



中间代码生成

Part5: 类型表达式及自动构造

李 诚

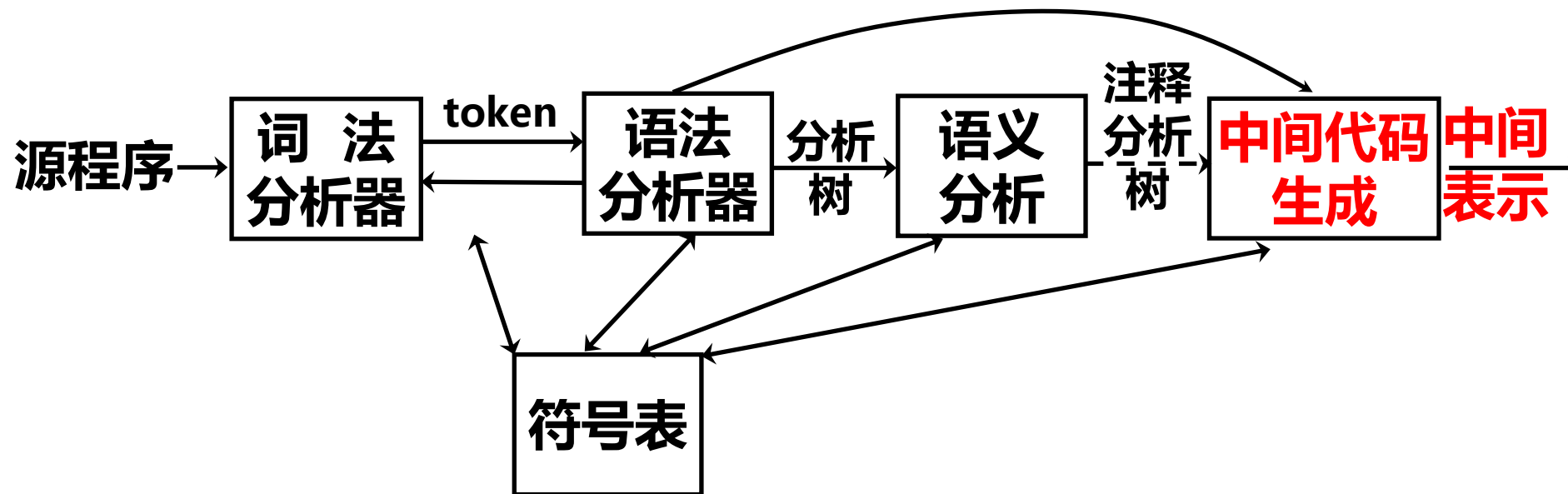
国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2023年11月01日



本节提纲



- 类型表达式
- 构造类型表达式的语法制导定义SDD
- 构造类型表达式的语法制导翻译SDT



- 类型可以是语法的一部分，因此也是结构的

考虑以下文法，**D代表声明语句**，S代表一般语句

$$P \rightarrow D ; S$$
$$D \rightarrow D ; D \mid \text{id} : T$$
$$T \rightarrow \text{boolean} \mid \text{integer} \mid \text{array} [\text{num}] \text{ of } T \mid \uparrow T \mid T \xrightarrow{\text{blue}} T$$



- 类型可以是语法的一部分，因此也是结构的

考虑以下文法，**D代表声明语句**，S代表一般语句

$$P \rightarrow D ; S$$
$$D \rightarrow D ; D \mid \text{id} : T$$
$$T \rightarrow \text{boolean} \mid \text{integer} \mid \text{array} [\text{num}] \text{ of } T \mid \uparrow T \mid T \xrightarrow{\text{blue}} T$$

数组

指针

函数

基本类型

复杂且可组合的类型



- 基本类型是类型表达式

- *integer*

- *real*

- *char*

- *boolean*

- *type_error* // 出错类型

在类型检查中
传递错误

- *void* // 无类型

语句的类型



类型表达式 (Type expression)



- 基本类型是类型表达式
- 可为类型表达式命名，**类名**也是类型表达式



类型表达式 (Type expression)



- 基本类型是类型表达式
- 可为类型表达式命名，**类名**也是类型表达式
- 将**类型构造算子**(type constructor)作用于类型表达式可以构成新的类型表达式
 - 数组类型构造算子 *array*
 - 如果 T 是类型表达式， N 是一个整数，则 $array(N, T)$ 是类型表达式



