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

4. 确定的自顶向下语法分析[predictive]

- 递归下降预测分析[Recursive Descent Predictive Parsing]
- LL(1)分析

5. 递归下降预测分析[Recursive Descent Predictive Parsing]

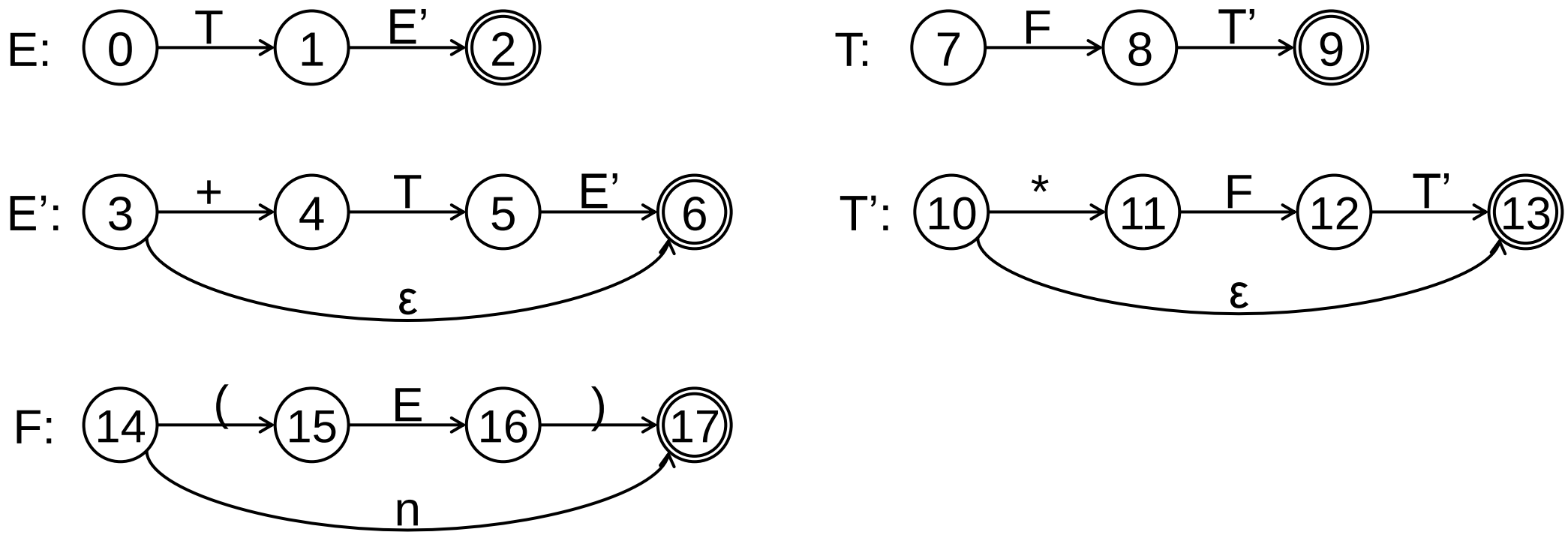
- 递归下降预测分析[Recursive Descent Predictive Parsing]
  - 基于**手写代码**实现，每个非终结符对应一个递归函数
  - 通过函数的递归调用模拟推导过程，**显式地编码产生式的选择**（通常通过if-else或switch分支实现）
  - 可以是LL(1)文法，也可以是更强的**LL(k)（通过向前看更多符号解决冲突）**
- 递归下降预测解析器的开发流程
  - 1. 消除语法中的二义性　　2. 消除语法中的左递归
  - 3. 提取左公因子　　4. 从语法构造转换图
  - 5. 简化转换图　　6. 使用转换图编写解析器作为蓝本

5. 递归下降预测分析[Recursive Descent Predictive Parsing]

- 转换图——可视化预测分析器
  - 对每个非终结符A
    - ✓ 创建初始状态和结束状态
    - ✓ 对于每个$A \rightarrow X_1X_2...X_n$，创建一条从初始状态到最终状态的路径，边标记为$X_1$，$X_2$，…，$X_n$。
    - ✓ 如果$A \rightarrow \varepsilon$：表示路径为ε。

