编译原理与技术

--自底向上的语法分析I

陈俊洁

天津大学智算学部

Outline

- 自底向上的语法分析基本问题
  - 移动 – 归约分析法
  - 用栈实现移动归约分析
- 算符优先分析法
  - 算符优先分析法定义、优先分析表的确定
  - 使用算符优先关系进行分析
  - 算符优先分析中的错误恢复
- LR分析法
- 语法分析器的自动产生工具Yacc

自底向上语法分析

- 自底向上的语法分析方法是一种“移进-归约”分析法。
  - 最易于实现的一种移进-归约分析方法，叫做算符优先分析法，
  - 而更一般的移进-归约分析方法叫做LR分析法，LR分析法可以用作许多自动的语法分析器的生成器。
- 移进-归约分析法为输入串构造分析树时是从叶结点（底端）开始，向根结点（顶端）前进。
  - 我们可以把该过程看成是把输入串w“归约”成文法开始符号的过程。
  - 在每一步归约中，如果一个子串和某个产生式的右部匹配，则用该产生式的左部符号代替该子串。
  - 如果每一步都能恰当的选择子串，我们就可以得到最右推导的逆过程。

移进-归约分析例子

假定文法为

(1) S→aAcBe
(2) A→b
(3) A→Ab
(4) B→d

将输入串**abbcde**归约到**S.**

步骤：	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
动作:	进 a	进 b	归 (2)	进 b	归 (3)	进 c	进 d	归 (4)	进 e	归 (1)
									e	
							d	B	B	
				b		c	c	c	c	
		b	A	A	A	A	A	A	A	
	a	a	a	a	a	a	a	a	a	S

关键问题

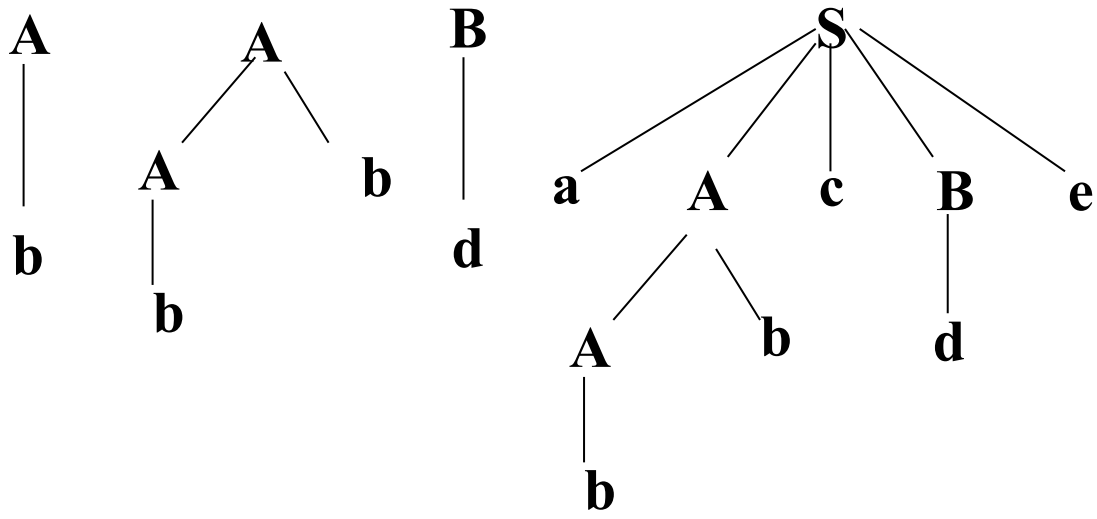
- 如何决定何时进行规约？即用哪一个产生式进行规约？

⇒ 精确定义“可规约串”

不同的可归约串形成了不同的自下而上分析法

- 算符优先分析法中用“最左素短语”；
- 规范归约分析法中用“句柄”；

语法分析过程中，可用一棵分析树来表示。在自下而上分析过程中，每一步归约都可以画出一棵子树来。归约完成，形成一棵分析树。



短语、直接短语、句柄的定义：

- 文法G[S]，αβδ是文法G的一个**句型**，$S \overset{*}{\Rightarrow} \alpha A \delta$且$A \overset{+}{\Rightarrow} \beta$

  则称β是句型αβδ相对于非终结符A的**短语**。
  若有A ⇒β则称β是句型αβδ相对于该规则A→β的**直接短语**。
- 一个句型的**最左直接短语**称为该句型的**句柄**。

