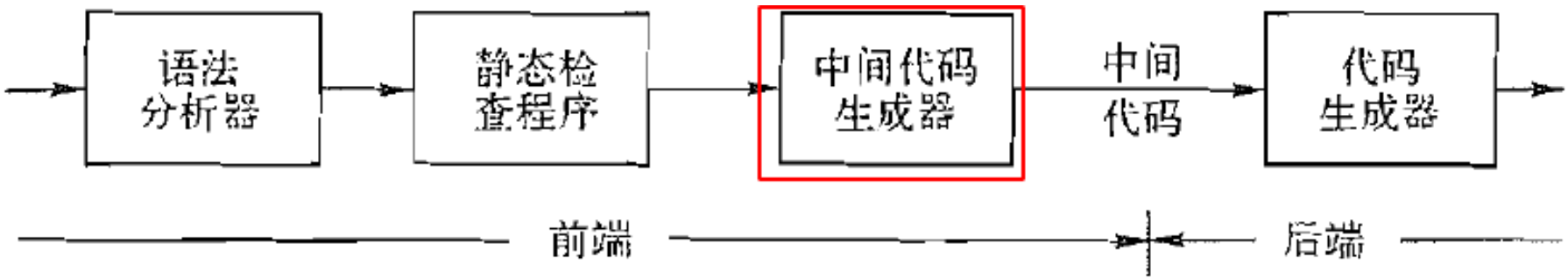
中间代码生成

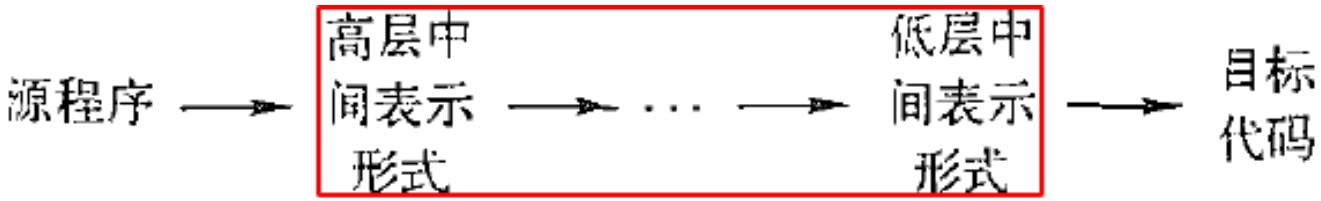
魏恒峰

hfwei@nju.edu.cn

2020 年 12 月 22 日


语法
分析器
静态检
查程序
中间代码
生成器
中间
代码
代码
生成器
前端
后端

Intermediate Representation (IR)



精确: 不能丢失源程序的信息

独立: 不依赖特定的源语言与目标语言

(如, 没有复杂的寻址方式)

Intermediate Representation (IR)

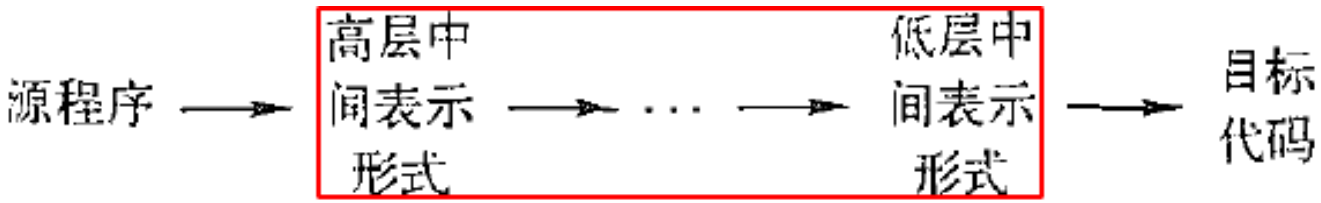
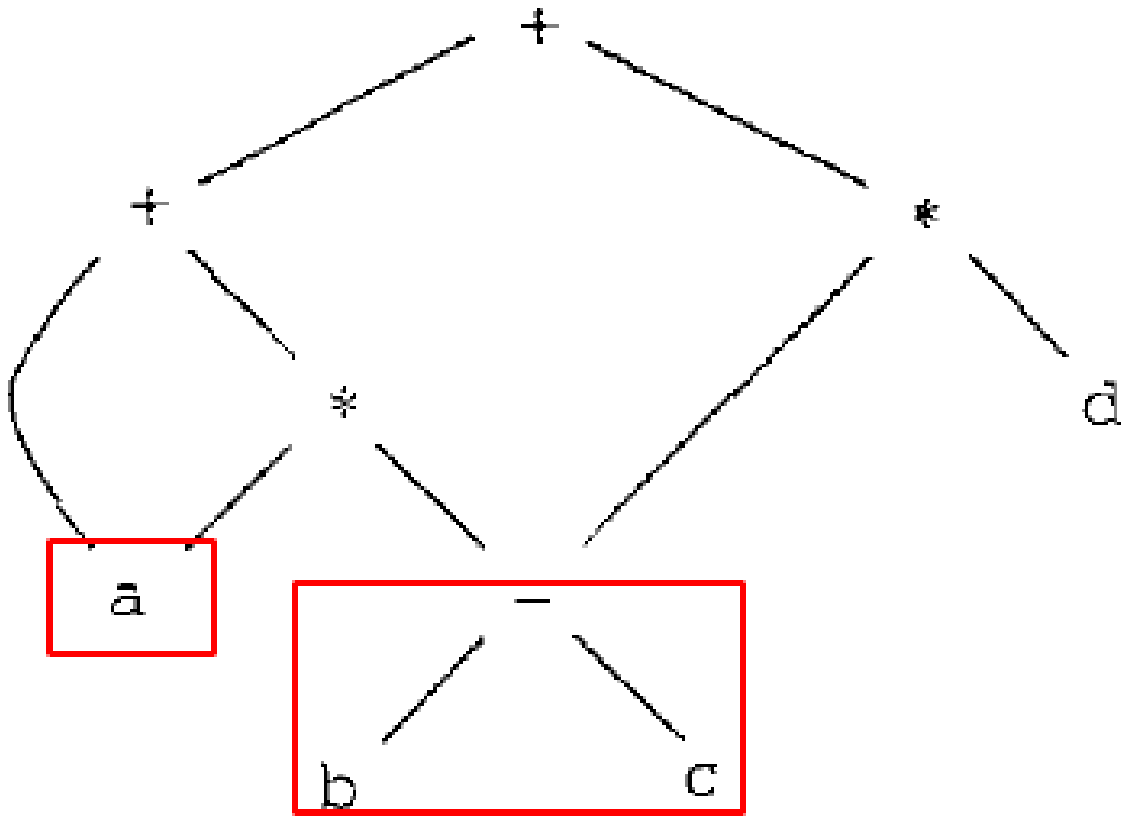


