



中间代码生成

Part3: 标号回填与布尔表达式

李 诚

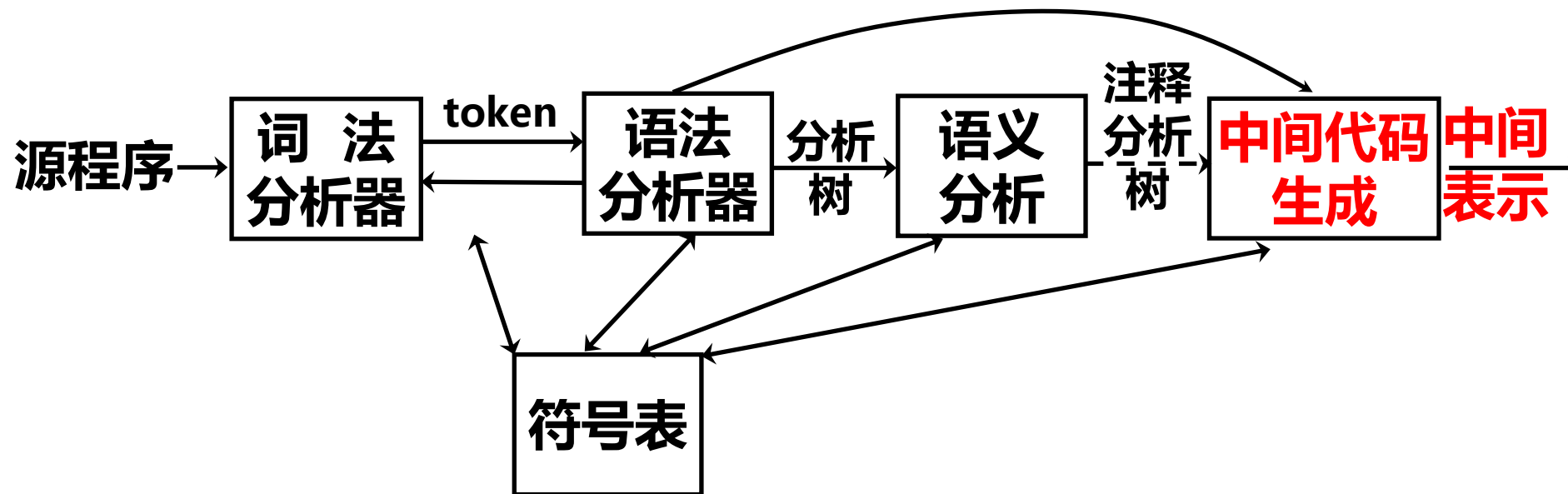
国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2023年10月18/30日



本节提纲



- 标号回填技术
- 基于标号回填的布尔表达式翻译
- 布尔表达式翻译举例



回填(backpatching)技术



- **问题:**

- 布尔表达式短路计算翻译中, 产生了转移目标不明确的条件或无条件代码;



回填(backpatching)技术



- **问题:**

- 布尔表达式短路计算翻译中，产生了转移目标不明确的条件或无条件代码；

- **解决方案:**

- 当生成跳转指令时，暂时不指定目标地址
- 当有关目标地址确定后，再填回到翻译代码中



• 问题:

- 布尔表达式短路计算翻译中, 产生了转移目标不明确的条件或无条件代码;

• 解决方案:

- 当生成跳转指令时, 暂时不指定目标地址
- 当有关目标地址确定后, 再填回到翻译代码中

• 具体实现:

- 将有相同转移目标的转移代码的**编号串起来形成链**, 可以方便回填目标地址。
- 该list变成了**综合属性**, 可以与LR结合
- **注: 后面的翻译均是**与LR结合的语法制导翻译方案



- **对布尔表达式而言，有两个综合属性：**
 - **B.truelist**：代码中所有转向真出口的代码指令链；
 - **B.falselist**：所有转向假出口的代码指令链；
 - 在生成B的代码时，跳转指令goto是不完整的，目标标号尚未填写，用truelist和falselist来管理



