中间代码生成
(1. 表达式的翻译与控制流的翻译)

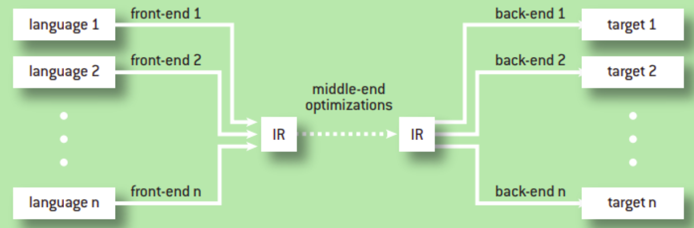
魏恒峰

hfwei@nju.edu.cn

2022 年 12 月 26 日

FIGURE 2

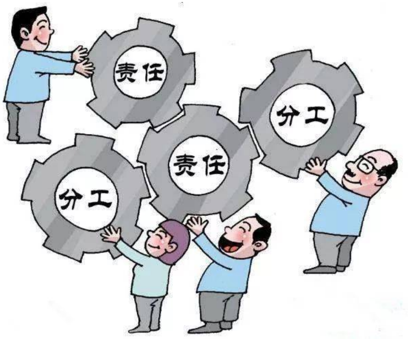
A Compiler System Supporting Multiple Languages and Multiple Targets



The Increasing Significance of Intermediate Representations in Compilers (Fred Chow; 2013)

腾讯视频出品
大局观
好羊奶 选蓝河
对有大局观对对对
知乎 @Genoa

分工　　合作



父节点为子节点准备跳转指令的目标标签

子节点通过**继承属性**确定跳转目标

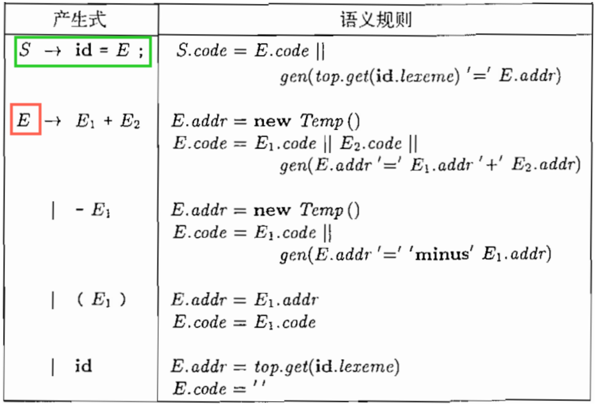
在自顶向下的分析过程中

为右部的每个 B 计算 $B.\text{true}$ 与 $B.\text{false}$

为右部的每个 S 计算 $S.\text{next}$

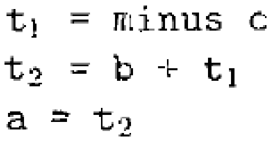
表达式的中间代码翻译

综合属性 $E.code$: 中间代码



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

$$a = b + -c$$



$t_1 = \text{minus } c$

$t_2 = b + t_1$

$a = t_2$

综合属性 $E.addr$: 变量名 (包括临时变量)、常量

```c
int main() {
  int a = 0, b = 1, c = 2;

  a = b + -c;

  return 0;
}
```

```llvm
%7 = sub nsw i32 0, %6
%8 = add nsw i32 %5, %7
store i32 %8, i32* %2, align 4
```

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 $a[1][2]$ 相对于**数组基地址** a 的**偏移地址**

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

	类型	宽度
a	$array(2, array(3, \text{integer}))$	24
$a[i]$	$array(3, \text{integer})$	12
$a[i][j]$	integer	4