图 (抽象语法树)、**三地址代码**、C 语言

表达式的有向无环图

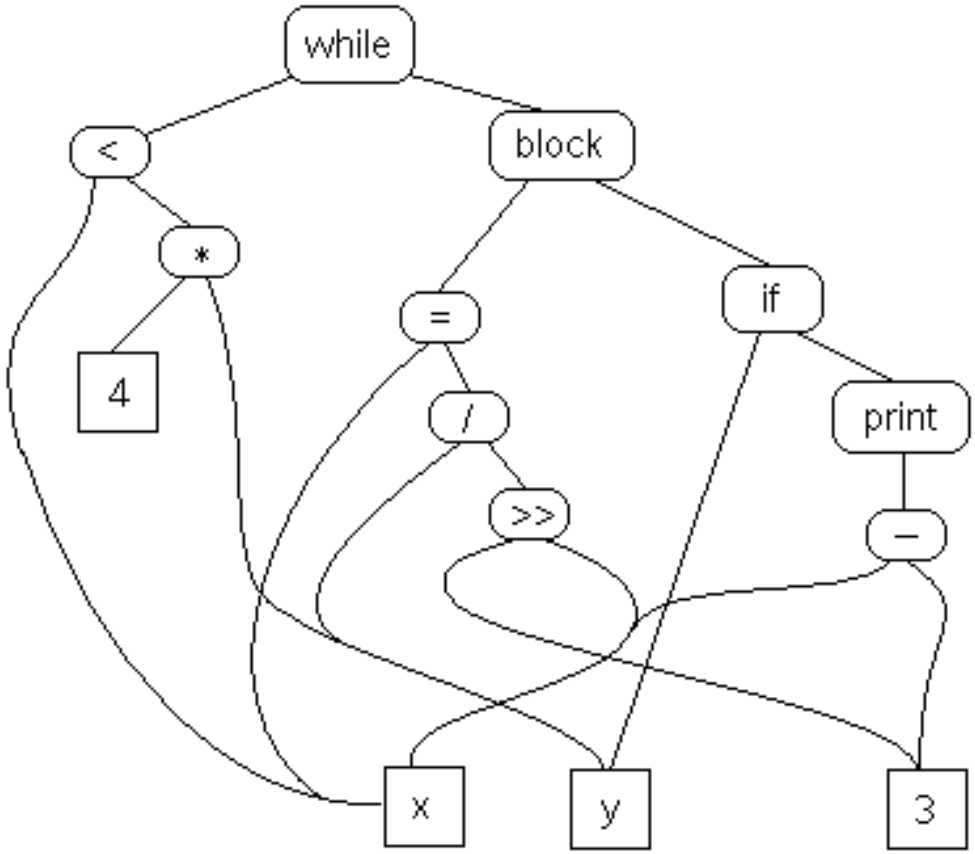


$$a + a * (b - c) + (b - c) * d$$

产生式		语义规则
1)	$E \to E_1 + T$	$E.node =$ **new** $Node('+', E_1.node, T.node)$
2)	$E \to E_1 - T$	$E.node =$ **new** $Node('-', E_1.node, T.node)$
3)	$E \to T$	$E.node = T.node$
	$T \to T_1 * F$	$T.node =$ **new** $Node('*', T_1.node, F.node)$
4)	$T \to (\ E\ )$	$T.node = E.node$
5)	$T \to$ **id**	$T.node =$ **new** $Leaf(\mathbf{id}, \mathbf{id}.entry)$
6)	$T \to$ **num**	$T.node =$ **new** $Leaf(\mathbf{num}, \mathbf{num}.val)$

在创建节点之前, 先判断是否已存在 (哈希表)

```
while (x < 4 * y) {
    x = y / 3 >> x;
    if (y) print x - 3;
}
```



Definition (三地址代码 (Three-Address Code (TAC; 3AC)))