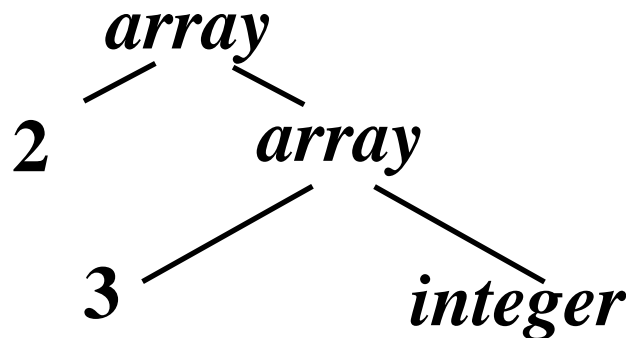
- 基本类型是类型表达式
- 可为类型表达式命名，**类名**也是类型表达式
- 将**类型构造算子**(type constructor)作用于类型表达式可以构成新的类型表达式
 - 数组类型构造算子 *array*
 - 如果T是类型表达式，N是一个整数，则 $array(N, T)$ 是类型表达式

类型	类型表达式
<code>int[3]</code>	<code>array(3, integer)</code>
<code>int[2][3]</code>	<code>array(2, array(3, integer))</code>



- 基本类型是类型表达式
- 可为类型表达式命名, **类名**也是类型表达式
- 将**类型构造算子**(type constructor)作用于类型表达式可以构成新的类型表达式
 - 数组类型构造算子 *array*
 - 如果T是类型表达式, N是一个整数, 则 $array(N, T)$ 是类型表达式

类型	类型表达式
<code>int[3]</code>	$array(3, integer)$
<code>int[2][3]</code>	$array(2, array(3, integer))$





类型表达式 (Type expression)



- 基本类型是类型表达式
- 可为类型表达式命名，**类名**也是类型表达式
- 将**类型构造算子**(type constructor)作用于类型表达式可以构成新的类型表达式

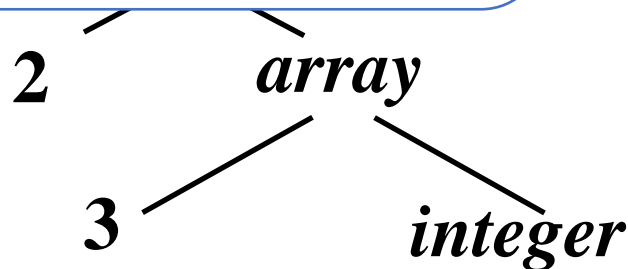
- 数组类型构造算子 *array*

- 如果T是类型表达式，N是一个整数，则 *array*(N, T)

类型	类型表达式
<code>int[3]</code>	<code>array(3, integer)</code>
<code>int[2][3]</code>	<code>array(2, array(3, integer))</code>

也可以写为

array ({0,...,2}, *integer*)
其中{0,...,2}代表索引集合
如首元素索引从1开始，则
为 *array* ({1,...,3}, *integer*)





- 基本类型是类型表达式
- 可为类型表达式命名，类名也是类型表达式
- 将类型构造算子(type constructor)作用于类型表达式可以构成新的类型表达式
 - 数组类型构造算子 *array*
 - 指针类型构造算子 *pointer*
 - 如果T是类型表达式，则 *pointer(T)* 是类型表达式



- 基本类型是类型表达式
- 可为类型表达式命名，类名也是类型表达式
- 将类型构造算子(type constructor)作用于类型表达式可以构成新的类型表达式
 - 数组类型构造算子 *array*
 - 指针类型构造算子 *pointer*
 - 笛卡尔乘积类型构造算子 \times
 - 如果 T_1 和 T_2 是类型表达式，则 $T_1 \times T_2$ 也是类型表达式
 - 主要用于描述列表和元组，如：表示函数的参数



- 基本类型是类型表达式
- 可为类型表达式命名，类名也是类型表达式
- 将类型构造算子(type constructor)作用于类型表达式可以构成新的类型表达式
 - 数组类型构造算子 *array*
 - 指针类型构造算子 *pointer*
 - 笛卡尔乘积类型构造算子 \times
 - 函数类型构造算子 \rightarrow
 - 若 T_1, T_2, \dots, T_n 和 R 是类型表达式，则 $T_1 \times T_2 \times \dots \times T_n \rightarrow R$ 也是

