

语法制导翻译

Part4: L属性定义的翻译方案

李诚

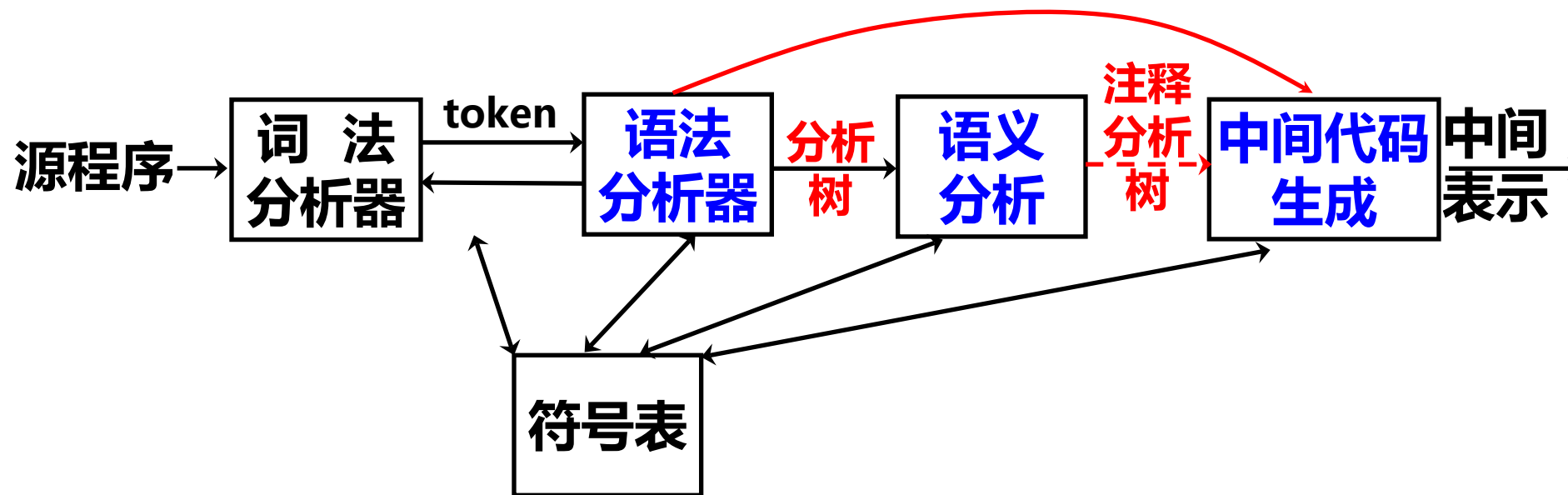
国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2024年10月16日



本节提纲



- L属性定义的SDT
- 实现方式1：与递归下降分析结合
- 实现方式2：与LR分析结合



- 如果每个产生式 $A \rightarrow X_1 \dots X_{j-1} X_j \dots X_n$ 的每条语义规则计算的属性是 A 的综合属性；或者是 X_j 的继承属性，但它仅依赖：
 - 该产生式中 X_j 左边符号 X_1, X_2, \dots, X_{j-1} 的属性；
 - A 的继承属性
 - X_j 的其他属性，且不能成环
- S属性定义属于L属性定义



- 变量类型声明的语法制导定义是一个 *L-SDD*

产生式	语义规则
$D \rightarrow TL$	$L.in = T.type$
$T \rightarrow \text{int}$	$T.type = \text{integer}$
$T \rightarrow \text{real}$	$T.type = \text{real}$
$L \rightarrow L_1, \text{id}$	$L_1.in = L.in;$ $\text{addType}(\text{id.entry}, L.in)$
$L \rightarrow \text{id}$	$\text{addType}(\text{id.entry}, L.in)$

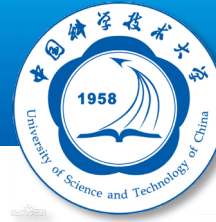
- 后缀SDT在这里并不适用



- 消除左递归的算术表达式语法制导定义是L-SDD

产生式	语义规则
$T \rightarrow FT'$	$T'.inh = F.val$ $T.val = T'.syn$
$T' \rightarrow *FT_1'$	$T_1'.inh = T'.inh \times F.val$ $T'.syn = T_1'.syn$
$T' \rightarrow \varepsilon$	$T'.syn = T'.inh$
$F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$

- 后缀SDT在这里并不适用



• 将L-SDD转换为SDT的规则

- 将计算一个产生式左部符号的综合属性的动作放置在这个产生式右部的最右端
- 将计算某个非终结符号A的继承属性的动作插入到产生式右部中紧靠在A的本次出现之前的位置上
 - 多个继承属性，要考虑次序，防止形成环



将L-SDD转换为SDT-举例



L-SDD

产生式	语义规则
$D \rightarrow TL$	$L.in = T.type$
$T \rightarrow \text{int}$	$T.type = \text{integer}$
$T \rightarrow \text{real}$	$T.type = \text{real}$
$L \rightarrow L_1, \text{id}$	$L_1.in = L.in;$ $addType(\text{id.entry}, L.in)$
$L \rightarrow \text{id}$	$addType(\text{id.entry}, L.in)$



将L-SDD转换为SDT-举例



L-SDD

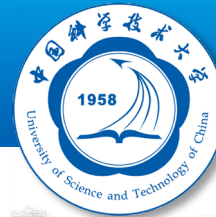
产生式	语义规则
$D \rightarrow TL$	$L.in = T.type$
$T \rightarrow \text{int}$	$T.type = \text{integer}$
$T \rightarrow \text{real}$	$T.type = \text{real}$
$L \rightarrow L_1, \text{id}$	$L_1.in = L.in;$ $\text{addType}(\text{id.entry}, L.in)$
$L \rightarrow \text{id}$	$\text{addType}(\text{id.entry}, L.in)$

L-SDT

$D \rightarrow T \{L.in = T.type\} L$
 $T \rightarrow \text{int} \{T.type = \text{integer}\}$
 $T \rightarrow \text{real} \{T.type = \text{real}\}$
 $L \rightarrow \{L_1.in = L.in\} L_1, \text{id} \{\text{addtype}(\text{id.entry}, L.in)\}$
 $L \rightarrow \text{id} \{\text{addtype}(\text{id.entry}, L.in)\}$