每个 **TAC** 指令**最多**包含三个操作数。

$$x = y \ \mathbf{op} \ z \quad (1)$$

$$x = \mathbf{op} \ y \quad (2)$$

$$x = y \quad (3)$$

$$\mathbf{goto} \ L \quad (4)$$

$$\mathbf{if} \ x \ \mathbf{goto} \ L \quad (5)$$

$$\mathbf{if \ False} \ x \ \mathbf{goto} \ L \quad (6)$$

$$\mathbf{if} \ x \ \mathbf{relop} \ y \ \mathbf{goto} \ L \quad (7)$$

Definition (三地址代码 (Three-Address Code (TAC; 3AC)))

每个 **TAC** 指令**最多**包含三个操作数。

$$\textbf{param}\ x \tag{8}$$
$$\textbf{call}\ p, n \tag{9}$$
$$y = \textbf{call}\ p, n \tag{10}$$
$$\textbf{return}\ y \tag{11}$$

```
param x_1
param x_2
...
param x_n
call p, n
```

$p(x_1, x_2, \ldots, x_n)$

Definition (三地址代码 (Three-Address Code (TAC; 3AC)))

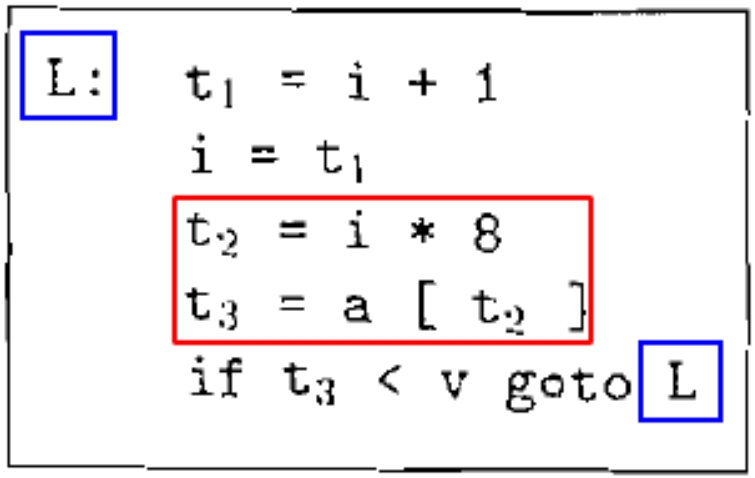
每个 **TAC** 指令**最多**包含三个操作数。

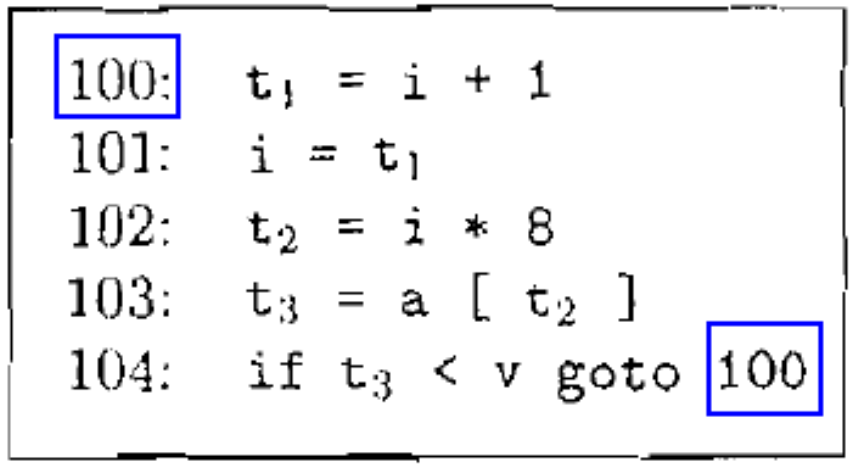
$$x = y[i] \tag{12}$$
$$x[i] = y \tag{13}$$

距离位置 y 处 i 个内存单元

$$x = \&y \tag{14}$$
$$x = *y \tag{15}$$
$$*x = y \tag{16}$$

```
do i = i + 1; while (a[i] < v);
```





三地址代码的四元式表示

Definition (四元式 (Quadruple))

一个四元式包含四个字段, 分别为 op、arg_1、arg_2 与 $result$。

$$a + a * (b - c) + (b - c) * d$$

```
t1 = minus c
t2 = b * t1
t3 = minus c
t4 = b * t3
t5 = t2 + t4
 a = t5
```

	op	arg_1	arg_2	result
0	minus	c		t_1
1	*	b	t_1	t_2
2	minus	c		t_3
3	*	b	t_3	t_4
4	+	t_2	t_4	t_5
5	=	t_5		a
		...		

$x = y[i]$	$=[\,]$	y	i	x
$x[i] = y$	$[\,]=$	i	y	x

$x = \&y$	$=\&$	y	x
$x = *y$	$=*$	y	x
$*x = y$	$*=$	y	x

表达式的中间代码翻译

产生式	语义规则
$S \rightarrow \mathbf{id} = E\ ;$	$S.code = E.code \ \|\|$ $gen(top.get(\mathbf{id}.lexeme)\ '='\ E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \ \|\| \ E_2.code \ \|\|$ $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid\ - E_1$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \ \|\|$ $gen(E.addr\ '='\ '\mathbf{minus}'\ E_1.addr)$
$\mid\ (\ E_1\ )$	$E.addr = E_1.addr$ $E.code = E_1.code$
$\mid\ \mathbf{id}$	$E.addr = top.get(\mathbf{id}.lexeme)$ $E.code = ''$

