第5章
自底向上优先分析

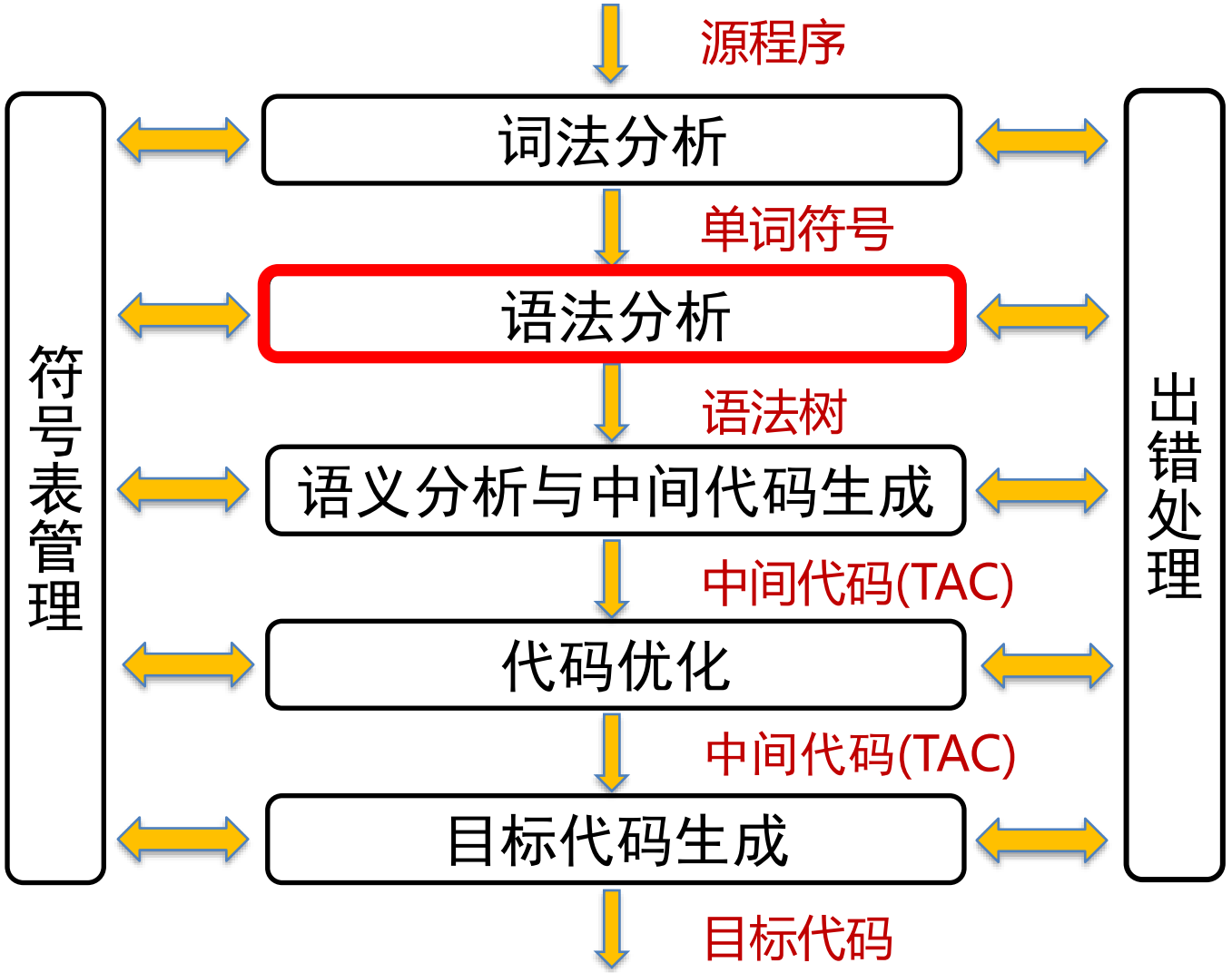
Chapter 5 Bottom-Up Parsing:

Precedence Grammar

主讲人：杨茂林 kylin@hust.edu.cn

编译程序的结构

- 语法分析在整个编译过程中所处的位置



语法分析方法

- 自上而下(Top-down)
  - 从文法的开始符号出发，反复使用各种产生式，寻找"匹配"的推导
  - 推导：根据文法的产生式规则，把串中出现的产生式的LHS符号替换成RHS
  - 从树的根开始，构造语法树
  - 递归下降分析法、预测分析i法
- 自下而上(Bottom-up)
  - 从输入串开始，逐步进行归约，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RHS替换成LHS符号
  - 从树叶节点开始，构造语法树
  - [简单|算符]优先分析法
    LR分析法

