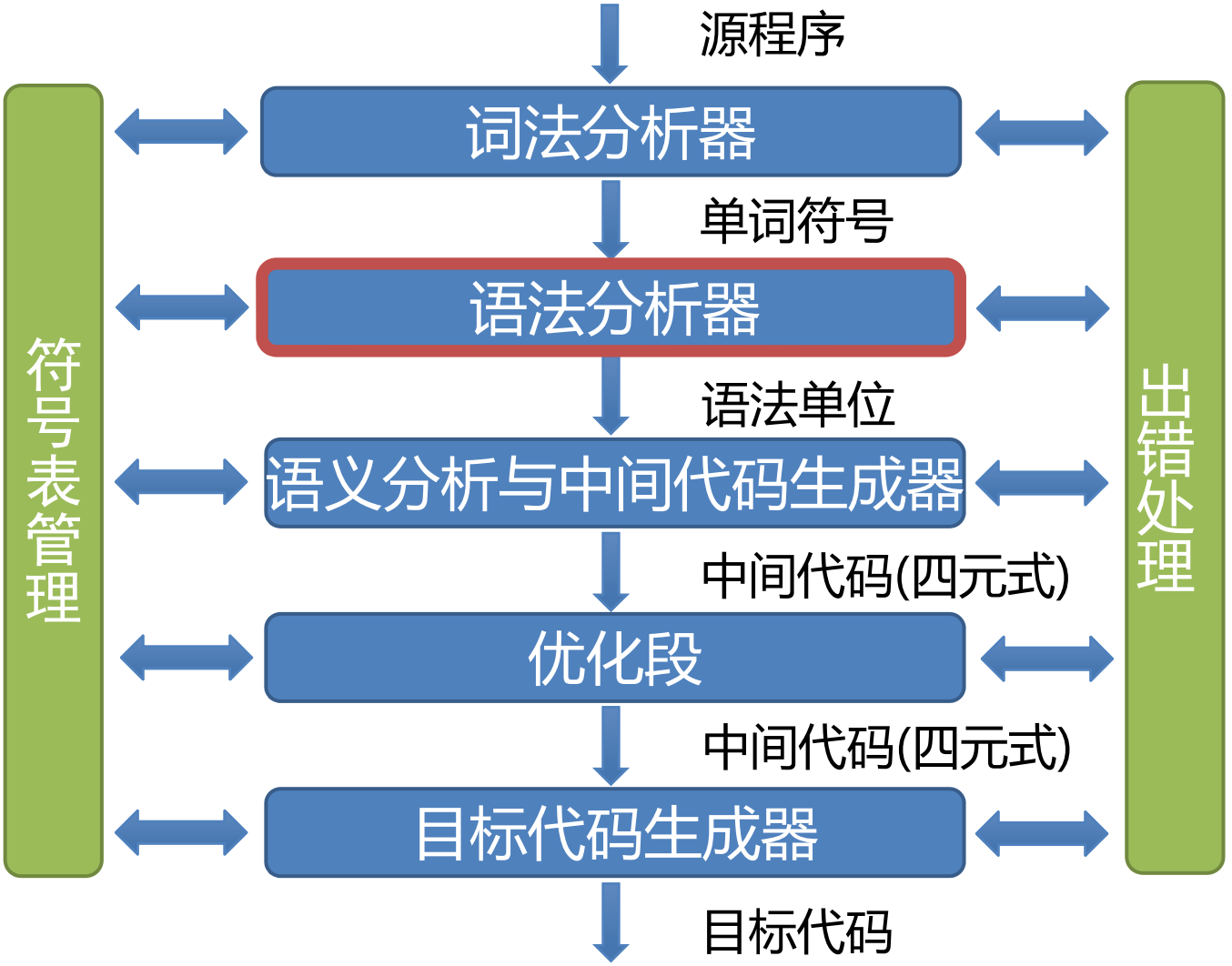
编译原理

LR分析法

编译原理

LR分析法概述

编译程序总框


源程序
词法分析器
单词符号
语法分析器
语法单位
语义分析与中间代码生成器
中间代码(四元式)
优化段
中间代码(四元式)
目标代码生成器
目标代码
符号表管理
出错处理

自下而上分析法(Bottom-up)

