中国科学技术大学
University of Science and Technology of China

语法制导的翻译 II

《编译原理和技术(H)》、《编译原理(H)》

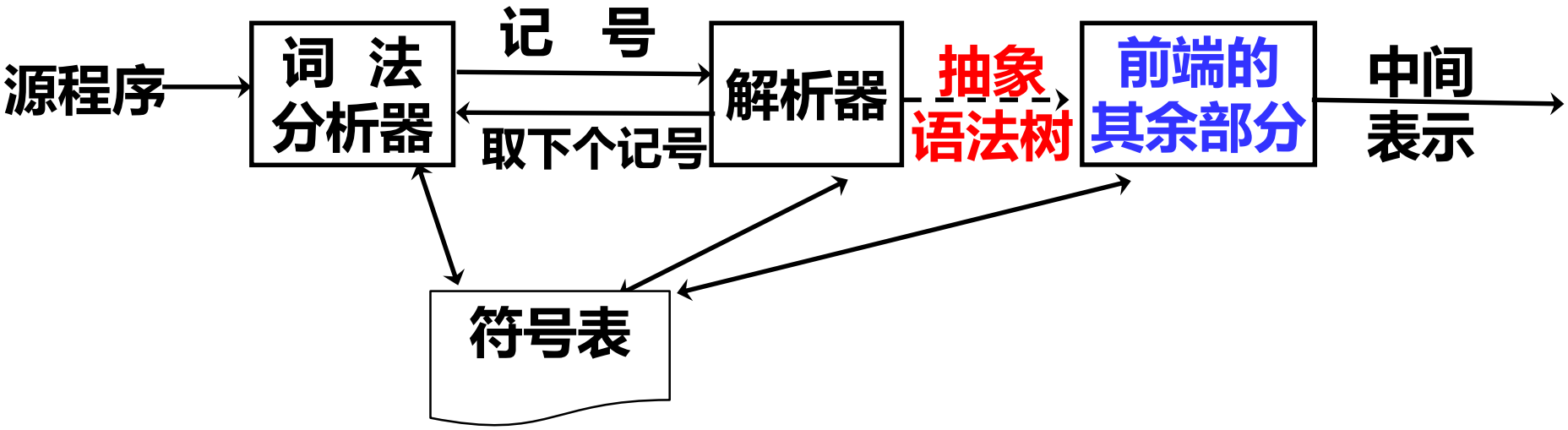
张昱

0551-63603804，yuzhang@ustc.edu.cn

中国科学技术大学

计算机科学与技术学院

本章内容



- 语义的描述：语法制导的定义、翻译方案
  - 语法制导：syntax-directed
    按语法结构来指导语义的定义和计算
  - 抽象语法树、注释分析树等
- 语法制导翻译的实现方法：自上而下、自下而上
  - 边语法分析边翻译

中国科学技术大学
University of Science and Technology of China

4.3 L属性定义的自上而下计算

- 翻译方案
- 预测翻译器的设计
- 用综合属性代替继承属性

翻译方案—内嵌无信息传播的语义动作

例 把有加和减的中缀表达式翻译成后缀表达式

如果输入是8+5 –2，则输出是8 5 + 2 –

$E \rightarrow T\ R$

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

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

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

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

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

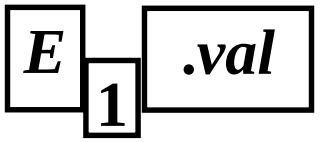
… {*print*(8)}{*print*(5)}{*print*(+)}addop T {*print*(–)}R

… {*print*(8)}{*print*(5)}{*print*(+)}{*print*(2)}{*print*(–)}

内嵌动作不涉及属性信息传播

L属性定义及其翻译方案

EQN程序示例　*E* sub 1 .val



ps-point size (继承属性); *ht*-height(综合属性)

产 生 式	语 义 规 则
$S \rightarrow B$	$B.ps = 10;\ S.ht = B.ht$
$B \rightarrow B_1\ B_2$	$B_1.ps = B.ps;\ B_2.ps = B.ps;$ $B.ht = max(B_1.ht, B_2.ht)$
$B \rightarrow B_1$ sub B_2	$B_1.ps = B.ps;\ B_2.ps = shrink(B.ps);$ $B.ht = disp\ (B_1.ht, B_2.ht)$
$B \rightarrow$ text	$B.ht =$ text$.h \times B.ps$

L属性定义及其翻译方案

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

产 生 式	语 义 规 则
$S \rightarrow B$	$B.ps = 10$; $S.ht = B.ht$
$B \rightarrow B_1\ B_2$	$B_1.ps = B.ps$; $B_2.ps = B.ps$; $B.ht = max(B_1.ht, B_2.ht)$
$B \rightarrow B_1$ sub B_2	$B_1.ps = B.ps$; $B_2.ps = shrink(B.ps)$; $B.ht = disp(B_1.ht, B_2.ht)$
$B \rightarrow$ text	$B.ht = text.h \times B.ps$

属性信息从左向右流动

- 产生式右部符号的继承属性必须在解析该符号之前计算
- 语义动作不能引用其右边符号的综合属性
- 产生式左部非终结符的综合属性只能在其依赖的所有属性都计算完后才能计算

