



中间代码生成

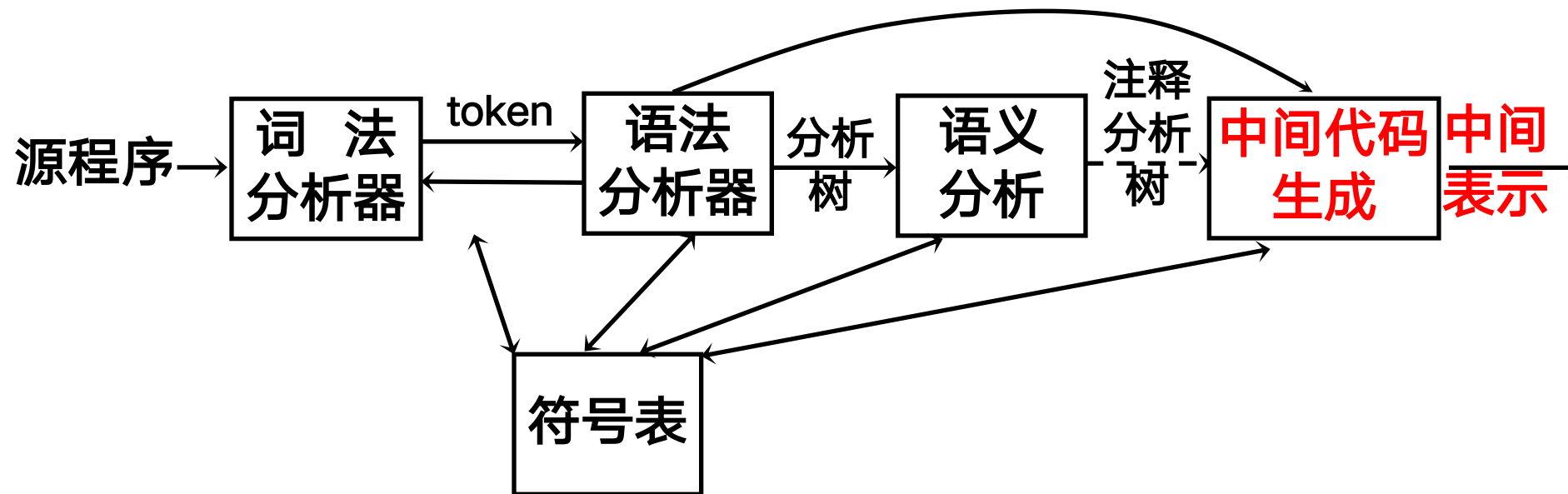
Part7：数组寻址的翻译

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

2025年04月03日



• 数组寻址的翻译

- 数组元素地址的计算
- 数组元素地址计算翻译方案
- 举例说明



- 数组类型的声明

e.g. Pascal的数组声明,

$A : \text{array} [\text{low}_1 \dots \text{high}_1, \dots, \text{low}_n \dots \text{high}_n] \text{ of integer ;}$

数组元素: $A[i, j, k, \dots]$ 或 $A[i][j][k] \dots$

(下界) $\text{low}_1 \leq i \leq \text{high}_1$ (上界) , ...

e.g. C的数组声明,

`int A [100][100][100];`

数组元素: $A[i][30][40] \quad 0 \leq i \leq (100-1)$



- 翻译的主要任务

- 输出(**gen/emit**)地址计算的指令
- “基址[偏移]”相关的中间指令: $t = b[o], b[o] = t$



- 一维数组A的第*i*个元素的地址计算

$$base + (i - low) \times w$$

base: 整个数组的基地址, 也是分配给该数组的内存块的相对地址

low: 下标的下界

w: 每个数组元素的宽度



数组元素的地址计算



- 一维数组A的第*i*个元素的地址计算

$$base + (i - low) \times w$$

base: 整个数组的基地址, 也是分配给该数组的内存块的相对地址

low: 下标的下界

w: 每个数组元素的宽度

可以变换成

$$i \times w + (base - low \times w)$$

low × *w* 是常量, 编译时计算, 减少了运行时计算



数组元素的地址计算



- 二维数组

A: array[1..2, 1..3] of T

- ❖ 列为主

A[1, 1], A[2, 1], A[1, 2], A[2, 2], A[1, 3], A[2, 3]

- ❖ 行为主

A[1, 1], A[1, 2], A[1, 3], A[2, 1], A[2, 2], A[2, 3]



数组元素的地址计算



• 二维数组

A: array[1..2, 1..3] of T

❖ 列为主

A[1, 1], A[2, 1], A[1, 2], A[2, 2], A[1, 3], A[2, 3]

❖ 行为主

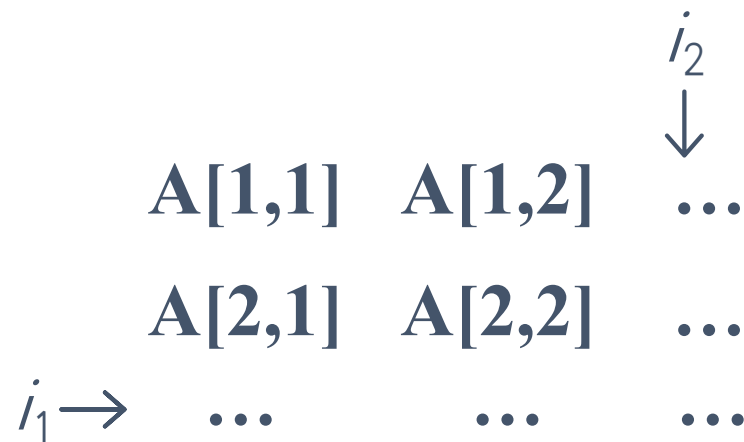
A[1, 1], A[1, 2], A[1, 3], A[2, 1], A[2, 2], A[2, 3]

