编译原理

第六章 属性文法和语法制导翻译

第六章 属性文法和语法制导翻译

- 属性文法
- 基于属性文法的处理方法
- S- 属性文法的自下而上计算
- L- 属性文法和自顶向下翻译

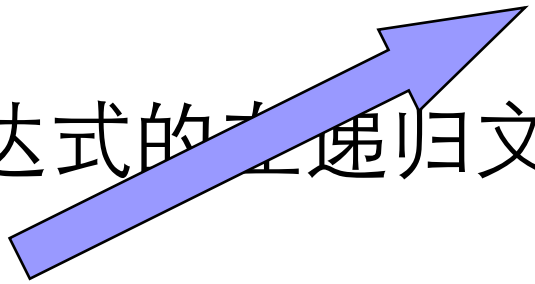
第六章 属性文法和语法制导翻译

- 属性文法
- 基于属性文法的处理方法
- S- 属性文法的自下而上计算
- L- 属性文法和自顶向下翻译

6.4.2 自顶向下翻译

- 动作是在处于相同位置上的符号被展开（匹配成功）时执行的
- 为了构造不带回溯的自顶向下语法分析，必须消除文法中的左递归
- 当消除一个翻译模式的基本文法的左递归时同时考虑属性——适合带综合属性的翻译模式

- 关于算术表达式的左递归文法模式



$E \rightarrow T\ R$
$R \rightarrow +\ T\ R_1$
$R \rightarrow -\ T\ R_1$
$R \rightarrow \varepsilon$
$T \rightarrow (\ E\ )$
$T \rightarrow num$

$E \rightarrow E_1+T$	$\{E.val:=E_1.val+T.val\}$
$E \rightarrow E_1-T$	$\{E.val:=E_1.val-T.val\}$
$E \rightarrow T$	$\{E.val:=T.val\}$
$T \rightarrow (E)$	$\{T.val:=E.val\}$
$T \rightarrow num$	$\{T.val:=num.val\}$