符号表条目

综合属性 $E.code$ **与** $E.addr$

产生式	语义规则
$S \rightarrow \mathbf{id} = E\ ;$	$S.code = E.code \mid\mid gen(top.get(\mathbf{id}.lexeme)\ '='\ E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \mid\mid E_2.code \mid\mid gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid\ - E_1$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \mid\mid gen(E.addr\ '='\ '\mathbf{minus}'\ E_1.addr)$
$\mid\ (\ E_1\ )$	$E.addr = E_1.addr$ $E.code = E_1.code$
$\mid\ \mathbf{id}$	$E.addr = top.get(\mathbf{id}.lexeme)$ $E.code = ''$

符号表条目

```
t1 = minus c
t2 = b + t1
a = t2
```

$$a = b + -c$$

表达式的中间代码翻译 (增量式)

$S \rightarrow$	id = E ;	{ $gen($ $top.get($id.$lexeme)$ '=' $E.addr)$; }
$E \rightarrow$	$E_1 + E_2$	{ $E.addr =$ **new** $Temp()$; $gen(E.addr$ '=' $E_1.addr$ '+' $E_2.addr)$; }
|	$- E_1$	{ $E.addr =$ **new** $Temp()$; $gen(E.addr$ '=' 'minus' $E_1.addr)$; }
|	(E_1)	{ $E.addr = E_1.addr$; }
|	id	{ $E.addr = top.get($id.$lexeme)$; }