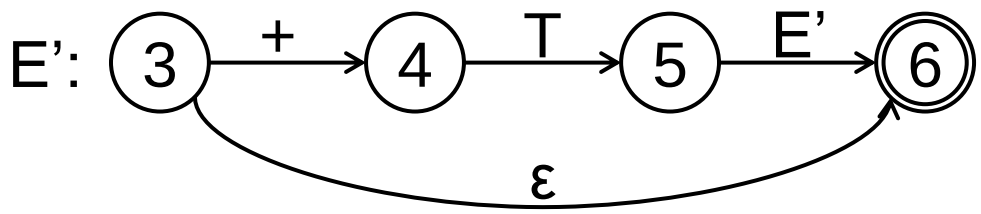
5. 递归下降预测分析[Recursive Descent Predictive Parsing]

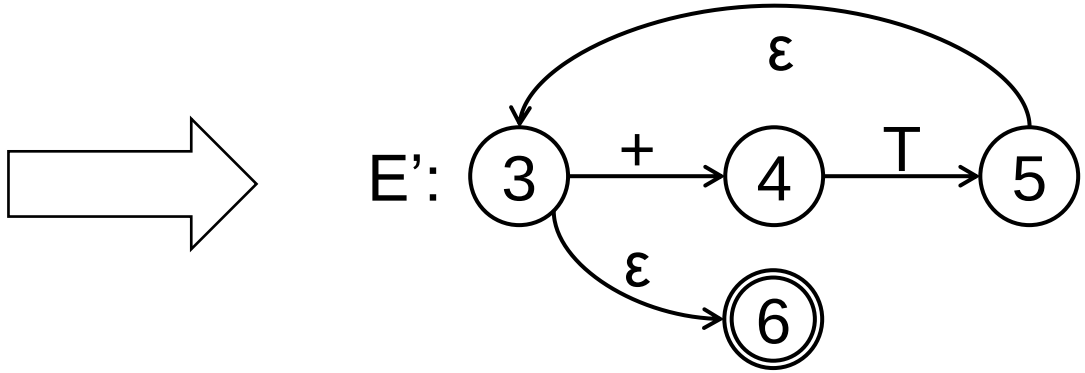
- 转换图——可视化预测分析器
  - 例：对文法G[E]：E→TE’，E’→+TE’|ε，T→FT’，T’→*FT’|ε，F→(E)|n构建转换图

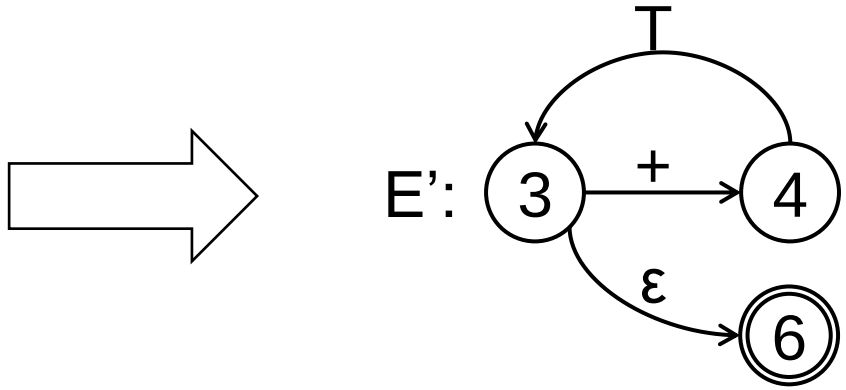


5. 递归下降预测分析[Recursive Descent Predictive Parsing]

