrow \text{other}$		
S'			$S' \rightarrow \epsilon$ $S' \rightarrow \text{else } S$			$S' \rightarrow \epsilon$
E					$E \rightarrow \text{bool}$	

$\text{FIRST}(\text{if } E \text{ then } S \ S') = \{\text{if}\}$

$\text{FIRST}(\text{else } S) = \{\text{else}\}$

$\text{FIRST}(\text{other}) = \{\text{other}\}$

$\text{FIRST}(\text{bool}) = \{\text{bool}\}$

$\text{FIRST}(\epsilon) = \{\epsilon\}$

$\text{FOLLOW}(S') = \{\text{else } \$\}$

2. 构造预测分析表

• 冲突[conflicts]

– 同一个单元格中存在多条产生式

– 例： $G[S]: S \rightarrow \text{if } E \text{ then } S \ S' | \text{other}, \ S' \rightarrow \epsilon | \text{else } S, \ E \rightarrow \text{bool}$

Non-terminal	lookahead					
	if	then	else	other	bool	\$
S	$S \rightarrow \text{if } E \text{ then } S \ S'$			$S \rightarrow \text{other}$		
S'			$S' \rightarrow \epsilon$ $S' \rightarrow \text{else } S$			$S' \rightarrow \epsilon$
E					$E \rightarrow \text{bool}$	