$base + ((i_1 - low_1) \times n_2 + (i_2 - low_2)) \times w$

(A[i₁, i₂])的地址, 其中 $n_2 = high_2 - low_2 + 1$

变换成 $((i_1 \times n_2) + i_2) \times w +$

$(base - ((low_1 \times n_2) + low_2) \times w)$





数组元素的地址计算



- 多维数组下标变量 $A[i_1, i_2, \dots, i_k]$ 的地址表达式

- 以行为主

$$((\dots ((i_1 \times n_2 + i_2) \times n_3 + i_3) \dots) \times n_k + i_k) \times w$$

$$+ \textit{base} - ((\dots ((\textit{low}_1 \times n_2 + \textit{low}_2) \times n_3 + \textit{low}_3) \dots) \times n_k + \textit{low}_k) \times w$$



数组元素的地址计算



- 多维数组下标变量 $A[i_1, i_2, \dots, i_k]$ 的地址表达式

- 以行为主

$$((\dots ((i_1 \times n_2 + i_2) \times n_3 + i_3) \dots) \times n_k + i_k) \times w$$

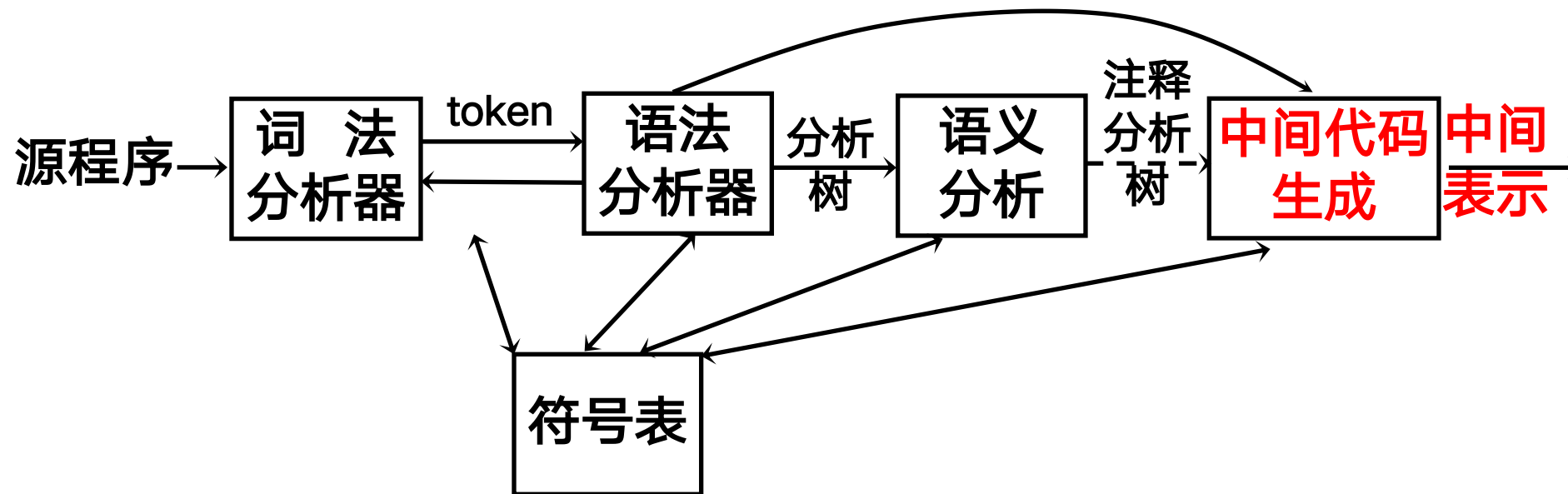
$$+ base - ((\dots ((low_1 \times n_2 + low_2) \times n_3 + low_3) \dots) \times n_k + low_k) \times w$$

红色部分是数组
访问翻译中的最
重要的内容

递推公式:

$$e_1 = i_1$$

$$e_m = e_{m-1} \times n_m + i_m$$



• 数组寻址的翻译

- 数组元素地址的计算
- 数组元素地址计算翻译方案
- 举例说明



- 下标变量访问的产生式

$$S \rightarrow L := E$$
$$L \rightarrow \text{id} [Elist] \mid \text{id}$$
$$Elist \rightarrow Elist, E \mid E$$
$$E \rightarrow L \mid \dots$$