int $a[2][3]$

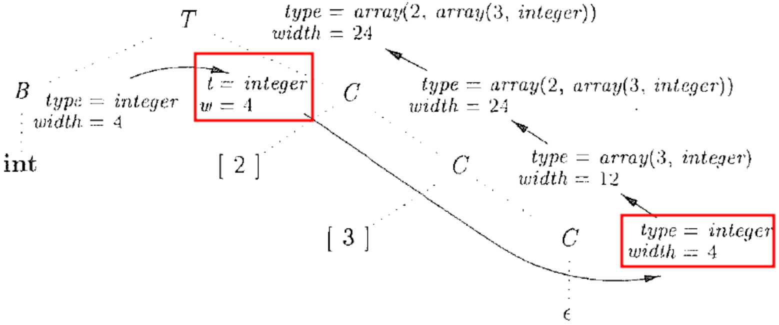
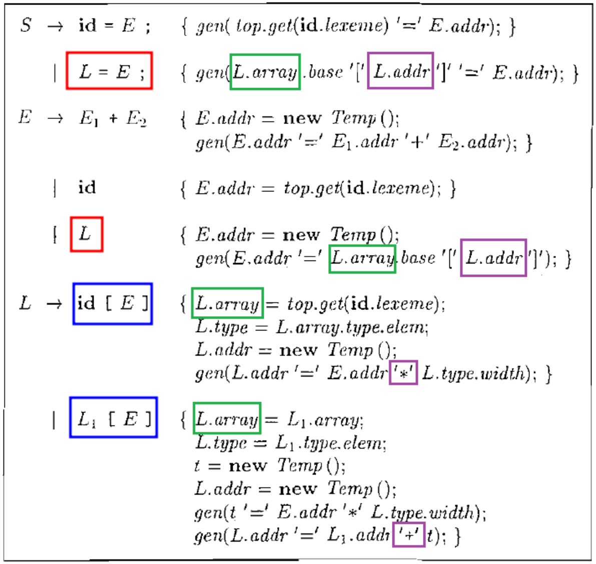


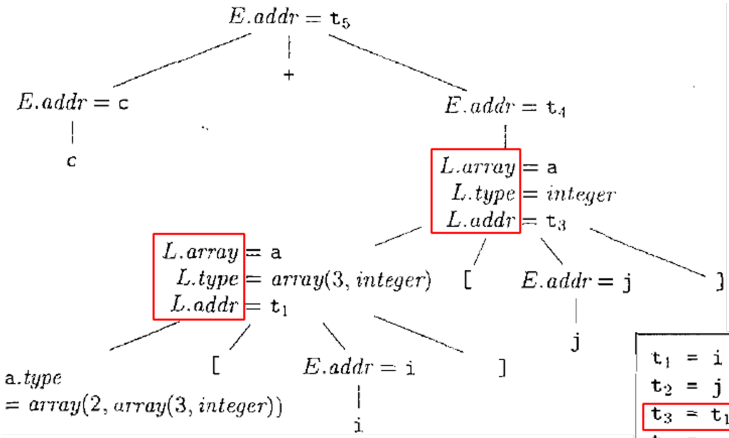
图 6-16 数组类型的语法制导翻译

综合属性 $L.array(.base)$: 数组基地址 (即, 数组名)



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

综合属性 $L.addr$: 偏移地址



```
t1 = i * 12
t2 = j * 4
t3 = t1 + t2
t4 = a [ t3 ]
t5 = c + t4
```

$$c + a[i][j]$$

```
%2 = alloca [2 x [3 x i32]], align 16
```

```c
int main() {
  int a[2][3] = { 0 };

  int i = 1, j = 2;
  int c = 10, d = 20;

  d = c + a[i][j];

  return 0;
}
```

```
%8 = load i32, i32* %5, align 4 %8: c
%9 = load i32, i32* %3, align 4 %9: i
%10 = sext i32 %9 to i64
%11 = getelementptr inbounds [2 x [3 x i32]], [2 x [3 x i32]]* %2, i64 0, i64 %10
%12 = load i32, i32* %4, align 4  %12: j
%13 = sext i32 %12 to i64
%14 = getelementptr inbounds [3 x i32], [3 x i32]* %11, i64 0, i64 %13
%15 = load i32, i32* %14, align 4  %15: a[i][j]
%16 = add nsw i32 %8, %15
store i32 %16, i32* %6, align 4
```