- 化简转换图

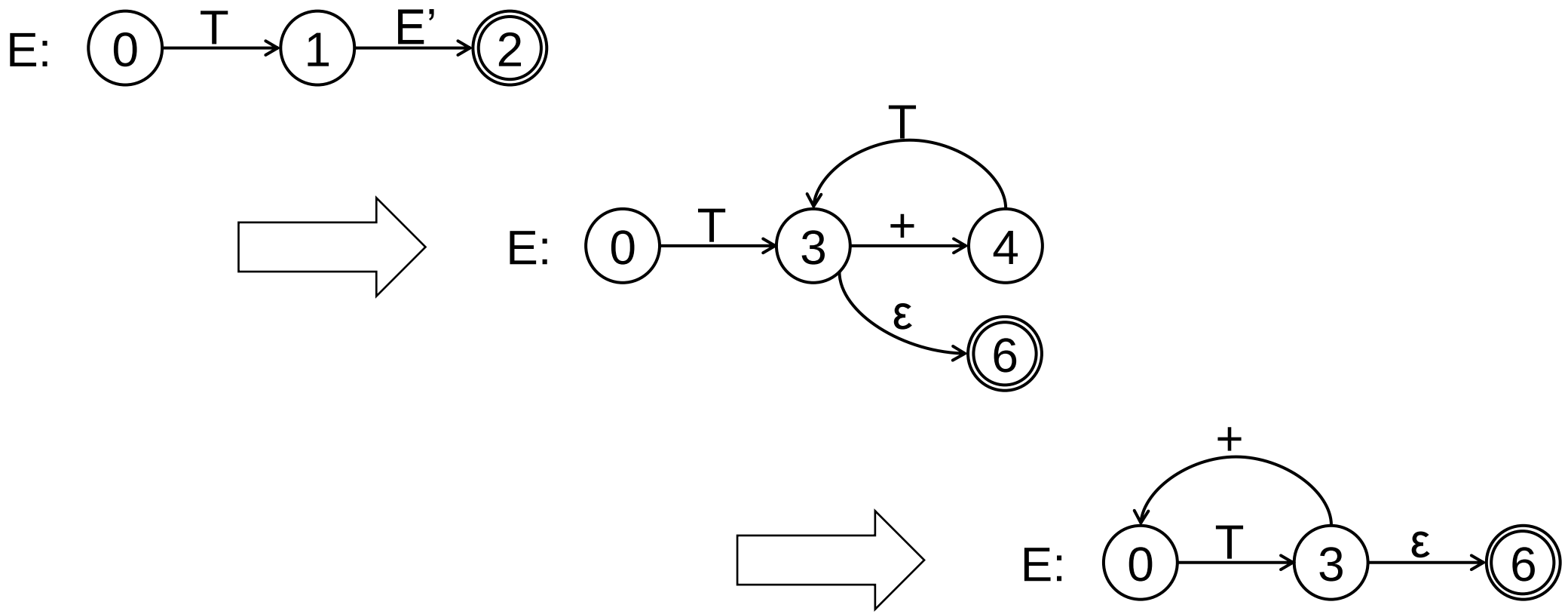






5. 递归下降预测分析[Recursive Descent Predictive Parsing]

- 化简转换图



5. 递归下降预测分析[Recursive Descent Predictive Parsing]

- 化简转换图

文法G[E]:

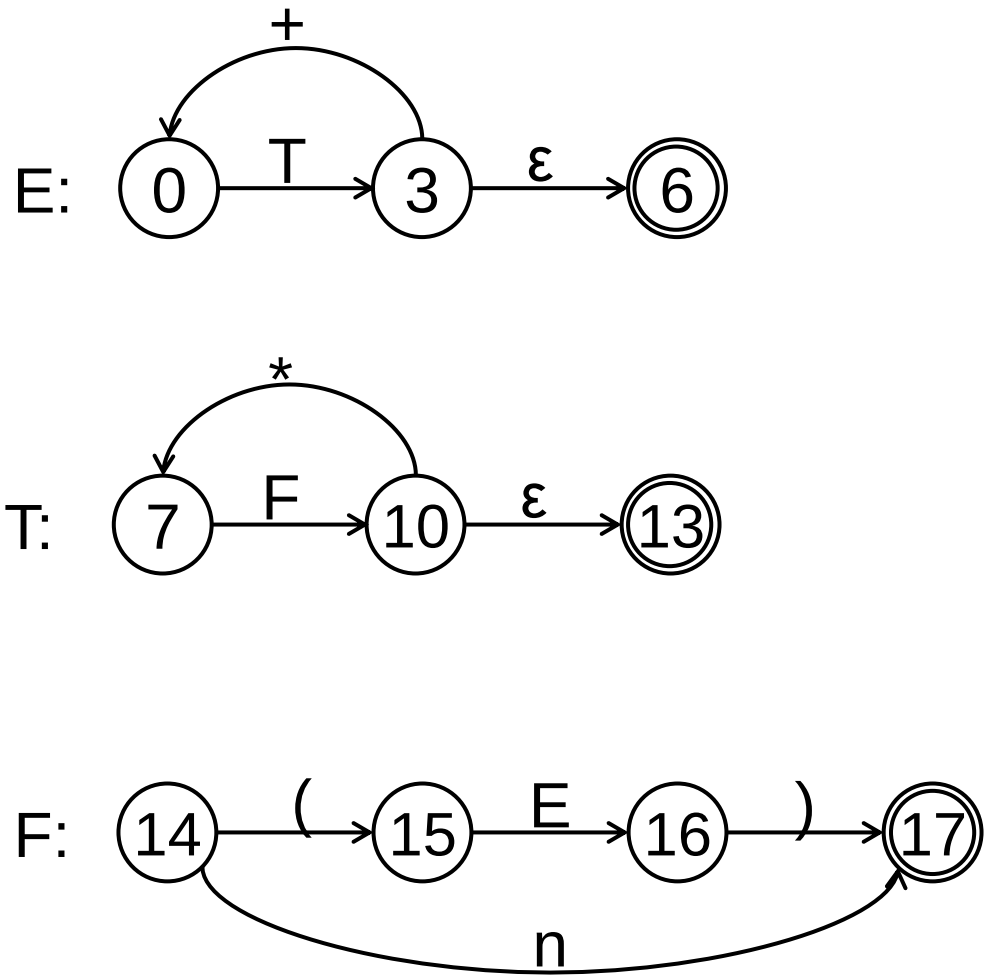
E→TE’

E’→+TE’|ε

T→FT’

T’→*FT’|ε

F→(E)|n

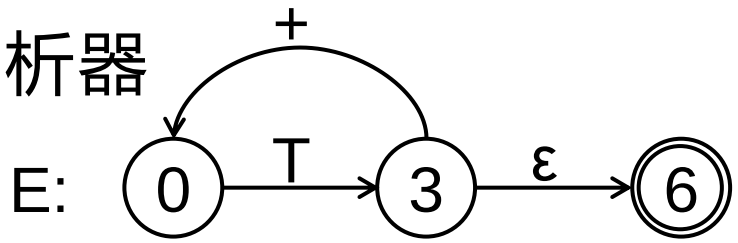


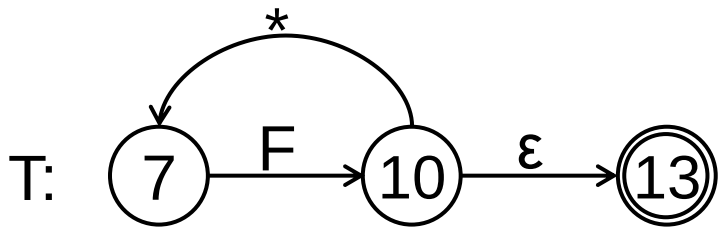
5. 递归下降预测分析[Recursive Descent Predictive Parsing]

- 编写解析器
  - 为每个图编写递归过程，**开始符号S**对应的转换图是程序的主要**main入口**
  - 设计程序的控制流，模仿图中的路径，考虑到前瞻
    - ✓ 如果路径中边的符号等于终结符，则执行**匹配[match]**动作
    - ✓ 如果路径中边的符号是非终结符，则执行**派生[derive]**操作，即调用非终结符的过程（递归）

5. 递归下降预测分析[Recursive Descent Predictive Parsing]