将L-SDD转换为SDT-举例



L-SDD

产生式	语义规则
$T \rightarrow FT'$	$T'.inh = F.val$ $T.val = T'.syn$
$T' \rightarrow *FT_1'$	$T_1'.inh = T'.inh \times F.val$ $T'.syn = T_1'.syn$
$T' \rightarrow \varepsilon$	$T'.syn = T'.inh$
$F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$



将L-SDD转换为SDT-举例



L-SDD

产生式	语义规则
$T \rightarrow FT'$	$T'.inh = F.val$ $T.val = T'.syn$
$T' \rightarrow *FT_1'$	$T_1'.inh = T'.inh \times F.val$ $T'.syn = T_1'.syn$
$T' \rightarrow \varepsilon$	$T'.syn = T'.inh$
$F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$

L-SDT

$T \rightarrow F \{T'.inh = F.val\} T' \{T.val = T'.syn\}$
 $T' \rightarrow *F \{T_1'.inh = T'.inh \times F.val\} T_1' \{T'.syn = T_1'.syn\}$
 $T' \rightarrow \varepsilon \{T'.syn = T'.inh\}$
 $F \rightarrow \text{digit} \{F.val = \text{digit.lexval}\}$



L属性定义的SDT-举例



- **例 把有加和减的中缀表达式翻译成后缀表达式输出**
如果输入是 $8+5-2$ ，则输出是 $8\ 5\ +\ 2\ -$ ，设计SDT

$$E \rightarrow T R$$

$$R \rightarrow \text{addop } T R_1 \mid \varepsilon$$

$$T \rightarrow \text{num}$$



L属性定义的SDT-举例



- 例 把有加和减的中缀表达式翻译成后缀表达式输出
如果输入是 $8+5-2$ ，则输出是 $8\ 5\ +\ 2\ -$ ，设计SDT

$E \rightarrow T R$

$R \rightarrow \text{addop } T \{ \textit{print}(\text{addop.lexeme}) \} R_1 \mid \varepsilon$

$T \rightarrow \text{num } \{ \textit{print}(\text{num.val}) \}$

$E \Rightarrow T R \Rightarrow \text{num } \{ \textit{print}(8) \} R$

$\Rightarrow \text{num} \{ \textit{print}(8) \} \text{addop } T \{ \textit{print}(+) \} R$

$\Rightarrow \text{num} \{ \textit{print}(8) \} \text{addop } \text{num} \{ \textit{print}(5) \} \{ \textit{print}(+) \} R$

$\dots \{ \textit{print}(8) \} \{ \textit{print}(5) \} \{ \textit{print}(+) \} \text{addop } T \{ \textit{print}(-) \} R$

$\dots \{ \textit{print}(8) \} \{ \textit{print}(5) \} \{ \textit{print}(+) \} \{ \textit{print}(2) \} \{ \textit{print}(-) \}$



L属性定义的SDT-举例



- 例 把有加和减的中缀表达式翻译成后缀表达式输出
如果输入是 $8+5-2$ ，则输出是 $8\ 5\ +\ 2\ -$ ，设计SDT

$E \rightarrow T R$

$R \rightarrow \text{addop } T R_1 \{ \textit{print}(\text{addop.lexeme}) \} \mid \varepsilon$

$T \rightarrow \text{num} \{ \textit{print}(\text{num.val}) \}$

语义动作不能随意放置



L属性定义的SDT-举例



- 例 把有加和减的中缀表达式翻译成后缀表达式输出
如果输入是 $8+5-2$ ，则输出是 $8\ 5\ +\ 2\ -$ ，设计SDT

$E \rightarrow T R$

$R \rightarrow \text{addop } T R_1 \{ \text{print}(\text{addop.lexeme}) \} \mid \varepsilon$

$T \rightarrow \text{num} \{ \text{print}(\text{num.val}) \}$

语义动作不能随意放置

$E \Rightarrow T R \Rightarrow \text{num} \{ \text{print}(8) \} R$

$\Rightarrow \text{num} \{ \text{print}(8) \} \text{addop } TR$

$\Rightarrow \text{num} \{ \text{print}(8) \} \text{addop num} \{ \text{print}(5) \} R$

$\dots \{ \text{print}(8) \} \{ \text{print}(5) \} \text{addop } TR$

$\dots \{ \text{print}(8) \} \{ \text{print}(5) \} \{ \text{print}(2) \} \{ \text{print}(-) \} \{ \text{print}(+) \}$



- 例 数学排版语言EQN, 设计SDT, 计算高度

$E \text{ sub } 1 \text{ .val}$

$S \rightarrow B$

$B \rightarrow B_1 B_2$

$B \rightarrow B_1 \text{ sub } B_2$

$B \rightarrow \text{text}$





文字排版中的符号属性



***S* : *S.ht*, 综合属性; 待排公式的整体高度**

***B* : *B.ps*, 继承属性; 公式 (文本) 中字体的大小
B.ht, 综合属性; 公式排版高度**

***text* : *text.h*, 文本高度**



文字排版中的符号属性



S : $S.ht$, 综合属性; 待排公式的整体高度

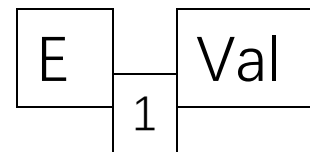
B : $B.ps$, 继承属性; 公式 (文本) 中字体的大小
 $B.ht$, 综合属性; 公式排版高度

$text$: $text.h$, 文本高度

$\max(B_1, B_2)$: 求两个排版公式的最大高度

$shrink(B)$: 将字体大小缩小为 B 的30%

$disp(B_1.ht, B_2.ht)$: 向下调整 B_2 的位置





L属性定义的SDT-举例



- 例 数学排版语言EQN (语法制导定义L-SDD)

E sub 1 .val

E_1 .val

ps-point size (继承属性); ht-height(综合属性)