The Often Misunderstood GEP Instruction

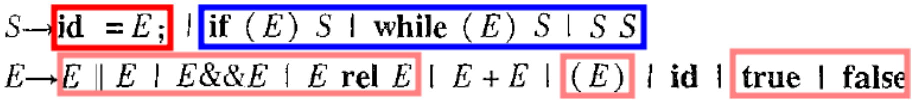
控制流语句与布尔表达式的中间代码翻译

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

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

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

布尔表达式的作用: 布尔值 *vs.* 控制流跳转

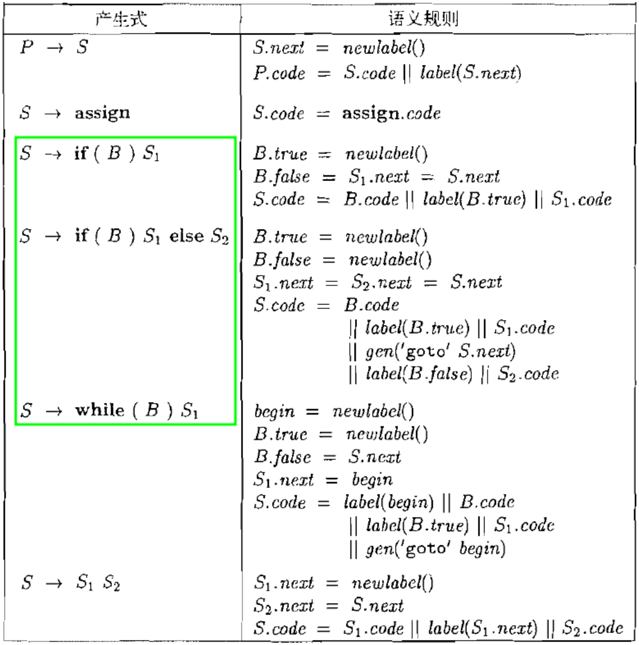


$$S \rightarrow \boxed{\textbf{id} = E;} \mid \boxed{\textbf{if}\ (E)\ S \mid \textbf{while}\ (E)\ S \mid S\ S}$$

$$E \rightarrow \boxed{E \parallel E \mid E \&\& E \mid E\ \textbf{rel}\ E} \mid E + E \mid \boxed{(E)} \mid \textbf{id} \mid \boxed{\textbf{true} \mid \textbf{false}}$$

我们先关注 “控制流跳转”

控制流语句与布尔表达式的中间代码翻译



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

继承属性 $S.next$

$$
P \rightarrow S \qquad \left| \begin{array}{lcl} \boxed{S.next} & = & newlabel() \\ P.code & = & S.code \;||\; \boxed{label(S.next)} \end{array} \right.
$$

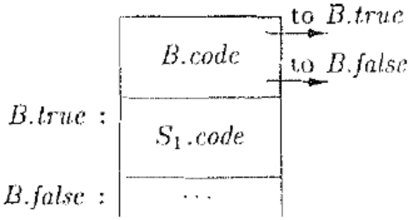
$S.next$ **为语句** S **指明了“跳出”** S **的目标**

$$S \rightarrow \mathbf{assign} \qquad \Big| \; S.code = \mathbf{assign}.code$$

代表了表达式的翻译, 包括数组引用

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

$$\begin{aligned} B.true &= newlabel() \\ B.false &= \boxed{S_1.next} = S.next \\ S.code &= B.code \,||\, label(B.true) \,||\, S_1.code \end{aligned}$$



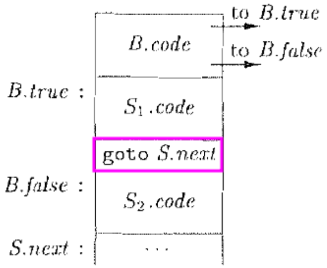
```
if (true)
  if (false) assign
```

$P \rightarrow S$

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

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

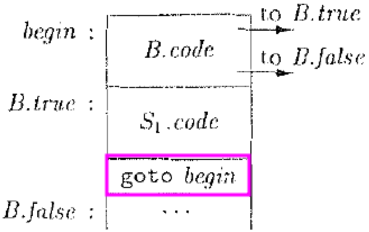
$$
S \rightarrow \textbf{if} (B) S_1 \textbf{ else } S_2 \quad
\begin{array}{|l}
B.true = newlabel() \\
B.false = newlabel() \\
\boxed{S_1.next = S_2.next = S.next} \\
S.code = B.code \\
\qquad\qquad \| \; label(B.true) \; \| \; S_1.code \\
\qquad\qquad \| \; \boxed{gen(\texttt{'goto'}\; S.next)} \\
\qquad\qquad \| \; label(B.false) \; \| \; S_2.code
\end{array}
$$



```
if (true)
  if (true) assign else assign
else
  assign
```