- 编写解析器



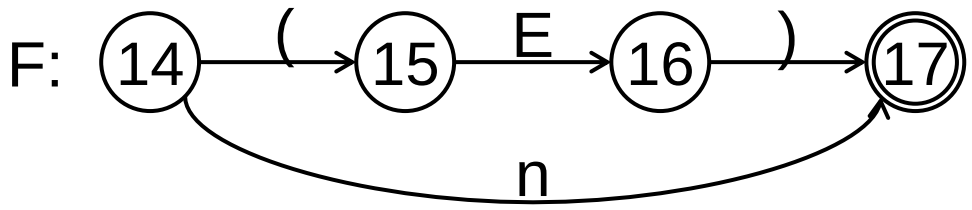



```
void E(){
   if(lookahead in FIRST(T)){
      T();
   }else error();
   while(lookahead=='+'){
      match('+');
      if(lookahead in FIRST(T)){
         T();
      }else error();
   }
   if(lookahead in FOLLOW(E)){
     // do nothing
   }else error();
}
```

```
void T(){
   if(lookahead in FIRST(F)){
      F();
   }else error();
   while(lookahead=='*'){
      match('*');
      if(lookahead in FIRST(F)){
         F();
      }else error();
   }
   if(lookahead in FOLLOW(T)){
     // do nothing
   }else error();
}
```

5. 递归下降预测分析[Recursive Descent Predictive Parsing]

- 编写解析器




```
void F(){
   if(lookahead=='('){
      match('(');
      if(lookahead in FIRST(E)){
         E();
      }else error();
      if(lookahead==')'){
         match(')');
      }else error();
   }else if(lookahead=='n'){
      match('n');
   }else error();
}
```

```
void match(Token tok){
   if(lookahead==tok){
      lookahead=scanner.getNextToken();
   }else error();
}
```

CONTENTS

目 录

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：
  - $FIRST(\alpha) \cap FIRST(\beta) = \Phi$
  - $\varepsilon \in FIRST(\beta) \Rightarrow FIRST(\alpha) \cap FOLLOW(A) = \Phi$

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→TE’

E’→+TE’|ε

T→FT’

T’→*FT’|ε

F→(E)|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→α:
    - ✓ 对于FIRST(α)中的每个终结符a，将A→α填入表M[A,a]中；
    - ✓ 如果ε∈FIRST(α)，则对于FOLLOW(A)中的每个终结符a，将A→α填入表M[A,a]中；如果$∈FOLLOW(A)，也将A→α填入表M[A,$]中。

2. 构造预测分析表

例：文法G[E]：

E→TE’

E’→+TE’|ε

T→FT’

T’→*FT’|ε

F→(E)|n

FIRST(E)={(, n}　FOLLOW(E)={), $}

FIRST(E’)={+, ε}　FOLLOW(E’)={), $}

FIRST(T)={(, n}　FOLLOW(T)={+,), $}

FIRST(T’)={*, ε}　FOLLOW(T’)={+,), $}

FIRST(F)={(, n}　FOLLOW(F)={*, +,), $}

- **对于FIRST(α)中的每个终结符a，将A→α填入表M[A,a]中；**
- **如果ε∈FIRST(α)，则对于FOLLOW(A)中的每个终结符a，将A→α填入表M[A,a]中；如果$∈FOLLOW(A)，也将A→α填入表M[A,$]中。**

Non-terminal	lookahead					
	n	**+**	*****	**(**	**)**	**$**
E	E→TE’			E→TE’		
E’		E'→+TE’			E'→ ε	E'→ε
T	T→ FT’			T→ FT’		
T’		T'→ ε	T'→*FT'		T'→ ε	T'→ε
F	F→ n			F→ (E)		

随堂练习（6）

- 填写以下文法的预测分析表
  - G[S]:S→if E then S S’|other，S’→ε|else S，E→bool

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

随堂练习（6）

- 填写以下文法的预测分析表
  - G[S]:S→if E then S S’|other，S’→ε|else S，E→bool

Non-terminal	lookahead					
	if	then	else	other	bool	$
S	S→if E then S S’			S→other		
S’			**S’→ε** **S’→else S**			S’→ε
E					E→bool	

FIRST(if E then S S’)={if}　　FIRST(other)={other}　　FIRST(ε)={ε}
FIRST(else S)={else}　　FIRST(bool)={bool}　　FOLLOW(S’)={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→if E then S S’|other，S’→ε|else S，E→bool