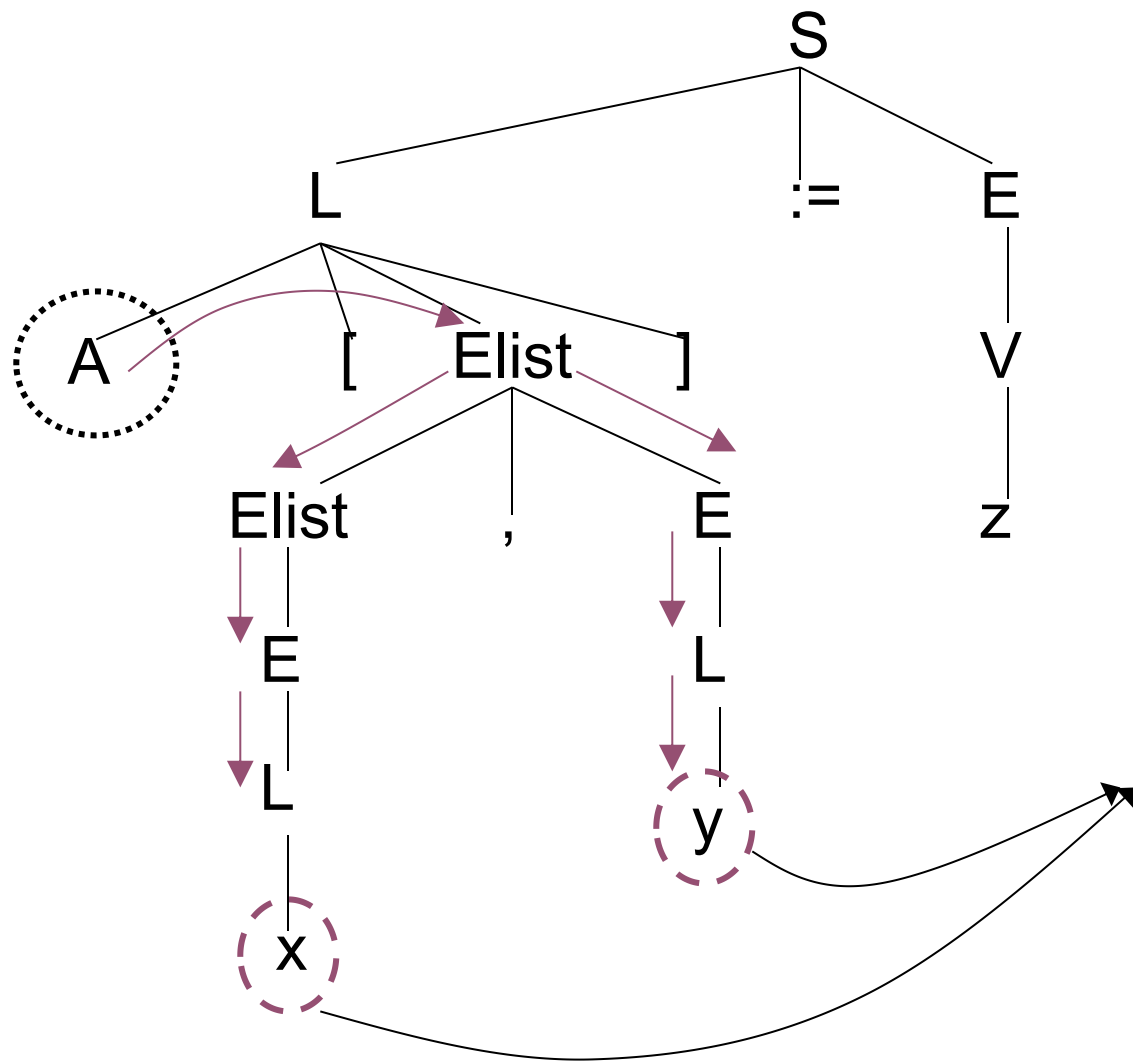
- 采用语法制导的翻译方案时存在的问题

$$Elist \rightarrow Elist, E \mid E$$

由Elist的结构只能得到各维的下标值，但无法获得数组的信息
(如各维的长度)



$A[x,y] := z$ 的分析树



当分析到下标（表达式） x 和 y 时，要计算地址中的“可变部分”。这时需要知晓数组 A 的有关属性，如 n_m ，类型宽度 w 等，而这些信息存于在结点 A 处。若想使用必须定义有关继承属性来传递之。