实现：数据结构及语义函数



- 将生成的指令放入一个指令数组，指令的标号即为数组下标

标号	指令数组
100	...
101	goto -
102	...
103	goto -
104	
105	
106	

➤ 假设100-103号指令都属于布尔表达式B



实现：数据结构及语义函数

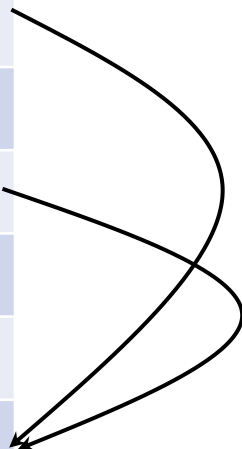


- 将生成的指令放入一个指令数组，指令的标号即为数组下标

标号	指令数组
100	...
101	goto -
102	...
103	goto -
104	
105	
106	

➤ 假设100-103号指令都属于布尔表达式B

➤ 101和103号指令都指向B真出口



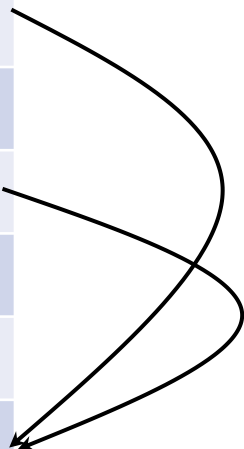


实现：数据结构及语义函数



- 将生成的指令放入一个指令数组，指令的标号即为数组下标

标号	指令数组
100	...
101	goto -
102	...
103	goto -
104	
105	
106	



- 假设100-103号指令都属于布尔表达式B
- 101和103号指令都指向B真出口
- B真出口是106，但还未生成
- `B.trueList = {101, 103}`

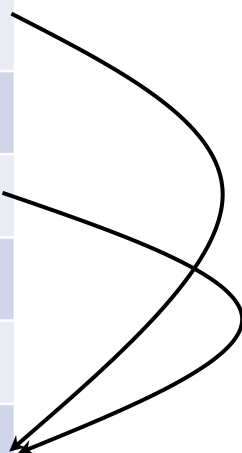


实现：数据结构及语义函数



- 将生成的指令放入一个指令数组，指令的标号即为数组下标

标号	指令数组
100	...
101	goto 106
102	...
103	goto 106
104	...
105	...
106	...



- 假设100-103号指令都属于布尔表达式B
- 101和103号指令都指向B真出口
- B真出口是106，但还未生成
- `B.trueList = {101, 103}`
- 回填时，将101和103补齐



实现：数据结构及语义函数



makelist(i)

- 创建含标号为i的指令的链表
- i不是目标指令，而是源指令，也就是那一些不完整的goto指令



实现：数据结构及语义函数



backpatch(instruction-list, target-label)