综合属性 $E.addr$

数组引用的中间代码翻译

声明：int $a[2][3]$

数组引用：$x = a[1][2];\ a[1][2] = x$

需要计算 $a[1][2]$ 的相对于**数组基地址** a 的**偏移地址**

数组引用的中间代码翻译

int $a[2][3]$

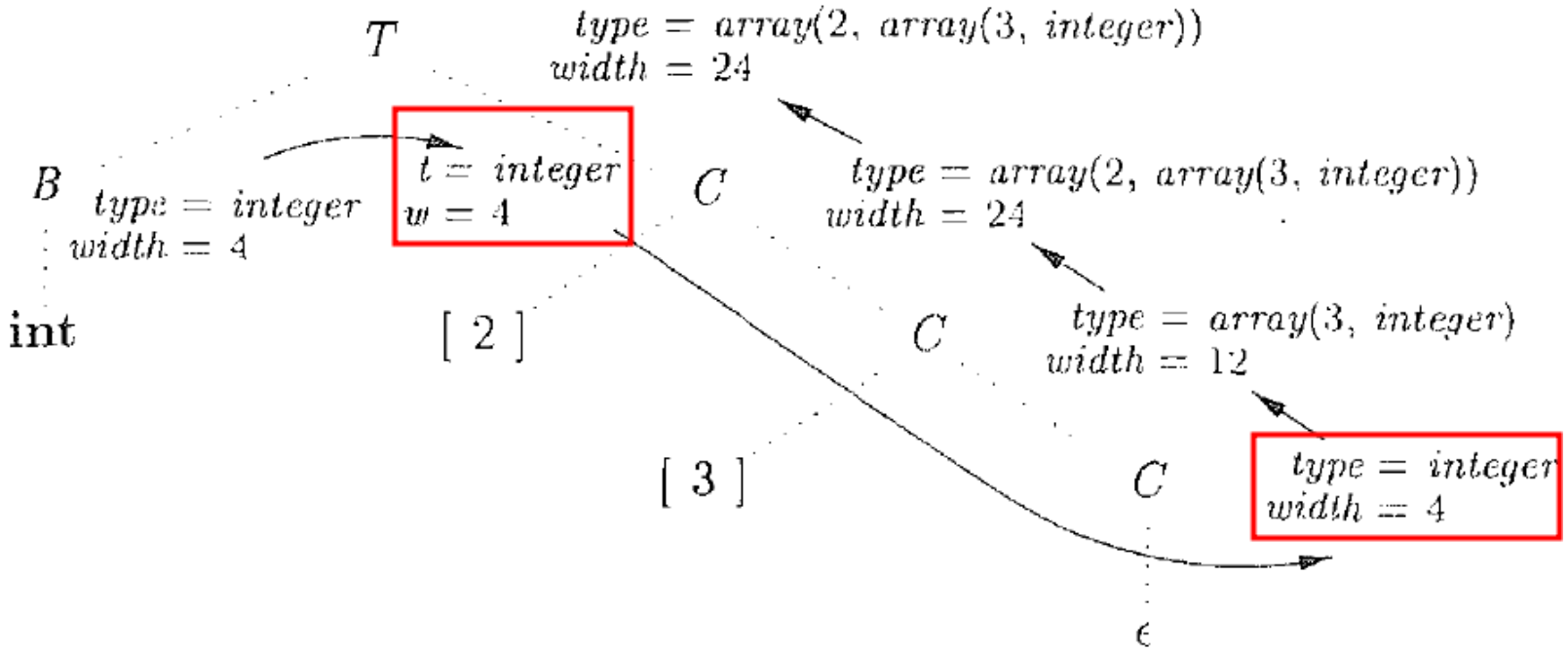


图 6-16　数组类型的语法制导翻译

数组类型声明

int $a[2][3]$

$$array(2, array(3, \text{integer}))$$

	元素类型	元素宽度
$a[\,]$	$array(3, \text{integer})$	12
$a[\,][\,]$	integer	4

$$addr(a[1][2]) = base + 1 \times 12 + 2 \times 4$$

$S \rightarrow$	**id** = E ;	{ $gen($ $top.get($**id**$.lexeme)$ $'='$ $E.addr)$; }
	L = E ;	{ $gen(L.array.base$ $'['$ $L.addr$ $']'$ $'='$ $E.addr)$; }
$E \rightarrow$	E_1 + E_2	{ $E.addr =$ **new** $Temp()$; $gen(E.addr$ $'='$ $E_1.addr$ $'+'$ $E_2.addr)$; }
	id	{ $E.addr = top.get($**id**$.lexeme)$; }
	L	{ $E.addr =$ **new** $Temp()$; $gen(E.addr$ $'='$ $L.array.base$ $'['$ $L.addr$ $']')$; }
$L \rightarrow$	**id** [E]	{ $L.array = top.get($**id**$.lexeme)$; $L.type = L.array.type.elem$; $L.addr =$ **new** $Temp()$; $gen(L.addr$ $'='$ $E.addr$ $'*'$ $L.type.width)$; }
	L_1 [E]	{ $L.array = L_1.array$; $L.type = L_1.type.elem$; $t =$ **new** $Temp()$; $L.addr =$ **new** $Temp()$; $gen(t$ $'='$ $E.addr$ $'*'$ $L.type.width)$; $gen(L.addr$ $'='$ $L_1.addr$ $'+'$ $t)$; }

int $a[2][3]$

综合属性 $L.array.base$：数组基地址（即，数组名）

$$
\begin{array}{lcll}
S & \rightarrow & \mathbf{id} = E\ ; & \{\ gen(\ top.get(\mathbf{id}.lexeme)\ '='\ E.addr);\ \} \\
 & | & L = E\ ; & \{\ gen(L.array.base\ '['\ L.addr\ ']'\ '='\ E.addr); \\
E & \rightarrow & E_1 + E_2 & \{\ E.addr = \mathbf{new}\ Temp\,();\ gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr);\ \} \\
 & | & \mathbf{id} & \{\ E.addr = top.get(\mathbf{id}.lexeme);\ \} \\
 & | & L & \{\ E.addr = \mathbf{new}\ Temp\,();\ gen(E.addr\ '='\ L.array.base\ '['\ L.addr\ ']');\ \}
\end{array}
$$

综合属性 $L.addr$：偏移地址

综合属性 $L.array$：数组名 id对应的符号表条目

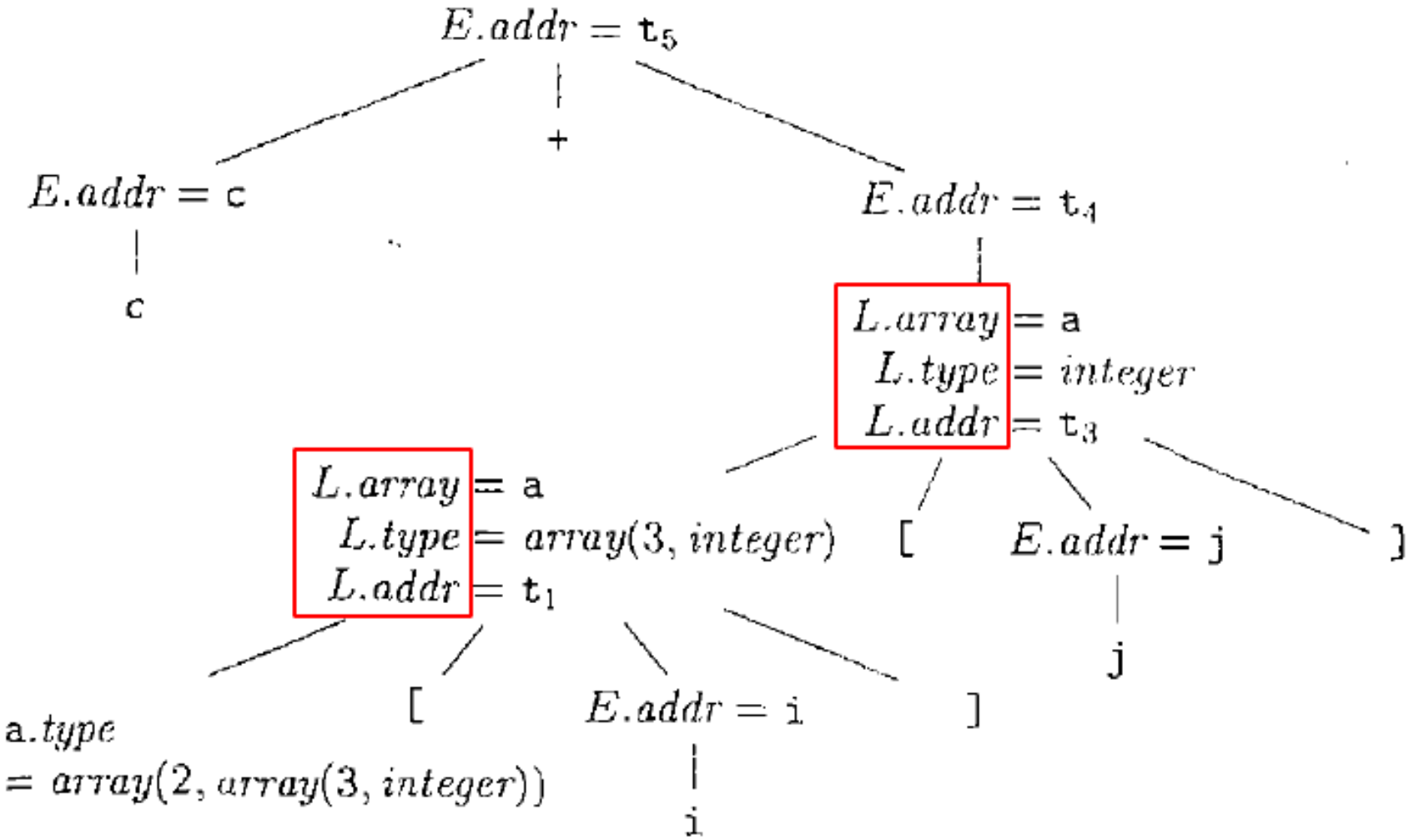
$$
\begin{aligned}
L \rightarrow\ & \mathbf{id}\ [\ E\ ] \quad \{\ L.array = top.get(\mathbf{id}.lexeme);\\
& \quad L.type = L.array.type.elem;\\
& \quad L.addr = \mathbf{new}\ Temp\,();\\
& \quad gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \}\\
\mid\ & L_1\ [\ E\ ] \quad \{\ L.array = L_1.array;\\
& \quad L.type = L_1.type.elem;\\
& \quad t = \mathbf{new}\ Temp\,();\\
& \quad L.addr = \mathbf{new}\ Temp\,();\\
& \quad gen(t\ '='\ E.addr\ '*'\ L.type.width);\\
& \quad gen(L.addr\ '='\ L_1.addr\ '+'\ t);\ \}
\end{aligned}
$$