但在移进－归约分析不适合继承属性的计算！



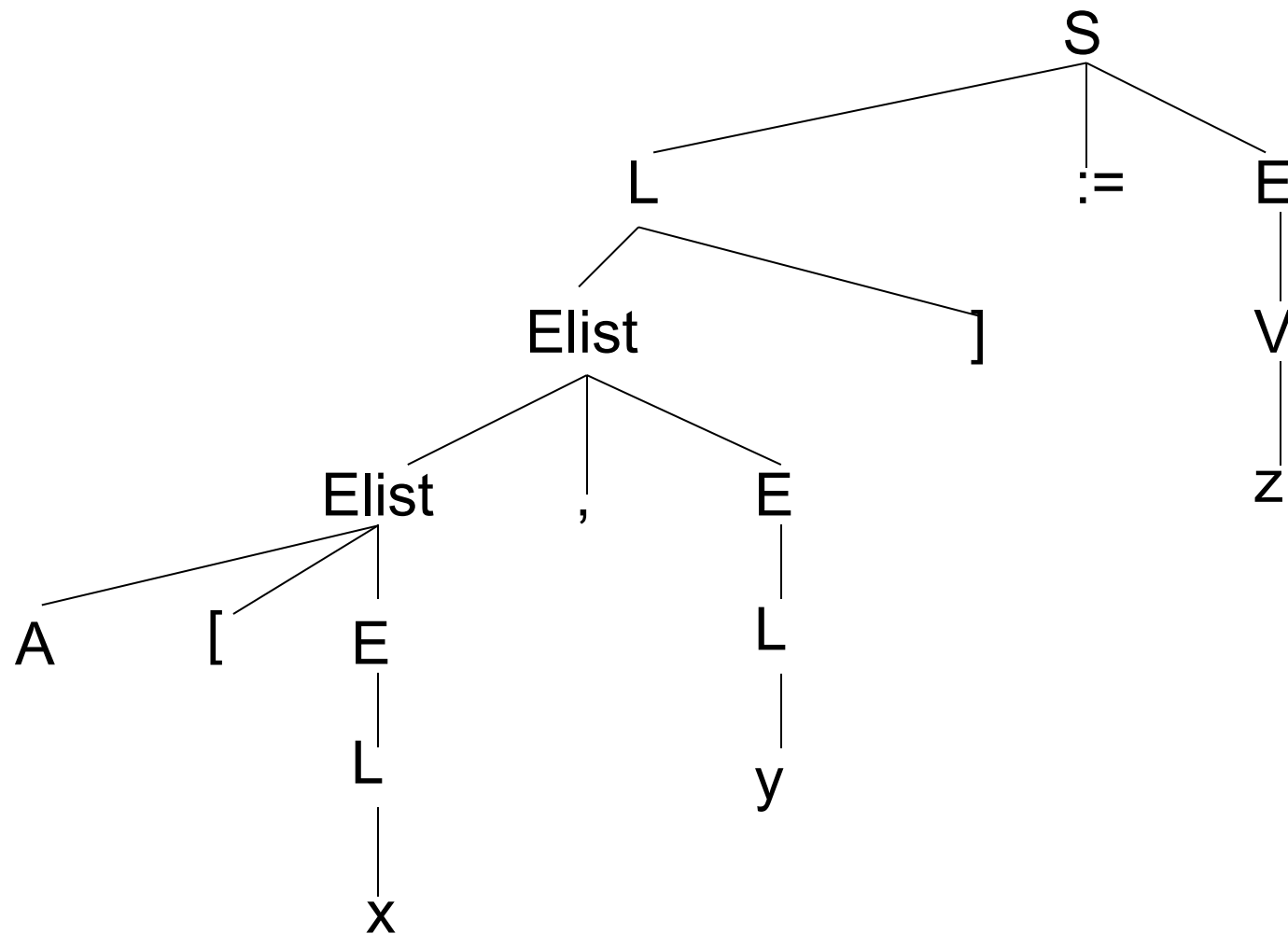
- 所有产生式

$$S \rightarrow L := E$$
$$E \rightarrow E + E$$
$$E \rightarrow (E)$$
$$E \rightarrow L$$
$$L \rightarrow Elist]$$
$$L \rightarrow id$$
$$Elist \rightarrow Elist, E$$
$$Elist \rightarrow id [E$$

修改文法，使数组名id成为Elist的子结点（类似于前面的类型声明），从而避免继承属性的出现



A[x,y] := z的分析树





相关符号属性定义：



L.place, L.offset :

- 若L是简单变量，L.place为其“值”的存放场所，而L.offset为空（null）；
- 当L表示数组元素时，L.place是其地址的“常量值”部分；而此时L.offset为数组元素地址中可变部分的“值”存放场所，数组元素的表示为：**L.place [L.offset]**



相关符号属性定义:



- 属性信息

Elist.place : 存放“可变部分”值(下标计算值)的地址

Elist.array : 数组名条目的指针, 比如可以查询base

Elist.ndim : 当前处理的维数

- 辅助函数:

limit(array, j) : 第j维的大小

width(array) : 数组元素的宽度

invariant(array) : 静态可计算的值, 即紫书7.4公式



- 翻译时重点关注三个表达式：

- $Elist \rightarrow id [E : \text{ 计算第1维}$
- $Elist \rightarrow Elist_1, E : \text{ 传递信息}$
- $L \rightarrow Elist] : \text{ 计算最终结果}$



数组元素的翻译



```
S → L := E {if L.offset == null then /* L是简单变量 */  
                gen (L.place, '=', E.place)  
            else  
                /*取数组元素的左值*/  
                gen (L.place , '[', L.offset, ']', '=', E.place) }
```



数组元素的翻译



```
Elist → id [ E {Elist.place = E.place;  
    /*第一维下标*/  
    Elist.ndim = 1;  
    Elist.array = id.place }
```



数组元素的翻译



```
Elist → Elist1, E {/*维度增加1*/  
    m = Elist1.ndim + 1;  
    /* 第m维的大小*/  
    nm = limit(Elist1.array, m);  
    t = newTemp();  
    /*计算公式7.6 em-1 * nm*/  
    gen(t, '=', Elist1.place, '*', nm);  
    /*计算公式7.6 em = em-1 * nm + im*/  
    gen (t, '=', t, '+', E.place);  
    Elist.array = Elist1.array;  
    Elist.place = t;  
    Elist.ndim = m}
```



$L \rightarrow Elist \mid \{ L.place = newTemp();$

*/*获取数组元素地址的常量值*/*

$gen (L.place, '=', invariant (Elist.array));$

$L.offset = newTemp();$

*/*获取数组元素地址的可变部分*/*

$gen (L.offset, '=', Elist.place, '*', width(Elist.array))\}$



数组元素的翻译



$L \rightarrow \text{id} \{L.place = \text{id}.place; L.offset = \text{null} \}$

$E \rightarrow L \{ \text{if } L.offset == \text{null} \text{ then } /* L是简单变量 */$