函数参数

函数返回值



- 基本类型是类型表达式
- 可为类型表达式命名，类名也是类型表达式
- 将类型构造算子(type constructor)作用于类型表达式可以构成新的类型表达式
 - 数组类型构造算子 *array*
 - 指针类型构造算子 *pointer*
 - 笛卡尔乘积类型构造算子 \times
 - 函数类型构造算子 \rightarrow
 - 记录类型构造算子 *record*
 - 若有标识符 N_1, N_2, \dots, N_n 以及对应的类型表达式 T_1, T_2, \dots, T_n ，则 $record((N_1 \times T_1) \times (N_2 \times T_2) \times \dots \times (N_n \times T_n))$ 也是类型表达式

记录中的
字段

字段对应的类
型表达式



类型表达式-例1



- 考虑C语言中数组double a[10][20], 写出a、a[0]、a[0][0]的类型表达式



类型表达式-例1



- 考虑C语言中数组double a[10][20], 写出a、a[0]、a[0][0]的类型表达式

a[0][0]: double



类型表达式-例1



- 考虑C语言中数组double a[10][20], 写出a、a[0]、a[0][0]的类型表达式

a[0][0]: double

a[0]: array(20,double);
pointer(double)



类型表达式-例1



- 考虑C语言中数组double a[10][20], 写出a、a[0]、a[0][0]的类型表达式

a[0][0]: double

a[0]: array(20,double);
pointer(double)

a: array(10,array(20,double));
pointer(array(20,double))



类型表达式-例2



- 为row、table和p分别写出类型表达式：

```
typedef struct{  
    int address;  
    char lexeme[15];  
} row;  
row table[101];  
row *p;
```



类型表达式-例2



- 为row、table和p分别写出类型表达式：

```
typedef struct{  
    int address;  
    char lexeme[15];  
} row;  
row table[101];  
row *p;
```

row的类型表达式：

$\text{record}((\text{address} \times \text{integer}) \times (\text{lexeme} \times (\text{array}(15, \text{char}))))$