- 消除左递归，构造新的翻译模式

$E \rightarrow T$ {R.i:=T.val}
R {E.val:=R.s}

$R \rightarrow +$
T {R_1.i:=R.i+T.val}
R_1 {R.s:=R_1.s}

$R \rightarrow -$
T {R_1.i:=R.i-T.val}
R_1 {R.s:=R_1.s}

$R \rightarrow \varepsilon$ {R.s:=R.i}

$T \rightarrow (E)$ {T.val:=E.val}

$T \rightarrow$ num {T.val:=num.val}

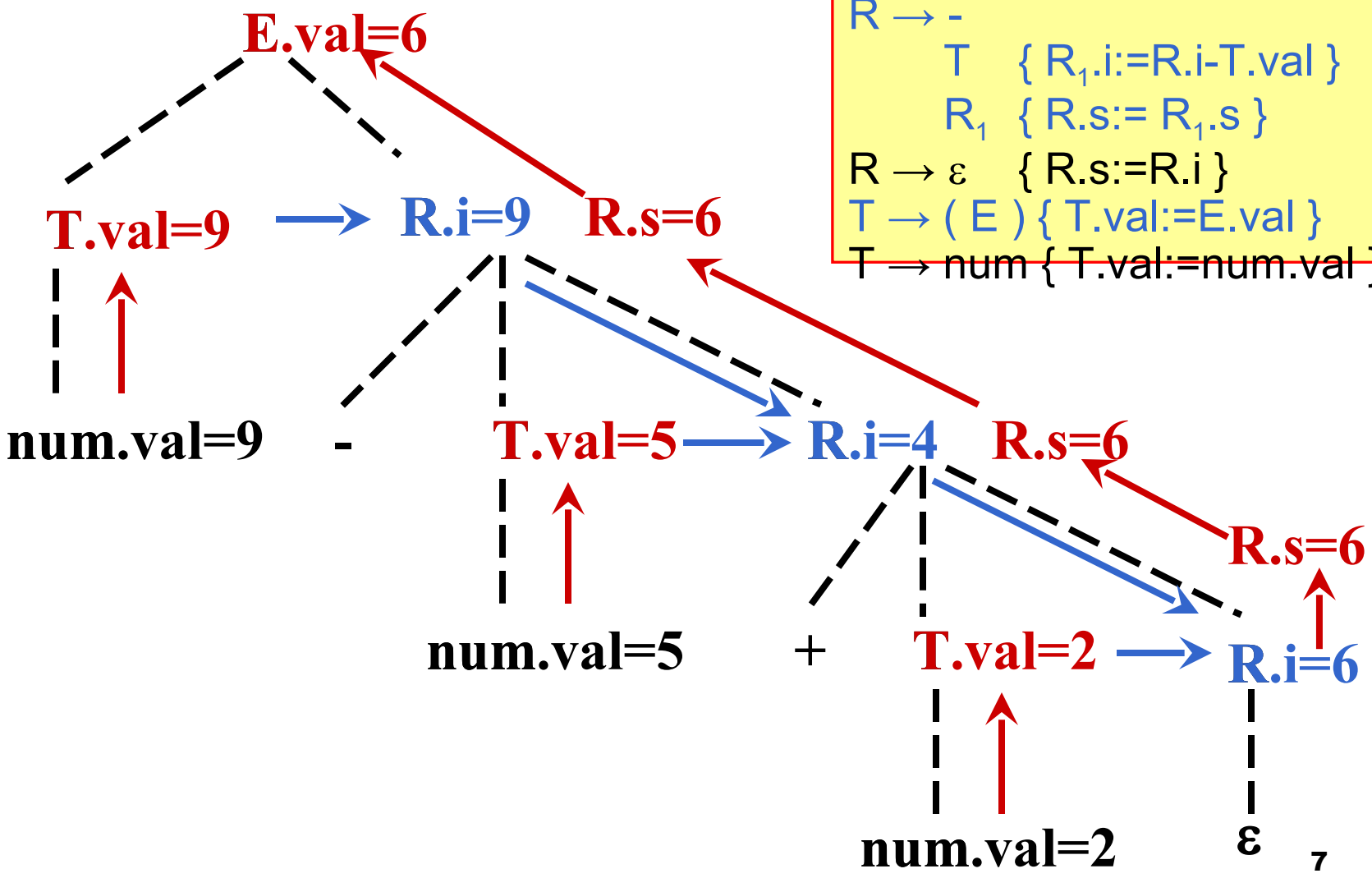
$E \rightarrow T\ R$
$R \rightarrow +TR_1$
$R \rightarrow -TR_1$
$R \rightarrow \varepsilon$
$T \rightarrow (E)$
$T \rightarrow$ num

R.i: R 前面子表达式的值

R.s: 分析完 R 时子表达式的值

计算表达式 9 – 5 + 2

$E \rightarrow T \quad \{ R.i := T.val \}$
$\quad\quad R \quad \{ E.val := R.s \}$
$R \rightarrow +$
$\quad\quad T \quad \{ R_1.i := R.i + T.val \}$
$\quad\quad R_1 \quad \{ R.s := R_1.s \}$
$R \rightarrow -$
$\quad\quad T \quad \{ R_1.i := R.i - T.val \}$
$\quad\quad R_1 \quad \{ R.s := R_1.s \}$
$R \rightarrow \varepsilon \quad \{ R.s := R.i \}$
$T \rightarrow (E) \quad \{ T.val := E.val \}$
$T \rightarrow num \quad \{ T.val := num.val \}$