$E.place = L.place$

$\text{else begin } E.place = \text{newTemp}();$

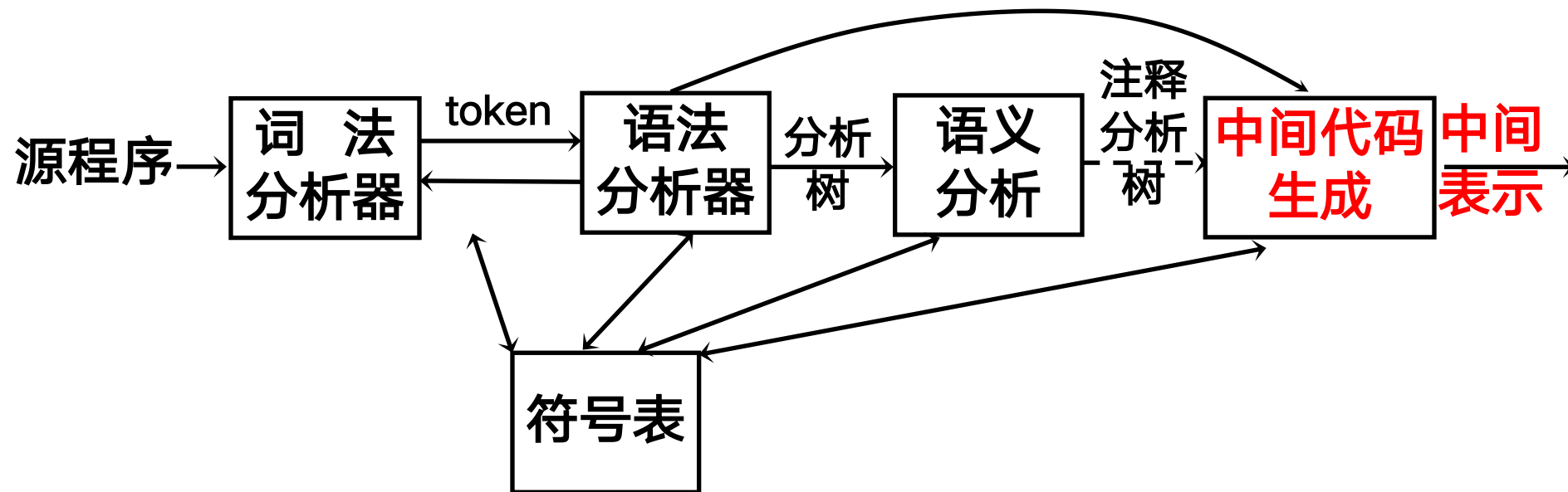
$\text{gen } (E.place, '=', L.place, '[', L.offset, ']') \text{ end } \}$

$E \rightarrow E_1 + E_2 \{ E.place = \text{newTemp}();$

$\text{gen } (E.place, '=', E_1.place, '+', E_2.place) \}$

$E \rightarrow (E_1) \{ E.place = E_1.place \}$

其他翻译同前



• 数组寻址的翻译

- 数组元素地址的计算
- 数组元素地址计算翻译方案
- 举例说明



数组元素的翻译-举例



- 数组A的定义为: $A[1...10, 1...20]$ of integer
- 数组的下界为1, 即low为1
- 为赋值语句 $x := A[y, z]$ 生成中间代码