内容提要

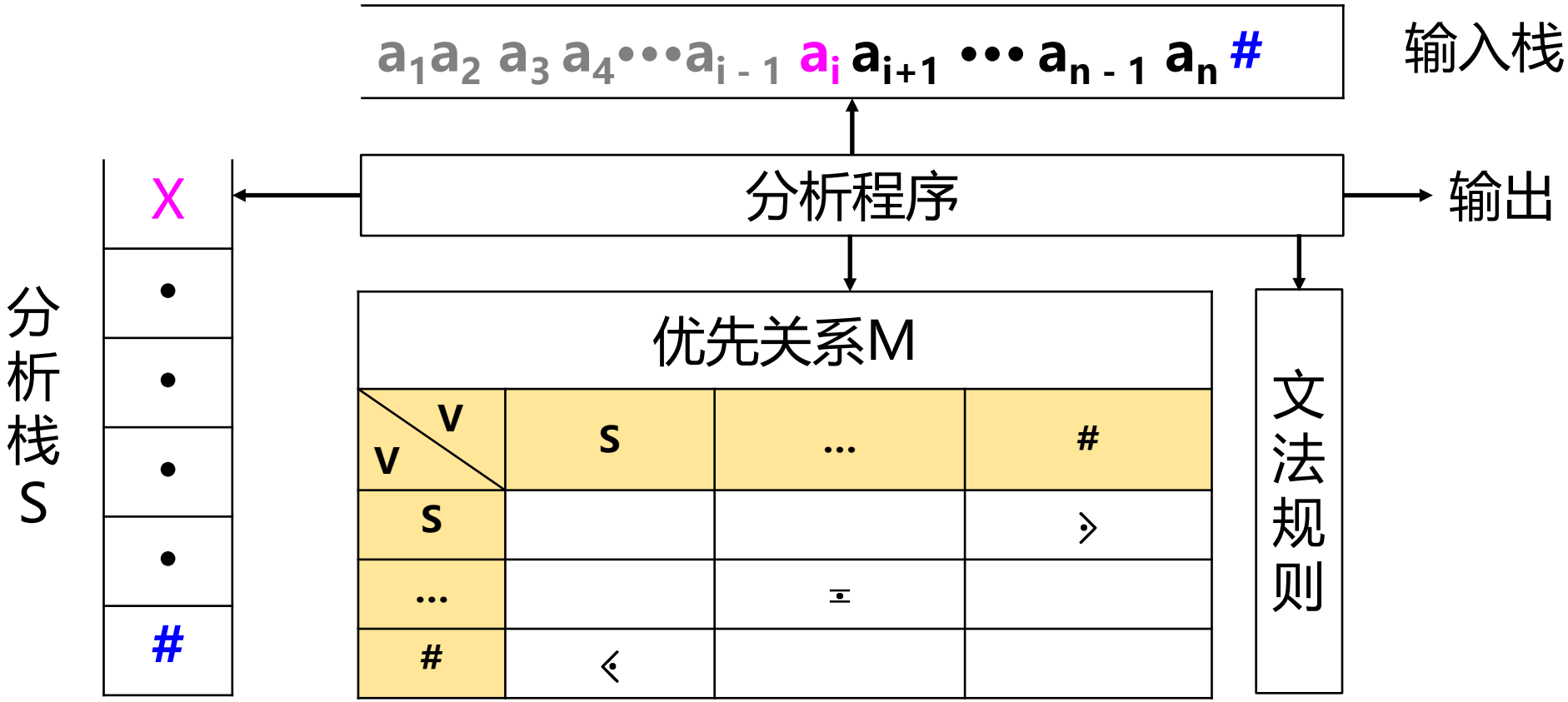
- 简单优先分析法和算符优先分析法
- 优先关系作用、优先关系计算方法、分析表的构造、优先分析法适用条件和语法分析程序结构及其分析算法。

本章主要内容

- **5.1　优先分析概述**
- **5.2　简单优先分析法**
- **5.3　算符优先分析法**

5.1 优先分析概述

- 移进-归约法：根据文法寻找句型的句柄
- 优先分析法：利用句型相邻两个符号之间的 “优先关系”确定句柄
- if ($X \lessdot a_i$) shift(a_i); else if ($X \gtrdot a_i$) reduce(A→…X); else error();



本章主要内容

- 5.1　优先分析概述
- 5.2　简单优先分析法
- 5.3　算符优先分析法

5.2 简单优先分析法

- 定义5.1 设文法G = (V_N，V_T，P，S)，则 ($V_N \cup V_T$)上的3种优先关系定义如下(二元关系):

  $X \doteq Y$ iff $A \to \ldots XY \ldots \in P$

  $X \lessdot Y$ iff $A \to \ldots XB \ldots \in P$，$B \overset{+}{\Rightarrow} Y \cdots$

  $X \gtrdot Y$ iff $A \to \ldots BD \ldots \in P$，$B \overset{+}{\Rightarrow} \ldots X$，$D \overset{*}{\Rightarrow} Y \ldots$

其中，X、Y、$D \in (V_N \cup V_T)$，A、$B \in V_N$，$\cdots \in (V_N \cup V_T)^*$

特别地，句子括号#的优先级低于其右邻符号，其左邻符号优先级高于#

优先关系在句型中相邻的符号，反映了归约时的优先关系:

$X \doteq Y$表示在句型中相邻出现的XY，同时被归约;

$X \lessdot Y$表示在句型中相邻出现的XY，X后于Y被归约;