- 将目标地址target-label填回instruction-list中每条指令
- 也就是将goto – 指令中不明确的目标补齐



实现：数据结构及语义函数

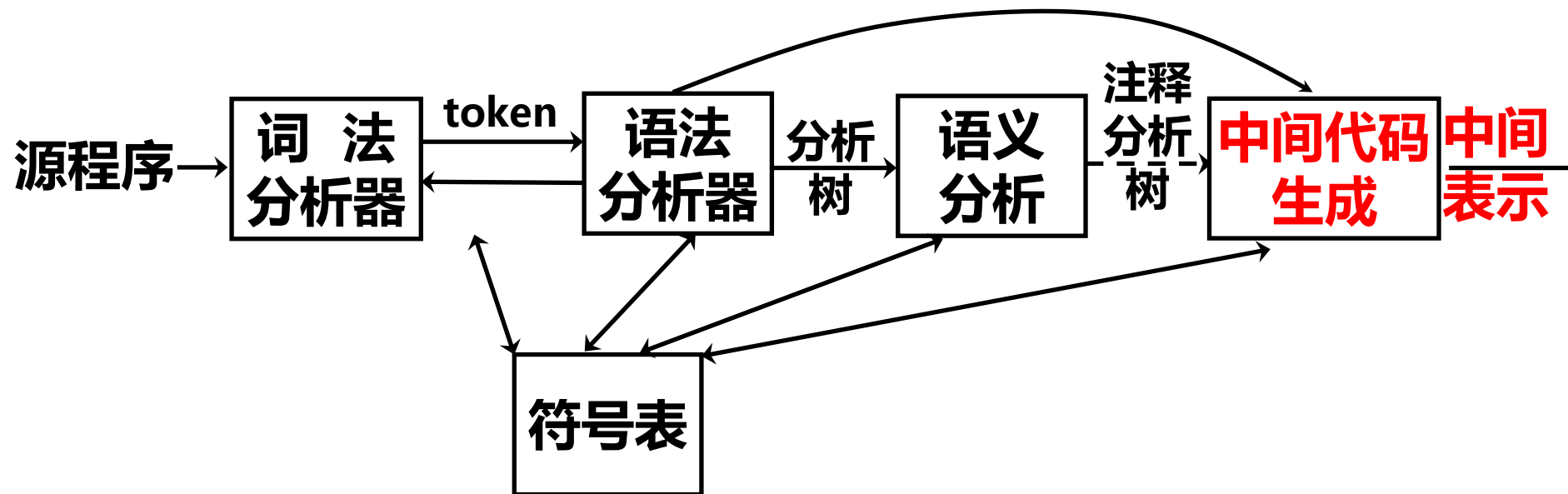


merge(instruction-list₁, instruction-list₂)

- 合并链list₁和list₂
- 要求list₁和list₂中每条指令都会跳转到同一条指令



本节提纲



- 标号回填技术
- 基于标号回填的布尔表达式翻译
- 布尔表达式翻译举例



短路计算及回填的翻译方案



$B \rightarrow \text{not } B_1 \{$ $B.\text{truelist} = B_1.\text{falselist};$
 $B.\text{falselist} = B_1.\text{truelist}; \}$

$B \rightarrow (B_1) \{$ $B.\text{truelist} = B_1.\text{truelist};$
 $B.\text{falselist} = B_1.\text{falselist}; \}$



短路计算及回填的翻译方案



B→ true {

gen(“goto” -); }



B → true {

B.truelist = makelist(**);**

/*为真时，执行无条件跳转指令，但是目标为空。当目标明确后，回填所对应的跳转指令*/

```
gen( "goto" - ); }
```

我们用变量
`nextinstr`保存了紧
跟着的下一条指令
的序号



短路计算及回填的翻译方案



B → true {

B.truelist = makelist(nextinstr);

**/*为真时，执行无条件跳转指令，但是目标为空。当目标明确后，回填
nextinstr所对应的跳转指令*/**

gen(“goto” -); }



短路计算及回填的翻译方案



B → true {

B.truelist = makelist(nextinstr);

**/*为真时，执行无条件跳转指令，但是目标为空。当目标明确后，回填
nextinstr所对应的跳转指令*/**

gen(“goto” -); }

B → false {

B.falselist = makelist(nextinstr);

**/*为假时，执行无条件跳转指令，但是目标为空。当目标明确后，回填
nextinstr所对应的跳转指令*/**

gen(“goto” -); }



短路计算及回填的翻译方案



$B \rightarrow E_1 \text{ relop } E_2 \{$

$\text{gen("if" } E_1.\text{place relop.op } E_2.\text{place "goto" -);}$

gen("goto" -);



短路计算及回填的翻译方案



$B \rightarrow E_1 \text{ relop } E_2 \{$

$B.\text{truelist} = \text{makelist}(\quad);$

$B.\text{falselist} = \text{makelist}(\quad);$

/*为真时，执行条件跳转指令，但是目标为空。当目标明确后，回填所对应的跳转指令*/

$\text{gen}(\text{"if"} E_1.\text{place relop.op } E_2.\text{place "goto"} -);$

/*为假时，执行无条件跳转指令，但是目标为空。当目标明确后，回填所对应的跳转指令*/

$\text{gen}(\text{"goto"} -); \}$



短路计算及回填的翻译方案



$B \rightarrow E_1 \text{ relop } E_2 \{$

$B.\text{truelist} = \text{makelist}(\text{nextinstr});$

$B.\text{falselist} = \text{makelist}(\text{nextinstr}+1);$

/*为真时，执行条件跳转指令，但是目标为空。当目标明确后，回填
nextinstr所对应的跳转指令*/

$\text{gen}(\text{"if"} E_1.\text{place } \text{relop.op } E_2.\text{place "goto"} -);$