左递归的消除引起继承属性

表达式语言的 LL文法

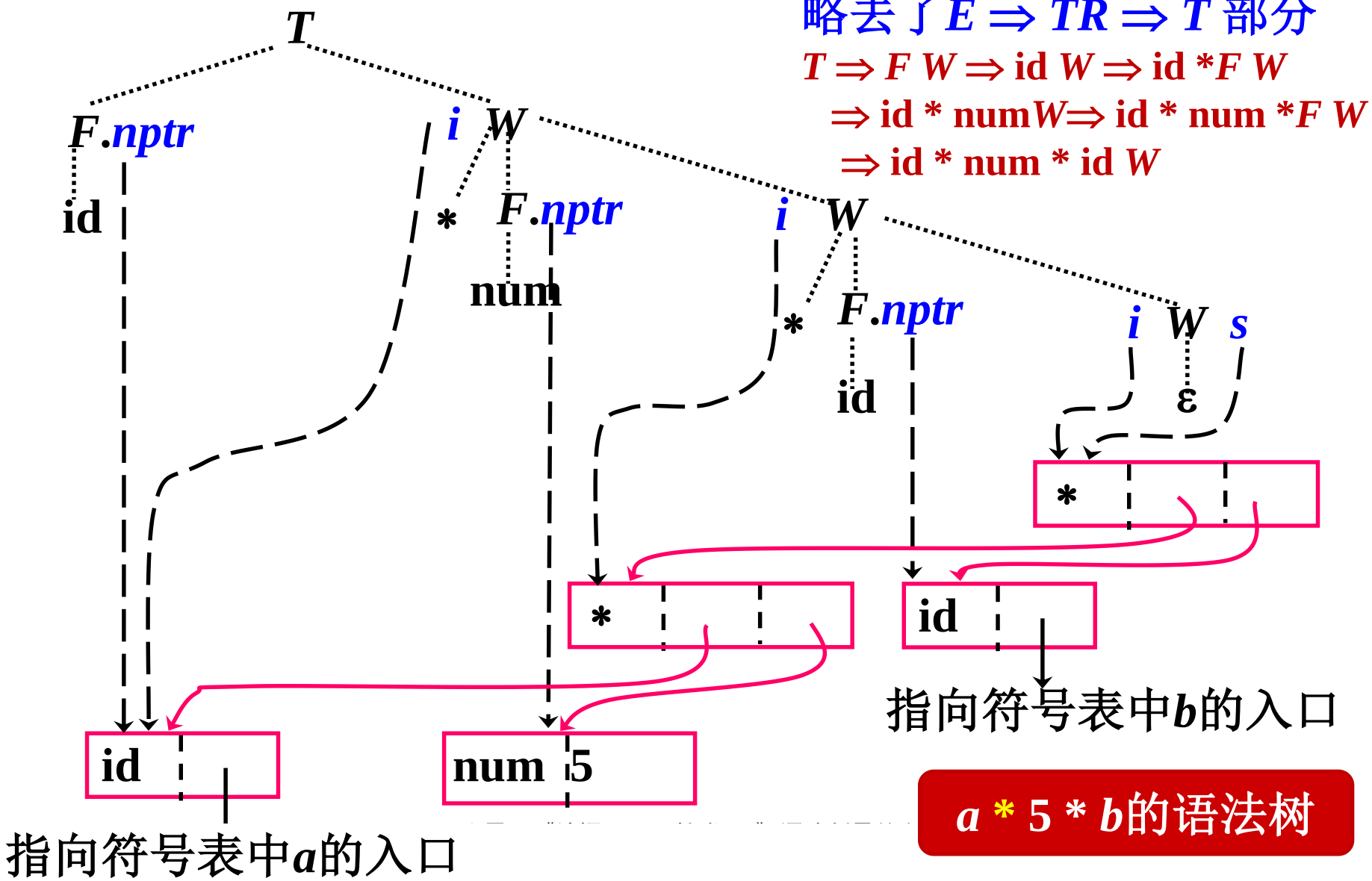
产 生 式	语 义 规 则
$E \to E_1 + T$	$E.nptr = mkNode(\text{'+'}, E_1.nptr, T.nptr)$
$E \to T$	$E.nptr = T.nptr$
$T \to T_1 * F$	$T.nptr = mkNode(\text{'*'}, T_1.nptr, F.nptr)$
$T \to F$	$T.nptr = F.nptr$
$F \to (E)$	$F.nptr = E.nptr$
$F \to \text{id}$	$F.nptr = mkLeaf(\text{id}, \text{id}.entry)$
$F \to \text{num}$	$F.nptr = mkLeaf(\text{num}, \text{num}.val)$

产 生 式	语 义 规 则
$E \to T\ R$	$R.i = T.nptr\ ;\ E.nptr = R.s$
$R \to +\ TR_1$	$R_1.i = mkNode\ (\text{'+'}, R.i, T.nptr);\ R.s = R_1.s$
$R \to \varepsilon$	$R.s = R.i$
$T \to FW$	$W.i = F.nptr\ ;\ T.nptr = W.s$
$W \to *\ FW_1$	$W_1.i = mkNode\ (\text{'+'}, W.i, F.nptr);\ W.s = W_1.s$
$W \to \varepsilon$	$W.s = W.i$
…	…

语法树的构造（LL文法）

略去了$E \Rightarrow TR \Rightarrow T$部分

$T \Rightarrow F\ W \Rightarrow \text{id}\ W \Rightarrow \text{id}\ *F\ W$
$\Rightarrow \text{id} * \text{num} W \Rightarrow \text{id} * \text{num}\ *F\ W$
$\Rightarrow \text{id} * \text{num} * \text{id}\ W$



a * 5 * b的语法树

消除左递归的语法树构造翻译方案

$E \to T$ $\quad \{R.i = T.nptr\}$ $\quad T + T + T + \ldots$

$\quad R$ $\quad \{E.nptr = R.s\}$

$R \to +$

$\quad T$ $\quad \{R_1.i = mkNode(\text{'+'}, R.i, T.nptr)\}$

$\quad R_1$ $\quad \{R.s = R_1.s\}$

$R \to \varepsilon$ $\quad \{R.s = R.i\}$

$T \to F$ $\quad \{W.i = F.nptr\}$

$\quad W$ $\quad \{T.nptr = W.s\}$

$W \to *$

$\quad F$ $\quad \{W_1.i = mkNode(\text{'*'}, W.i, F.nptr)\}$

$\quad W_1$ $\quad \{W.s = W_1.s\}$

$W \to \varepsilon$ $\quad \{W.s = W.i\}$

产 生 式	语 义 规 则
$E \to E_1 + T$	$E.nptr = mkNode(\text{'+'}, E_1.nptr, T.nptr)$
$E \to T$	$E.nptr = T.nptr$
$T \to T_1 * F$	$T.nptr = mkNode(\text{'*'}, T_1.nptr, F.nptr)$
$T \to F$	$T.nptr = F.nptr$
$F \to (E)$	$F.nptr = E.nptr$
$F \to \text{id}$	$F.nptr = mkLeaf(\text{id}, \text{id}.entry)$
$F \to \text{num}$	$F.nptr = mkLeaf(\text{num}, \text{num}.val)$

继承属性的计算内嵌在产生式右部的某文法符号之前，表示在分析该文法符号之前计算

F 的产生式部分不再给出

中国科学技术大学
University of Science and Technology of China

4.3 L属性定义的自上而下计算

- 翻译方案
- 预测翻译器的设计
- 用综合属性代替继承属性

预测翻译器的设计

方法：将预测解析器的构造方法推广到翻译方案的实现（*LL*文法）

产生式$R \rightarrow +TR \mid \varepsilon$ 的分析过程

```
void R( ) {
    if (lookahead == '+' ) {          // First(+TR)
            match ( '+' ); T( ); R( );
    }
    else if (lookahead == ')' || lookahead == '$' )  ; // Follow(R)
    else error( );
}
```

预测翻译器的设计


```
syntaxTreeNode * R (syntaxTreeNode * i) {
  //继承属性作为参数, 综合属性作为返回值
    syntaxTreeNode *nptr, *i1, *s1, *s;
    char addoplexeme;

    if (lookahead == '+' ) {
        addoplexeme = lexval;
        match('+' ); nptr = T( );
        i1 = mkNode(addoplexeme, i , nptr);
        s1 = R (i1); s = s1;
    }
    else if (lookahead == ')' || lookahead == '$')  s = i;
    else error( );
    return s;
}
```

```
void R( ) {
    if (lookahead == '+' ) {
        match ( '+' ); T( ); R( );
    }
    else if (lookahead == ')' || lookahead == '$' )  ;
    else error( );
}
```

R* : *i*, *s
T* : *nptr
+ : *addoplexeme*

中国科学技术大学
University of Science and Technology of China

4.3 L属性定义的自上而下计算

- 翻译方案
- 预测翻译器的设计
- 用综合属性代替继承属性

非L属性定义

例 Pascal语言的变量声明，如 m, n : integer

$D \rightarrow L : T$	$L.in = T.type$
$L \rightarrow L_1$, id | id	$L_1.in = L.in$, …
$T \rightarrow$ integer | char	$T.type$ = …