!!注意两个条件缺一不可

例如文法：

$$E \longrightarrow T|E+T$$
$$T \longrightarrow F|T*F$$
$$F \longrightarrow (E)|a$$

a1*a2+a3是文法的句型，
找出此文法的短语，直接短语和句柄。

例子

- 考虑右边所示文法的一个句型**a1*a2+a3:**
  - **a1,a2,a3,a1*a2, a1*a2+a3**都是句型**a1*a2+a3**的短语,
  - **a1,a2**和**a3**是直接短语,
  - **a1**是最左直接短语,
  - **a2+a3**不是句型**a1*a2+a3**短语, 因为有**E=>a2+a3**但不存在从文法的开始符号**E**到**a1*E**的推导.

$$E \longrightarrow T|E+T$$
$$T \longrightarrow F|T*F$$
$$F \longrightarrow (E)|a$$

- 文法G[S]，αβδ是文法G的一个**句型**，$S \overset{*}{=>} \alpha A \delta$且$A \overset{+}{=>} \beta$ 则称β是句型αβδ相对于非终结符A的**短语**。
  若有$A \Rightarrow \beta$则称β是句型αβδ相对于该规则A→β的**直接短语**。
- 一个句型的**最左直接短语**称为该句型的**句柄**。

用子树解释短语，直接短语，句柄:

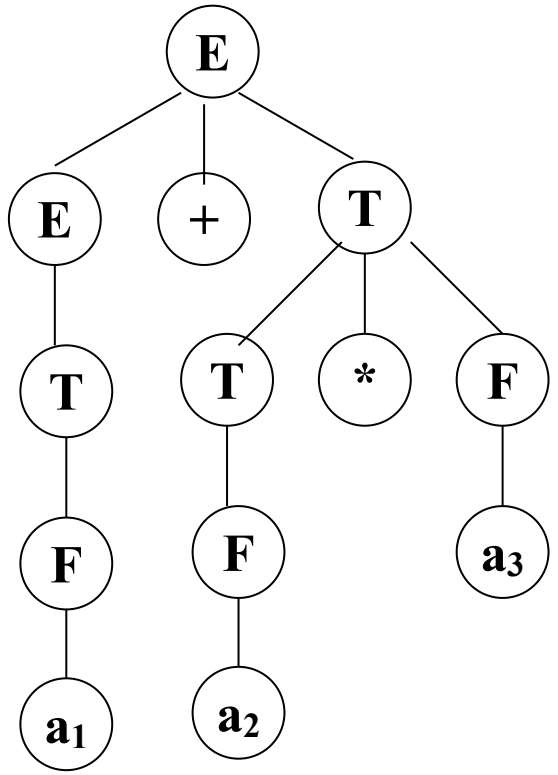
- 短语：一棵子树的所有叶子自左至右排列起来形成一个相对于子树根的短语。
- 直接短语：仅有父子两代的一棵子树，它的所有叶子自左至右排列起来所形成的符号串。
- 句柄：一个句型的分析树中最左那棵只有父子两代的子树的所有叶子的自左至右排列。

例子

对表达式文法G[E]和句子a1+a2*a3，挑选出推导过程中产生的句型中的短语，直接短语，句柄。

$$E \longrightarrow T|E+T$$
$$T \longrightarrow F|T*F$$
$$F \longrightarrow (E)|a$$

E	短语
⇒E+T	E+T
⇒E+T*F	T*F,E+T*F
⇒E+T*a_3	a_3,T*a_3,E+T*a_3
⇒ E+F*a_3	a_3,F,F*a_3,E+F*a_3
⇒ E+a_2*a_3	a_3,a_2,a_2*a_3, E+a_2*a_3
⇒ T+a_2 *a_3	a_3,a_2,T, a_2*a_3, T+a_2*a_3
⇒ F+a_2*a_3	a_3,a_2,F, a_2*a_3, F+a_2*a_3
⇒a_1+a_2 *a_3	a_1, a_2, a_3, a_2 * a_3 a_1+ a_2 *a_3



规范归约定义

假定a是文法G的一个句子, 称序列$a_n,a_{n-1},\ldots,a_0$ 是a的一个规范归约,此序列应满足:

1) $a_n=a$

2) a_0为文法的开始符, 即$a_0=S$

3) 对任何$i, 0<i<=n$, a_{i-1}是从a_i经把句柄替换为相应产生式的左部符号而得到的

规范归约是关于a的一个最右推导的逆过程。 规范归约也称最左归约。

在形式语言中，最右推导常被称为规范推导，由规范推导所得的句型称为规范句型。规范归约的实质是：在移进过程中,当发现栈顶呈现句柄时，就用相应产生式的左部符号进行替换。