table的类型表达式：

$\text{array}(101, \text{row})$ //此处row是类型名，因此也是类型表达式

p的类型表达式：

$\text{pointer}(\text{row})$



类型表达式-例3



- 考虑下面的函数f，写出其类型表达式。

`int *f(char a, char b);`

f的类型表达式：

`(char×char) → pointer(integer)`



类型表达式-例3



- 考虑下面的函数f，写出其类型表达式。

```
int *f(char a, char b);
```

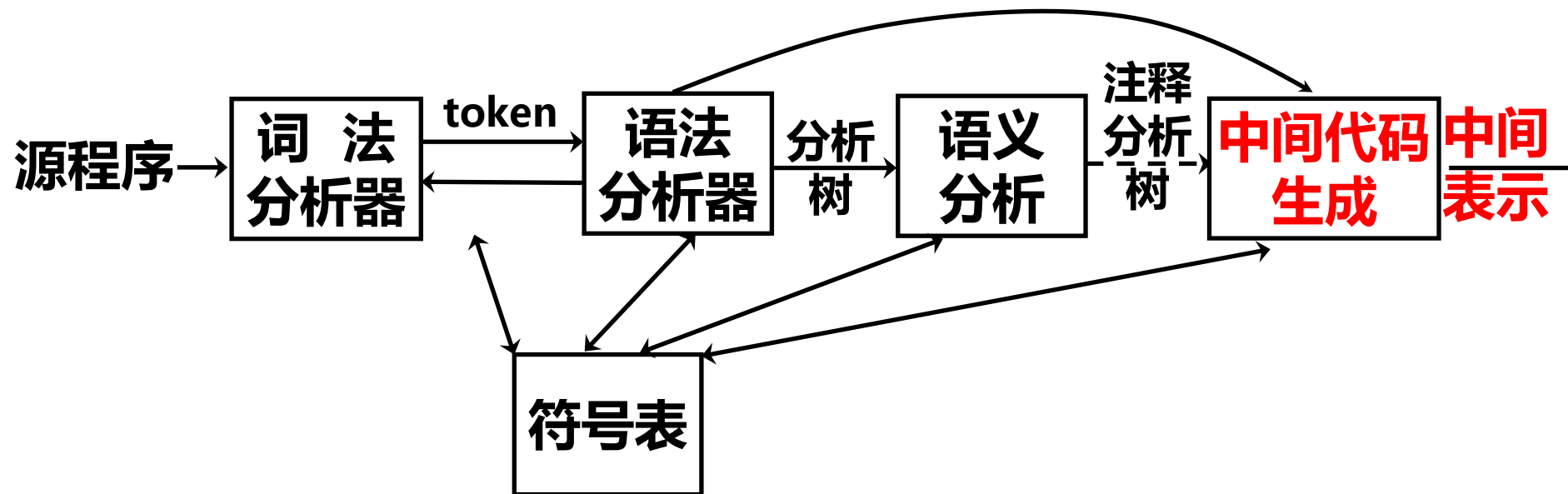
f的类型表达式：

$(\text{char} \times \text{char}) \rightarrow \text{pointer}(\text{integer})$

问题：可否自动化地实现类型表达式的生成？



本节提纲



- 类型表达式
- 构造类型表达式的语法制导定义SDD
- 构造类型表达式的语法制导翻译SDT



- 为以下文法制定构造类型表达式的语义规则

产 生 式	语 义 规 则
$T \rightarrow B C$	
$B \rightarrow \text{int}$	
$B \rightarrow \text{float}$	
$C \rightarrow [\text{num}] C_1$	
$C \rightarrow \varepsilon$	



- 为以下文法制定构造类型表达式的语义规则

产 生 式	语 义 规 则
$T \rightarrow B C$	
$B \rightarrow \text{int}$	
$B \rightarrow \text{float}$	
$C \rightarrow [\text{num}] C_1$	
$C \rightarrow \varepsilon$	

- 为每个文法符号设置**综合属性** t 和**继承属性** b
 - t : 该符号对应的类型表达式
 - b : 将类型信息从左到右传递



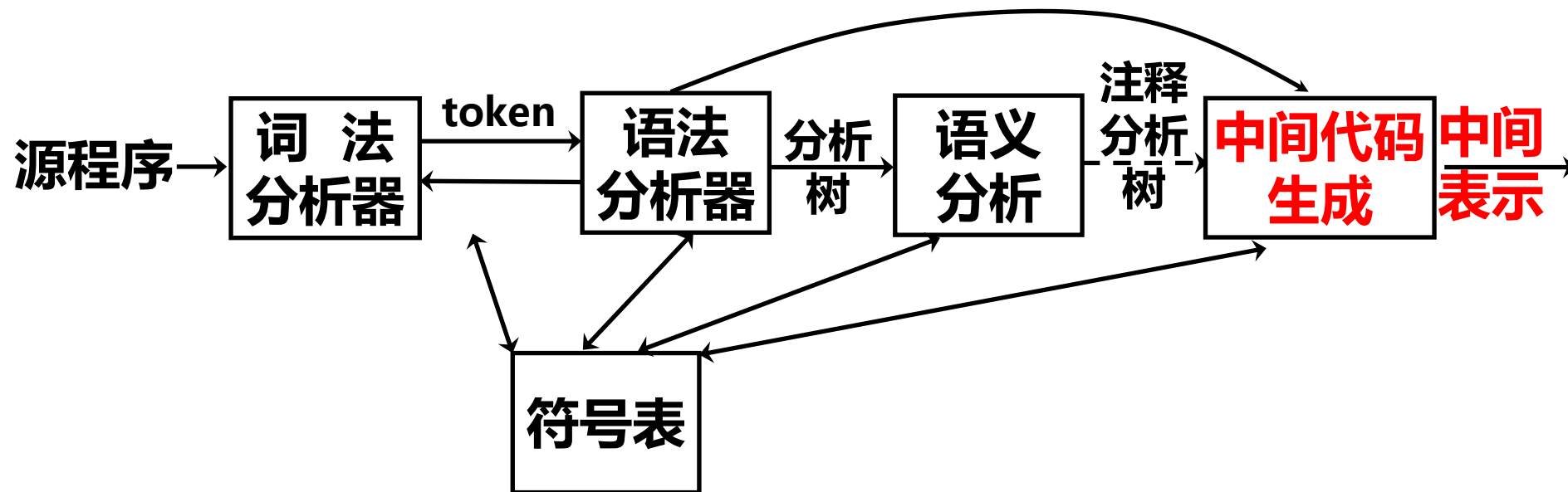
- 为以下文法制定构造类型表达式的语义规则

产 生 式	语 义 规 则
$T \rightarrow B C$	$T.t = C.t; C.b = B.t$
$B \rightarrow \text{int}$	$B.t = \text{integer}$
$B \rightarrow \text{float}$	$B.t = \text{float}$
$C \rightarrow [\text{num}] C_1$	$C.t = \text{array}(\text{num.val}, C_1.t); C_1.b = C.b$
$C \rightarrow \varepsilon$	$C.t = C.b$