该语法制导定义非L属性定义

信息从右向左流，归约从左向右，两者不一致

非L属性定义：改写文法

例 Pascal的声明，如m, n : integer

$D \rightarrow L : T$ $\quad L.in = T.type$ (非L属性定义：属性信息从右流向左)

$L \rightarrow L_1$, id | id $\quad L_1.in = L.in, \ldots$

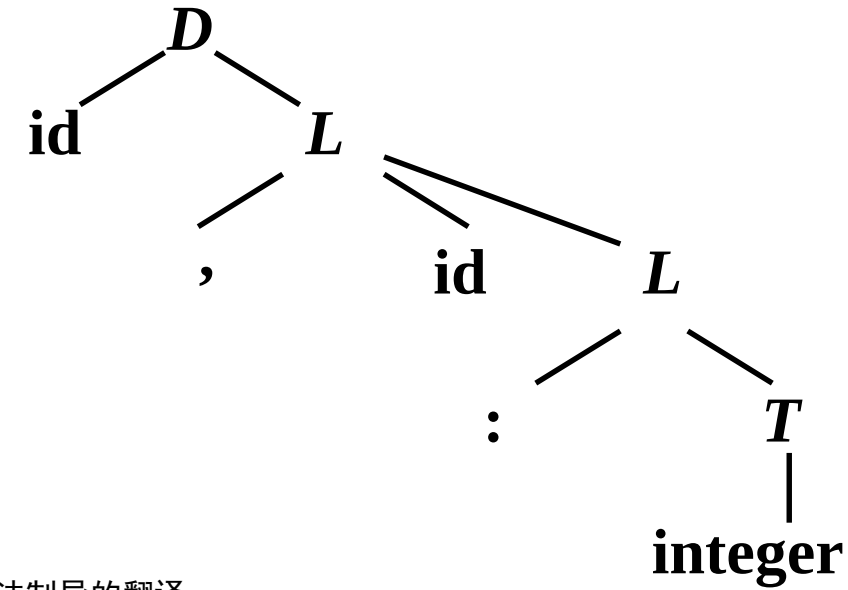
$T \rightarrow$ integer | char $\quad T.type = \ldots$

等所需信息获得后再归约，改成从右向左归约

$D \rightarrow$ id L （S属性定义）

$L \rightarrow$, id L | : T

$T \rightarrow$ integer | char



用综合属性代替继承属性

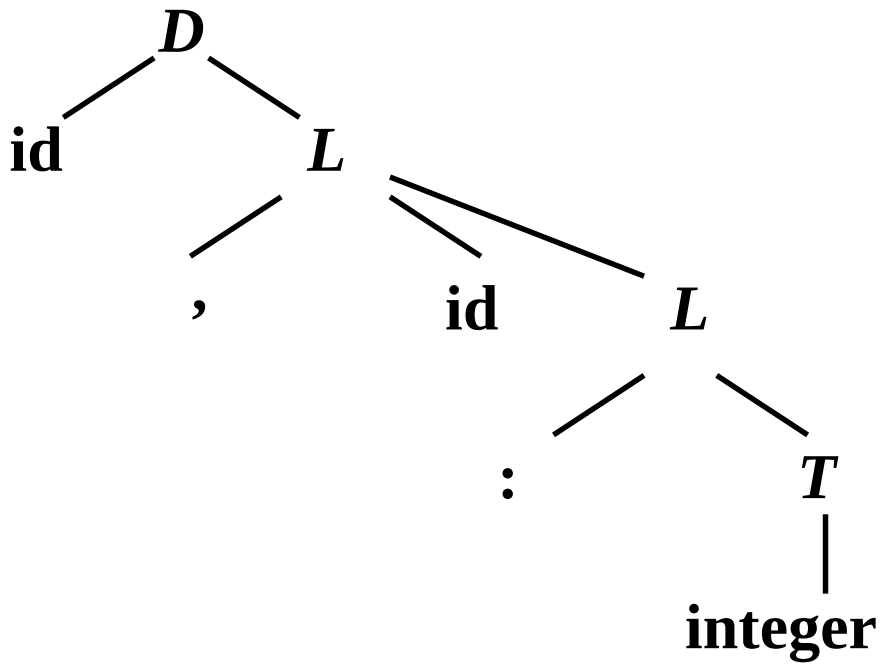
$D \rightarrow$ id L { *addtype* (id. *entry*, *L. type*);}

$L \rightarrow$, id L_1 { *L. type* = L_1. *Type*;
addtype (id. *entry*, L_1. *type*) ;}

$L \rightarrow$: T {*L. type* = *T. type*;}

$T \rightarrow$ integer {*T. type* = *integer*;}

$T \rightarrow$ real {*T. type* = *real*;}



中国科学技术大学
University of Science and Technology of China

4.4 L属性定义的自下而上计算

- 删除翻译方案中的内嵌动作
- 继承属性的计算

*L*属性的自下而上计算

在自下而上分析的框架中实现*L*属性定义的方法

- 它能实现任何基于LL(1)文法的*L*属性定义
- 也能实现许多（但不是所有的）基于LR(1) 的*L*属性定义

删除翻译方案中的内嵌动作

中缀表达式翻译成后缀表达式

$E \rightarrow T\ R$

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

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

- 在文法中加入推出ε的标记非终结符，让每个内嵌动作由不同的标记非终结符M代表，并把该动作移到产生式$M \rightarrow \varepsilon$的右端 （继承属性=>综合属性）

$E \rightarrow T\ R$

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

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

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

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

YACC会按这种方法来处理输入的文法，即为内嵌的语义动作引入ε产生式

L属性的自下而上计算

bison-examples: config/exprL.y

```
input   : …
   | input{ lineno ++; printf("Line %d:\t", lineno);}  line { printf("*"); } ;
```

- $$表示产生式LHS(left-hand side)符号的语义值，$1, $2…依次为RHS(right-hand side)中符号的语义值，本例中line的语义值通过$3 来引用
- Bison自动为上述2个蓝色的嵌入动作引入2个产生ε的标记非终结符，对应case 3和4

src/exprL.tab.c

```
 case 4:
/* Line 1806 of yacc.c  */
#line 36 "config/exprL.y"
    { printf("*"); }
    break;
```

```
yyreduce:
  /* yyn is the number of a rule to reduce with.  */
 …
 YY_REDUCE_PRINT (yyn);
 switch (yyn) { …
    case 3:
    /* Line 1806 of yacc.c  */
    #line 32 "config/exprL.y"
            { lineno ++; printf("Line %d:\t", lineno);
              }
            break;
```