一般方法

- 假设有翻译模式：

  $A \rightarrow A_1 Y$ {A.a:=g(A_1.a, Y.y) }

  $A \rightarrow X$ {A.a:=f(X.x) }

  它的每个文法符号都有一个综合属性，用小写字母表示，g 和 f 是任意函数。

- 消除左递归：

  $A \rightarrow XR$

  $R \rightarrow YR \mid \varepsilon$

R.i: R 前面子表达式的值

R.s: 分析完 R 时子表达式的值

- 翻译模式变为：

  $A \rightarrow X$ {R.i:=f (X.x) }

  R {A.a:=R.s}

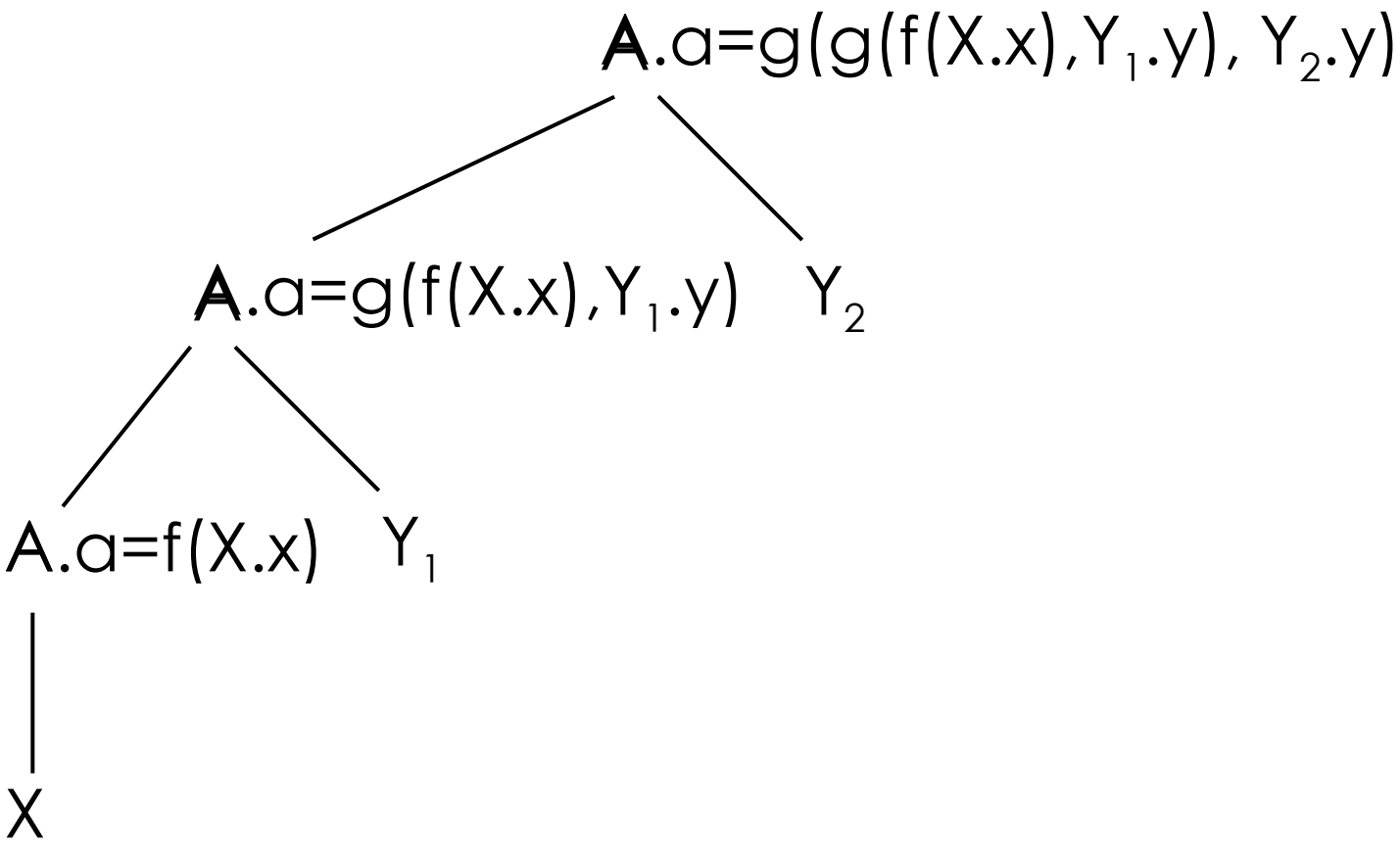
  $R \rightarrow Y$ {R_1.i:=g(R.i, Y.y) }