$X \gtrdot Y$表示在句型中相邻出现的XY，X先于Y被归约。

5.2 简单优先分析法

- 定义5.2 文法G的符号集V中,任意两个符号之间至多存在 $\doteq$ 、$\lessdot$ 和 $\gtrdot$ 三种简单优先关系之一，且没有相同右部的规则，则文法G称为简单优先文法。
- 关于简单优先文法，可以证明下列几个结论:

① 设文法G为简单优先文法，句型：

$\#a_1\ a_2\cdots a_{i-1}\ a_i\ a_{i+1}\cdots a_{j-1}\ a_j\ a_{j+1}\cdots a_n\ \#$，存在下列关系:

$a_{i-1} \lessdot a_i \doteq a_{i+1} \doteq \ldots \doteq a_{j-1} \doteq a_j \gtrdot a_{j+1}$，则

子串$a_i\ a_{i+1}\cdots a_{j-1}\ a_j$是句型的直接短语。

特别地，如果这个子串是句型最左子串，则该子串就是句型的句柄。

② 设文法G为简单优先文法，如果输入串或归约后的符号串中相邻的两个符号之间不存在任何一种简单优先关系，则输入串不是文法的句子。

③ 如果文法G是简单优先文法，则文法G是无二义性的文法。

5.2 简单优先分析法

- 简单优先分析法的思想：

① 初始化：#入分析栈S，w#入输入栈I，#在栈底;

② X=top(S); a = top(I);

③ if (X $\lessdot$ $\doteq$ a) shift(a); else if (X $\gtrdot$ a) reduce(A→…X); else error();

在S栈顶寻找以下关系序列,以确定句柄的首部:

$a_{i-1} \lessdot a_i \doteq a_{i+1} \doteq \ldots \doteq \ldots \doteq X \gtrdot a$ ，确定句柄为$a_i \ldots X$

再用A→ $a_i \ldots X$归约

④ 重复②③，直到归约出文法开始符S（#S)

- 简单优先文法中任何两个符号之间优先关系保存在优先关系矩阵M中。M[X，Y]存放3个优先关系符号 $\doteq$ 、 $\lessdot$ 、 $\gtrdot$ 和空白之一。

5.2 简单优先分析法

- 例5.1 设文法G[S]为:

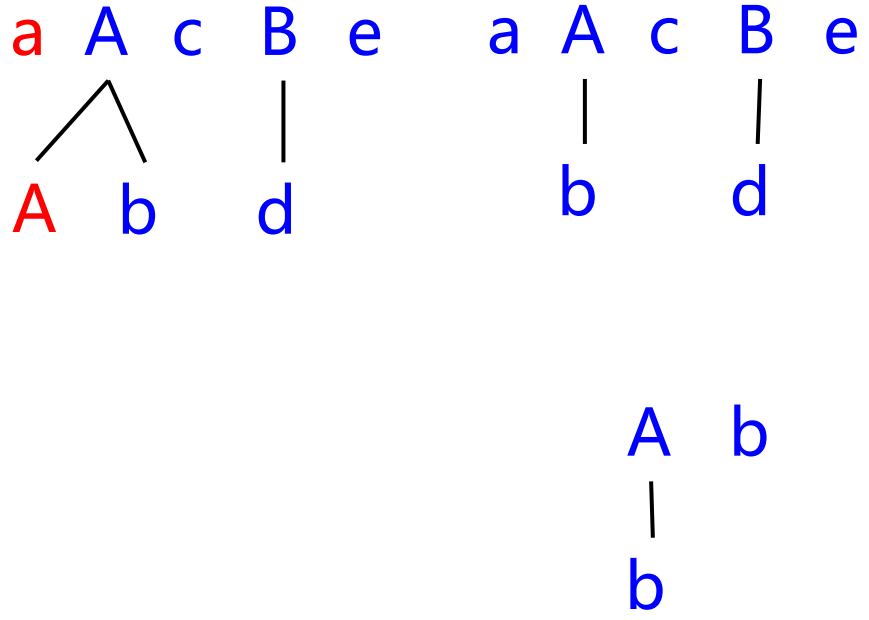
  (1) S→aAcBe

  (2) A→b

  (3) A→Ab

  (4) B→d

  判断文法G[S]是否简单优先文法。



解：依产生式(1)有: a $\doteq$ A，

依产生式(1)和(3)，有 : a $\lessdot$ A

∵ 文法符号a与A之间存在两种不同的优先关系，

∴ 文法G[S]不是简单优先文法。

5.2 简单优先分析法

- 例5.2 文法G[S]:

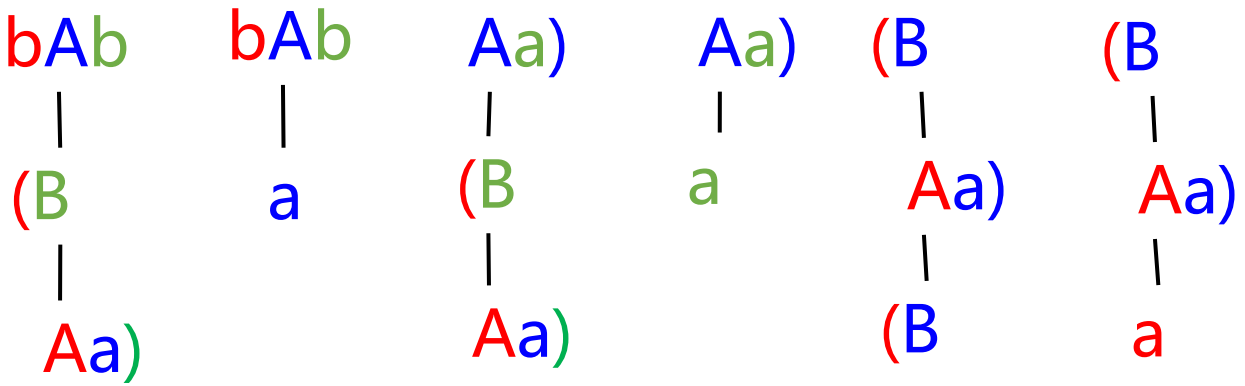
① S→bAb

② A→(B

③ A→ a

④ B→Aa)

试分析串w=b((aa)a)b是否文法的句子。



第1步:
简单优先关系矩阵(表)

V \ V	S	A	B	a	b	(	)	#
S								⋗
A				≐	≐			
B				⋗	⋗			
a				⋗	⋗		≐	
b		≐		⋖		⋖		⋗
(		⋖	≐	⋖		⋖		
)				⋗	⋗			
#	⋖				⋖			≐

5.2 简单优先分析法

- 第2步:分析串w=b((aa)a)b

V V	S	A	B	a	b	(	)	#
S								⋗
A				≐	≐			
B				⋗	⋗			
a				⋗	⋗		≐	
b		≐		⋖		⋖		⋗
(		⋖	≐	⋖		⋖		
)				⋗	⋗			
#	⋖				⋖			≐

① S→bAb
② A→(B
③ A→ a
④ B→Aa)

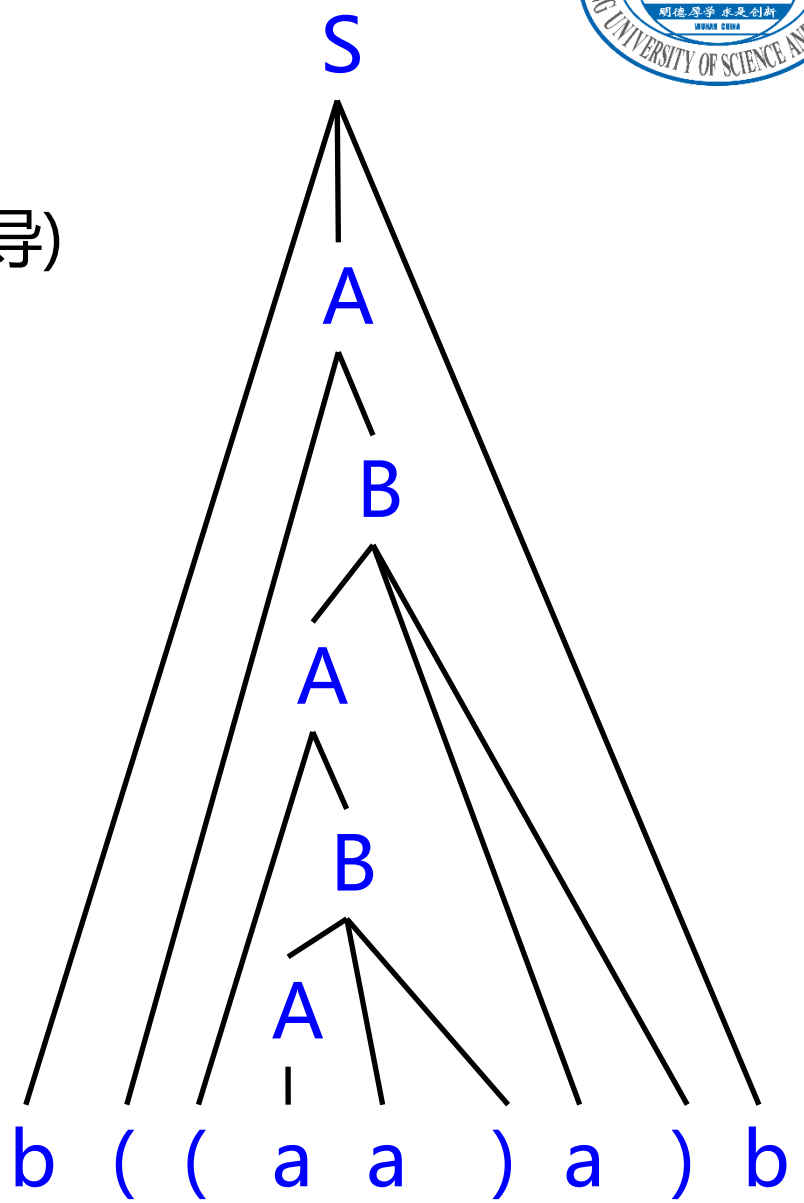
步骤	分析栈S	输入串I	动作
1	#	b((aa)a)b#	Shift,∵# ⋖ b
2	#b	((aa)a)b#	Shift,∵b ⋖ (
3	#b(	(aa)a)b#	Shift,∵(⋖ (
4	#b((	aa)a)b#	Shift,∵(⋖ a
5	#b((a	a)a)b#	Reduce, A→ a(∵a ⋗ a)
6	#b((A	a)a)b#	Shift,∵A ≐ a
7	#b((Aa	)a)b#	Shift,∵a ≐)
8	#b((Aa)	a)b#	Reduce, B→Aa) (∵) ⋗ a)
9	#b((B	a)b#	Reduce, A→(B (∵B ⋗ a)
10	#b(A	a)b#	Shift, ∵A ≐ a
11	#b(Aa	)b#	Shift, ∵a ≐)
12	#b(Aa)	b#	Reduce, B→Aa) (∵) ⋗ b)
13	#b(B	b#	Reduce, A→(B (∵B ⋗ b)
14	#bA	b#	Shift, ∵A ≐ b
15	#bAb	#	Reduce, S→bAb (∵B ⋗ #)
16	#S	#	accept