综合属性 $L.type$：(当前) 元素类型

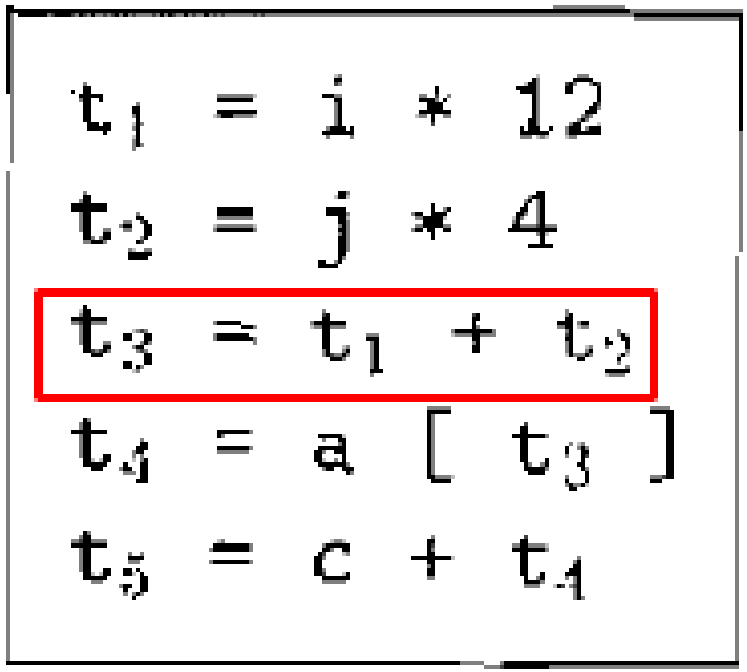
$$
\begin{aligned}
L \rightarrow\ & \mathbf{id}\ [\ E\ ] \quad \{\ L.array = top.get(\mathbf{id}.lexeme);\\
& \quad L.type = L.array.type.elem;\\
& \quad L.addr = \mathbf{new}\ Temp\,();\\
& \quad gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \}\\
\\
|\ \ & L_1\ [\ E\ ] \quad \{\ L.array = L_1.array;\\
& \quad L.type = L_1.type.elem;\\
& \quad t = \mathbf{new}\ Temp\,();\\
& \quad L.addr = \mathbf{new}\ Temp\,();\\
& \quad gen(t\ '='\ E.addr\ '*'\ L.type.width);\\
& \quad gen(L.addr\ '='\ L_1.addr\ '+'\ t);\ \}
\end{aligned}
$$

综合属性 $L.addr$：(当前) 偏移地址

$$
\begin{aligned}
L \rightarrow\ & \mathbf{id}\ [\ E\ ] \quad \{\ L.array = top.get(\mathbf{id}.lexeme);\\
& \quad L.type = L.array.type.elem;\\
& \quad L.addr = \mathbf{new}\ Temp\,();\\
& \quad gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \}\\
\mid\ & L_1\ [\ E\ ] \quad \{\ L.array = L_1.array;\\
& \quad L.type = L_1.type.elem;\\
& \quad t = \mathbf{new}\ Temp\,();\\
& \quad L.addr = \mathbf{new}\ Temp\,();\\
& \quad gen(t\ '='\ E.addr\ '*'\ L.type.width);\\
& \quad gen(L.addr\ '='\ L_1.addr\ '+'\ t);\ \}
\end{aligned}
$$