产生式	语义规则
$S \rightarrow B$	$B.ps = 10; S.ht = B.ht$
$B \rightarrow B_1 B_2$	$B_1.ps = B.ps; B_2.ps = B.ps;$ $B.ht = \max(B_1.ht, B_2.ht)$
$B \rightarrow B_1 \text{ sub } B_2$	$B_1.ps = B.ps; B_2.ps = \text{shrink}(B.ps);$ $B.ht = \text{disp}(B_1.ht, B_2.ht)$
$B \rightarrow \text{text}$	$B.ht = \text{text}.h \times B.ps$



L属性定义的SDT-举例



- 例 数学排版语言EQN (翻译方案SDT)

$S \rightarrow \{B.ps = 10\}$

$B \quad \{S.ht = B.ht\}$

**B 继承属性的计算
位于 B 的左边**



L属性定义的SDT-举例



- 例 数学排版语言EQN (翻译方案SDT)

$S \rightarrow \{B.ps = 10\}$

$B \quad \{S.ht = B.ht\}$

**B 综合属性的计算
放在右部末端**



- 例 数学排版语言EQN (翻译方案SDT)

$$S \rightarrow \{B.ps = 10\}$$

$$B \quad \{S.ht = B.ht\}$$

$$B \rightarrow \quad \{B_1.ps = B.ps\}$$

$$B_1 \quad \{B_2.ps = B.ps\}$$

$$B_2 \quad \{B.ht = \max(B_1.ht, B_2.ht)\}$$



• 例 数学排版语言EQN (翻译方案SDT)

$$S \rightarrow \{ B.ps = 10 \}$$

$$B \quad \{ S.ht = B.ht \}$$

$$B \rightarrow \quad \{ B_1.ps = B.ps \}$$

$$B_1 \quad \{ B_2.ps = B.ps \}$$

$$B_2 \quad \{ B.ht = \max(B_1.ht, B_2.ht) \}$$

$$B \rightarrow \quad \{ B_1.ps = B.ps \}$$

$$B_1$$

$$\text{sub} \quad \{ B_2.ps = \text{shrink}(B.ps) \}$$

$$B_2 \quad \{ B.ht = \text{disp}(B_1.ht, B_2.ht) \}$$



• 例 数学排版语言EQN (翻译方案SDT)

$$S \rightarrow \{B.ps = 10\}$$

$$B \quad \{S.ht = B.ht\}$$

$$B \rightarrow \quad \{B_1.ps = B.ps\}$$

$$B_1 \quad \{B_2.ps = B.ps\}$$

$$B_2 \quad \{B.ht = \max(B_1.ht, B_2.ht)\}$$

$$B \rightarrow \quad \{B_1.ps = B.ps\}$$

$$B_1$$

$$\text{sub} \quad \{B_2.ps = \text{shrink}(B.ps)\}$$

$$B_2 \quad \{B.ht = \text{disp}(B_1.ht, B_2.ht)\}$$

$$B \rightarrow \text{text} \quad \{B.ht = \text{text.h} \times B.ps\}$$



- 例 翻译while循环语句，生成代码
产生式 $S \rightarrow \text{while } (C) S_1$

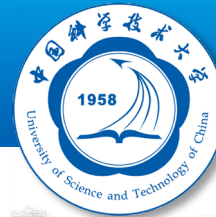


$S \rightarrow \text{while } (C) S_1$	<pre>L1 = new(); L2 = new(); S₁.next = L1; C.false = S.next; C.true = L2; S.code = label L1 C.code label L2 S₁.code</pre>
---------------------------------------	--

- *next*: 继承属性，语句结束后应跳转到的标号
- *true*、*false*: C为真/假时应该跳转到的标号
- *code*: 综合属性，表示代码



L属性定义的SDT-举例



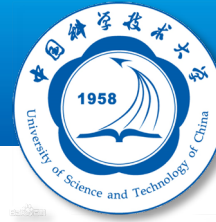
• 根据语义动作的放置规则得到如下SDT:



$S \rightarrow \text{while } (C) S_1$	$L1 = \text{new}();$	} 临时变量
	$L2 = \text{new}();$	
S_1 的继承属性 \rightarrow	$S_1.\text{next} = L1;$	
	$C.\text{false} = S.\text{next};$	} C的继承属性
	$C.\text{true} = L2;$	
S 的综合属性 \rightarrow	$S.\text{code} = \mathbf{label} \parallel L1 \parallel C.C\text{code} \parallel \mathbf{label} \parallel L2 \parallel S_1.\text{code}$	



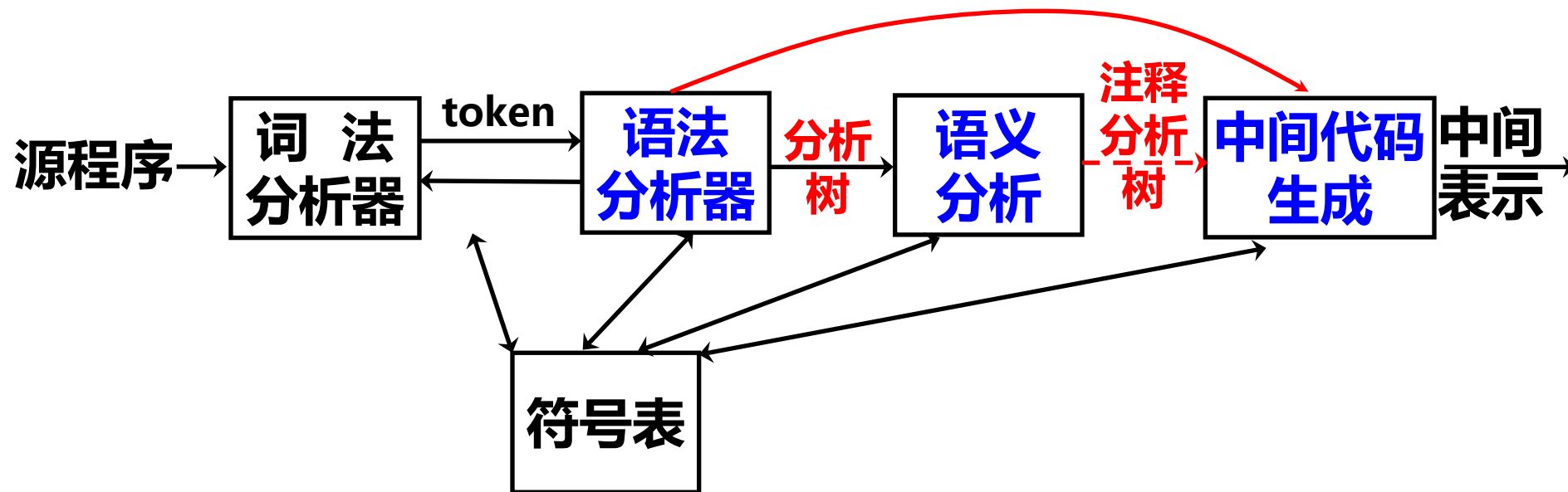
$S \rightarrow \text{while } ($	$\{L1 = \text{new}(); L2 = \text{new}(); C.\text{false} = S.\text{next}; C.\text{true} = L2;\}$
$C)$	$\{S_1.\text{next} = L1;\}$
S_1	$\{S.\text{code} = \mathbf{label} \parallel L1 \parallel C.C\text{code} \parallel \mathbf{label} \parallel L2 \parallel S_1.\text{code}\}$