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

$$
\begin{aligned}
begin &= newlabel() \\
B.true &= newlabel() \\
B.false &= S.next \\
\boxed{S_1.next} &= begin \\
S.code &= label(begin) \;||\; B.code \\
&\quad ||\; label(B.true) \;||\; S_1.code \\
&\quad ||\; \boxed{gen(\text{'goto'}\ begin)}
\end{aligned}
$$

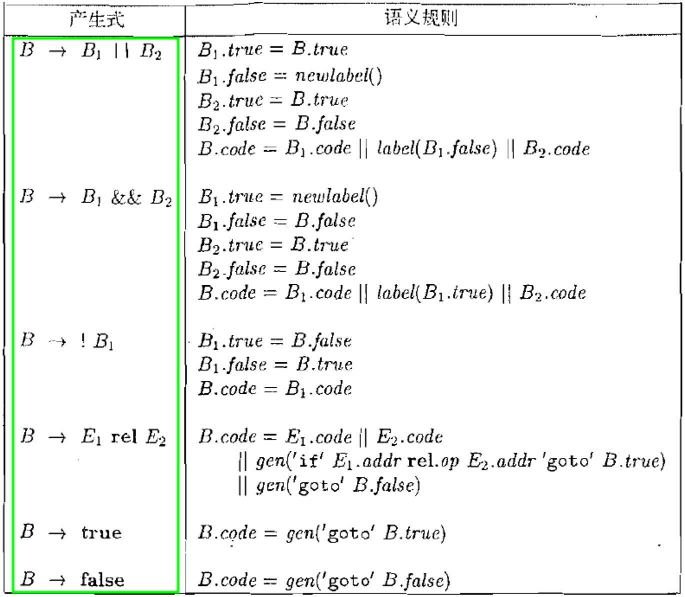


```
while (true)
  if (false) assign else assign
```

$$S \rightarrow S_1 \; S_2 \qquad \begin{array}{|l} \boxed{\begin{array}{l} S_1.next \\ S_2.next \end{array}} \begin{array}{l} = newlabel() \\ = S.next \end{array} \\ S.code = S_1.code \; || \; label(S_1.next) \; || \; S_2.code \end{array}$$

`if (true)` **assign** `else` **assign** **assign**

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	$B.code = E_1.code \mid\mid E_2.code$ $\mid\mid gen($'if' $E_1.addr$ **rel**.op $E_2.addr$ 'goto' $B.true)$ $\mid\mid gen($'goto' $B.false)$
$B \rightarrow$ **true**	$B.code = gen($'goto' $B.true)$
$B \rightarrow$ **false**	$B.code = gen($'goto' $B.false)$

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

`if (true)` **assign**

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

`if (false)` **assign**

$B \rightarrow \; ! \; B_1$

$$\boxed{\begin{array}{l} B_1.true \\ B_1.false \end{array}} \begin{array}{l} = B.false \\ = B.true \end{array}$$
$$B.code = B_1.code$$

`if (!true)` **assign**

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

$$\begin{array}{lll} B.true & = & newlabel() \\ B.false & = & \boxed{S_1.next} = S.next \\ S.code & = & B.code \; || \; label(B.true) \; || \; S_1.code \end{array}$$

`if (!false)` **assign**

短路求值

$$
B \rightarrow B_1 \;||\; B_2 \quad \left|\; \begin{array}{l}
\boxed{B_1.\mathit{true}} = B.\mathit{true} \\
B_1.\mathit{false} = \mathit{newlabel}() \\
\boxed{B_2.\mathit{true}} = B.\mathit{true} \\
B_2.\mathit{false} = B.\mathit{false} \\
B.\mathit{code} = B_1.\mathit{code} \;||\; \mathit{label}(B_1.\mathit{false}) \;||\; B_2.\mathit{code}
\end{array}\right.
$$

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

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

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

短路求值

$$
B \to B_1 \;\&\&\; B_2 \quad \left|
\begin{array}{l}
B_1.true = newlabel() \\
\boxed{B_1.false} = B.false \\
B_2.true = B.true \\
\boxed{B_2.false} = B.false \\
B.code = B_1.code \;||\; label(B_1.true) \;||\; B_2.code
\end{array}\right.
$$