- 基本思想
  - 从输入串开始，逐步归约，直到文法的开始符号
  - 归约：根据文法的产生式规则，把串中出现的产生式的右部替换成左部符号
  - 从树叶节点开始，构造语法树
- 算符优先分析法
  - 按照算符的优先关系和结合性质进行语法分析
  - 适合分析表达式
- LR分析法
  - 规范归约：句柄作为可归约串

LR分析法

- 1965年 由Knuth提出
- L：从左到右扫描输入串
- R：自下而上进行归约

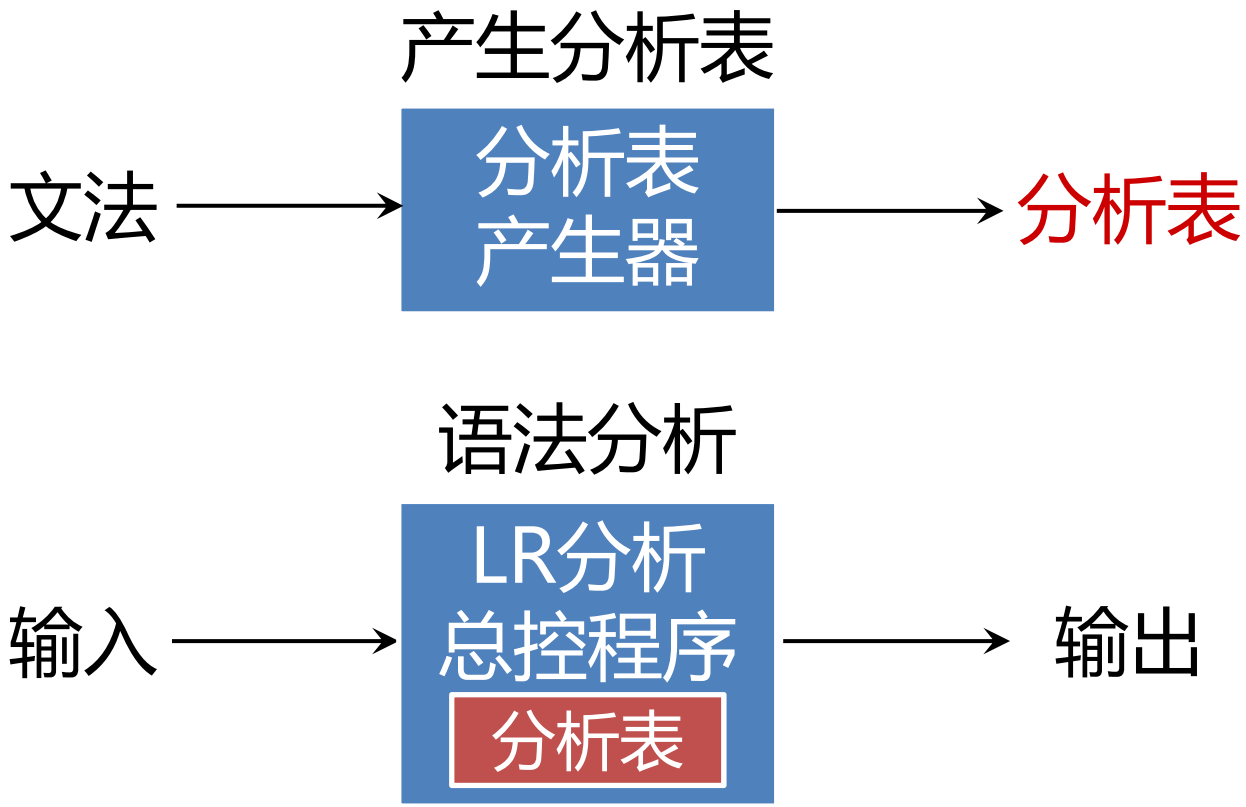
Donald Ervin Knuth

For his major contributions to the analysis of algorithms and the design of programming languages, and in particular for his contributions to "the art of computer programming" through his well-known books in a continuous series by this title.