/*为假时，执行无条件跳转指令，但是目标为空。当目标明确后，回
填nextinstr+1所对应的跳转指令*/

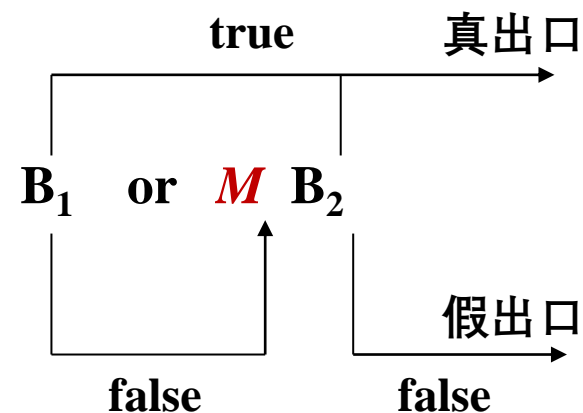
$\text{gen}(\text{"goto"} -); \}$



短路计算及回填的翻译方案



$B \rightarrow B_1 \text{ or } M B_2$



/*获取下一三地址代码（语句）的编号（作为转移目标来回填），
在自底向上的语法分析中传递信息；

在分析 B_2 之前做，因此可以保存 B_2 开始的第一条指令的地址*/

$M \rightarrow \epsilon \quad \{ M.instr = nextinstr \}$



短路计算及回填的翻译方案

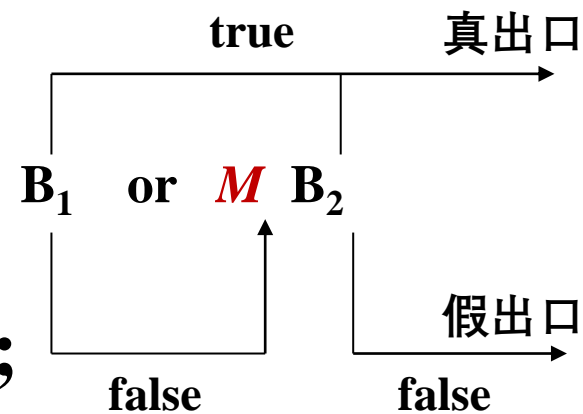


$B \rightarrow B_1 \text{ or } M B_2$

{ backpatch(B_1 .falselist, M .instr);

B .truelist = merge(B_1 .truelist, B_2 .truelist);

B .falselist = B_2 .falselist; }



/*获取下一三地址代码（语句）的编号（作为转移目标来回填），
在自底向上的语法分析中传递信息；

在分析 B_2 之前做，因此可以保存 B_2 开始的第一条指令的地址*/

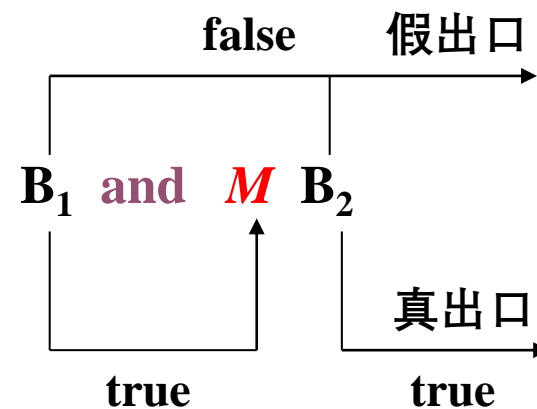
$M \rightarrow \epsilon$ { M .instr = nextinstr }



短路计算及回填的翻译方案



$B \rightarrow B_1 \text{ and } M B_2$



$M \rightarrow \varepsilon \{ M.instr = nextinstr \}$ // 在分析 B_2 之前做，因此可以保存 B_2 开始的第一条指令的地址