  R_1 {R.s:=R_1.s}

  $R \rightarrow \varepsilon$ {R.s:=R.i}

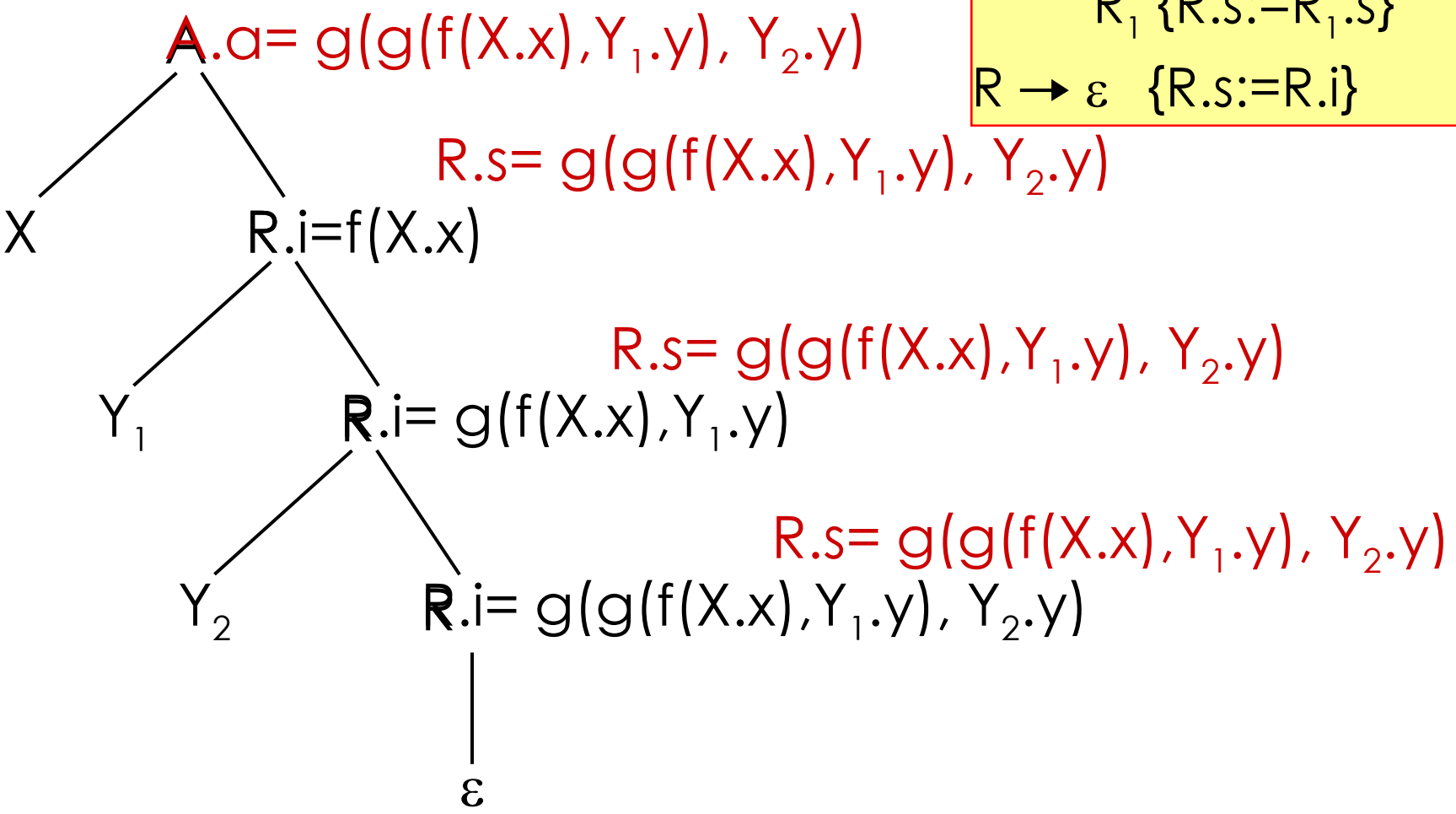
XYY