LR分析法

- 计算思维的典型方法
  - 知识与控制的分离
  - 自动化

- 工作框架



编译原理

句柄和规范归约

短语、直接短语和句柄

- 定义：令G是一个文法，S是文法的开始符号，假定αβδ是文法G的一个句型，如果有

$$S \overset{*}{\Rightarrow} \alpha A \delta \text{ 且 } A \overset{+}{\Rightarrow} \beta$$

则β称是句型αβδ相对于非终结符A的短语。

- 如果有$A \Rightarrow \beta$,则称β是句型αβδ相对于规则$A \rightarrow \beta$的直接短语。
- 一个句型的最左直接短语称为该句型的句柄。

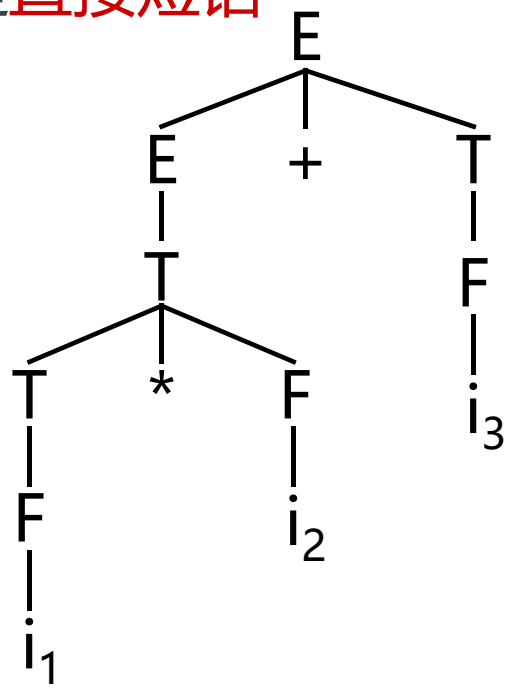
短语、直接短语和句柄

- 在一个句型对应的语法树中
  - 以某非终结符为根的两代以上的子树的所有末端结点从左到右排列就是相对于该非终结符的一个短语
  - 如果子树只有两代，则该短语就是直接短语
  - 最左两代子树末端就是句柄

短语：i_1，i_2，i_3，i_1*i_2，$i_1*i_2+i_3$

直接短语：i_1，i_2，i_3

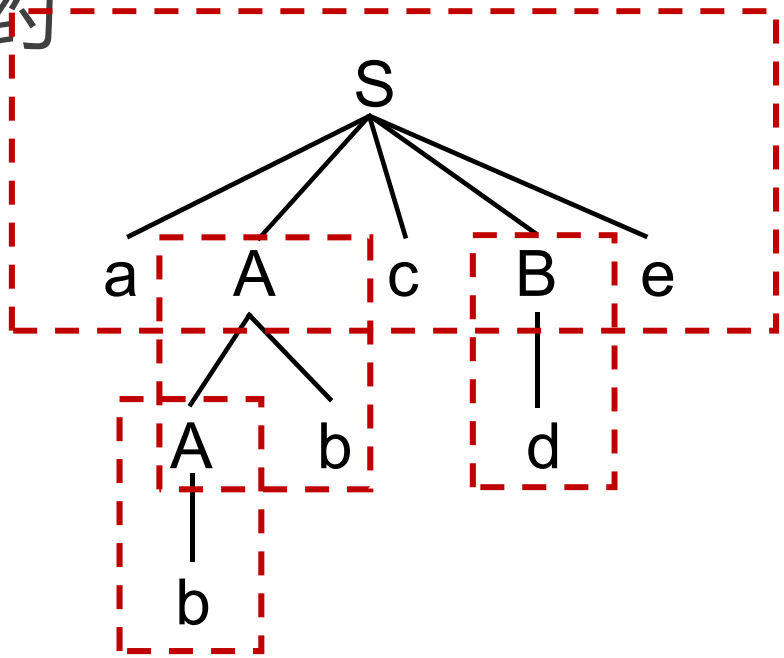
句柄：i_1



用句柄归约

- 可用句柄来对句子进行归约

句型	归约规则
abbcde	(2) A → b
aAbcde	(3) A → Ab
aAcde	(4) B → d
aAcBe	(1) S → aAcBe
S	



规范归约

- 定义：假定α是文法G的一个句子，我们称序列α_n，α_{n-1}，…，α_0 是α的一个规范归约，如果此序列满足：

1. $\alpha_n = \alpha$
2. α_0为文法的开始符号，即$\alpha_0 = S$
3. 对任何i，$0 \le i \le n$，α_{i-1}是从α_i经把句柄替换成为相应产生式左部符号而得到的