- **结合SDT，考虑在语法分析过程中进行翻译**
 - 自顶向下计算
 - 递归下降分析器
 - 自底向上计算，考虑与LR分析器的结合
 - 删除翻译方案中嵌入的动作
 - 继承属性在分析栈中的计算



本节提纲



- L属性定义的SDT
- 实现方式1：与递归下降分析结合
- 实现方式2：与LR分析结合



• 递归下降翻译器的设计

- 为每个非终结符A构造一个函数
 - A的每个继承属性对应该函数的一个形参
 - 函数的返回值是A的综合属性值



• 递归下降翻译器的设计

- 为每个非终结符A构造一个函数
 - A的每个继承属性对应该函数的一个形参
 - 函数的返回值是A的综合属性值
- 在函数体中
 - 首先选择适当的A的产生式
 - 用局部变量保存产生式中文法符号的属性
 - 对产生式体中的终结符号，读入符号并获取其综合属性（由词法分析得到）
 - 对产生式体中的非终结符，调用相应函数，记录返回值



L属性定义计算-递归下降



$$\begin{array}{l} E \rightarrow T \quad \{R.i = T.nptr\} \quad T + T + T + \dots \\ \quad R \quad \{E.nptr = R.s\} \\ R \rightarrow + \\ \quad T \quad \{R_1.i = mkNode ('+', R.i, T.nptr)\} \\ \quad R_1 \quad \{R.s = R_1.s\} \\ R \rightarrow \varepsilon \quad \{R.s = R.i\} \\ T \rightarrow F \quad \{W.i = F.nptr\} \\ \quad W \quad \{T.nptr = W.s\} \\ W \rightarrow * \\ \quad F \quad \{W_1.i = mkNode ('*', W.i, F.nptr)\} \\ \quad W_1 \quad \{W.s = W_1.s\} \\ W \rightarrow \varepsilon \quad \{W.s = W.i\} \end{array}$$

F 产生式部分不再给出

文法是LL(1), 所以适合
自顶向下分析



产生式 $R \rightarrow +TR \mid \varepsilon$ 的递归下降分析过程

```
void R() {  
    if (lookahead == '+' ) {  
        match ( '+' );  
        T();  
        R();  
    }  
    else /* 什么也不做 */  
}
```



L属性定义计算-递归下降



```
syntaxTreeNode* R (syntaxTreeNode* i) {  
    syntaxTreeNode *nptr, *i1, *s1, *s;  
    char addoplexeme;  
  
    if (lookahead == '+') { /* 产生式  $R \rightarrow +TR$  */  
        addoplexeme = lexval;  
        match('+');  
        nptr = T();  
        i1 = mkNode(addoplexeme, i, nptr);  
        s1 = R(i1);  
        s = s1;  
    }  
    else s = i; /* 产生式  $R \rightarrow \varepsilon$  */  
    return s;  
}
```