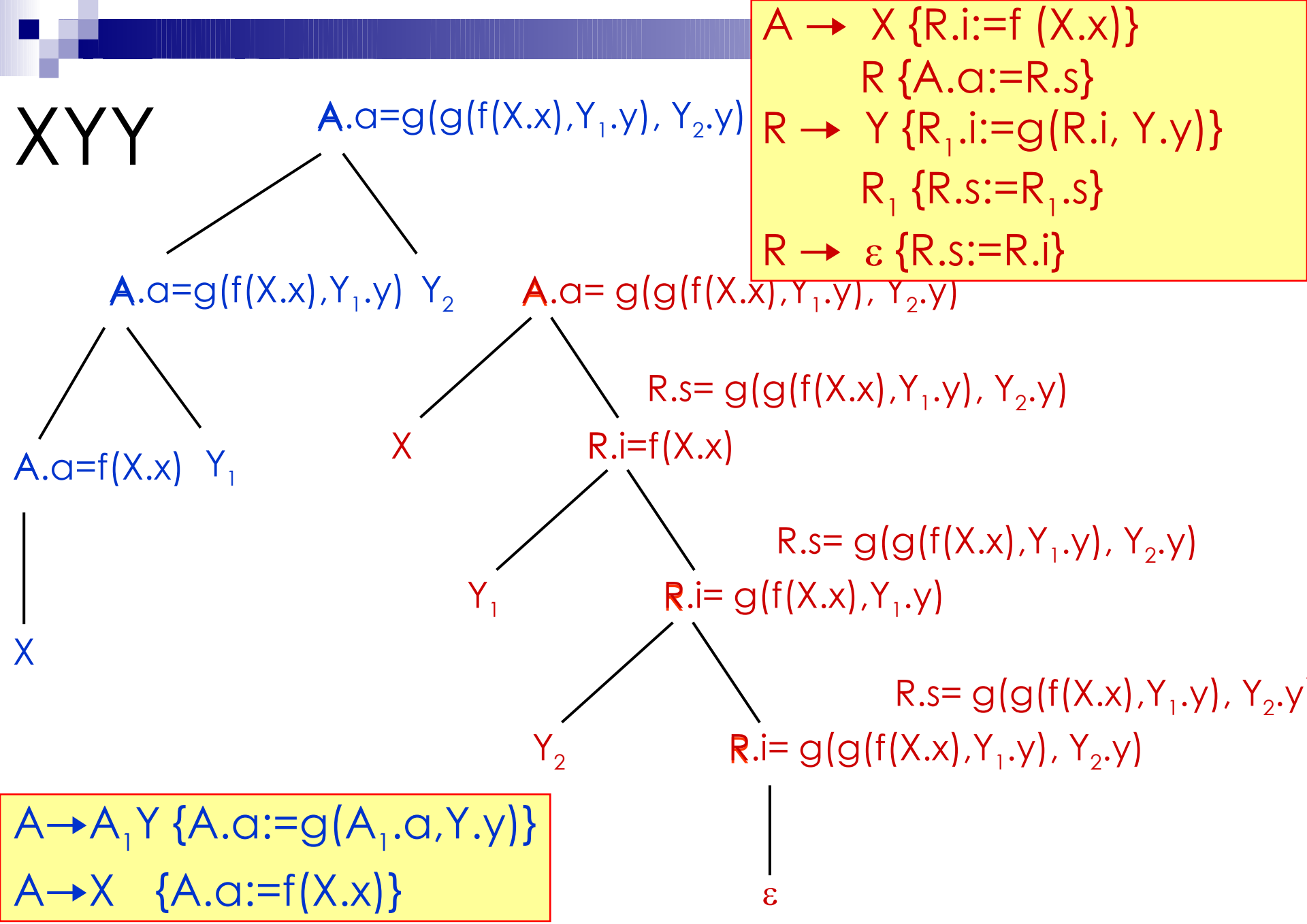
$A \rightarrow A_1 Y$	$\{A.a := g(A_1.a, Y.y)\}$
$A \rightarrow X$	$\{A.a := f(X.x)\}$