产生式编号

中国科学技术大学
University of Science and Technology of China

4.4 L属性定义的自下而上计算

- 删除翻译方案中的内嵌动作
- 继承属性的计算

继承属性在分析栈中

情况1 属性位置可预测

例 int p, q, r

$D \rightarrow T$ {*L.in* = *T.type*}

L

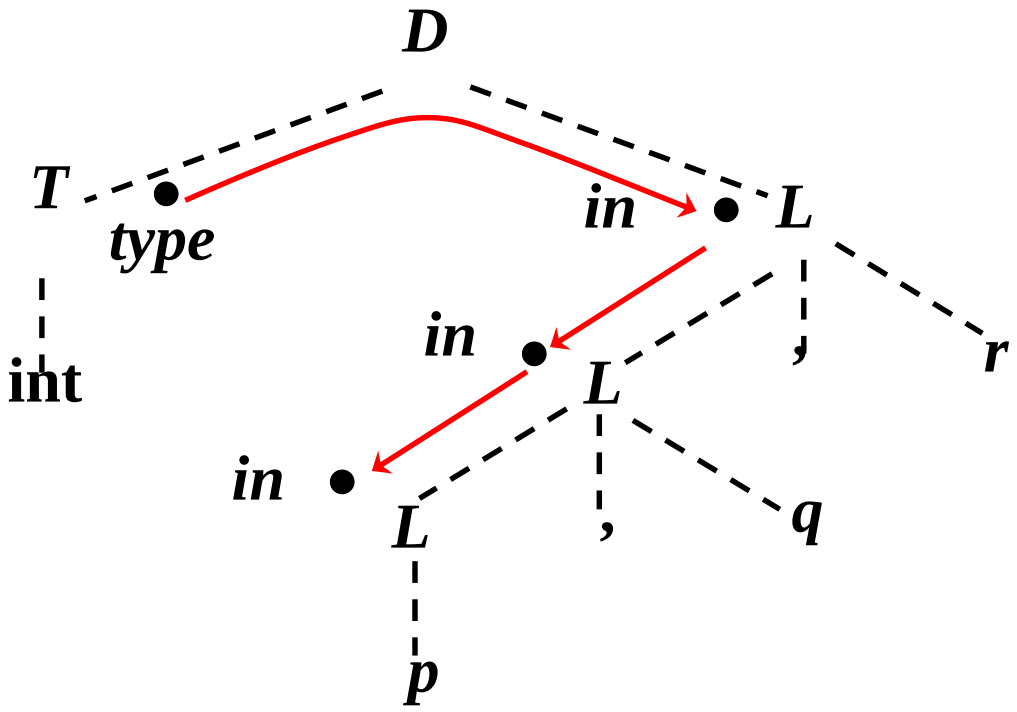
$T \rightarrow$ int {*T. type* = *integer*}

$T \rightarrow$ real {*T. type* = *real*}

$L \rightarrow$ {L_1.*in* = *L.in* }

L_1, id {*addtype* (id.*entry*, *L.in*)}

$L \rightarrow$ id {*addtype* (id.*entry*, *L.in*)}



继承属性值
已在分析栈中

继承属性在分析栈中

情况1 属性位置可预测

例 int p, q, r

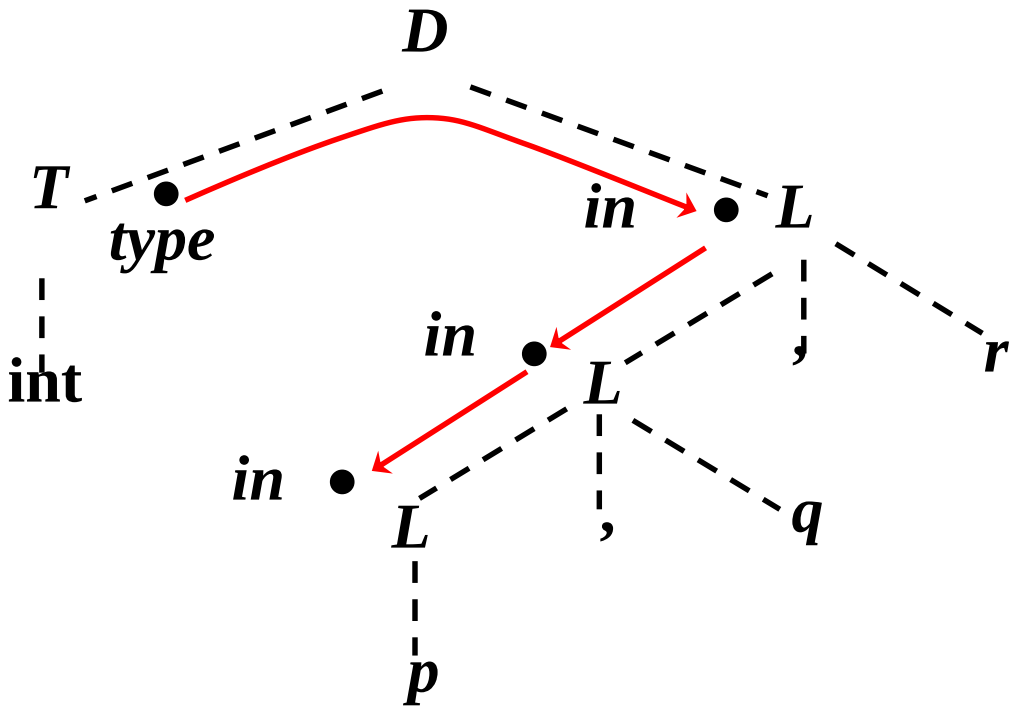
$D \rightarrow T$ {~~*L.in = T.type*~~}
L

$T \rightarrow$ int {*T. type = integer*}

$T \rightarrow$ real {*T. type = real*}

$L \rightarrow$ {~~L_1.*in = L.in*~~}
L_1, id {*addtype* (id.*entry*, *L.in*)}

$L \rightarrow$ id {*addtype* (id.*entry*, *L.in*)}

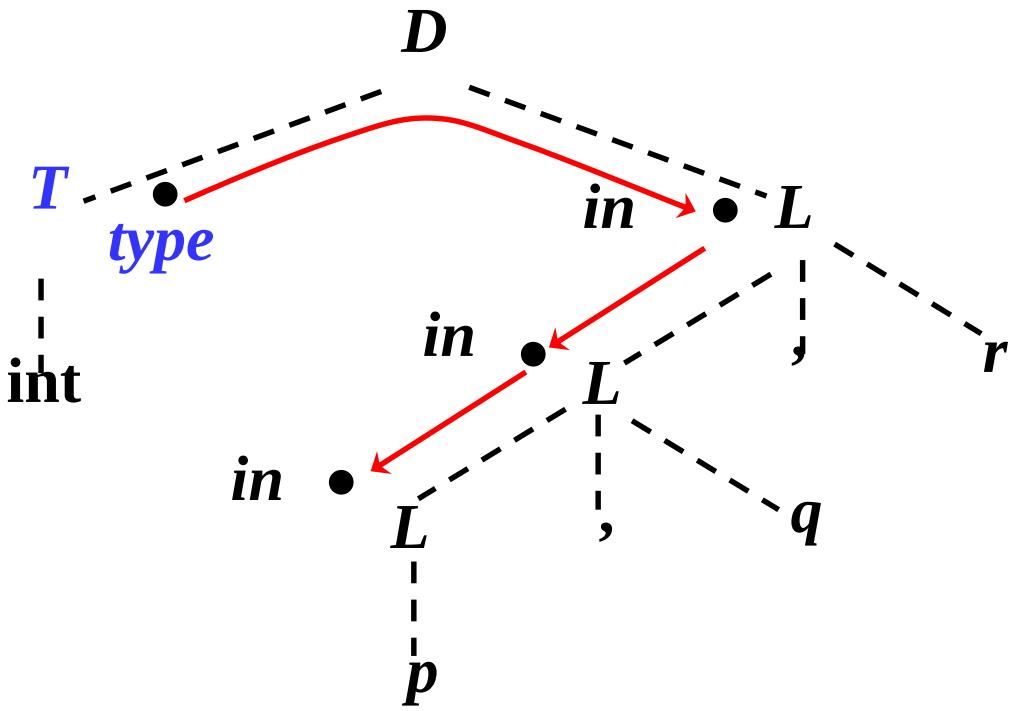


略去继承属性的计算
引用继承属性的地方改成
引用其他符号的综合属性

继承属性在分析栈中

情况1 属性位置可预测

产 生 式	代 码 段
$D \rightarrow TL$	
$T \rightarrow$ int	$val[top] = integer$
$T \rightarrow$ real	$val[top] = real$
$L \rightarrow L_1$, id	$addType(val[top], val[top-3])$;
$L \rightarrow$ id	$addType(val[top], val[top-1])$;