E.addr = t5
+
E.addr = c
c
E.addr = t4
L.array = a
L.type = integer
L.addr = t3
L.array = a
L.type = array(3, integer)
L.addr = t1
[
E.addr = j
]
j
a.type
= array(2, array(3, integer))
[
E.addr = i
]
i

int $a[2][3]$



int $a[2][3]$

控制流语句的中间代码翻译

$$S \longrightarrow \textbf{if}\ (B)\ S_1$$

$$S \longrightarrow \textbf{if}\ (B)\ S_1\ \textbf{else}\ S_2$$

$$S \longrightarrow \textbf{while}\ (B)\ S_1$$

产生式	语义规则
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow$ **assign**	$S.code =$ **assign**$.code$
$S \rightarrow$ **if** (B) S_1	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$S \rightarrow$ **if** (B) S_1 **else** S_2	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ S.next)$ $\mid\mid label(B.false) \mid\mid S_2.code$
$S \rightarrow$ **while** (B) S_1	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ begin)$
$S \rightarrow S_1\ S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code$

继承属性 $S.next$: S 的下一条指令

$$P \rightarrow S \quad \left| \begin{array}{l} S.next = newlabel() \\ P.code = S.code \,||\, label(S.next) \end{array} \right.$$

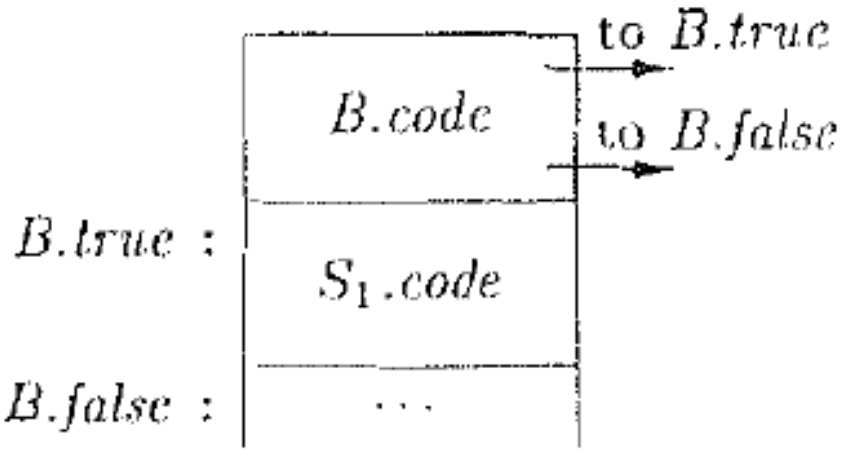
$S.next$ 为语句 S 指明了 “跳出” S 的目标

$$S \rightarrow \textbf{assign} \quad \Big| \quad S.code = \textbf{assign}.code$$

表达式语句的翻译, 包括数组引用

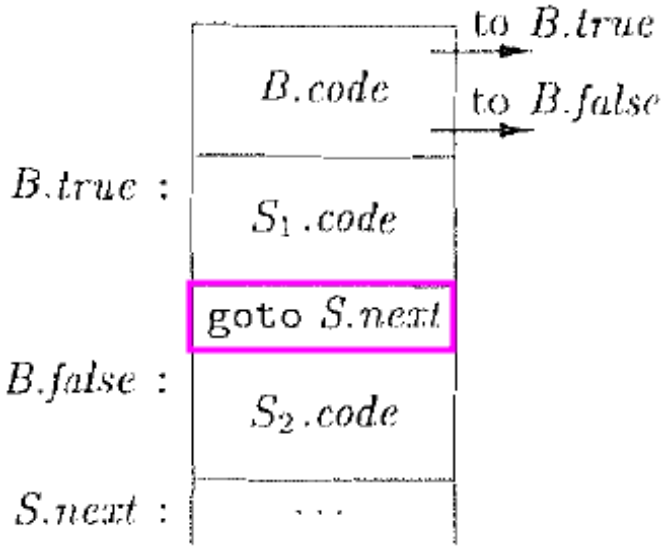
$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$

$B.true = newlabel()$
$B.false = S_1.next = S.next$
$S.code = B.code \,||\, label(B.true) \,||\, S_1.code$




```
if (B)
  if (B) assign
```

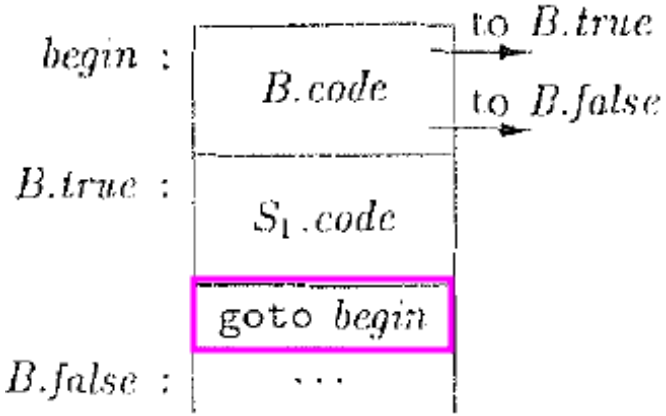
$S \rightarrow$ **if** (B) S_1 **else** S_2	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\quad \| \| label(B.true) \| \| S_1.code$ $\quad \| \| gen('\texttt{goto}'\ S.next)$ $\quad \| \| label(B.false) \| \| S_2.code$




```
if (B)
  if (B) assign else assign
else
  assign
```