因存在二义性导致冲突

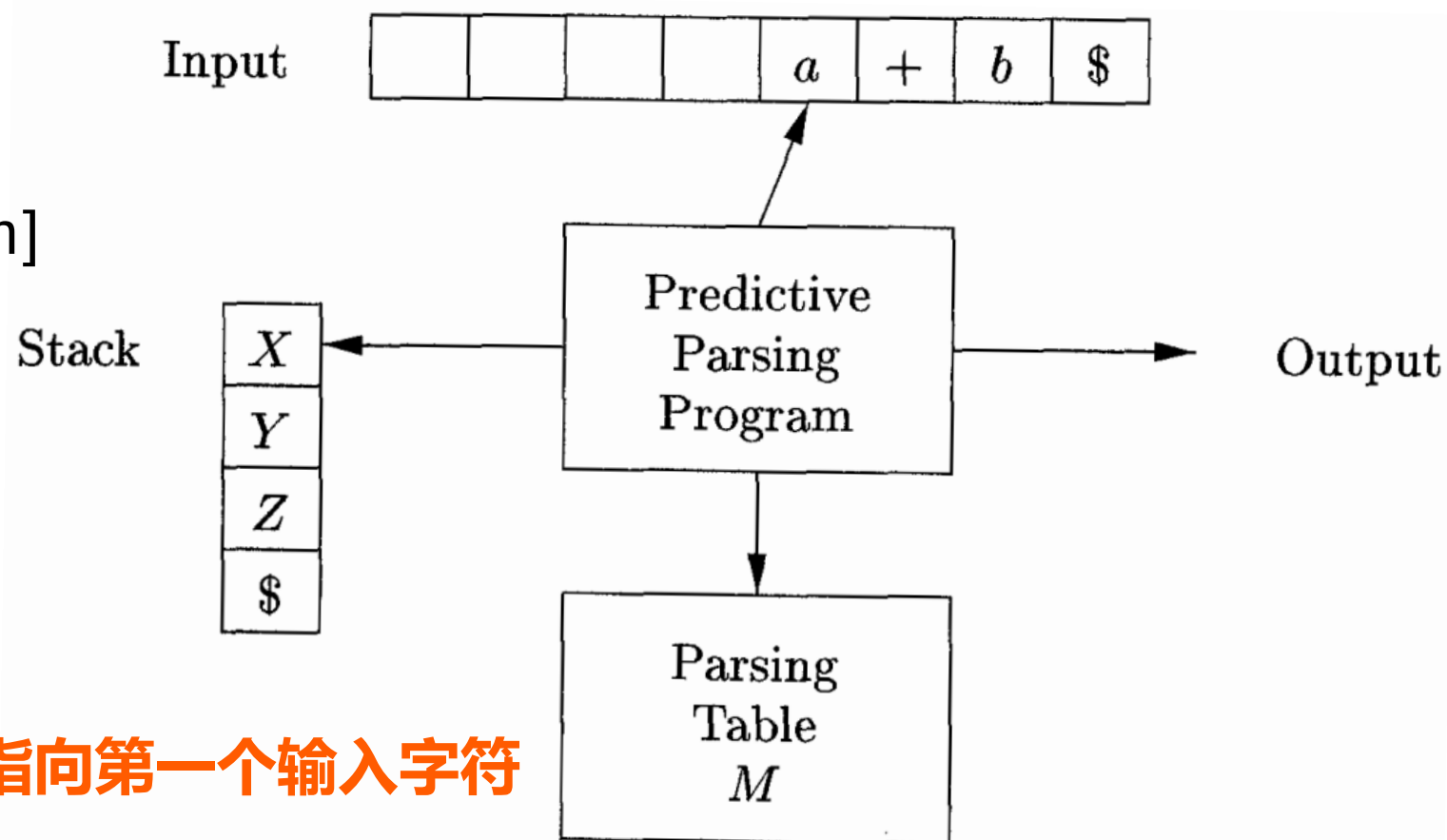
– 可能引起冲突的情况：左递归、左公因子、二义性 应消除这些情况后再建表

3. Table-Driven Parser

- Parser由预测分析表/解析表[Parsing Table]驱动

- Parser的构成:

- 预测分析表
- 解析程序[Parsing Program]
- 栈[Stack]
- 输入缓冲区
- 输出流



初始:

Stack: \$S; lookahead: 指向第一个输入字符

接受:

Stack: \$; lookahead: 指向\$

3. Table-Driven Parser

• LL(1)预测分析算法

令 a =输入符号串 w 的第一个符号; 令 X =栈顶符号;

while($X \neq \$$){ //栈非空

if($X == a$){ 弹出栈顶符号 X ; 令 a 指向下一个输入符号; } //a字符匹配

else if(X 是一个终结符) error();

else if($M[X,a]$ 是一个报错条目) error();

else if($M[X,a] == X \rightarrow Y_1 Y_2 \dots Y_k$){

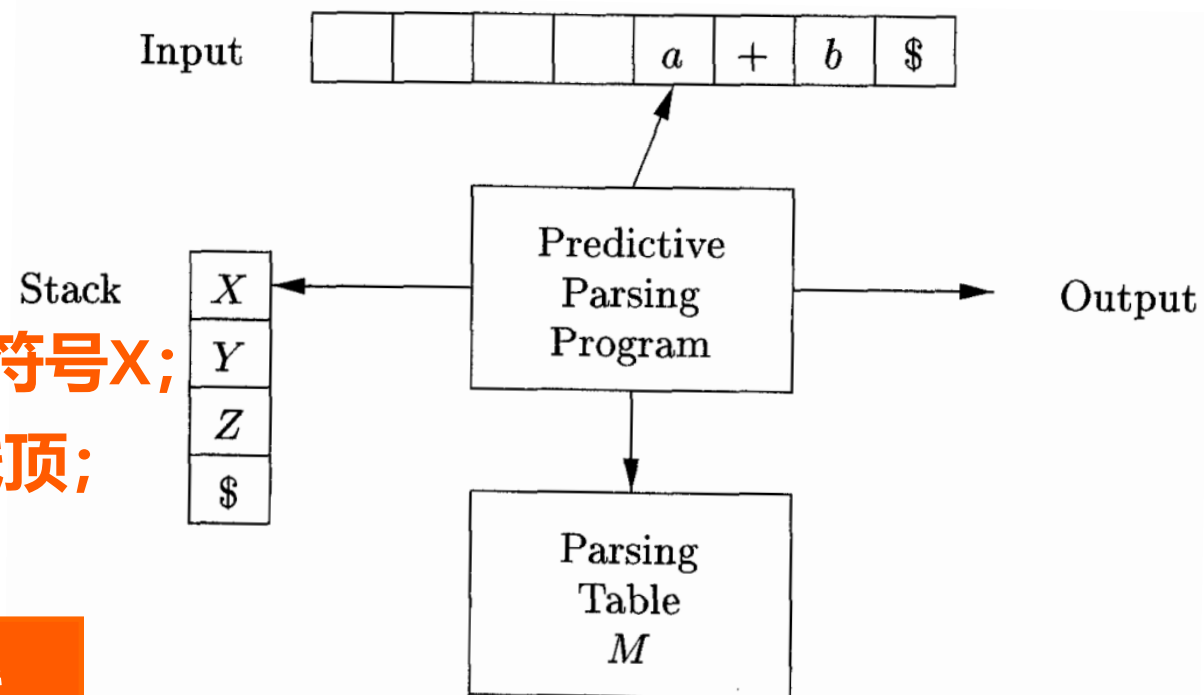
输出产生式 $X \rightarrow Y_1 Y_2 \dots Y_k$; 弹出栈顶符号 X ;