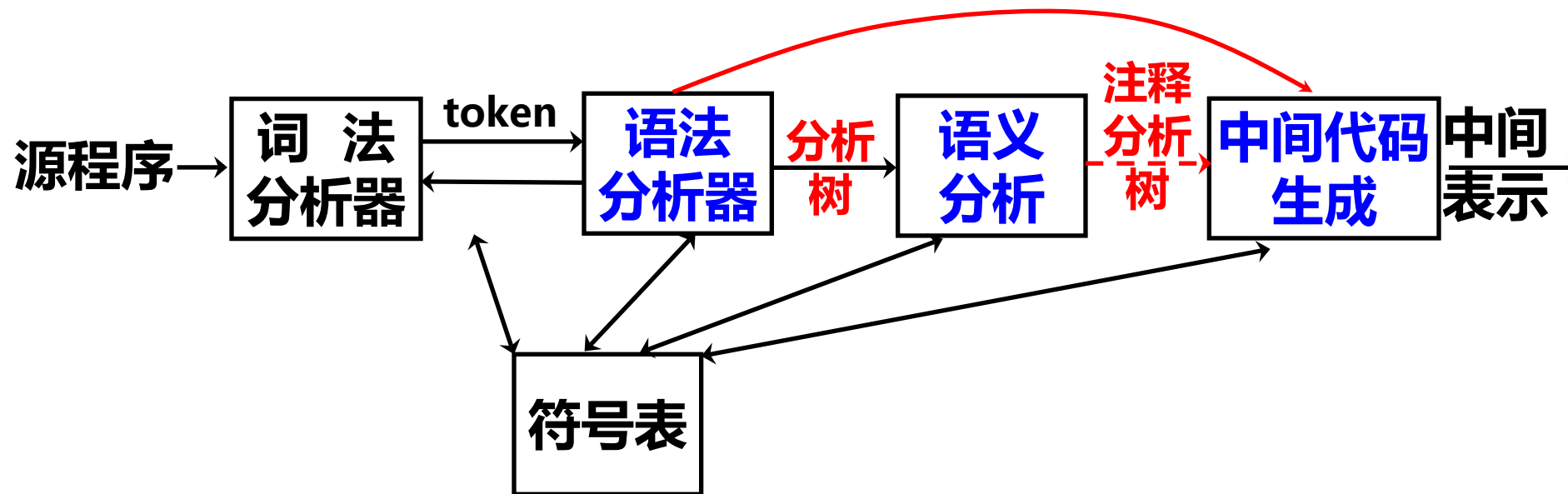
$R : i, s$

$T : nptr$

$+ : addoplexeme$



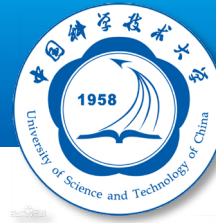
本节提纲



- L属性定义的SDT
- 实现方式1：与递归下降分析结合
- 实现方式2：与LR分析结合



- 可否直接将带有继承属性的L-SDD直接与LR结合起来?
- 考虑 $A \rightarrow BC$ 产生式, 假设 $B.i = A.i$
 - 当对B进行归约时, 由于没有看到C, 不能确定会使用该产生式, $A.i$ 无法得知, 此时可以推迟计算
 - 等按照BC归约成为A之后, 我们仍然不能确定包含了A的产生式, $A.i$ 仍无法得知, 继续推迟计算
 - 最终退化为“先造分析树, 后翻译”的策略
- 解决方案:
 - 可将产生式中嵌入的动作删除, 挪到产生式最右端



- 将L-SDD转换为SDT
- 对于产生式 $A \rightarrow \alpha \{a\} \beta$, a 是语义动作
 - 引入新的非终结符M, 代替 $\{a\}$, 形成 $A \rightarrow \alpha M \beta$
 - 引入新的产生式 $M \rightarrow \epsilon$
 - 修改 a 得到 a' :
 - 将 a 需要的A或者 α 中的属性作为M的继承属性进行复制
 - 按照 a 中的方法计算各属性, 将这些属性作为M的综合属性保存起来
 - 将 $\{a'\}$ 与 $M \rightarrow \epsilon$ 关联起来



- 假设LL文法对应的SDT有如下片段

$$A \rightarrow \{B.i = f(A.i)\}BC$$

- 修改后的SDT为

$$A \rightarrow M B C$$

$$M \rightarrow \varepsilon \{M.i = A.i; M.s = f(M.i)\}$$

分析栈的设计可以保证M的语义规则中可以使用A.i，因为A.i在M下方紧邻的位置。



- 例：删除翻译方案中嵌入的动作

$$E \rightarrow T R$$
$$R \rightarrow + T \{print ('+')\} R_1 \mid - T \{print ('-')\} R_1 \mid \varepsilon$$
$$T \rightarrow num \{print (num.val)\}$$

这些动作的一个重要特点：
没有引用原来产生式文法符号
的属性，即只涉及虚拟属性



• 例：删除翻译方案中嵌入的动作

$$E \rightarrow T R$$

$$R \rightarrow + T \{print\ ('+')\} R_1 \mid - T \{print\ ('-')\} R_1 \mid \varepsilon$$

$$T \rightarrow num \{print\ (num.val)\}$$

加入产生 ε 的标记非终结符，让每个嵌入动作由不同标记非终结符 M 代表，并把该动作放在产生式 $M \rightarrow \varepsilon$ 的右端

$$E \rightarrow T R$$

$$R \rightarrow + T M R_1 \mid - T N R_1 \mid \varepsilon$$

$$T \rightarrow num \{print\ (num.val)\}$$

$$M \rightarrow \varepsilon \{print\ ('+')\}$$

$$N \rightarrow \varepsilon \{print\ ('-')\}$$

这些动作的一个重要特点：
没有引用原来产生式文法符号的属性，即只涉及虚拟属性



• 例 数学排版语言EQN

$$S \rightarrow \{ B.ps = 10 \}$$

$$B \quad \{ S.ht = B.ht \}$$

$$B \rightarrow \quad \{ B_1.ps = B.ps \}$$

$$B_1 \quad \{ B_2.ps = B.ps \}$$

$$B_2 \quad \{ B.ht = \max(B_1.ht, B_2.ht) \}$$

$$B \rightarrow \quad \{ B_1.ps = B.ps \}$$

$$B_1$$

$$\text{sub} \quad \{ B_2.ps = \text{shrink}(B.ps) \}$$

$$B_2 \quad \{ B.ht = \text{disp}(B_1.ht, B_2.ht) \}$$

$$B \rightarrow \text{text} \quad \{ B.ht = \text{text.h} \times B.ps \}$$



文字排版引入标记符号



为了自底向上的计算:

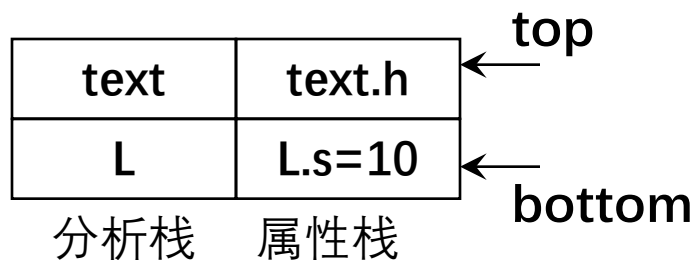
$B \rightarrow \text{text} \quad \{ B.ht := \text{text.h} \times B.ps \}$

必须确定继承属性B.ps的（“属性栈”）位置。为引入标记非终结符L、M和N及其属性，包括相应的空产生式和有关属性规则。这样B.ps即可在紧靠“句柄”text下方的位置上找到。（L的综合属性置为B.ps的初值）