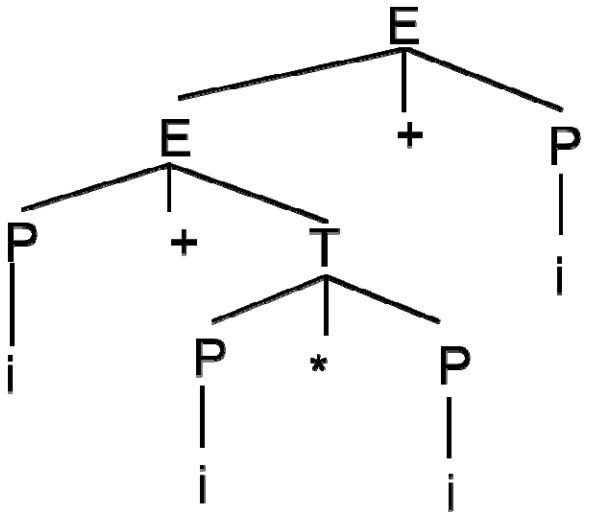
算符优先分析 vs. 规范归约

- 算符优先分析一般并不等价于规范归约

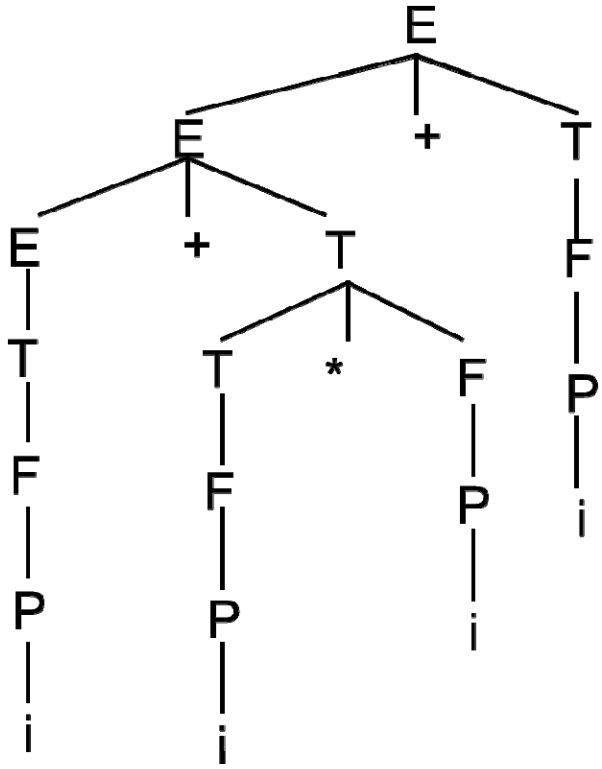
考虑文法G(E):

(1) E→E+T | T
(2) T→T*F | F
(3) F→P ↑ F | P
(4) P→(E) | i

的句子i+i*i+i



算符优先分析结果



规范归约分析结果

规范句型

- 规范归约是最左归约
- 规范归约的逆过程就是最右推导

$S \Rightarrow aAcBe \Rightarrow aAcde \Rightarrow aAbcde \Rightarrow abbcde$

- 最右推导也称为规范推导
- 由规范推导推出的句型称为规范句型

规范句型

句型	归约规则
a<u>b</u>bcde	(2) A → b
a<u>Ab</u>cde	(3) A → Ab
aAc<u>d</u>e	(4) B → d
<u>aAcBe</u>	(1) S → aAcBe
S	

编译原理

LR分析法

规范归约 vs. 句柄

- 规范归约的关键问题是寻找句柄.