回顾例子

考虑文法：

(1) S→aAcBe

(2) A→b

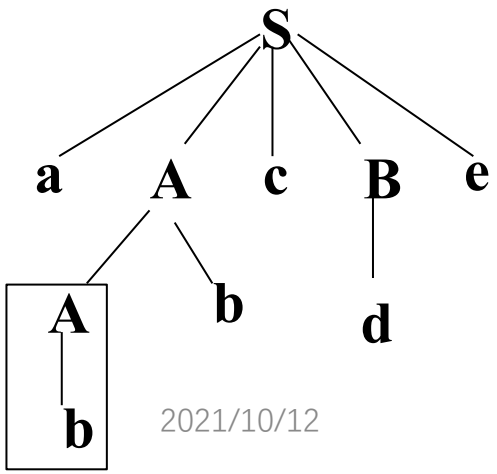
(3) A→Ab

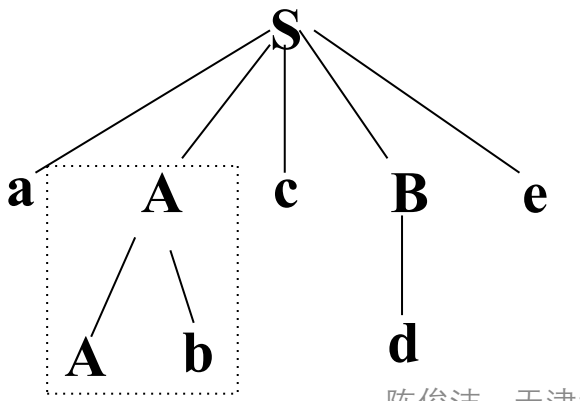
(4) B→d

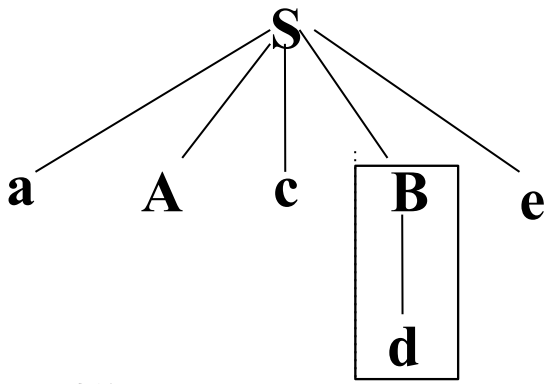
将输入串**abbcde**归约到**S.**

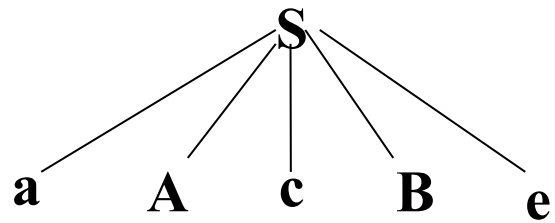
对该文法的句子**abbcde**逐步寻找句柄, 并用相应产生式的左部符号去替换, 得到如下归约过程: (画底线的部分是句柄).

句型	归约过程
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	









“移进-归约”分析法的栈实现

- 移进-归约分析器使用符号栈和输入缓冲区。用“#”作为栈底。分析器的工作过程中，对符号栈的使用有四类操作:“移进”，“归约”，“接受”,和“出错处理”。
- 输入串i1*i2+i3的分析(规范归约)步骤如下:

$$E \longrightarrow E+E$$
$$E \longrightarrow E*E$$
$$E \longrightarrow (E)|i$$

步骤	符号栈	输入串	动作
0	#	i1*i2+i3#	进
1	# i1	*i2+i3#	归E→i
2	#E	*i2+i3#	进
3	#E*	i2+i3#	进
4	#E*i2	+i3#	归E→i
5	#E*E	+i3#	归, E→E*E
6	#E	+i3#	进,
7	#E+	i3#	进
8	#E+i3	#	归,E →I
9	#E+E	#	归,E→E+E
10	#E	#	接受

注意:任何可归约串的出现必须在栈顶.

$E \longrightarrow E+E$

$E \longrightarrow E*E$

$E \longrightarrow (E)|i$

练习

规范规约：

考虑文法G[S]：

(1)S → aABe
(2)A→ b
(3)A→ Abc
(4)B→ d

最右推导：

S =>aABe
=>aAde
=>aAbcde
=>abbcde

步骤	符号栈	输入符号串	动作

移进归约分析中需要解决的问题

- 定位右句型中将要归约的子串（可归约串）
  - 在用堆栈实现时，这个子串总是在栈顶。语法分析器不需要深入到栈中去寻找句柄
- 选择产生式对选定的串进行归约
  - 如果选定的子串是多个产生式的右部，要确定选择哪个产生式进行归约
- 移进归约分析过程中的冲突
  - 根据栈中的内容和下一个输入符号不能决定是移动还是归约（移进-归约冲突）
  - 不能决定按哪一个产生式进行归约（归约-归约冲突）