5.2 简单优先分析法

- 分析串w=b((aa)a)b
- 分析的过程，同时自底向上构造了语法树（最右推导）

① S→bAb
② A→(B
③ A→ a
④ B→Aa)

步骤	分析栈S	输入串I	动作
5	#b((a	a)a)b#	Reduce, A→ a(∵a ⋗ a)
8	#b((Aa)	a)b#	Reduce, B→Aa) (∵) ⋗ a)
9	#b((B	a)b#	Reduce, A→(B (∵B ⋗ a)
12	#b(Aa)	b#	Reduce, B→Aa) (∵) ⋗ b)
13	#b(B	b#	Reduce, A→(B (∵B ⋗ b)
15	#bAb	#	Reduce, S→bAb (∵B ⋗ #)

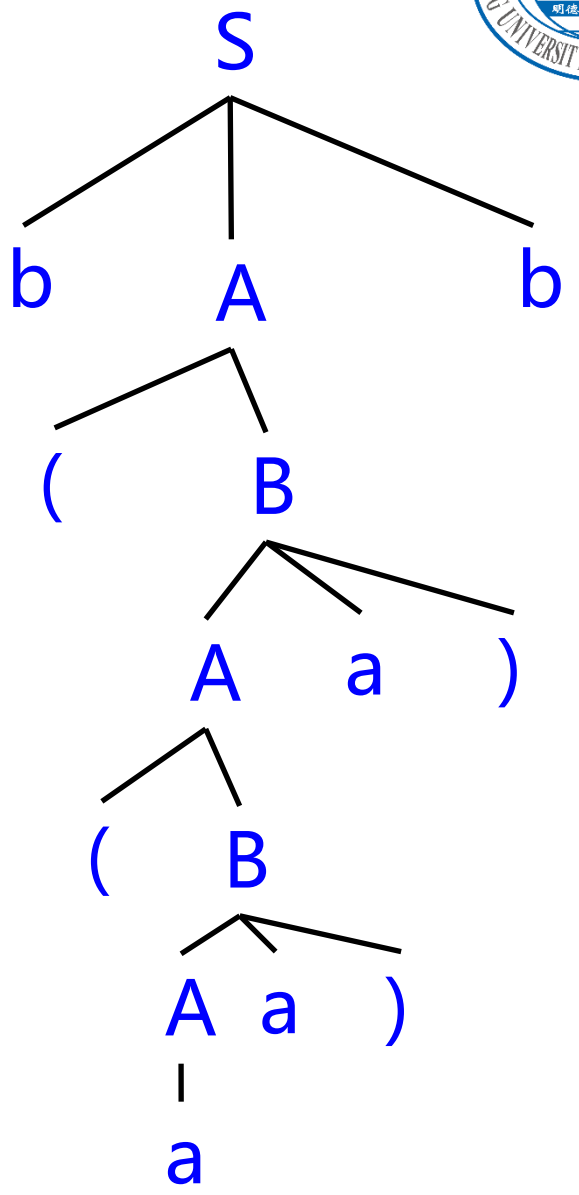


5.2 简单优先分析法

- 分析串w=b((aa)a)b
- 分析的过程，同时自底向上构造了语法树（最右推导）

① S→bAb
② A→(B
③ A→ a
④ B→Aa)

步骤	分析栈S	输入串I	动作
5	#b((a	a)a)b#	Reduce, A→ a(∵a ⋗ a)
8	#b((Aa)	a)b#	Reduce, B→Aa) (∵) ⋗ a)
9	#b((B	a)b#	Reduce, A→(B (∵B ⋗ a)
12	#b(Aa)	b#	Reduce, B→Aa) (∵) ⋗ b)
13	#b(B	b#	Reduce, A→(B (∵B ⋗ b)
15	#bAb	#	Reduce, S→bAb (∵B ⋗ #)



本章主要内容

- 5.1　优先分析概述
- 5.2　简单优先分析法
- **5.3　算符优先分析法**

5.3 算符优先分析法

- 与简单优先分析法不同，算符优先分析法仅仅利用相邻的两个终结符之间优先关系寻找归约子串。这个归约子串不是句柄，但它是一种特殊的直接短语，被称为“最左素短语”。
- 这种分析法源于对表达式处理。表达式是计算机语言中最重要的、最核心的组成部分，其语法特点是表达式运算符和运算对象组成。表达式组成是否符合语法，通常是以运算符的计算优先顺序和运算符需要运算对象个数进行判别的。