- 为每个文法符号设置**综合属性** t 和**继承属性** b
 - t : 该符号对应的类型表达式
 - b : 将类型信息从左到右传递



本节提纲



- 类型表达式
- 构造类型表达式的语法制导定义SDD
- 构造类型表达式的语法制导翻译SDT



- 将SDD改造为SDT

$$T \rightarrow B \{C.b = B.t\} C \{T.t = C.t; \}$$
$$B \rightarrow \text{int} \{B.t = \text{integer}\}$$
$$B \rightarrow \text{float} \{B.t = \text{float}\}$$
$$C \rightarrow [\text{num}] \{C_1.b = C.b\} C_1 \{C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$C \rightarrow \varepsilon \{C.t = C.b\}$$

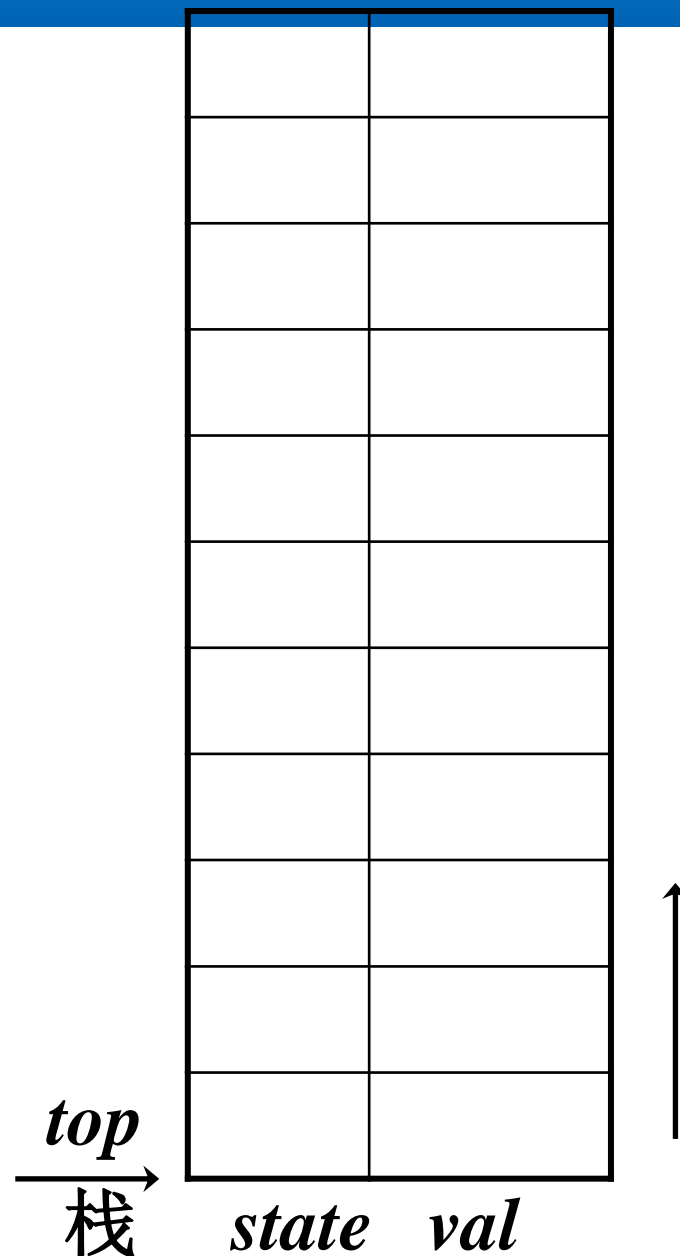
- 但是继承属性的计算与LR分析方法不适配
- 因此，如果要使用LR，就需要改造文法