$S \rightarrow \mathbf{L} B$

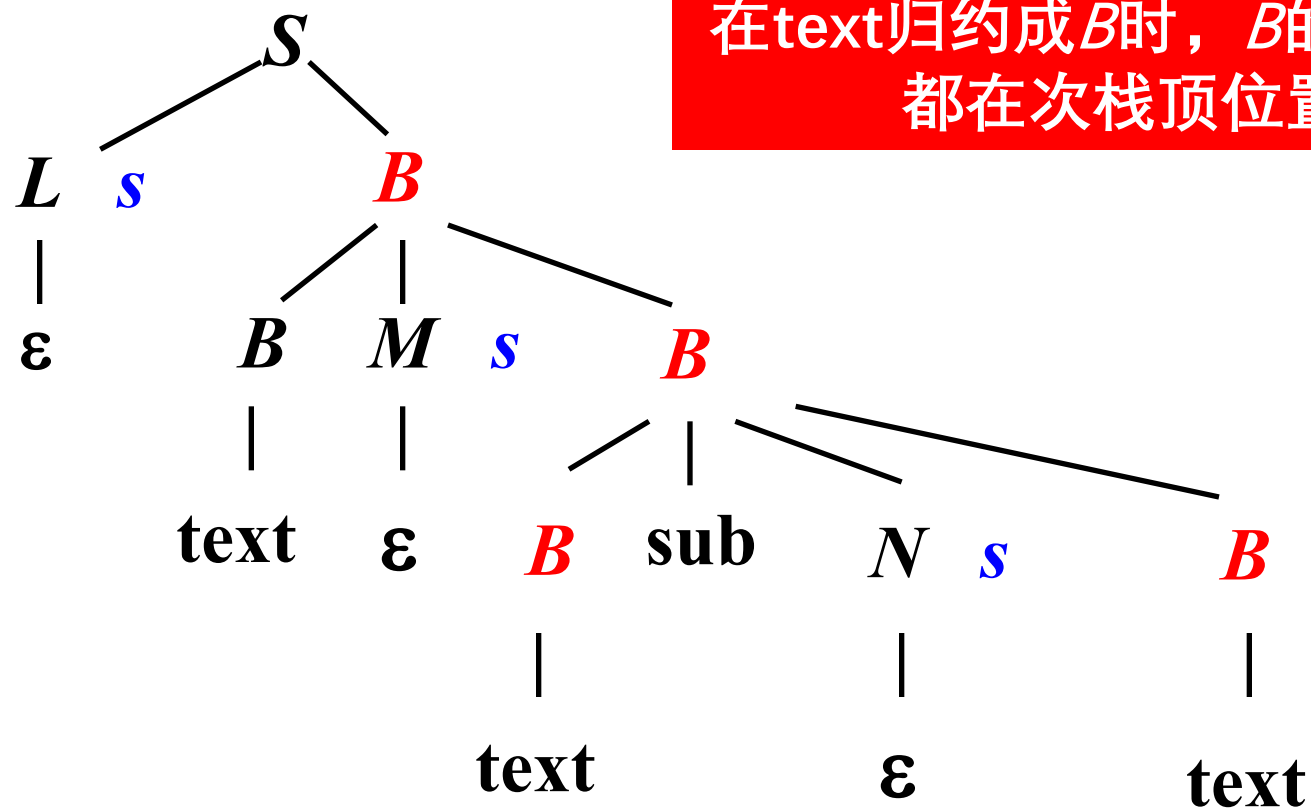
$B \rightarrow B_1 \mathbf{M} B_2$

$B \rightarrow B_1 \text{ sub } \mathbf{N} B_2$





L属性定义的自底向上计算





L属性定义的自底向上计算



产生式	语义规则
$S \rightarrow LB$	$B.ps = L.s; S.ht = B.ht$
$L \rightarrow \varepsilon$	$L.s = 10$ 将 $B.ps$ 存入栈中, 便于引用
$B \rightarrow B_1 MB_2$	$B_1.ps = B.ps; M.i = B.ps;$ $B_2.ps = M.s; B.ht = \max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text}.h \times B.ps$



L属性定义的自底向上计算



产生式	语义规则
$S \rightarrow LB$	$B.ps = L.s; S.ht = B.ht$
$L \rightarrow \varepsilon$	$L.s = 10$ 将 $B.ps$ 存入栈中, 便于引用
$B \rightarrow B_1 MB_2$	$B_1.ps = B.ps; M.i = B.ps;$ $B_2.ps = M.s; B.ht = \max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$ 单纯为了属性位置可预测
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text.h} \times B.ps$



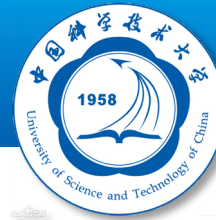
L属性定义的自底向上计算



产生式	语义规则
$S \rightarrow LB$	$B.ps = L.s; S.ht = B.ht$
$L \rightarrow \varepsilon$	$L.s = 10$ 将 $B.ps$ 存入栈中, 便于引用
$B \rightarrow B_1 MB_2$	$B_1.ps = B.ps; M.i = B.ps;$ $B_2.ps = M.s; B.ht = \max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$ 单纯为了属性位置可预测
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$ 兼有计算功能
$B \rightarrow \text{text}$	$B.ht = \text{text.h} \times B.ps$



L属性定义的自底向上计算



产生式	语义规则
$S \rightarrow LB$	$B.ps = L.s; S.ht = B.ht$
$L \rightarrow \varepsilon$	$L.s = 10$
$B \rightarrow B_1 MB_2$	$B_1.ps = B.ps; M.i = B.ps;$ $B_2.ps = M.s; B.ht = \max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text.h} \times B.ps$

继承属性的值等于栈中某个综合属性的值，因此栈中只保存综合属性的值



L属性定义的自底向上计算



产生式	代码段
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$L.s = 10$
$B \rightarrow B_1 MB_2$	$B_1.ps = B.ps; M.i = B.ps;$ $B_2.ps = M.s; B.ht = \max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text.h} \times B.ps$

$B.ps = L.s; S.ht = B.ht$



L属性定义的自底向上计算



产生式	代码段
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1 MB_2$	$B_1.ps = B.ps; M.i = B.ps;$ $B_2.ps = M.s; B.ht = \max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text}.h \times B.ps$

L.s = 10



L属性定义的自底向上计算



产生式	代码
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1 MB_2$	$B_1.ps = B.ps; M.i = B.ps;$ $B_2.ps = M.s; B.ht = \max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text.h} \times B.ps$

归约时, 弹出0个元素,
随后将L压栈, 因此
top为top+1

L.s = 10



L属性定义的自底向上计算



产生式	代码段
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1 MB_2$	$val[top-2] = \max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text.h} \times B.ps$