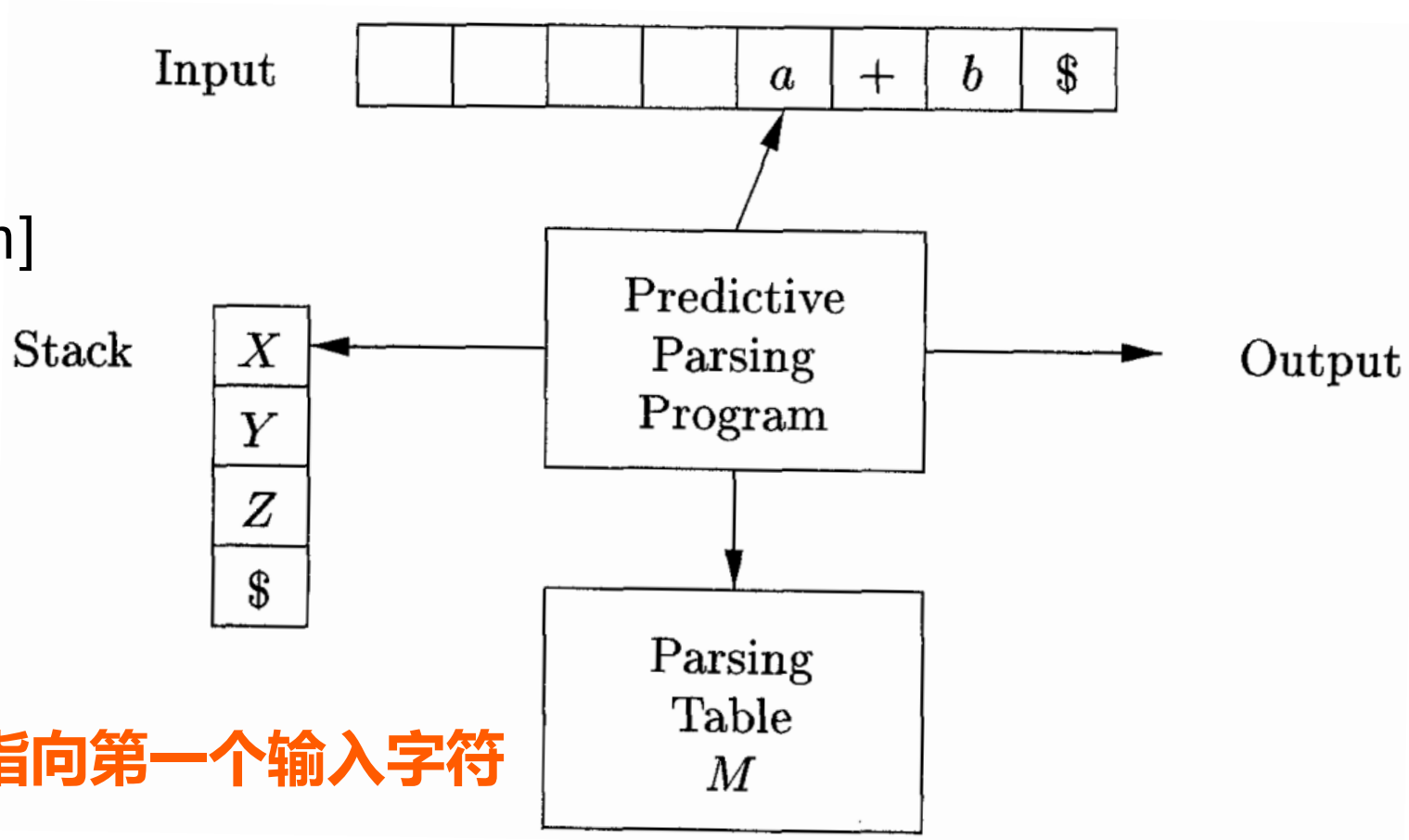
Non-terminal	lookahead					
	if	then	else	other	bool	$
S	S→if E then S S’			S→other		
S’			**S’→ε S’→else S**			S’→ε
E					E→bool	