举例: $x := A[y, z]$



$L.place = x$
 $L.offset = \text{null}$
|
 x

$A[1...10, 1...20]$ of integer



举例: $x := A[y, z]$



$L.place = x$
 $L.offset = \text{null}$
|
 x
 $:=$

$A[1...10, 1...20]$ of integer



举例: $x := A[y, z]$



$L.place = x$
 $L.offset = \text{null}$
|
 x

$:=$

A $[$

$E.place = y$
|
 $L.place = y$
 $L.offset = \text{null}$
|
 y

$A[1...10, 1...20]$ of integer

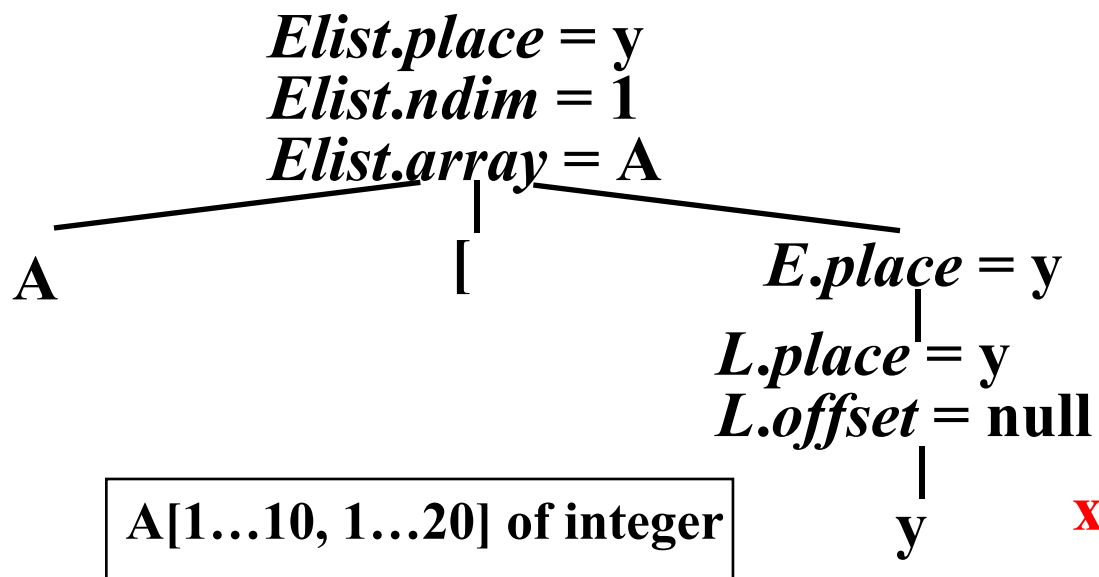


举例: $x := A[y, z]$



$L.place = x$
 $L.offset = \text{null}$
|
 x