$B_1.ps = B.ps; M.i = B.ps; B_2.ps = M.s; B.ht = \max(B_1.ht, B_2.ht)$



L属性定义的自底向上计算



产生式	代码段
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1 MB_2$	$val[top-2] = \max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$val[top+1] = val[top-1]$
$B \rightarrow B_1 \text{ sub } NB_2$	$B_1.ps = B.ps; N.i = B.ps;$ $B_2.ps = N.s; B.ht = \text{disp}(B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text}.h \times B.ps$

$M.i = B.ps; M.s = M.i$



L属性定义的自底向上计算



产生式	代码段
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1 MB_2$	$val[top-2] = \max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$val[top+1] = val[top-1]$
$B \rightarrow B_1 \text{ sub } NB_2$	$val[top-3] = \text{disp}(val[top-3], val[top])$
$N \rightarrow \varepsilon$	$N.s = \text{shrink}(N.i)$
$B \rightarrow \text{text}$	$B.ht = \text{text}.h \times B.ps$

$B1.ps=B.ps; N.i = B.ps; B2.ps= N.s; B.ht= \text{disp}(B1.ht,B2.ht)$



L属性定义的自底向上计算

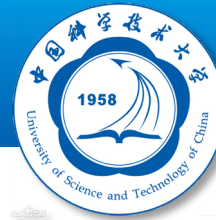


产生式	代码段
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1 MB_2$	$val[top-2] = \max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$val[top+1] = val[top-1]$
$B \rightarrow B_1 \text{ sub } NB_2$	$val[top-3] = \text{disp}(val[top-3], val[top])$
$N \rightarrow \varepsilon$	$val[top+1] = \text{shrink}(val[top-2])$
$B \rightarrow \text{text}$	$B.ht = \text{text}.h \times B.ps$

$N.i = B.ps; N.s = \text{shrink}(N.i)$



L属性定义的自底向上计算



产生式	代码段
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1 MB_2$	$val[top-2] = \max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$val[top+1] = val[top-1]$
$B \rightarrow B_1 \text{ sub } NB_2$	$val[top-3] = \text{disp}(val[top-3], val[top])$
$N \rightarrow \varepsilon$	$val[top+1] = \text{shrink}(val[top-2])$
$B \rightarrow \text{text}$	$val[top] = val[top] \times val[top-1]$