5.3 算符优先分析法

- 定义5.3 设文法G =(V_N，V_T，P，S），如果G中没有形如A→…BC…的规则，其中A、B、C∈V_N，则称文法G为算符文法（Operater Grammar），简称OG文法。
- 例如：表达式文法(有二义性的，无二义性但有左递归）均是OG文法

OG文法其实质是对规则进行了限制，也就是规则右部不得出现两个非终结符直接相邻。它是对表达式进行形式化和推广，终结符均视为算符，非终结符视为运算结果。

- 性质1 在OG文法中任何句型都不包含两个相邻的非终结符。
- 性质2 如果Ab或bA出现在OG文法的句型γ中，其中A∈V_N，b∈V_T，则句型γ中任何包含此b的短语必含有A。

5.3 算符优先分析法

■ 性质1 OG文法中任何句型都不包含两个相邻的非终结符

证明：按推导长度(步数)n，用归纳法证明。设γ为句型,$S \overset{*}{\Rightarrow} \gamma$.

$$S \Rightarrow \omega_1 \Rightarrow \omega_2 \Rightarrow \ldots \Rightarrow \omega_k \Rightarrow \omega_{k+1} = \gamma$$

当n=1时，必存在产生式A→γ，由OG文法的定义， γ满足性质1。

假设当n=k(>1)时， ω_k满足性质1。

不妨设$\omega_k = \alpha A \delta$, $A \in V_N$。

由假设，α的尾符号与δ的首符号都不会是非终结符，否则与假设矛盾。

设A →β是文法的产生式(依定义， β必不含两个相邻的非终结符)，则第k+1步推导有：

$$S \Rightarrow \ldots \Rightarrow \omega_k \Rightarrow \omega_{k+1} = \alpha\beta\delta = \gamma ,$$

显然也不会有两个相邻的非终结符。满足性质1，证毕。

5.3 算符优先分析法

- 性质2 如果Ab或bA出现在OG文法的句型γ中，其中A∈V_N，b∈V_T，则句型γ中任何包含此b的短语必含有A。

证明：用反证法。设

$S \overset{*}{\Rightarrow} \gamma = \ldots\alpha bA\beta\ldots$

假设有短语含b不含A，则必存在某个非终结符，比如$B \overset{*}{\Rightarrow} \ldots b$，则必有：

$S \overset{*}{\Rightarrow} \ldots BA\beta$

这样，在句型中就存在两个相邻的非终结符B和A，与性质1矛盾，故假设不成立，证毕。

- 注意：含b的短语必含A，含A的短语不一定含b。

5.3 算符优先分析法

- 例5.3 有文法G[S]:
  - ① S→BbA
  - ② B→b
  - ③ A→c
- 文法G[S]是否OG文法?
- $S \overset{*}{\Rightarrow} bbA$? bbA是句型吗?
- b是句型bbA的短语吗?
- 短语b不含A，与性质2是否矛盾?
- 记句型bbA为b_1b_2A，显有b_1为b_1b_2A的短语，且是句柄。性质2的意思是含b_2的短语必含A，此b非彼b。

5.3 算符优先分析法

- 定义5.4 设算符文法$G=(V_N, V_T, P, S)$，V_T上的3种算符优先关系定义如下：
  - $a \doteq b$ iff $A\to\dots ab\dots\in P$ 或$A\to\dots aBb\dots\in P$
  - $a \lessdot b$ iff $A\to\dots aB\dots\in P$，$B \overset{+}{\Rightarrow} b\dots$或$B \overset{+}{\Rightarrow} Cb\dots$
  - $a \gtrdot b$ iff $A\to\dots Bb\dots\in P$，$B \overset{+}{\Rightarrow} \dots a$ ，$B\overset{+}{\Rightarrow}\dots aC$

  其中，$a,b\in V_T$，$A,B,C\in V_N$，$\cdots\in(V_N\cup V_T)^*$
- 算符优先关系在句型中相邻的符号，反映了归约时的优先关系。即：
  - $a \doteq b$ 在句型中相邻出现的ab或aBb，同时被归约
  - $a \lessdot b$ 在句型中相邻出现的ab或aBb，a晚于b被归约
  - $a \gtrdot b$ 在句型中相邻出现的ab或aBb，a早于b被归约

5.3 算符优先分析法

- 定义5.5 设不含空规则的文法G为OG文法，如果任意两个终结符之间至多存在 $\doteq$ 、$\lessdot$ 和 $\gtrdot$ 三种算符优先关系之一，则称文法G为算符优先文法（Operator Precedence Grammar），简称OPG文法。

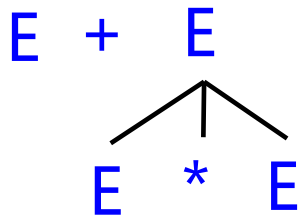
- 例5.4 表达式文法G[E]:

  ① E→E+E

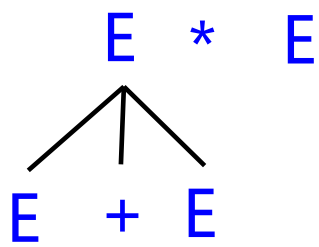
  ② E→E*E

  ③ E→i

  是算符文法，但不是算符优先文法。



图a



图b

∵ 由图 a知: + $\lessdot$ *，由图b知: + $\gtrdot$ *, 两个终结符之间的优先关系不唯一

∴ 文法G[E]不是算符优先文法。

- 算符优先文法是无二义性文法

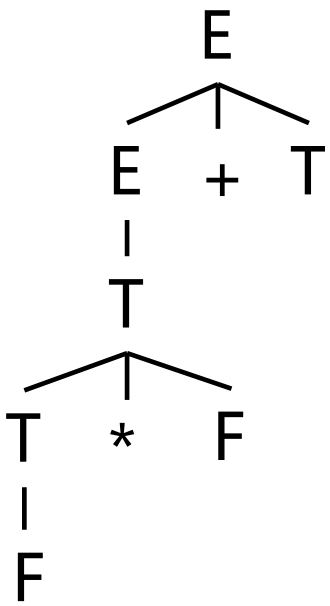
5.3 算符优先分析法

- 定义5.6 设文法G[S]，其句型的素短语是满足下列两个条件的短语：
  ① 至少含有一个终结符；
  ② 除自身外不包含其它素短语。

特别地，最左边的素短语称为最左素短语。

- 例5.5 表达式文法G[E]：E→E+T | T，T→T*F | F，F→(E) | i，求句型 F*F + T的短语，素短语，最左素短语。

  - 短语有F, F*F, F*F+T，其中F还是句柄；
  - F不是素短语(∵不含终结符)
  - F*F是素短语，且为最左素短语
  - F*F+T不是素短语，因为它包含素短语F*F



5.3 算符优先分析法

- 最左素短语定理:一个算符优先文法G的任何句型

  $\#N_1a_1N_2\ a_2\cdots N_{i-1}a_{i-1}\ N_ia_i\ N_{i+1}a_{i+1}\cdots N_{j-1}a_{j-1}\ N_ja_j\ N_{j+1}a_{j+1}\cdots N_na_n\ \#$

  (其中，a_i都是终结符，N_i是可有可无的非终结符)

的最左素短语是满足如下条件的最左子串 $N_ia_i...N_ja_jN_{j+1}$，

$$a_{i-1} \lessdot a_i \doteq a_{i+1} \doteq \dots \doteq a_{j-1} \doteq a_j \gtrdot a_{j+1}$$

- 设文法G为算符优先文法，如果输入串或归约后的符号串中，相邻的两个符号之间，不存在任何一种算符优先关系，则输入串不是文法的句子。
- 如果文法G是算符优先文法，则文法G是无二义性的文法。

5.3 算符优先分析法

- 定义5.7 设文法G = (V_N，V_T，P，S)，则FIRSTTV(B)和LASTTV(B)定义如下:

  FIRSTVT(B) = {a | $B \overset{+}{\Rightarrow} a \ldots$ 或 $B \overset{+}{\Rightarrow} C\ a \ldots$，$a \in V_T$且$C \in V_N$ }

  LASVT(B) = { a | $B \overset{+}{\Rightarrow} \ldots a$ 或 $B \overset{+}{\Rightarrow} \ldots a\ C$ ，$a \in V_T$且$C \in V_N$ }

- FIRSTVT(B)集计算方法:

  ① FIRSTVT(B)=ϕ

  ② 扫描文法规则，对形如B→a…或B→Ca…规则，$a \in V_T$，$C \in V_N$，则 FIRSTVT(B) ∪ = {a};

  ③ 扫描文法规则，对形如B→C…规则，$C \in V_N$，则 FIRSTVT(B) ∪ = FIRSTVT(C);

  ④ 重复③，直到FIRSTVT集不再扩大为止。

5.3 算符优先分析法

- LASTVT(B)集计算方法:
  - ① LASTVT(B)=ϕ
  - ② 扫描文法规则，对形如B→…a或B→…aC规则，$a\in V_T$，$C\in V_N$，则 LASTVT(B) ∪= {a};
  - ③ 扫描文法规则，对形如B→…C规则，$C\in V_N$，则 LASTVT(B) ∪= LASTVT(C);
  - ④ 重复③，直到LASTVT集不再扩大为止。
- 基于FIRSTVT(B)和LASTVT(B)的算符优先关系计算

  如果A→…ab…∈P或A→…aBb…∈P，a，$b\in V_T$，$B\in V_N$，则$a \doteq b$;

  如果A→…aB…∈P，$b\in$FIRSTVT(B)，$a\in V_T$，$B\in V_N$，则$a \lessdot b$;

  如果A→…Bb…∈P，$a\in$LASTVT(B)，$b\in V_T$，$B\in V_N$，则$a \gtrdot b$。

5.3 算符优先分析法

- 例5.6 文法G(E)：
  - (1) E→E+T | T
  - (2) T→T*F | F
  - (3) F→P ↑ F | P
  - (4) P→(E) | i