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!=$){ //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]==$X \rightarrow Y_1 Y_2 \ldots Y_k$){

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

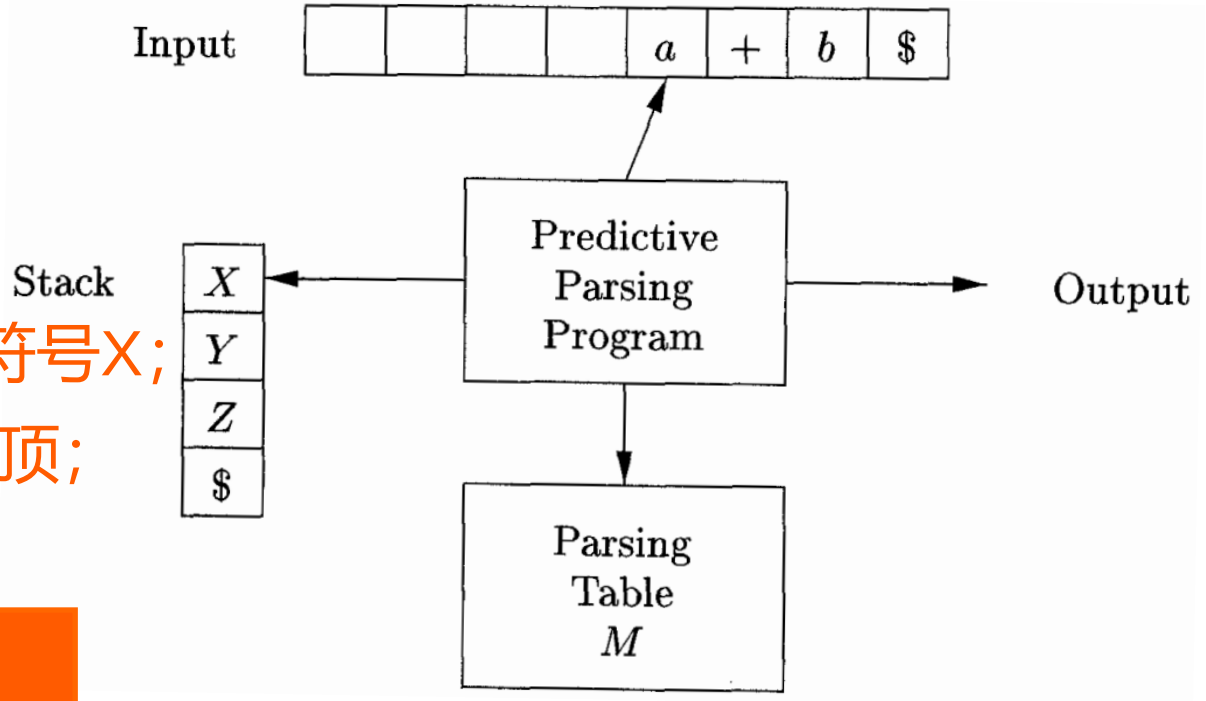
将$Y_1 Y_2 \ldots Y_k$压入栈中，其中Y_1位于栈顶；

}

令X=栈顶符号；

}

反序压栈



3. Table-Driven Parser

- 例：

	n	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→ FT'			T→ FT'		
T'		T'→ ε	T'→* FT'		T'→ ε	T'→ ε
F	F→ n			F→ (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→TE’
2	$E’T	n+n*n$	输出 T→FT’
3	$E’T’F	n+n*n$	输出 F→n
4	$E’T’n	n+n*n$	n 匹配
5	$E’T’	+n*n$	输出 T’→ ε
6	$E’	+n*n$	输出 E’→+TE’
7	$E’T+	+n*n$	+ 匹配

3. Table-Driven Parser

- 例:

	n	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→ FT'			T→ FT'		
T'		T'→ ε	T'→* FT'		T'→ ε	T'→ ε
F	F→ n			F→ (E)		