XYY

$A \rightarrow X \ \{R.i := f(X.x)\}$
$\quad R \ \{A.a := R.s\}$
$R \rightarrow Y \ \{R_1.i := g(R.i, Y.y)\}$
$\quad R_1 \ \{R.s := R_1.s\}$
$R \rightarrow \varepsilon \ \{R.s := R.i\}$




XYY
A.a=g(g(f(X.x),Y1.y), Y2.y)
A.a=g(f(X.x),Y1.y) Y2
A.a= g(g(f(X.x),Y1.y), Y2.y)
A.a=f(X.x) Y1
X
R.s= g(g(f(X.x),Y1.y), Y2.y)
X
R.i=f(X.x)
Y1
R.s= g(g(f(X.x),Y1.y), Y2.y)
R.i= g(f(X.x),Y1.y)
Y2
R.s= g(g(f(X.x),Y1.y), Y2.y
R.i= g(g(f(X.x),Y1.y), Y2.y)
ε
A → X {R.i:=f (X.x)}
R {A.a:=R.s}
R → Y {R1.i:=g(R.i, Y.y)}
R1 {R.s:=R1.s}
R → ε {R.s:=R.i}
A→A1Y {A.a:=g(A1.a,Y.y)}
A→X {A.a:=f(X.x)}