1. 计算文法G的FIRSTVT和LASTVT；
2. 构造优先关系表;
3. G是算符优先文法吗?
4. 分析串 i+i*i#

5.3 算符优先分析法

(1) E→E+T | T

(2) T→T*F | F

(3) F→P ↑ F | P

(4) P→(E) | i

1. 计算文法G的FIRSTVT和LASTVT；

FIRSTVT(E)={+,*,↑,(,i}

FIRSTVT(T)={*,↑,(,i}

FIRSTVT(F)={↑,(,i}

FIRSTVT(P)={(,i}

LASTVT(E)={+,*, ↑,),i}

LASTVT(T)={*, ↑,),i}

LASTVT(F)={↑,),i}

LASTVT(P)={),i}

2. 构造优先关系表;

3. G[E]是算符优先文法；

	+	*	↑	i	(	)	#
+	⋗	⋖	⋖	⋖	⋖	⋗	⋗
*	⋗	⋗	⋖	⋖	⋖	⋗	⋗
↑	⋗	⋗	⋖	⋖	⋖	⋗	⋗
i	⋗	⋗	⋗			⋗	⋗
(	⋖	⋖	⋖	⋖	⋖	⋍	
)	⋗	⋗	⋗			⋗	⋗
#	⋖	⋖	⋖	⋖	⋖		⋍

5.3 算符优先分析法

4. 分析串i+i*i#

	+	*	↑	i	(	)	#
+	>	<	<	<	<	>	>
*	>	>	<	<	<	>	>
↑	>	>	<	<	<	>	>
i	>	>	>			>	>
(	<	<	<	<	<	=	
)	>	>	>			>	>
#	<	<	<	<	<		=

(1) E→E+T | T

(2) T→T*F | F

(3) F→P ↑ F | P

(4) P→(E) | i

步	栈S	关系	当前	剩余符号	动作
1	#	<	i	+i*i#	shift
2	#i	>	+	i*i#	reduce
3	#N	<	+	i*i#	shift
4	#N+	<	i	*i#	shift
5	#N+i	>	*	i#	reduce
6	#N+N	<	*	i#	shift
7	#N+N*	<	i	#	shift
8	#N+N*i	>	#		reduce
9	#N+N*N	>	#		reduce
10	#N+N	>	#		reduce
11	#N	=	#		accept

- 算符优先分析结果不一定是语法树

5.3 算符优先分析法

- 定义5.8 设文法$G=(V_N, V_T, P, S)$，算符优先关系为$\doteq$、$\lessdot$ 和$\gtrdot$，定义在$V_T \rightarrow N^+$上的函数f和g满足下列条件，则文法G的f和g称为算符优先函数。

  ① $a \doteq b$ iff $f(a)=g(b)$

  ② $a \lessdot b$ iff $f(a)<g(b)$

  ③ $a \gtrdot b$ iff $f(a)>g(b)$

- 优先函数与算符优先矩阵： $2(|V_T|+1)$ vs $(|V_T|+1)^2$

5.3 算符优先分析法

- 算符优先函数的构造

①对a∈V_T(含#)，置f(a)=g(a)=1；

② 对每个关系偶对<a，b>，按下列情况分别计算：

如果a $\doteq$ b，f(a)≠g(b)，则 min{f(a),g(b)}=max{ f(a),g(b)}；

如果a $\lessdot$ b，f(a)≥g(b)，则 g(b)=f(a)＋1；

如果a $\gtrdot$ b，f(a)≤g(b)，则 f(a)=g(b)＋1；

③ 重复②步骤，直到f和g函数不再变化（或称过程收敛），或者f和g函数有值大于2｜V_T｜为止。后一种情况判定f和g函数不存在。

5.3 算符优先分析法

- 优先函数构造过程

	+	*	↑	i	(	)	#
+	>	<	<	<	<	>	>
*	>	>	<	<	<	>	>
↑	>	>	<	<	<	>	>
i	>	>	>			>	>
(	<	<	<	<	<	=	
)	>	>	>			>	>
#	<	<	<	<	<		=

	+	*	↑	i	(	)	#
f	1	1	1	1	1	1	1
g	1	1	1	1	1	1	1

	+	*	↑	i	(	)	#
f	2	4	4	6	1	6	1
g	2	3	5	5	5	1	1

	+	*	↑	i	(	)	#
f	3	5	5	7	1	7	1
g	2	4	6	6	6	1	1

	+	*	↑	i	(	)	#
f	3	5	5	7	1	7	1
g	2	4	6	6	6	1	1

5.3 算符优先分析法

- 算符优先分析法的优缺点：
  - 优点：简单，快速
  - 缺点：可能错误接受非法句子
- 算符优先分析法的主要应用场景：
  - 用于分析各类表达式

小结

- 优先分析法的共性
- 简单优先分析法
  - 优先关系计算-简单优先表(矩阵)
  - 简单优先文法
  - 简单优先文法的语法分析方法，规范归约，最右推导的逆过程
- 算符优先分析法
  - 算符文法，算符优先方法
  - FIRSTVT(),LASTVT()的计算；算符优先矩阵
  - 算符优先分析法，最左素短语，不是规范归约，分析结果未必是语法树

小结

- 重点掌握：
  - 计算优先关系
  - 优先文法的判定
  - 构造优先分析表
  - 优先分析算法