8	$E’T	n*n$	输出 T→FT’
9	$E’T’F	n*n$	输出 F→n
10	$E’T’n	n*n$	n 匹配
11	$E’T’	*n$	输出T’→*FT’
12	$E’T’F*	*n$	* 匹配
13	$E’T’F	n$	输出 F→n
14	$E’T’n	n$	n 匹配
15	$E’T’	$	输出 T’→ ε
16	$E’	$	输出 E’→ ε
17	$	$	ACCEPT

3. Table-Driven Parser

- 例:
  - **缺少操作数**

  如：n+*n

	n	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→ FT'			T→ FT'		
T'		T'→ ε	T'→* FT'		T'→ ε	T'→ ε
F	F→ n			F→ (E)		

步骤	分析栈	剩余输入串	动作
1	$E	**n+*n**$	输出 E→TE’
2	$E’T	n+*n$	输出 T→FT’
3	$E’T’F	n+*n$	输出 F→n
4	$E’T’n	n+*n$	n 匹配
5	$E’T’	+*n$	输出 T’→ ε
6	$E’	+*n$	输出 E’→+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→TE'			E→TE'		
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→ FT'			T→ FT'		
T'		T'→ ε	T'→* FT'		T'→ ε	T'→ ε
F	F→ n			F→ (E)		

步骤	分析栈	剩余输入串	动作
1	$E	**nn*n**$	输出 E→TE’
2	$E’T	nn*n$	输出 T→FT’
3	$E’T’F	nn*n$	输出 F→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→TE'			E→TE'		
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→ FT'			T→ FT'		
T'		T'→ ε	T'→* FT'		T'→ ε	T'→ ε
F	F→ n			F→ (E)		

步骤	分析栈	剩余输入串	动作
1	$E	**n+n)**$	输出 E→TE’
2	$E’T	n+n)$	输出 T→ FT’
3	$E’T’F	n+n)$	输出 F→n
4	$E’T’n	n+n)$	n 匹配
5	$E’T’	+n)$	输出 T’→ ε
6	$E’	+n)$	输出 E’→+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→ FT’
9	$E’T’F	n)$	输出 F→n
10	$E’T’n	n)$	n 匹配
11	$E’T’	)$	输出 T’→ ε
12	$E’	)$	输出 E’→ ε
13	$	)$	**ERROR**

随堂练习（7）

- 对文法G[S]：S→aBa，B→bB|ε，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→aBa		
B	B→ε	B→bB	

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

随堂练习（7）

- 对文法G[S]：S→aBa，B→bB|ε，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→aBa		
B	B→ε	B→bB	

步骤	分析栈	剩余输入串	动作
1	$S	aabb$	输出 S→aBa
2	$aBa	aabb$	a 匹配
3	$aB	abb$	输出 B→ε
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，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∈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lookhead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→TE'			E→TE'	**synch**	**synch**
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→ FT'	**synch**		T→ FT'	**synch**	**synch**
T'		T'→ ε	T'→* FT'		T'→ ε	T'→ ε
F	F→ i	**synch**	**synch**	F→ (E)	**synch**	**synch**

FOLLOW(E)={), $}　　FOLLOW(E')={), $}

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

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→TE'			E→TE'	**synch**	**synch**
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→ FT'	**synch**		T→ FT'	**synch**	**synch**
T'		T'→ ε	T'→*FT'		T'→ ε	T'→ ε
F	F→ i	**synch**	**synch**	F→ (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→TE’
3	$E’T	i*+i$	输出 T→ FT’
4	$E’T’F	i*+i$	输出 F→ i
5	$E’T’i	i*+i$	i 匹配
6	$E’T’	*+i$	输出 T'→*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'→ ε

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→TE'			E→TE'	**synch**	**synch**
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→ FT'	**synch**		T→ FT'	**synch**	**synch**
T'		T'→ ε	T'→*FT'		T'→ ε	T'→ ε
F	F→ i	**synch**	**synch**	F→ (E)	**synch**	**synch**

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

第四章课后作业（1）

- 对文法G[S]：S→(L)|a，L→L,S|S，构造预测分析表（不含恐慌模式），并对输入符号串(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1日晚上12:00**；
  - **重要提示：不得抄袭！**