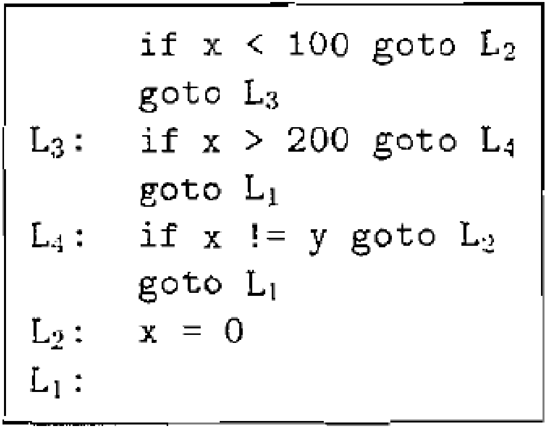
`if (true && false) assign`

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

`if (false && true) assign`

$$
\begin{array}{lll|l}
B & \rightarrow & E_1 \ \mathbf{rel}\ E_2 & B.code = E_1.code \ \|\ E_2.code \\
 & & & \qquad \|\ \boxed{gen}('\texttt{if}'\ E_1.addr\ \mathbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true) \\
 & & & \qquad \|\ \boxed{gen}('\texttt{goto}'\ B.false)
\end{array}
$$

```
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 :
```

布尔表达式的作用: **布尔值** *vs.* **控制流跳转**

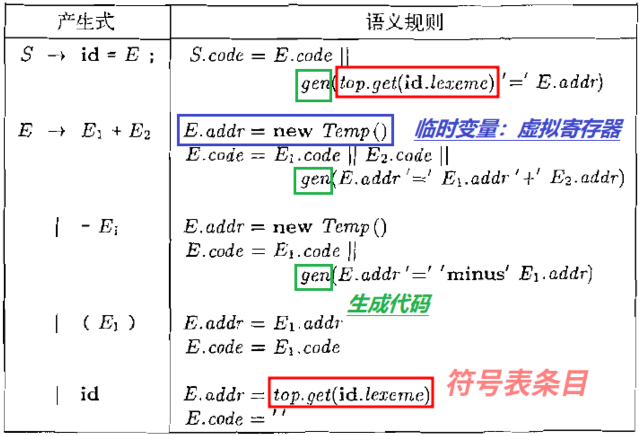
$$S \rightarrow \boxed{\textbf{id} = E;} \mid \boxed{\textbf{if}\ (E)\ S \mid \textbf{while}\ (E)\ S \mid S\ S}$$

$$E \rightarrow \boxed{E \parallel E \mid E \&\& E \mid E\ \textbf{rel}\ E} \mid E + E \mid \boxed{(E)} \mid \textbf{id} \mid \boxed{\textbf{true} \mid \textbf{false}}$$

根据 E 所处的上下文判断 E 所扮演的角色, 调用不同的代码生成函数

函数 $jump(t, f)$: 生成控制流代码

函数 $rvalue()$: 生成计算布尔值的代码, 并将结果存储在临时变量中

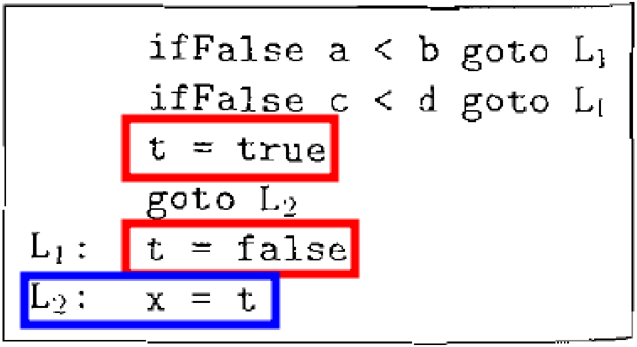

临时变量：虚拟寄存器
生成代码
符号表条目

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

$$E \rightarrow E_1 \&\& E_2$$

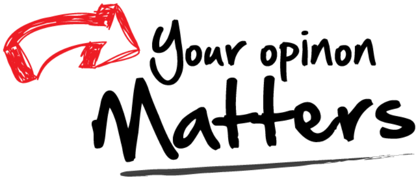
为 E 生成**跳转代码**, 在**真假出口处**将 **true** 或 **false** 存储到临时变量

`x = a < b && c < d`



```
        ifFalse a < b goto L1
        ifFalse c < d goto L1
        t = true
        goto L2
L1 :    t = false
L2 :    x = t
```

Thank
You!



Office 926

hfwei@nju.edu.cn