将 $Y_1 Y_2 \dots Y_k$ 压入栈中, 其中 Y_1 位于栈顶;

}

令 X =栈顶符号;

}



反序压栈

3. Table-Driven Parser

• 例：

	n	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow n$			$F \rightarrow (E)$		

反序压栈

步骤	分析栈	剩余输入串	动作
1	$\$E$	$n+n*n\$$	输出 $E \rightarrow TE'$
2	$\$E'T$	$n+n*n\$$	输出 $T \rightarrow FT'$
3	$\$E'T'F$	$n+n*n\$$	输出 $F \rightarrow n$
4	$\$E'T'n$	$n+n*n\$$	n 匹配
5	$\$E'T'$	$+n*n\$$	输出 $T' \rightarrow \epsilon$
6	$\$E'$	$+n*n\$$	输出 $E' \rightarrow +TE'$
7	$\$E'T+$	$+n*n\$$	+ 匹配

3. Table-Driven Parser

• 例:

	n	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow * FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow n$			$F \rightarrow (E)$		

8	$\$E'T$	$n*n\$$	输出 $T \rightarrow FT'$
9	$\$E'T'F$	$n*n\$$	输出 $F \rightarrow n$
10	$\$E'T'n$	$n*n\$$	n 匹配
11	$\$E'T'$	$*n\$$	输出 $T' \rightarrow *FT'$
12	$\$E'T'F*$	$*n\$$	* 匹配
13	$\$E'T'F$	$n\$$	输出 $F \rightarrow n$
14	$\$E'T'n$	$n\$$	n 匹配
15	$\$E'T'$	$\$$	输出 $T' \rightarrow \varepsilon$
16	$\$E'$	$\$$	输出 $E' \rightarrow \varepsilon$
17	$\$$	$\$$	ACCEPT

3. Table-Driven Parser

- 例：
 - 缺少操作数
 - 如：n+*n

	n	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow n$			$F \rightarrow (E)$		

步骤	分析栈	剩余输入串	动作
1	\$E	n+*n\$	输出 $E \rightarrow TE'$
2	\$E'T	n+*n\$	输出 $T \rightarrow FT'$
3	\$E'T'F	n+*n\$	输出 $F \rightarrow n$
4	\$E'T'n	n+*n\$	n 匹配
5	\$E'T'	+*n\$	输出 $T' \rightarrow \epsilon$
6	\$E'	+*n\$	输出 $E' \rightarrow +TE'$
7	\$E'T+	+*n\$	+ 匹配
8	\$E'T	*n\$	ERROR

[T, *]=empty

3. Table-Driven Parser

- 例：
 - 缺少操作符
 - 如：nn*n

	n	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow n$			$F \rightarrow (E)$		

步骤	分析栈	剩余输入串	动作
1	\$E	nn*n\$	输出 $E \rightarrow TE'$
2	\$E'T	nn*n\$	输出 $T \rightarrow FT'$
3	\$E'T'F	nn*n\$	输出 $F \rightarrow n$
4	\$E'T'n	nn*n\$	n 匹配
5	\$E'T'	n*n\$	ERROR