略去继承属性的计算
引用继承属性的地方改成
引用其他符号的综合属性

YACC中的继承属性定义

在内嵌动作代码中设置该文法符号的语义值

bison-examples: config/exprL1.y

```
line   : …
       | NUMBER { …
                  $<val>lineno = $1;   // val是%union中声明的语义值类型
                  // $<val>$ = $1;        // 该语义动作代码未指定名字时
                  …
                  } [lineno]
       exp EOL { …
                  printf("Line %d: %g\n", (int) $<val>lineno, $3);
                  …
       }
```

指明lineno的语义值所在的共用体的域

给内嵌语义动作对应的标记非终结符命名

YACC中的继承属性定义

在内嵌代码中使用存储在栈中任意固定相对位置的语义值

bison-examples: config/midrule.y

```
exp: a_1 a_2 { $<val>$ = 3; } { $<val>$ = $<val>3 + 1; } a_5
    sum_of_the_five_previous_values
   {
     USE (($1, $2, $<foo>3, $<foo>4, $5));
     printf ("%d\n", $6);
   }

sum_of_the_five_previous_values:
   {
     $$ = $<val>0 + $<val>-1 + $<val>-2 + $<val>-3 + $<val>-4;
   }
```

$<val>0、 $<val>-1、$<val>-2、$<val>-3、$<val>-4分别表示栈中a_5、{ $<val>$ = $<val>3 + 1; }、{ $<val>$ = 3; }、a_2、a_1文法符号的语义值

继承属性在分析栈中

情况2 属性位置不可预测

继承属性值
已在分析栈中

$S \to aAC$ $\quad C.i = A.s$

$S \to bABC$ $\quad C.i = A.s$

$C \to c$ $\quad C.s = g(C.i)$

*A*和*C*之间可能有B，也可能没有B，*C.i* 的值有2种可能

□ 增加标记非终结符，使得位置可以预测

$S \to aAC$ $\quad C.i = A.s$

$S \to bABMC$ $\quad M.i = A.s;\ C.i = M.s$

$C \to c$ $\quad C.s = g(C.i)$

$M \to \varepsilon$ $\quad M.s = M.i$

模拟继承属性的计算

- 继承属性是综合属性的函数

$S \rightarrow aAC$　　$C.i = f(A.s)$

$C \rightarrow c$　　$C.s = g(C.i)$

继承属性不直接等于某个综合属性

- 增加标记非终结符，把$f(A.s)$的计算移到对标记非终结符归约时进行

$S \rightarrow aANC$　　$N.i = A.s;\ C.i = N.s$

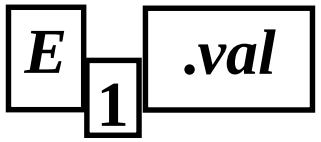
$N \rightarrow \varepsilon$　　$N.s = f(N.i)$

$C \rightarrow c$　　$C.s = g(C.i)$

L属性定义的自下而上计算

例 数学排版语言EQN

E sub 1 .val



$S \rightarrow B$

$B \rightarrow B_1\ B_2$

$B \rightarrow B_1$ sub B_2

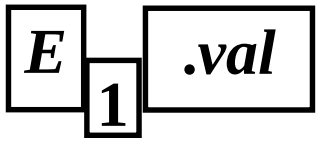
$B \rightarrow$ text

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
$B \rightarrow B_1\ MB_2$	$B_1.ps = B.ps$; $M.i = B.ps$; $B_2.ps = M.s$;$B.ht = max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1$ sub NB_2	$B_1.ps = B.ps$; $N.i = B.ps$; $B_2.ps = N.s$; $B.ht = disp\ (B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = shrink(N.i)$
$B \rightarrow$ text	$B.ht =$ text.$h \times B.ps$

L属性定义的自下而上计算

例 数学排版语言EQN

E sub 1 .val



$S \rightarrow B$

$B \rightarrow B_1\ B_2$

$B \rightarrow B_1$ sub B_2

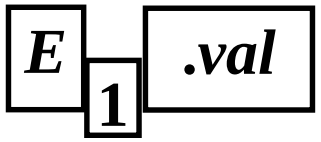
$B \rightarrow$ text

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
$B \rightarrow B_1$ sub NB_2	$B_1.ps = B.ps$; $N.i = B.ps$; $B_2.ps = N.s$; $B.ht = disp\ (B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = shrink(N.i)$
$B \rightarrow$ text	$B.ht =$ text.$h \times B.ps$

L属性定义的自下而上计算

例 数学排版语言EQN

E sub 1 .val



$S \rightarrow B$

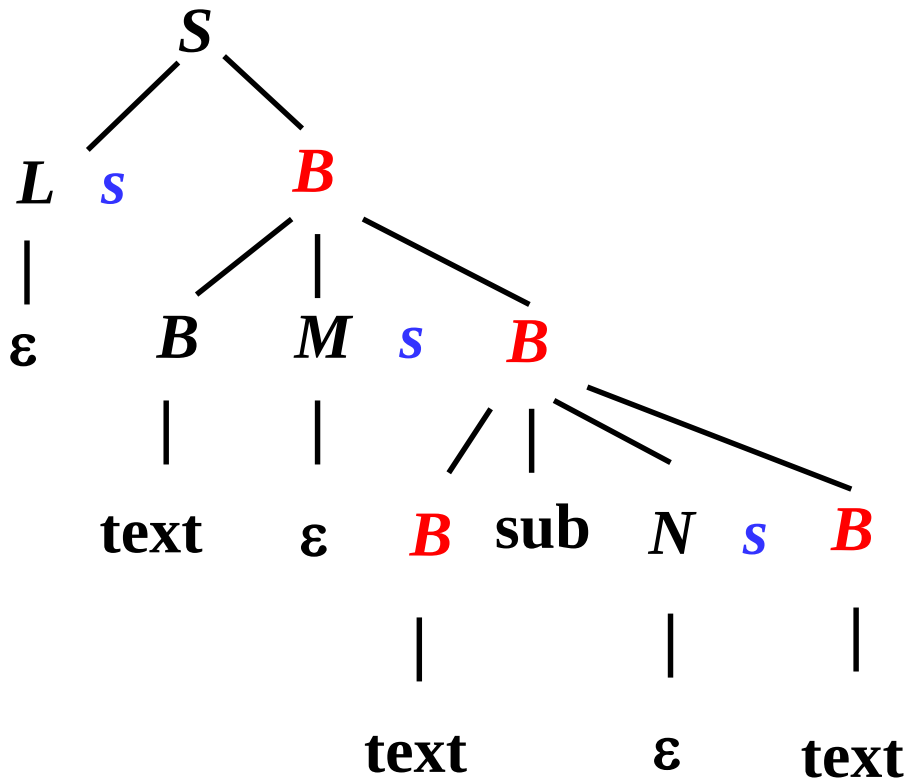
$B \rightarrow B_1\ B_2$

$B \rightarrow B_1$ sub B_2

$B \rightarrow$ text

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
$B \rightarrow B_1$ sub NB_2	$B_1.ps = B.ps$; $N.i = B.ps$; $B_2.ps = N.s$; $B.ht = disp\ (B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = shrink(N.i)$ 兼有计算功能
$B \rightarrow$ text	$B.ht =$ text.$h \times B.ps$