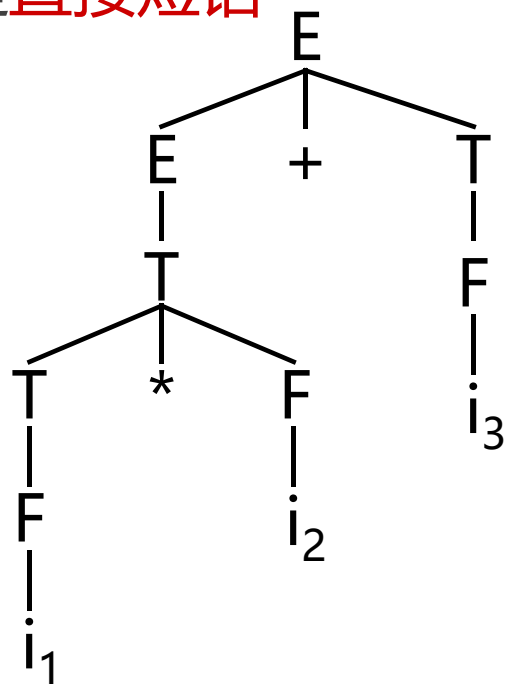
短语、直接短语和句柄

- 在一个句型对应的语法树中
  - 以某非终结符为根的两代以上的子树的所有末端结点从左到右排列就是相对于该非终结符的一个短语
  - 如果子树只有两代，则该短语就是直接短语
  - 最左两代子树末端就是句柄