[T', n]=empty

3. Table-Driven Parser

- 例：
 - 缺少操作符
 - 如：n+n)

	n	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow n$			$F \rightarrow (E)$		

步骤	分析栈	剩余输入串	动作
1	\$E	n+n)\$	输出 $E \rightarrow TE'$
2	\$E'T	n+n)\$	输出 $T \rightarrow FT'$
3	\$E'T'F	n+n)\$	输出 $F \rightarrow n$
4	\$E'T'n	n+n)\$	n 匹配
5	\$E'T'	+n)\$	输出 $T' \rightarrow \epsilon$
6	\$E'	+n)\$	输出 $E' \rightarrow +TE'$
7	\$E'T+	+n)\$	+ 匹配

步骤	分析栈	剩余输入串	动作
8	\$E'T	n)\$	输出 $T \rightarrow FT'$
9	\$E'T'F	n)\$	输出 $F \rightarrow n$
10	\$E'T'n	n)\$	n 匹配
11	\$E'T'	)\$	输出 $T' \rightarrow \epsilon$
12	\$E'	)\$	输出 $E' \rightarrow \epsilon$
13	\$	)\$	ERROR

- 对文法 $G[S]: S \rightarrow aBa, B \rightarrow bB | \epsilon$, 构造预测分析表, 并对输入符号串 $abba$ 及 $aabb$ 分别进行 LL(1) 预测分析, 写出分析的全过程 (表格形式)。

Non-terminal	lookahead		
	a	b	\$
S	$S \rightarrow aBa$		
B	$B \rightarrow \epsilon$	$B \rightarrow bB$	

步骤	分析栈	剩余输入串	动作
1	\$S	abba\$	输出 $S \rightarrow aBa$
2	\$aBa	abba\$	a 匹配
3	\$aB	bba\$	输出 $B \rightarrow bB$
4	\$aBb	bba\$	b 匹配
5	\$aB	ba\$	输出 $B \rightarrow bB$
6	\$aBb	ba\$	b 匹配
7	\$aB	a\$	输出 $B \rightarrow \epsilon$
8	\$a	a\$	a 匹配
9	\$	\$	ACCEPT

随堂练习 (7)

- 对文法 $G[S]: S \rightarrow aBa, B \rightarrow bB | \epsilon$, 构造预测分析表, 并对输入符号串 $abba$ 及 $aabb$ 分别进行 LL(1) 预测分析, 写出分析的全过程 (表格形式)。

Non-terminal	lookahead		
	a	b	\$
S	$S \rightarrow aBa$		
B	$B \rightarrow \epsilon$	$B \rightarrow bB$	

步骤	分析栈	剩余输入串	动作
1	$\$S$	$aabb\$$	输出 $S \rightarrow aBa$
2	$\$aBa$	$aabb\$$	a 匹配
3	$\$aB$	$abb\$$	输出 $B \rightarrow \epsilon$
4	$\$a$	$abb\$$	a 匹配
5	$\$$	$bb\$$	ERROR

4. LL(1)预测分析中的错误恢复[Error Recovery]

- Parser在Error情况下应该做什么?
 - 给出Error信息
 - ✓ Error信息越详尽越好
 - 从Error中恢复
 - ✓ 能够继续对其余输入进行解析
- Error类型
 - 终结符不匹配：栈顶符号和下一个输入符号不匹配
 - $M[A, a]$ 为空：无候选产生式
 - 栈为空，输入符号仍有剩余

4. LL(1)预测分析中的错误恢复[Error Recovery]

• 恐慌模式[Panic Mode]

- 是一种错误恢复策略
- 恐慌：遇到危险先跑再说（**三十六计走为上计**），不尝试精确修复错误，直接跳过
- 目标：让parser尽快回到可以继续分析的状态，而不是纠结于错误的细节
- 基本思想：当parser遇到一个无法处理的错误时，采取“紧急避险”的方式，简单粗暴地**忽略/跳过**输入中的一些符号，直到输入中出现一些特定的符号（**同步符号synch**）