短路计算及回填的翻译方案

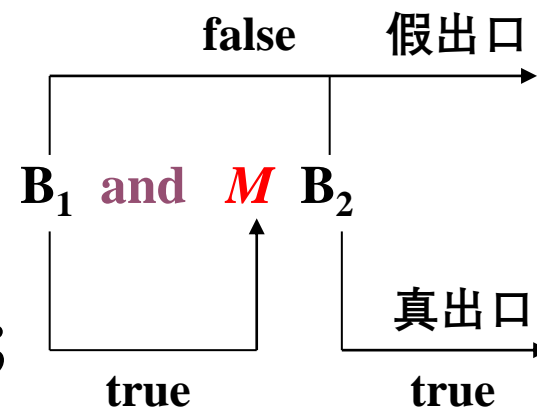


$B \rightarrow B_1 \text{ and } M B_2$

{ backpatch(B_1 .truelist, M .instr);

B .falselist = merge(B_1 .falselist, B_2 .falselist);

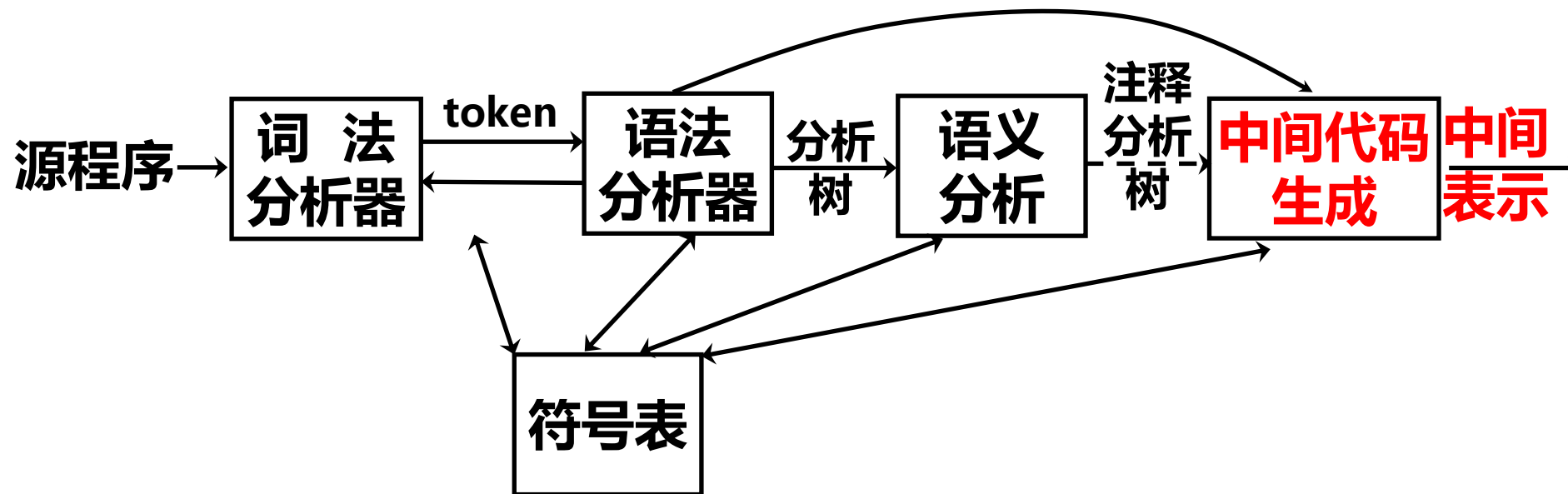
B .truelist = B_2 .truelist; }



$M \rightarrow \epsilon$ { M .instr = nextinstr } // 在分析 B_2 之前做, 因此可以保存 B_2 开始的第一条指令的地址