短语： i_1、i_2、i_3、i_1*i_2、$i_1*i_2+i_3$

直接短语： i_1、i_2、i_3

句柄： i_1

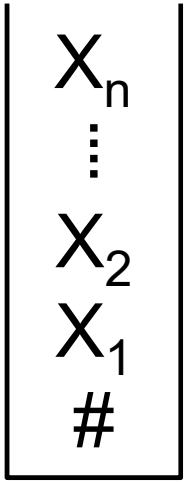


规范归约 vs. 句柄

- 规范归约的关键问题是寻找句柄.
  - 历史：已移入符号栈的内容
  - 展望：根据产生式推测未来可能遇到的输入符号
  - 现实：当前的输入符号

文法G(E)：

(1) E→E+T | T
(2) T→T*F | F
(3) F→P ↑ F | P
(4) P→(E) | i



分析栈

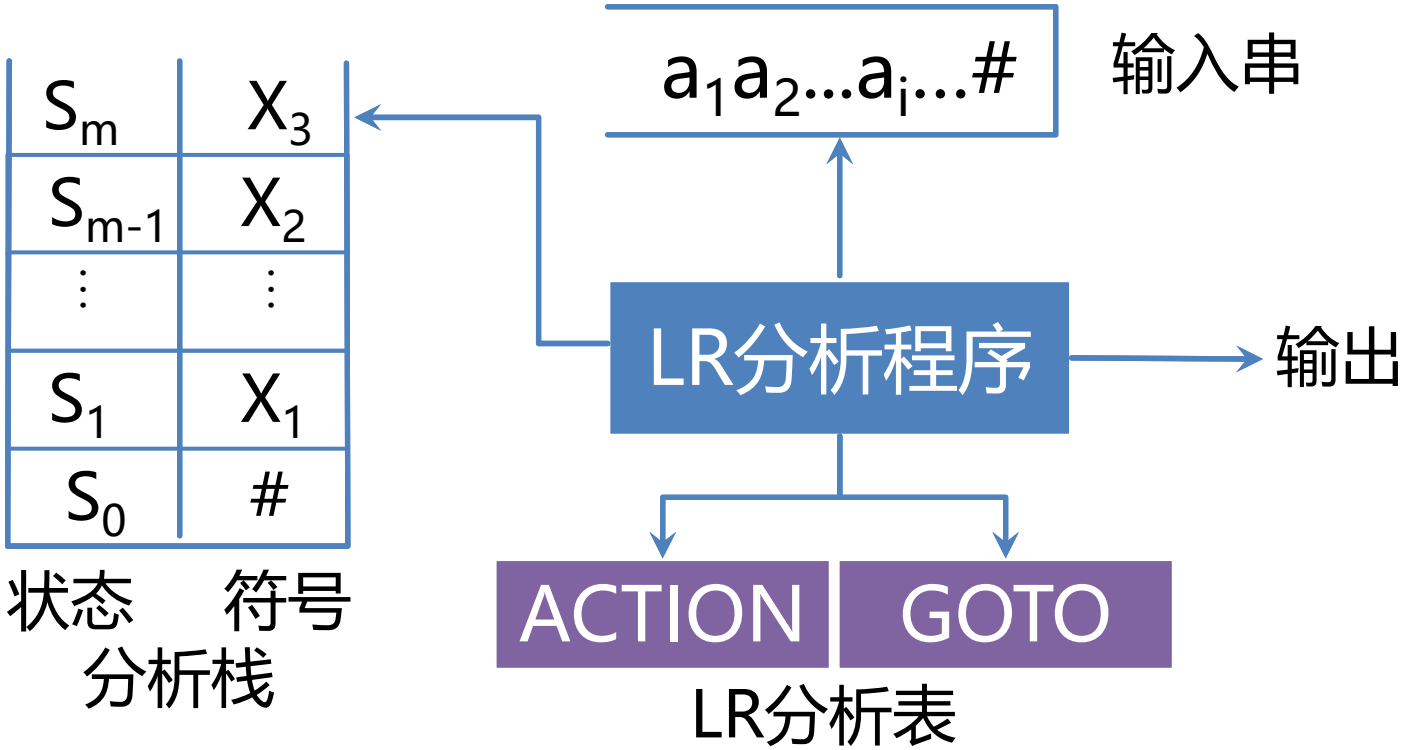
输入串

a……#

LR分析器的结构

- 计算思维的典型方法
  - 知识与控制的分离
  - 自动化

- LR分析方法：把"历史"及"展望"综合抽象成状态；由栈顶的状态和现行的输入符号唯一确定每一步工作



LR分析表

- LR分析器的核心是一张分析表
  - ACTION[s，a]：当状态s面临输入符号a时，应采取什么动作.
  - GOTO[s，X]：状态s面对文法符号X时，下一状态是什么

	ACTION						GOTO		
状态	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10

LR分析表

移进(shift)：把(s，a)的下一状态s′和输入符号a推进栈，下一输入符号变成现行输入符号

	ACTION						GOTO		
状态	i		*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

LR分析表

归约(reduce)：用某产生式$A \rightarrow \beta$进行归约。 假若β的长度为r， 去除栈顶r个项，使状态s_{m-r}变成栈顶状态，然后把下一状态$s'=GOTO[s_{m-r}, A]$和文法符号A推进栈

	ACTION						GOTO		
状态	i	+			)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

LR分析表

接受：宣布分析成功，停止分析器工作

	ACTION						GOTO		
状态	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

LR分析表

报错

	ACTION						GOTO		
状态	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

LR分析过程

- 分析开始时:

  状态栈　　符号栈　　输入串

  $(s_0,$　　$\#,$　　$a_1a_2 \ldots a_n \#)$

- 以后每步的结果可以表示为:

$(s_0\ s_1 \ldots s_m,$　$\#\ X_1 \ldots X_m,$　$a_ia_{i+1} \ldots a_n \#)$

LR分析过程

$(s_0\ s_1\ \dots\ s_m\ ,\ \#\ X_1\ \dots\ X_m\ ,\ a_i a_{i+1}\ \dots\ a_n\ \#)$

$(s_0\ s_1 \dots\ s_{m-r}\ s_{m-r+1}\ \dots\ s_m\ ,\ \#\ X_1 \dots X_{m-r}\ X_{m-r+1} \dots X_m\ ,\ a_i a_{i+1}\ \dots\ a_n\ \#)$

$(s_0\ s_1 \dots\ s_{m-r}\ ,\ \#\ X_1 \dots X_{m-r},\ a_i a_{i+1}\ \dots\ a_n\ \#)$

$(s_0\ s_1 \dots\ s_{m-r},\ \#\ X_1 \dots X_{m-r} A\ ,\ a_i a_{i+1}\ \dots\ a_n\ \#)$

$(s_0\ s_1 \dots\ s_{m-r} s\ ,\ \#\ X_1 \dots X_{m-r} A\ ,\ a_i a_{i+1}\ \dots\ a_n\ \#)$

$(s_0\ s_1\ \dots\ s_{m-r} s\ ,\ \#\ X_1\ \dots\ X_{m-r} A\ ,\ a_i a_{i+1}\ \dots\ a_n\ \#)$

此处, s=GOTO(s_{m-r}, A), r为β的长度, $\beta = X_{m-r+1} \dots X_m$

3. 若ACTION(s_m , a_i)为“接受”，则格局变化过程终止，宣布分析成功。

4. 若ACTION(s_m , a_i)为“报错”，则格局变化过程终止，报告错误。

编译原理

LR分析示例

LR分析示例

- 根据文法G(E)：
  (1) E→E＋T
  (2) E→T
  (3) T→T*F
  (4) T→F
  (5) F→(E)
  (6) F→I

分析输入串i*i+i

	ACTION						GOTO		
状态	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

步骤	状态	符号	输入串
(1)	0	#	i*i+i#
(2)	05	#i	*i+i#
(3)	03	#F	*i+i#
(4)	02	#T	*i+i#
(5)	027	#T*	i+i#