4. LL(1)预测分析中的错误恢复[Error Recovery]

- 恐慌模式[Panic Mode]

- 规则:

- ✓ 将FOLLOW(A)的所有符号都放到A的同步集合中, 即, 对于 $a \in \text{FOLLOW}(A)$, 预测分析表中M[A, a]处均填写 **synch**
 - ✓ 如果 **M[top, lookahead]==empty**, 则跳过lookahead字符, 栈保持不变
 - ✓ 如果 **M[top, lookahead]==synch**, 则跳过栈顶top, 将栈顶弹出, 输入字符不变
 - ✓ 如果 **top!=lookahead**, 则同样跳过栈顶top, 将栈顶弹出, 输入字符不变

4. LL(1)预测分析中的错误恢复[Error Recovery]

• 恐慌模式[Panic Mode]

– 例：

	i	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$	synch	synch
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$	synch		$T \rightarrow FT'$	synch	synch
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow * FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow i$	synch	synch	$F \rightarrow (E)$	synch	synch

$\text{FOLLOW}(E) = \{), \$\}$

$\text{FOLLOW}(E') = \{), \$\}$

$\text{FOLLOW}(T) = \{+,), \$\}$

$\text{FOLLOW}(T') = \{+,), \$\}$

$\text{FOLLOW}(F) = \{*, +,), \$\}$

4. LL(1)预测分析中的错误恢复[Error Recovery]

- 恐慌模式[Panic Mode]
 - 例：

	i	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$	synch	synch
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$	synch		$T \rightarrow FT'$	synch	synch
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow i$	synch	synch	$F \rightarrow (E)$	synch	synch

步骤	分析栈	剩余输入串	动作
1	\$E	*i*+i\$	empty, skip input, 输出error
2	\$E	i*+i\$	输出 $E \rightarrow TE'$
3	\$E'T	i*+i\$	输出 $T \rightarrow FT'$
4	\$E'T'F	i*+i\$	输出 $F \rightarrow i$
5	\$E'T'i	i*+i\$	i 匹配
6	\$E'T'	*+i\$	输出 $T' \rightarrow *FT'$
7	\$E'T'F*	*+i\$	* 匹配
8	\$E'T'F	+i\$	synch, skip top, 输出error
9	\$E'T'	+i\$	输出 $T' \rightarrow \epsilon$

4. LL(1)预测分析中的错误恢复[Error Recovery]

- 恐慌模式[Panic Mode]
 - 例：

	i	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$	synch	synch
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$	synch		$T \rightarrow FT'$	synch	synch
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow i$	synch	synch	$F \rightarrow (E)$	synch	synch

步骤	分析栈	剩余输入串	动作
10	$\$E'$	$+i\$$	输出 $E' \rightarrow +TE'$
11	$\$E'T+$	$+i\$$	$+$ 匹配
12	$\$E'T$	$i\$$	输出 $T \rightarrow FT'$
13	$\$E'T'F$	$i\$$	输出 $F \rightarrow i$
14	$\$E'T'i$	$i\$$	i 匹配
15	$\$E'T'$	$\$$	输出 $T' \rightarrow \epsilon$
16	$\$E'$	$\$$	输出 $E' \rightarrow \epsilon$
17	$\$$	$\$$	end

第四章课后作业（1）

- 对文法 $G[S]: S \rightarrow (L)|a, L \rightarrow L,S|S$ ，构造预测分析表（**含恐慌模式**），并对输入符号串 $(a,(a,a))$ 、 $(a,)$ 和 $(a,a$ 分别进行LL(1)预测分析，写出分析的全过程（表格形式）。
- 提交要求：
 - 文件命名：学号-姓名-第四章作业(1)；
 - 文件格式：.pdf文件；
 - 手写版、电子版均可；若为手写版，则拍照后转成pdf提交，但**须注意将照片旋转为正常角度，且去除照片中的多余信息**；电子版如word等转成pdf提交；
 - 提交到超算习堂（第四章作业(1)）处；
 - 提交ddl：**4月6日晚上12:00**；
 - **重要提示：不得抄袭！**