本节提纲



- 标号回填技术
- 基于标号回填的布尔表达式翻译
- 布尔表达式翻译举例

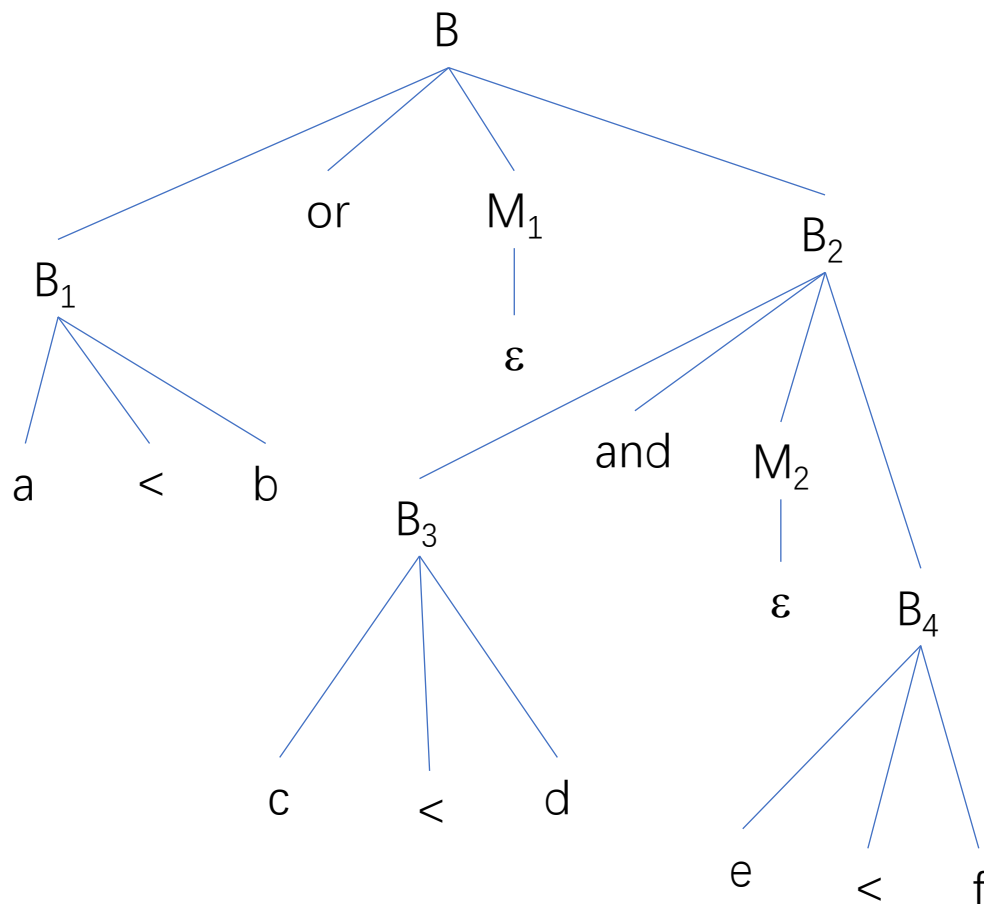


布尔表达式的翻译



$a < b$ or $c < d$ and $e < f$

假设 $\text{nextinstr} = 100$





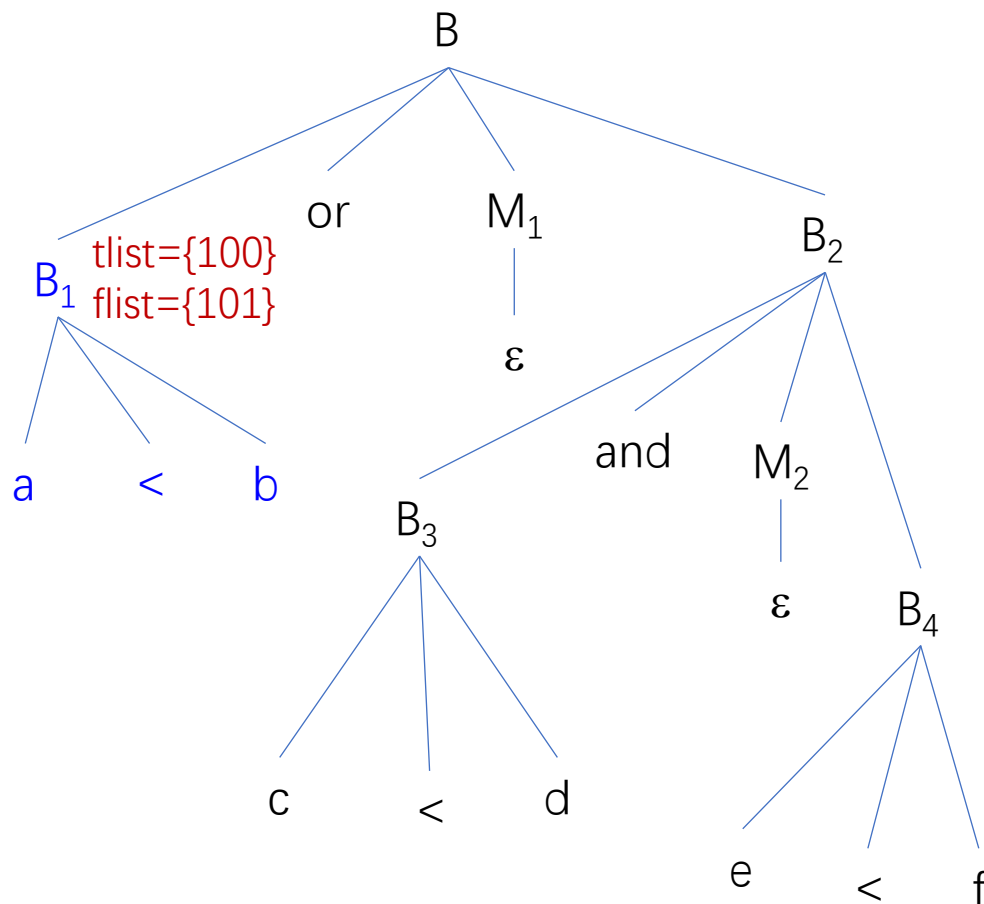
布尔表达式的翻译