EQN：自下而上计算的实现

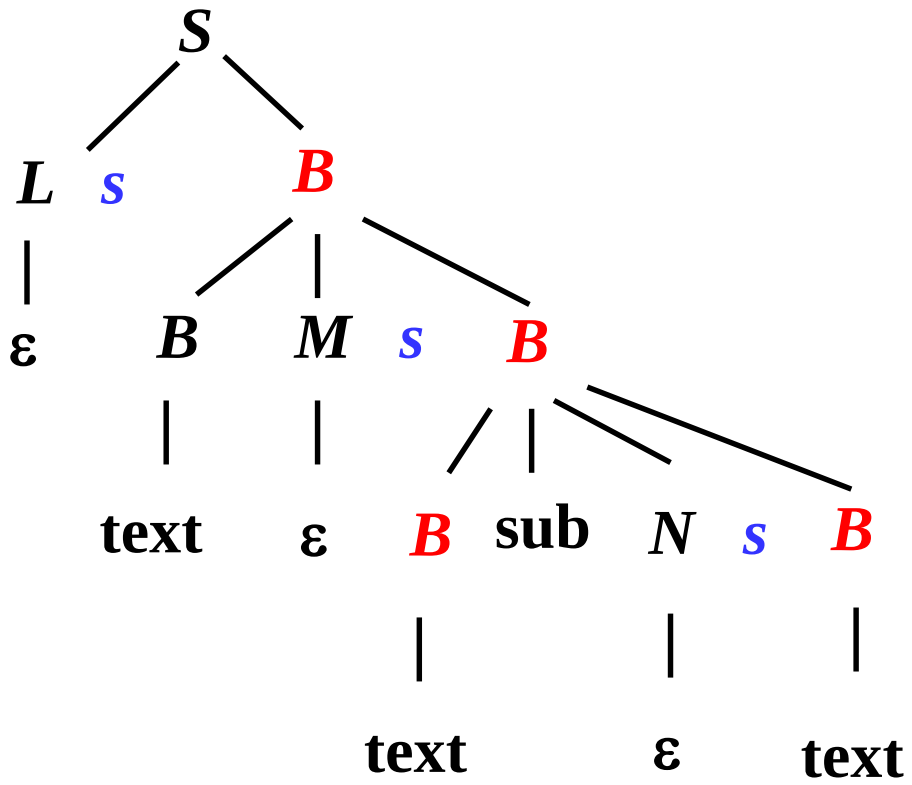


产 生 式	语 义 规 则
$S \rightarrow LB$	$B.ps = L.s$; $S.ht = B.ht$
$L \rightarrow \varepsilon$	$L.s = 10$
$B \rightarrow B_1\ MB_2$	$B_1.ps = B.ps$; $M.i = B.ps$; $B_2.ps = M.s$; $B.ht = max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1$ sub NB_2	$B_1.ps = B.ps$; $N.i = B.ps$; $B_2.ps = N.s$; $B.ht = disp\ (B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = shrink(N.i)$
$B \rightarrow$ text	$B.ht =$ text.$h \times B.ps$

在text归约成B时，B的ps属性都在次栈顶位置

继承属性的值为栈中某个综合属性的值时，栈中只保存综合属性的值

EQN：自下而上计算的实现

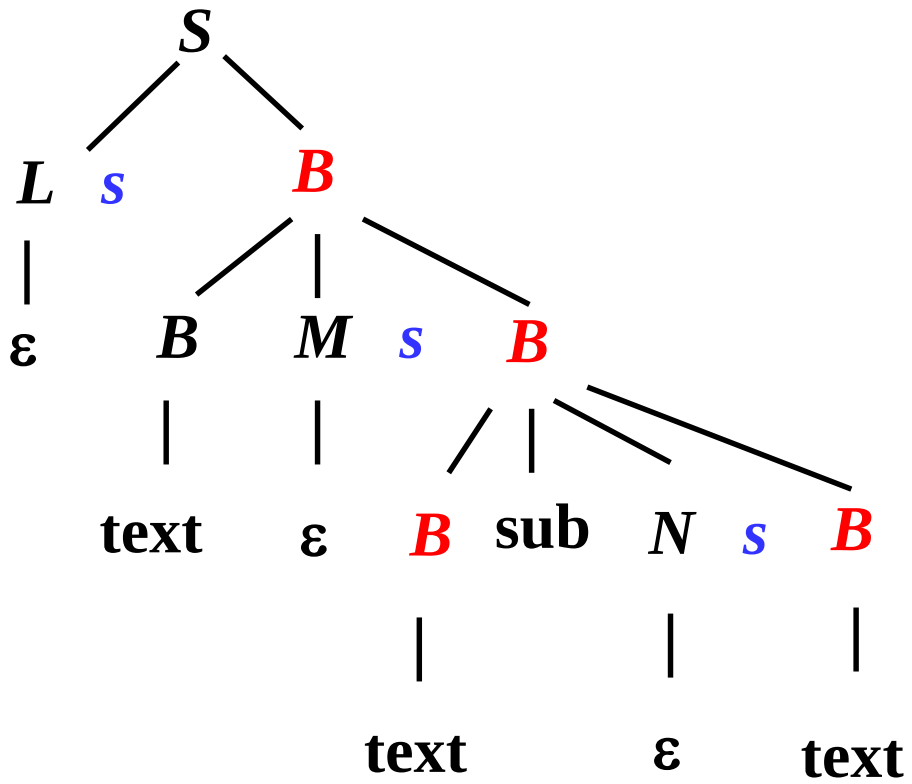


产 生 式	语 义 规 则
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$L.s = 10$
$B \rightarrow B_1\ MB_2$	$B_1.ps = B.ps;\ \ M.i = B.ps;$ $B_2.ps = M.s;B.ht = max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1$ sub NB_2	$B_1.ps = B.ps;\ \ N.i = B.ps;$ $B_2.ps = N.s;\ B.ht = disp\ (B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = shrink(N.i)$
$B \rightarrow$ text	$B.ht =$ text$.h \times B.ps$