文法G(E):
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→i

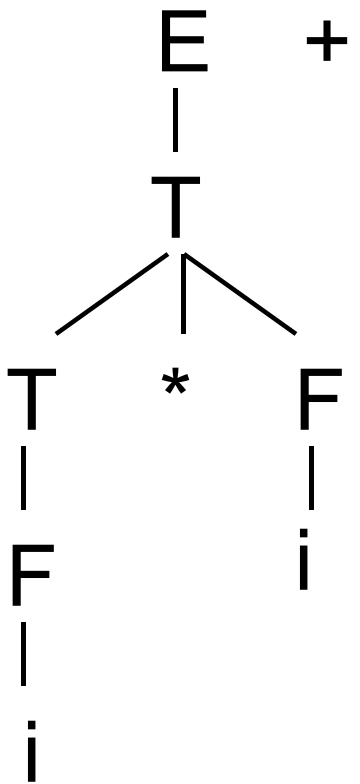
	ACTION						GOTO		
状态	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

T　　*
|
F
|
i

步骤	状态	符号	输入串
(5)	027	#T*	i+i#
(6)	0275	#T*i	+i#
(7)	02710	#T*F	+i#
(8)	02	#T	+i#
(9)	01	#E	+i#
(10)	016	#E+	i#

文法G(E):
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→i

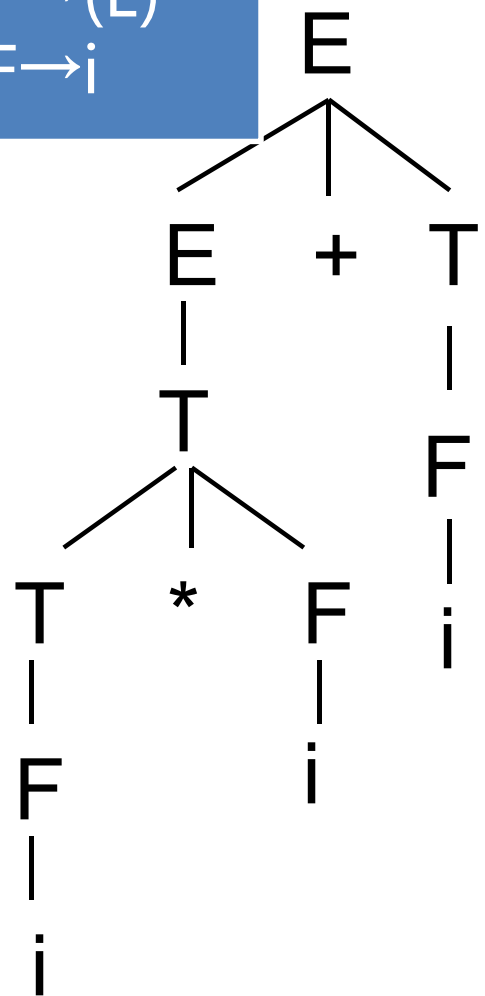
	ACTION						GOTO		
状态	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			



步骤	状态	符号	输入串
(10)	016	#E+	i#
(11)	0165	#E+i	#
(12)	0163	#E+F	#
(13)	0169	#E+T	#
(14)	01	#E	#
(15)	接受		

文法G(E):
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→i

	ACTION						GOTO		
状态	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			



编译原理

LR文法

LR文法

- 定义：对于一个文法，如果能够构造一张分析表，使得它的每个入口均是唯一确定的，则这个文法就称为LR文法。
- 定义：一个文法，如果能用一个每步顶多向前检查k个输入符号的LR分析器进行分析，则这个文法就称为LR(k)文法.

LR文法与二义文法

- LR文法不是二义的，二义文法肯定不会是LR的
- LR文法⊂ 无二义文法
- 非LR结构
  - S → iCtS | iCtSeS

栈	输入
#…iCtS	e…#

小结

- LR分析器的工作原理
- LR分析器的性质
  - 栈内的符号串和扫描剩下的输入符号串构成了一个规范句型
  - 一旦栈的顶部出现可归约串(句柄)，则进行归约