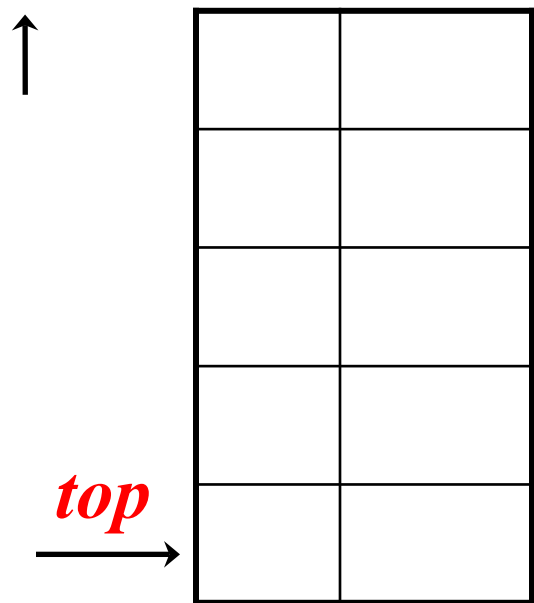
B.ht= text.h × B.ps



举例说明

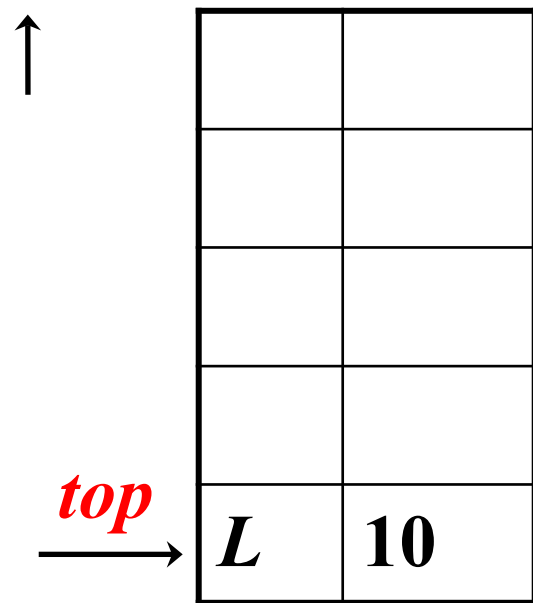


- 考虑字体大小10，基准高度2，输入串a sub 1



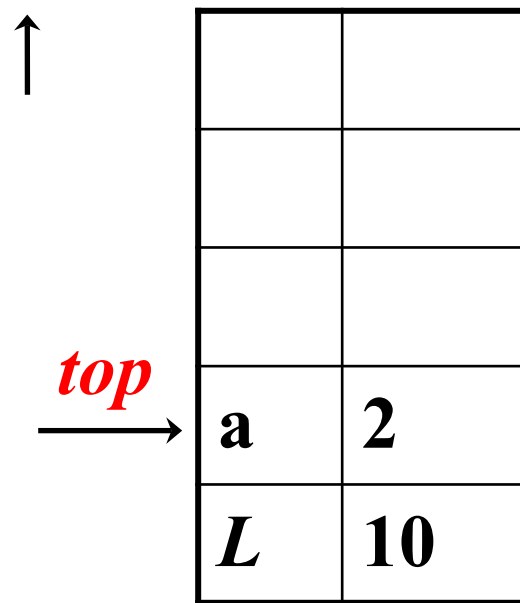
栈 *state val*

初始状态



栈 *state val*

空归约



栈 *state val*

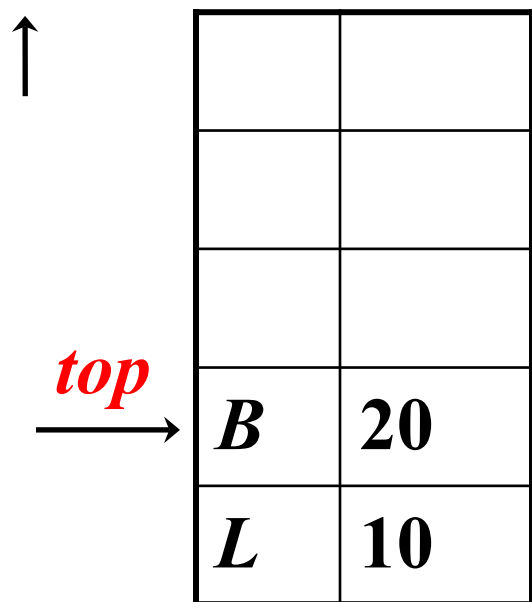
移进



举例说明

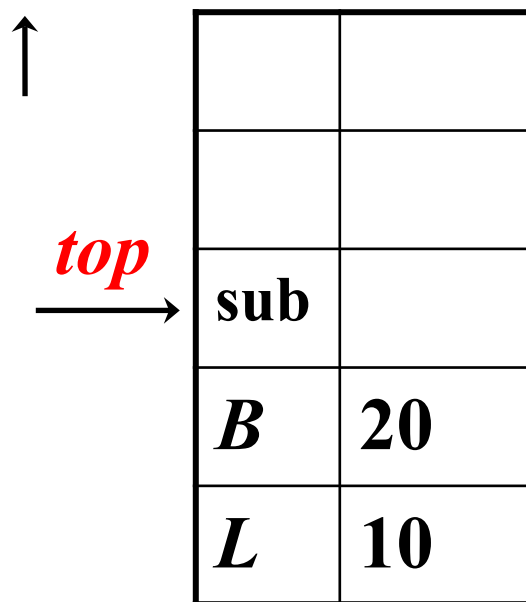


- 考虑字体大小10，基准高度2，输入串a sub 1



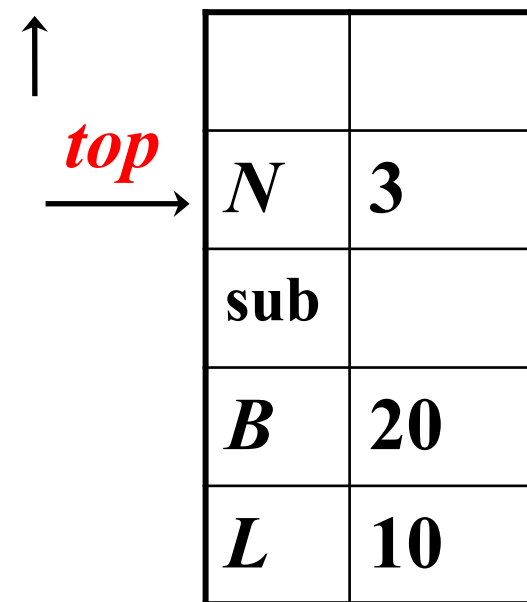
栈 *state val*

text归约



栈 *state val*

移进关键字



栈 *state val*

空归约，缩放字体



- 考虑字体大小10，基准高度2，输入串a sub 1

top ↑ →

1	2
N	3
sub	
B	20
L	10

栈 *state val*

移进text

top ↑ →

B	6
N	3
sub	
B	20
L	10

栈 *state val*

text归约，使用缩放后的字体大小

↑

top →

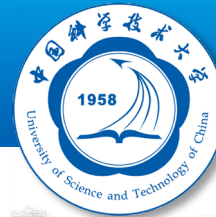
B	23
L	10

栈 *state val*

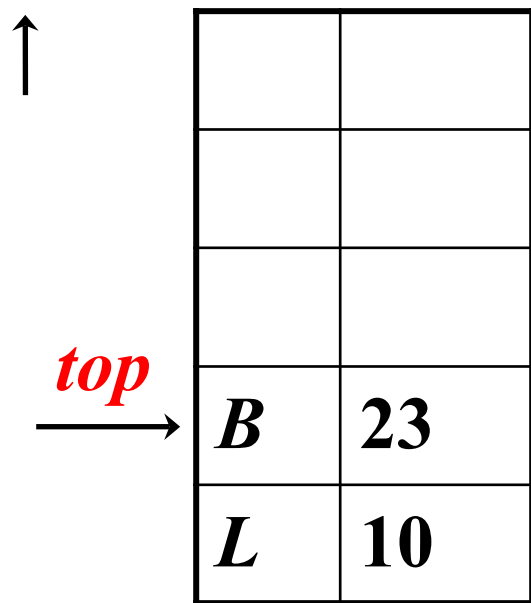
归约，重排高度
假设disp的结果为23



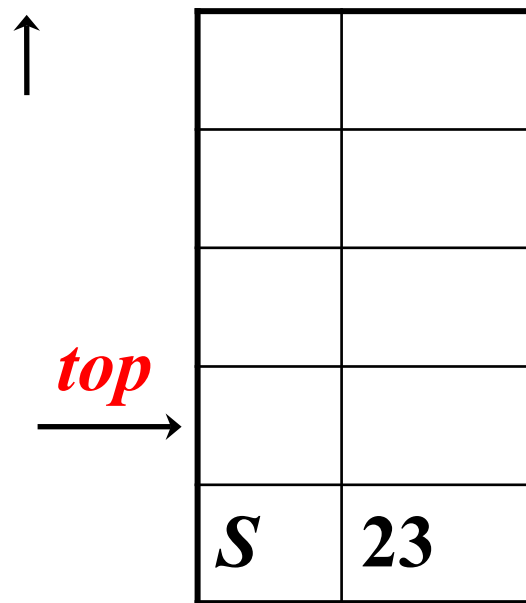
举例说明



- 考虑字体大小10，基准高度2，输入串a sub 1



栈 *state val*



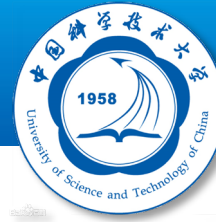
栈 *state val*



本章要点



- **语义规则的两种描述方法：语法制导的定义和翻译方案**
- **设计简单问题的语法制导定义和翻译方案，这是本章的重点和难点**
- **语法制导定义和翻译方案的实现**
 - 构造分析树
 - 与分析器结合



一起努力 打造国产基础软硬件体系!

李诚

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2024年10月16日