在text归约成B时，B的ps属性都在次栈顶位置

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

EQN：自下而上计算的实现

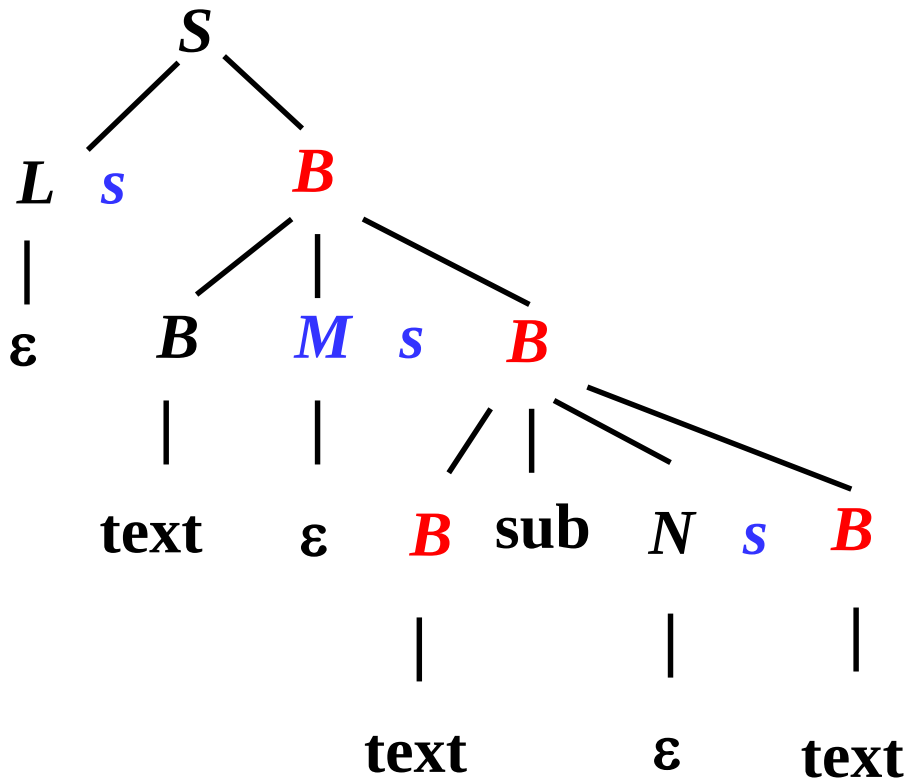


产 生 式	语 义 规 则
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1\ MB_2$	$B_1.ps = B.ps;\ \ M.i = B.ps;$ $B_2.ps = M.s;B.ht = max(B_1.ht, B_2.ht)$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1$ sub NB_2	$B_1.ps = B.ps;\ \ N.i = B.ps;$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

在text归约成B时，B的ps属性都在次栈顶位置

$L.s = 10$

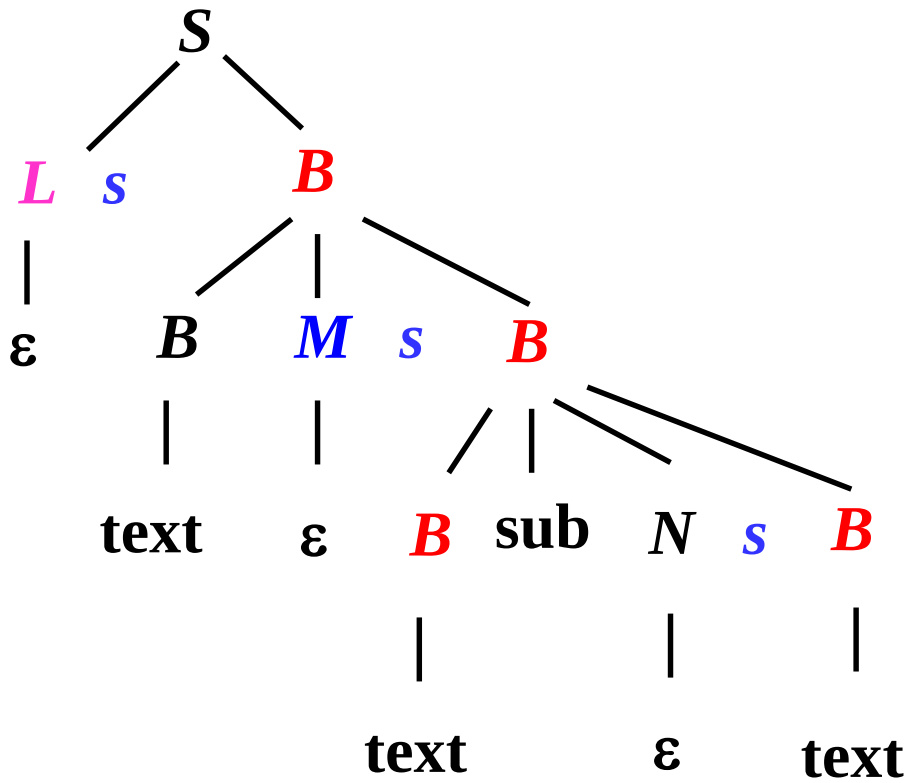
EQN：自下而上计算的实现



产 生 式	语 义 规 则
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1\ MB_2$	$val[top-2] = max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$M.s = M.i$
$B \rightarrow B_1$ sub NB_2	$B_1.ps = B.ps;\ \ N.i = B.ps;$ $B_2.ps = N.s;\ B.ht = disp\ (B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = shrink(N.i)$
$B \rightarrow$ text	$B.ht = \text{text}.h \times B.ps$

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

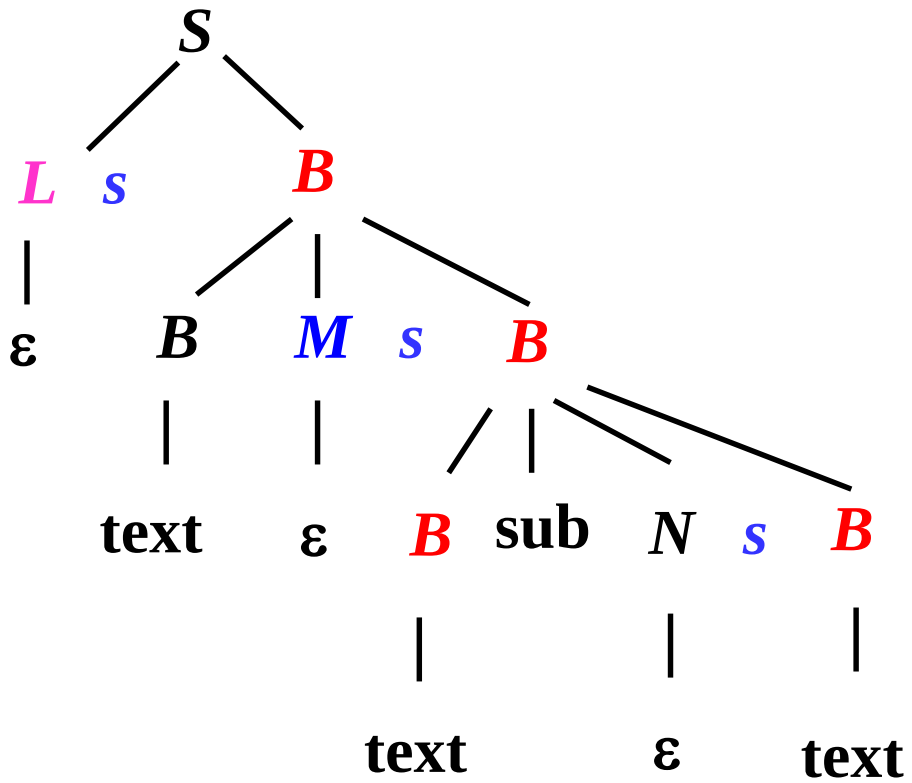
EQN：自下而上计算的实现



产 生 式	语 义 规 则
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1\ MB_2$	$val[top-2] = max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$val[top+1] = val[top-1]$
$B \rightarrow B_1$ sub NB_2	$B_1.ps = B.ps;\ \ N.i = B.ps;$ $B_2.ps = N.s;\ B.ht = disp\ (B_1.ht, B_2.ht)$
$N \rightarrow \varepsilon$	$N.s = shrink(N.i)$
$B \rightarrow$ text	$B.ht = \text{text}.h \times B.ps$