构造抽象语法树的属性文法定义转化成翻译模式

$E \rightarrow E_1+T$ {E.nptr:=mknode(‘+’ ,E_1.nptr,T.nptr)}

$E \rightarrow E_1-T$ {E.nptr:=mknode(‘-’ ,E_1.nptr,T.nptr)}

$E \rightarrow T$ {E.nptr:=T.nptr}

$A \rightarrow A_1 Y$ {A.a:=g(A_1.a,Y.y) }

$A \rightarrow X$ {A.a:=f(X.x) }

$A \rightarrow X$ {R.i:=f (X.x) }
R {A.a:=R.s}

$R \rightarrow Y$ {R_1.i:=g(R.i, Y.y)}
R_1 {R.s:=R_1.s}

$R \rightarrow \varepsilon$ {R.s:=R.i}

构造抽象语法树的属性文法定义转化成翻译模式

$E \rightarrow T$ {R.i:=T.nptr}
R {E.nptr:=R.s}

$R \rightarrow +$
T {R_1.i:=mknode(‘+’ , R.i, T.nptr)}
R_1 {R.s:=R_1.s}

$R \rightarrow -$
T {R_1.i:=mknode(‘ – ’ , R.i, T.nptr)}
R_1 {R.s:=R_1.s}

$R \rightarrow \varepsilon$ {R.s:=R.i}

$T \rightarrow (E)$ {T.nptr:=E.nptr}

T → id {T.nptr:=mkleaf(id, id.entry)}

T → num {T.nptr:=mkleaf(num, num.val)}

使用继承属性构造 a – 4 + c 的抽象语法

$E \rightarrow T\ \{R.i:=T.nptr\}\ R\ \{E.nptr:=R.s\}$

$R \rightarrow +\ T\ \{R_1.i:=mknode('+',R.i,T.nptr)\}\ R_1\ \{R.s:=R_1.s\}$

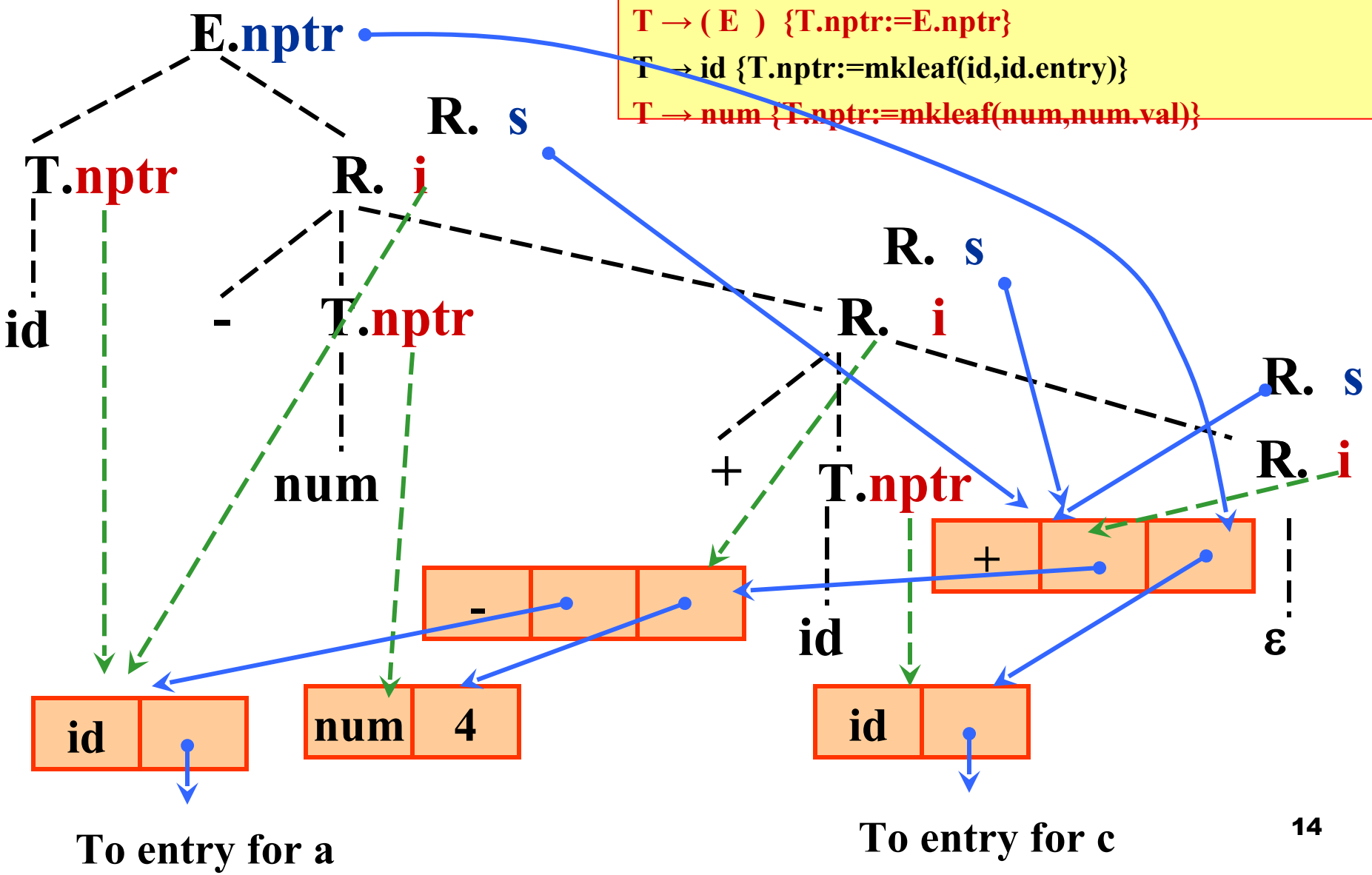
$R \rightarrow -\ T\ \{R_1.i:=mknode('-',R.i,T.nptr)\}\ R_1\ \{R.s:=R_1.s\}$

$R \rightarrow \varepsilon\ \{R.s:=R.i\}$

$T \rightarrow (\ E\ )\ \{T.nptr:=E.nptr\}$

$T \rightarrow id\ \{T.nptr:=mkleaf(id,id.entry)\}$

$T \rightarrow num\ \{T.nptr:=mkleaf(num,num.val)\}$



6.4.3 递归下降翻译器的设计

- 递归下降分析器的设计
  - 分析程序由一组递归过程组成，文法中每个非终结符对应一个过程；所以这样的分析程序称为递归下降分析器

R→addop TR|ε 的递归下降分析过程

```
procedure R;
 begin
   if sym=addop then begin
        advance;T; R
      end
   else begin /*do nothing*/
      end
 end;
```

如何在递归下降分析中实现翻译模式，构造递归下降翻译器?