$a < b$ or $c < d$ and $e < f$

假设 $\text{nextinstr} = 100$

(100) if $a < b$ goto -
(101) goto -





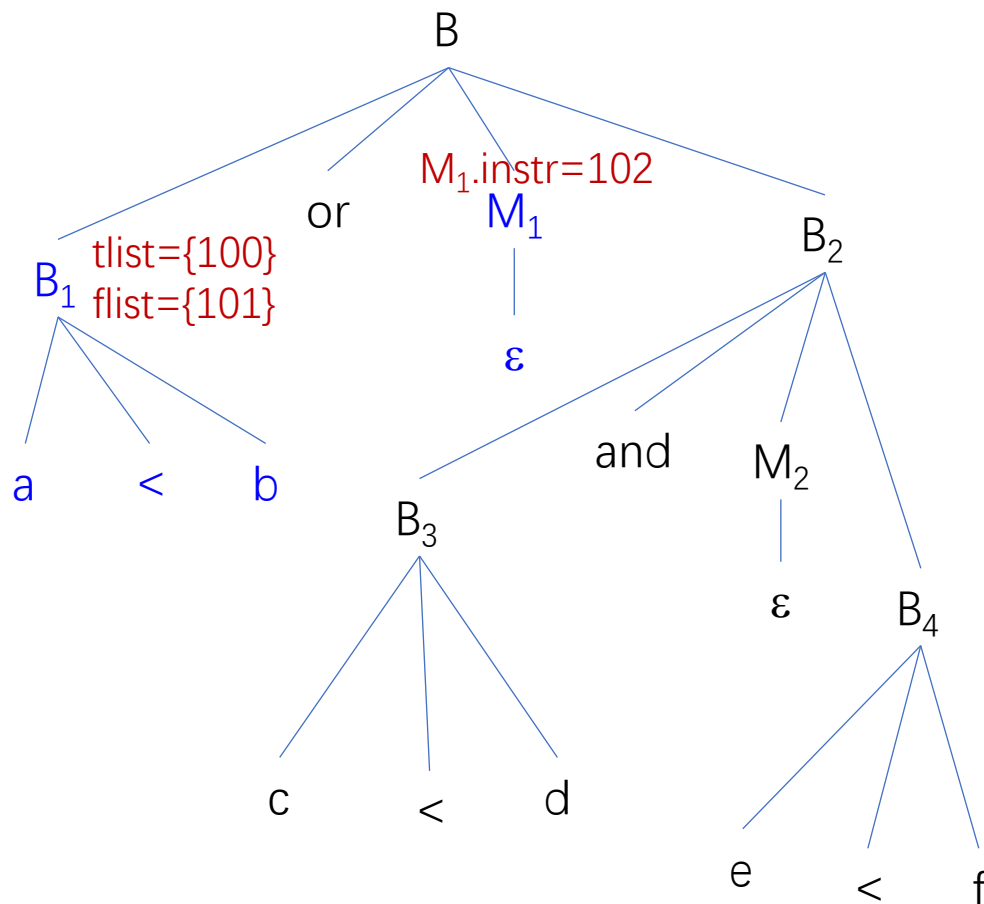
布尔表达式的翻译



$a < b$ or $c < d$ and $e < f$

假设 $\text{nextinstr} = 100$