$S \rightarrow$ **while** (B) S_1

$begin = newlabel()$
$B.true = newlabel()$
$B.false = S.next$
$S_1.next = begin$
$S.code = label(begin) \,||\, B.code$
$\,||\, label(B.true) \,||\, S_1.code$
$\,||\, gen('\texttt{goto}'\ begin)$




```
while (B)
  if (B) assign else assign
```

$$
\begin{array}{lll}
S \rightarrow S_1\ S_2 & \boxed{S_1.next} = newlabel() \\
& \boxed{S_2.next} = S.next \\
& S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code
\end{array}
$$

产生式	语义规则
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow$ **assign**	$S.code =$ **assign**$.code$
$S \rightarrow$ **if** (B) S_1	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$S \rightarrow$ **if** (B) S_1 **else** S_2	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ S.next)$ $\mid\mid label(B.false) \mid\mid S_2.code$
$S \rightarrow$ **while** (B) S_1	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ begin)$
$S \rightarrow S_1\ S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code$

布尔表达式的中间代码翻译

产生式	语义规则
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \;\&\&\; B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1$ **rel** E_2	$B.code = E_1.code \mid\mid E_2.code$ $\mid\mid gen(\texttt{'if'}\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ \texttt{'goto'}\ B.true)$ $\mid\mid gen(\texttt{'goto'}\ B.false)$
$B \rightarrow$ **true**	$B.code = gen(\texttt{'goto'}\ B.true)$
$B \rightarrow$ **false**	$B.code = gen(\texttt{'goto'}\ B.false)$

$B \rightarrow$ true		$B.code = gen('\texttt{goto}'\ B.true)$
$B \rightarrow$ false		$B.code = gen('\texttt{goto}'\ B.false)$

if (true) assign

$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$		$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$

if (false) assign

$$B \rightarrow \; ! \; B_1 \qquad \left| \begin{array}{l} B_1.true = B.false \\ B_1.false = B.true \\ B.code = B_1.code \end{array} \right.$$

if (!true) assign

$$S \rightarrow \textbf{if} \; (\; B \;) \; S_1 \qquad \left| \begin{array}{l} B.true = newlabel() \\ B.false = S_1.next = S.next \\ S.code = B.code \; || \; label(B.true) \; || \; S_1.code \end{array} \right.$$

if (!false) assign

短路求值

$$
\begin{array}{l|l}
B \rightarrow B_1 \;||\; B_2 & B_1.true = B.true \\
 & B_1.false = newlabel() \\
 & B_2.true = B.true \\
 & B_2.false = B.false \\
 & B.code = B_1.code \;||\; label(B_1.false) \;||\; B_2.code
\end{array}
$$

```
if (true || false) assign
```

$$
\begin{array}{l|l}
S \rightarrow \textbf{if}\ (\ B\ )\ S_1 & B.true = newlabel() \\
 & B.false = S_1.next = S.next \\
 & S.code = B.code \;||\; label(B.true) \;||\; S_1.code
\end{array}
$$

```
if (false || true) assign
```

短路求值

$$
\begin{array}{l|l}
B \rightarrow B_1 \;\&\&\; B_2 & B_1.true = newlabel() \\
 & B_1.false = B.false \\
 & B_2.true = B.true \\
 & B_2.false = B.false \\
 & B.code = B_1.code \;||\; label(B_1.true) \;||\; B_2.code
\end{array}
$$

```
if (true && false) assign
```

$$
\begin{array}{l|l}
S \rightarrow \textbf{if}\ (\ B\ )\ S_1 & B.true = newlabel() \\
 & B.false = S_1.next = S.next \\
 & S.code = B.code \;||\; label(B.true) \;||\; S_1.code
\end{array}
$$

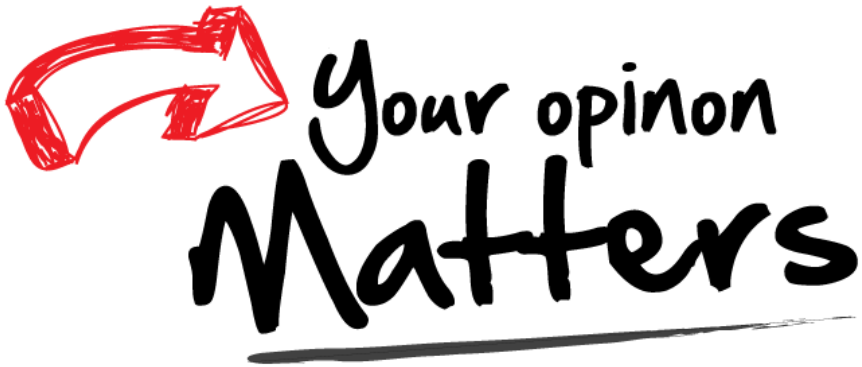
```
if (false && true) assign
```

$$
\begin{array}{lll}
B \rightarrow E_1 \ \mathbf{rel} \ E_2 & \Big| & B.code = E_1.code \ || \ E_2.code \\
& & \quad || \ gen(\text{'}\mathtt{if}\text{'} \ E_1.addr \ \mathbf{rel}.op \ E_2.addr \ \text{'}\mathtt{goto}\text{'} \ B.true) \\
& & \quad || \ gen(\text{'}\mathtt{goto}\text{'} \ B.false)
\end{array}
$$

```
if (x < 100 || x > 200 && x != y) x = 0;
```

```
        if x < 100 goto L2
        goto L3
L3:     if x > 200 goto L4
        goto L1
L4:     if x != y goto L2
        goto L1
L2:     x = 0
L1:
```


Thank
You!



Office 926

hfwei@nju.edu.cn