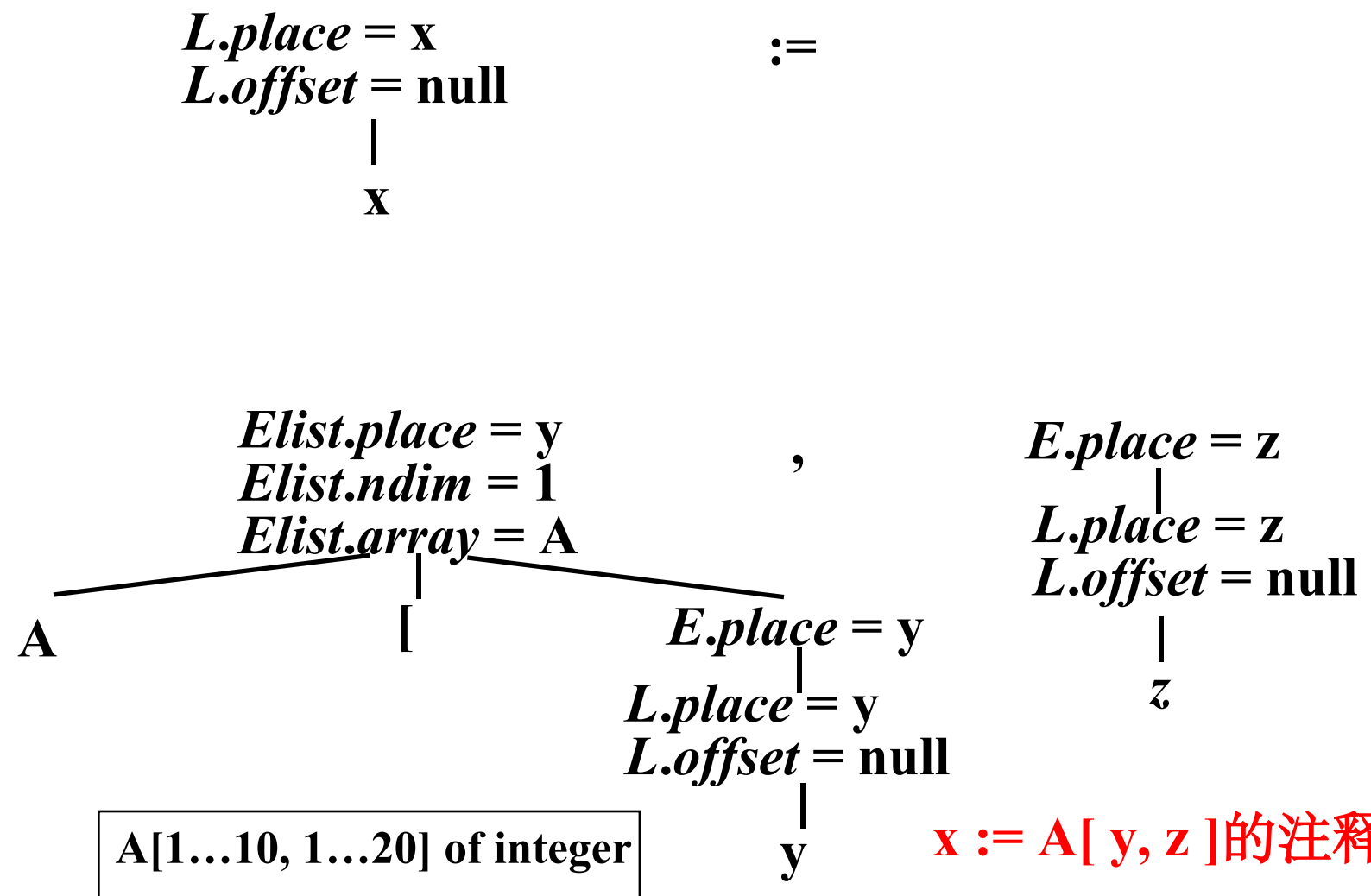
$:=$



$x := A[y, z]$ 的注释分析树



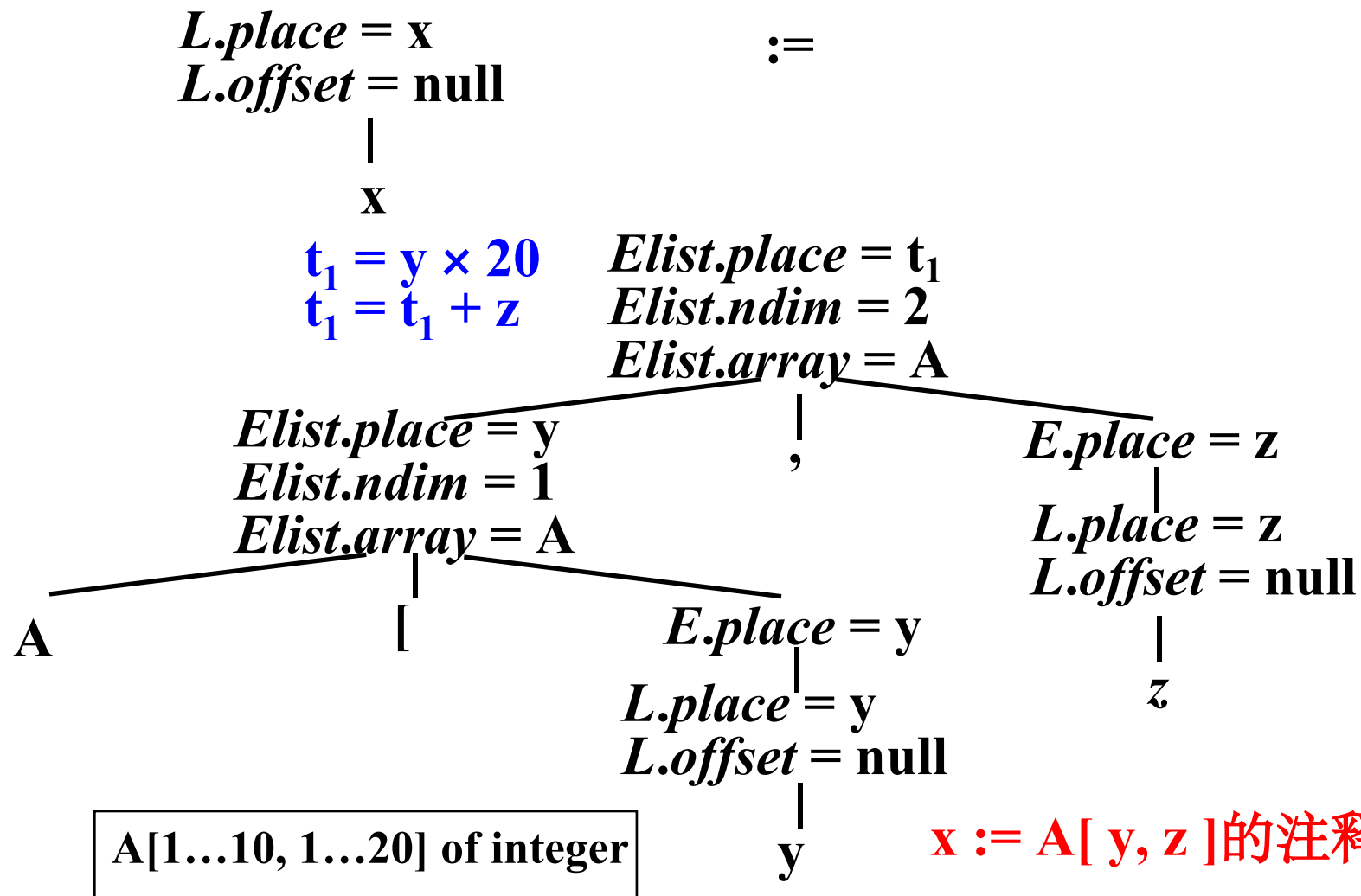
举例: $x := A[y, z]$



$x := A[y, z]$ 的注释分析树



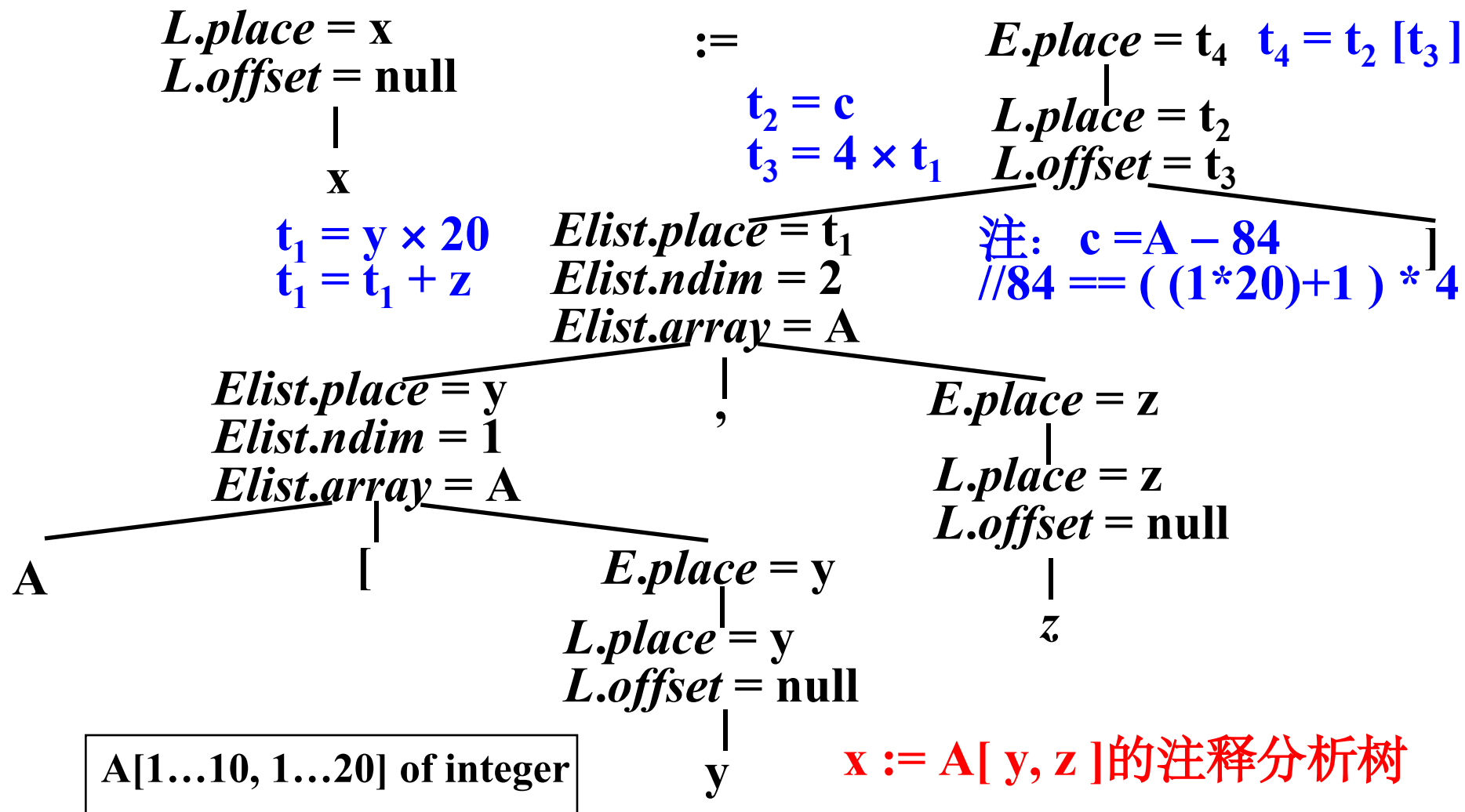
举例: $x := A[y, z]$



$x := A[y, z]$ 的注释分析树

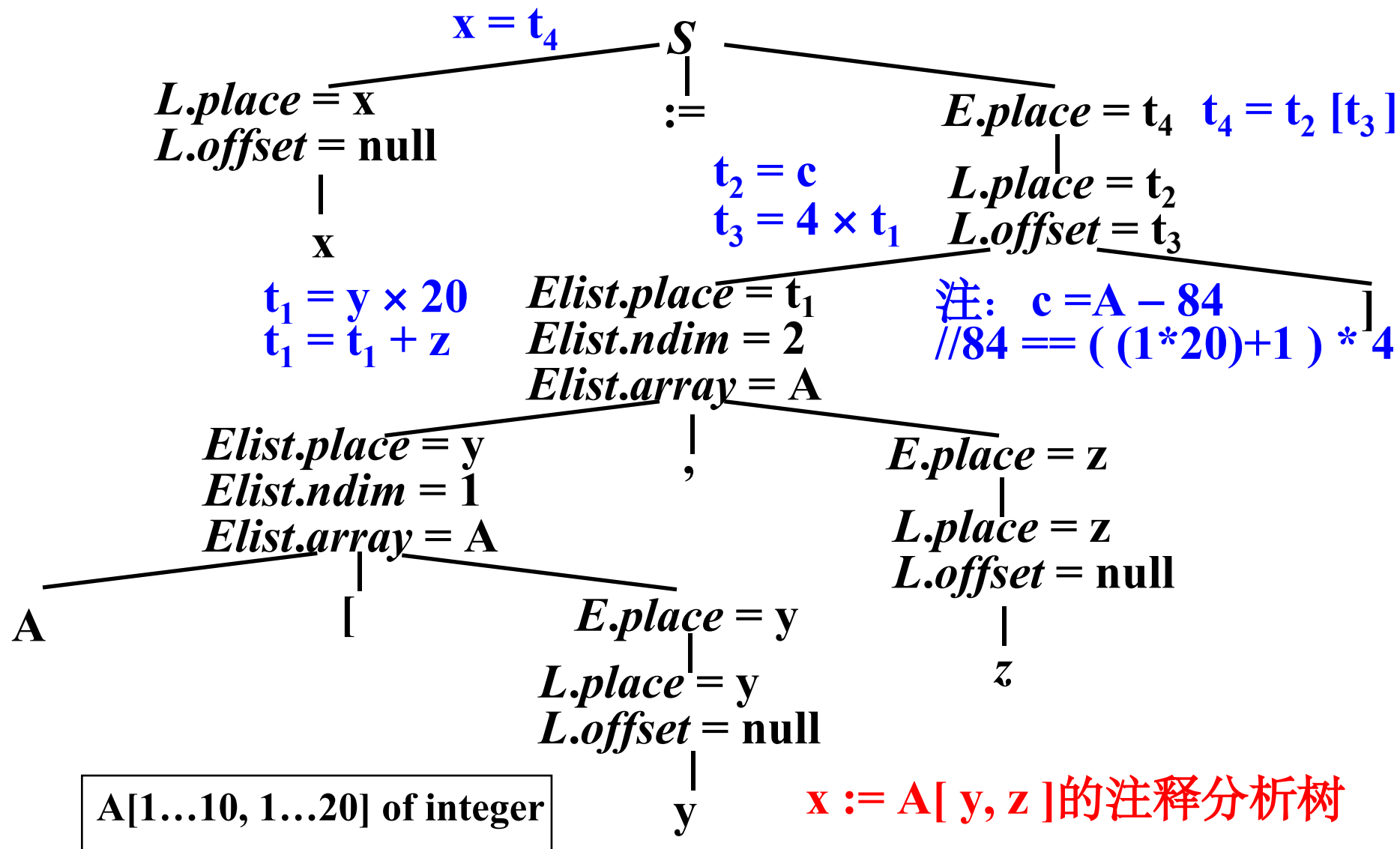


举例: $x := A[y, z]$





举例: $x := A[y, z]$



$x := A[y, z]$ 的注释分析树



举例: $A[i, j] := B[i, j] * k$



- 数组A: $A[1..10, 1..20]$ of integer;

数组B: $B[1..10, 1..20]$ of integer;

$w : 4$ (integer)

- TAC如下:

(1) $t_1 := i * 20$

(2) $t_1 := t_1 + j$

(3) $t_2 := A - 84 \quad // \quad 84 == ((1 * 20) + 1) * 4$

(4) $t_3 := t_1 * 4 \quad // \text{ 以上 } A[i, j] \text{ 的 (左值) 翻译}$



举例: $A[i, j] := B[i, j] * k$



TAC如下 (续) :

(5) $t_4 := i * 20$

(6) $t_4 := t_4 + j$

(7) $t_5 := B - 84$

(8) $t_6 := t_4 * 4$

(9) $t_7 := t_5[t_6]$

//以上计算 $B[i, j]$ 的右值

TAC如下 (续) :

(10) $t_8 := t_7 * k$

//以上整个右值表达

//式计算完毕

(11) $t_2[t_3] := t_8$

// 完成数组元素的赋值

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

2025年04月03日