R → addop
T {R_1.i:= mknode(addop.lexme, R.i, T.nptr)}
R_1 {R.s:=R_1.s}
R→ ε {R.s:=R.i}

设计递归下降翻译器的方法

- 定义参数和变量
  - 对每个非终结符 A 构造一个函数过程，对 A 的每个继承属性设置一个形式参数
  - 函数的返回值为 A 的综合属性
    - 作为记录，或指向记录的一个指针，记录中有若干域，每个属性对应一个域）
    - 为了简单，我们假设每个非终结只有一个综合属性
  - A 对应的函数过程中，为出现在 A 的产生式中的每一个文法符号的每一个属性都设置一个局部变量

设计递归下降翻译器的方法

- 函数的功能
  - 非终结符 A 对应的函数过程中，根据当前的输入符号决定使用哪个产生式候选

设计递归下降翻译器的方法

- 每个产生式对应的程序代码
  - 按照从左到右的次序，对于单词符号（终结符）、非终结符和语义动作分别作以下工作
    - 对于带有综合属性 x 的终结符 X，把 x 的值存入为 X.x 设置的变量中。然后产生一个匹配 X 的调用，并继续读入一个输入符号。
    - 对于每个非终结符 B，产生一个右边带有函数调用的赋值语句 c=B(b1,b2,…,bk)，其中，b1,b2,…,bk 是为 B 的继承属性设置的变量，c 是为 B 的综合属性设置的变量。
    - 对于语义动作，把动作的代码抄进分析器中，用代表属性的变量来代替对属性的每一次引用。

构造抽象语法树的属性文法定义转化成翻译模式

E →	T	{R.i:=T.nptr}
	R	{E.nptr:=R.s}
R →	+	
	T	{R_1.i:=mknode(‘+’ , R.i, T.nptr)}
	R_1	{R.s:=R_1.s}
R →	-	
	T	{R_1.i:=mknode(‘ – ’ ,R.i, T.nptr)}
	R_1	{R.s:=R.s}
R →	ε	{R.s:=R.i}
T →	(E)	{T.nptr:=E.nptr}
T →	id	{T.nptr:=mkleaf(id, id.entry)}
T →	num	{T.nptr:=mkleaf(num, num.val)}

- 非终结符 E 、 R 、 T 的函数及其参数类型

  function E:↑AST-node;

  function R(in:↑AST-node): ↑AST-node;

  function T: ↑AST-node;

- 用 addop 代表 + 和 –

R → addop

T {R_1.i := mknode(addop.lexme, R.i,T.nptr) }

R_1 {R.s:=R_1.s}

R→ ε { R.s:=R.i }

产生式 R→addop TR|ε 的分析过程

```
procedure R;
 begin
   if sym=addop then begin
         advance;T; R
       end
    else begin /*do nothing*/
       end
end;
```

R → addop
T {R_1.i:= mknode(addop.lexme, R.i, T.nptr)}
R_1 {R.s:=R_1.s}
R→ ε {R.s:=R.i}

```
function R (in:↑AST-node): ↑AST-node;
  var nptr, i1,s1,s: ↑AST-node;
  addoplexeme: char;
begin
   if sym=addop then begin
        /* 产生式       R→addop TR */
        addoplexeme:=lexval;
        advance;
        nptr:=T;
        i1:=mknode (addoplexeme, in, nptr);
        s1:=R (i1)
        s:=s1
    end
   else  s:= in;
   return  s
end;
```

小结

- 翻译模式的改造——消除左递归
- 自顶向下翻译——构造递归下降翻译器

作业

- P165 - 11， 12 （选作）

第六章 属性文法和语法制导翻译

- 属性文法与翻译模式
- 基于属性文法的的处理方法
  - 依赖图
  - 树遍历
  - 一遍扫描
- 抽象语法树
- S- 属性文法的自下而上计算
- L- 属性文法和自顶向下翻译