• 通过改造文法，与LR适配

- 引入标记M，C归约时可在栈顶以下位置找到B.t
- 引入标记N，把继承属性C.b当做综合属性记录

$$T \rightarrow B M C \{T.t = C.t;\}$$
$$M \rightarrow \varepsilon \{M.t = B.t\}$$
$$B \rightarrow \text{int} \{B.t = \text{integer}\}$$
$$B \rightarrow \text{float} \{B.t = \text{float}\}$$
$$C \rightarrow [\text{num}] N C_1 \{C.t = \text{array}(\text{num.val}, C_1.t);\}$$
$$N \rightarrow \varepsilon \{N.t = C.b\}$$
$$C \rightarrow \varepsilon \{C.t = C.b\}$$


$$\begin{aligned} T &\rightarrow B \ M \ C \ \{T.t = C.t;\} \\ M &\rightarrow \varepsilon \ \{M.t = B.t\} \\ B &\rightarrow \text{int} \ \{B.t = \text{integer}\} \\ B &\rightarrow \text{float} \ \{B.t = \text{float}\} \\ C &\rightarrow [\text{num}] \ N \ C_1 \\ &\quad \{C.t = \text{array}(\text{num.val}, C_1.t);\} \\ N &\rightarrow \varepsilon \ \{N.t = C.b\} \\ C &\rightarrow \varepsilon \ \{C.t = C.b\} \end{aligned}$$




• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1$$
$$\{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

top
→

栈

int	

state val





• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1$$
$$\{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

$\xrightarrow{\text{top}}$
栈

B	integer

$\text{state} \quad \text{val}$





• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1 \\ \{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

$\xrightarrow{\text{top}}$

M	integer
B	integer



栈 state val



• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1 \\ \{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

top
→

]	
num	2
[
M	integer
B	integer



栈 *state* *val*



• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1 \\ \{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

top
→

<i>N</i>	<i>integer</i>
]	
num	2
[
<i>M</i>	<i>integer</i>
<i>B</i>	<i>integer</i>



栈 *state* *val*



• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1$$
$$\{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

top
→

]	
num	3
[
N	integer
]	
num	2
[
M	integer
B	integer



栈 state val



• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1$$
$$\{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

top
→

<i>N</i>	<i>integer</i>
]	
num	3
[
<i>N</i>	<i>integer</i>
]	
num	2
[
<i>M</i>	<i>integer</i>
<i>B</i>	<i>integer</i>



栈 *state* *val*



• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{T.t = C.t;\}$$
$$M \rightarrow \varepsilon \{M.t = B.t\}$$
$$B \rightarrow \text{int} \{B.t = \text{integer}\}$$
$$B \rightarrow \text{float} \{B.t = \text{float}\}$$
$$C \rightarrow [\text{num}] N C_1$$
$$\{C.t = \text{array}(\text{num.val}, C_1.t);\}$$
$$N \rightarrow \varepsilon \{N.t = C.b\}$$
$$C \rightarrow \varepsilon \{C.t = C.b\}$$

top
→

<i>C</i>	<i>integer</i>
<i>N</i>	<i>integer</i>
<i>]</i>	
num	3
<i>[</i>	
<i>N</i>	<i>integer</i>
<i>]</i>	
num	2
<i>[</i>	
<i>M</i>	<i>integer</i>
<i>B</i>	<i>integer</i>



栈 *state* *val*



• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1$$
$$\{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

完成第一次归约
 $C \rightarrow [\text{num}] N C_1$

top →

<i>C</i>	<i>array(3, integer)</i>
<i>N</i>	<i>integer</i>
<i>]</i>	
<i>num</i>	<i>2</i>
<i>[</i>	
<i>M</i>	<i>integer</i>
<i>B</i>	<i>integer</i>

栈 *state* *val*



• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1$$
$$\{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

完成第二次归约

$$C \rightarrow [\text{num}] N C_1$$

<i>C</i>	<i>array(2, array(3, integer))</i>
<i>M</i>	<i>integer</i>
<i>B</i>	<i>integer</i>

↑

栈 *state* *val*



• 分析int[2][3]的LR栈操作

$$T \rightarrow B M C \{ T.t = C.t; \}$$
$$M \rightarrow \varepsilon \{ M.t = B.t \}$$
$$B \rightarrow \text{int} \{ B.t = \text{integer} \}$$
$$B \rightarrow \text{float} \{ B.t = \text{float} \}$$
$$C \rightarrow [\text{num}] N C_1$$
$$\{ C.t = \text{array}(\text{num.val}, C_1.t); \}$$
$$N \rightarrow \varepsilon \{ N.t = C.b \}$$
$$C \rightarrow \varepsilon \{ C.t = C.b \}$$

完成第三次归约

$$T \rightarrow B M C$$

top

<i>T</i>	<i>array(2,</i> <i>array(3,i</i> <i>nteger))</i>

栈 *state* *val*

2023年秋季学期 《编译原理和技术》



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

2023年11月01日