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

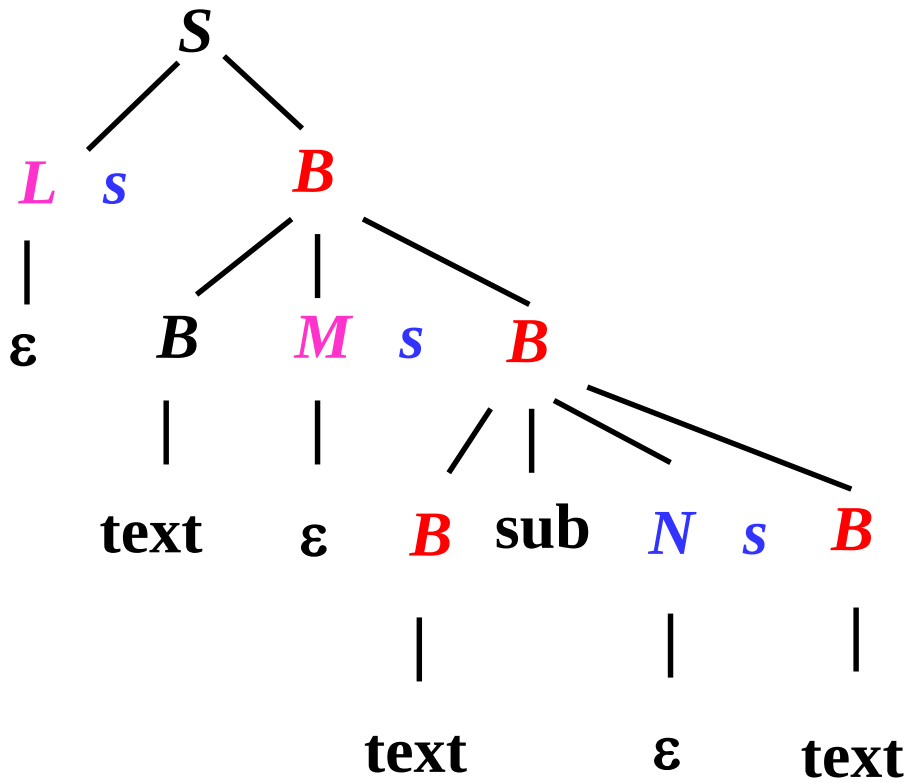
EQN：自下而上计算的实现



产 生 式	语 义 规 则
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1\ MB_2$	$val[top-2] = max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$val[top+1] = val[top-1]$
$B \rightarrow B_1$ sub NB_2	$val[top-3] = disp\ (val[top-3], val[top])$
$N \rightarrow \varepsilon$	$N.s = shrink(N.i)$
$B \rightarrow$ text	$B.ht =$ text$.h \times B.ps$

$B_1.ps = B.ps;\ N.i = B.ps; B_2.ps = N.s;\ B.ht = disp\ (B_1.ht, B_2.ht)$

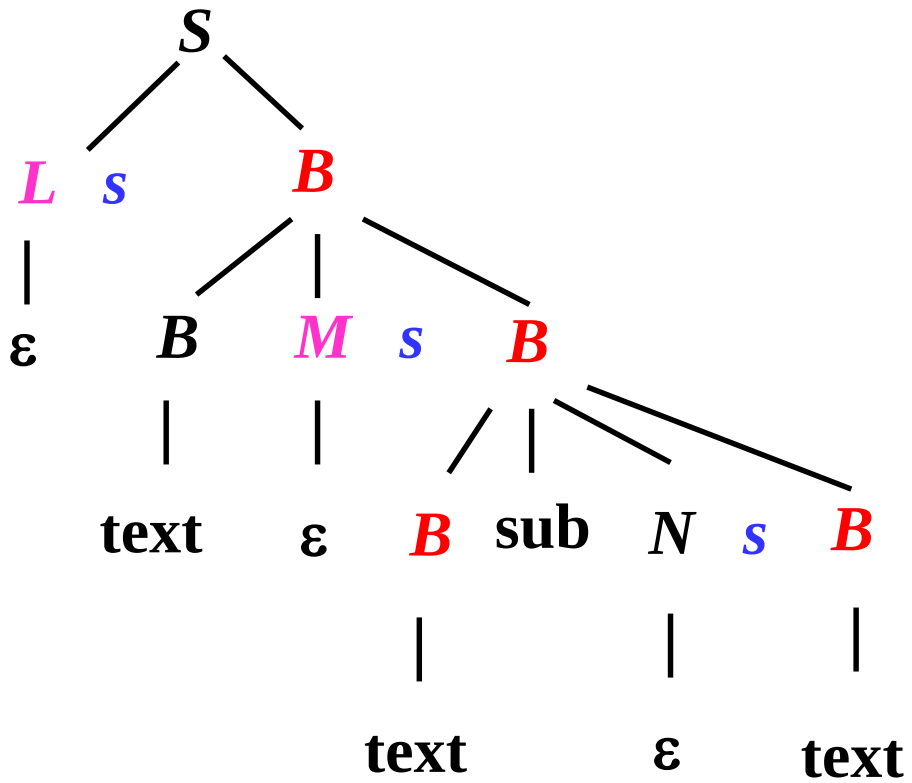
EQN：自下而上计算的实现



产 生 式	语 义 规 则
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1\ MB_2$	$val[top-2] = max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$val[top+1] = val[top-1]$
$B \rightarrow B_1$ sub NB_2	$val[top-3] = disp\ (val[top-3], val[top])$
$N \rightarrow \varepsilon$	$val[top+1] = shrink(val[top-2])$
$B \rightarrow$ text	$B.ht = \text{text}.h \times B.ps$

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

EQN：自下而上计算的实现



产 生 式	语 义 规 则
$S \rightarrow LB$	$val[top-1] = val[top]$
$L \rightarrow \varepsilon$	$val[top+1] = 10$
$B \rightarrow B_1\ MB_2$	$val[top-2] = max(val[top-2], val[top])$
$M \rightarrow \varepsilon$	$val[top+1] = val[top-1]$
$B \rightarrow B_1$ sub NB_2	$val[top-3] = disp\ (val[top-3], val[top])$
$N \rightarrow \varepsilon$	$val[top+1] = shrink(val[top-2])$
$B \rightarrow$ text	$val[top] = val[top] \times val[top-1]$

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

中国科学技术大学

University of Science and Technology of China

下期预告：语义分析