(100) if $a < b$ goto -
(101) goto -





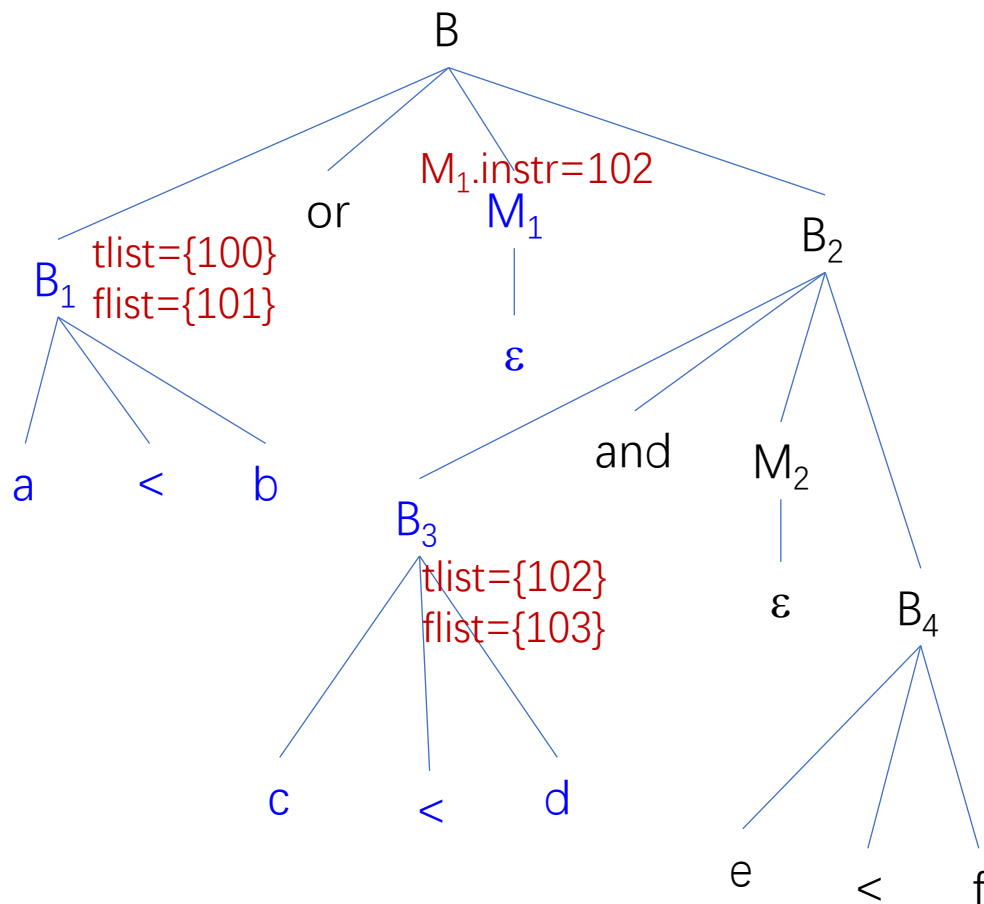
布尔表达式的翻译



$a < b$ or $c < d$ and $e < f$

假设 $\text{nextinstr} = 100$

(100) if $a < b$ goto -
(101) goto -
(102) if $c < d$ goto -
(103) goto -





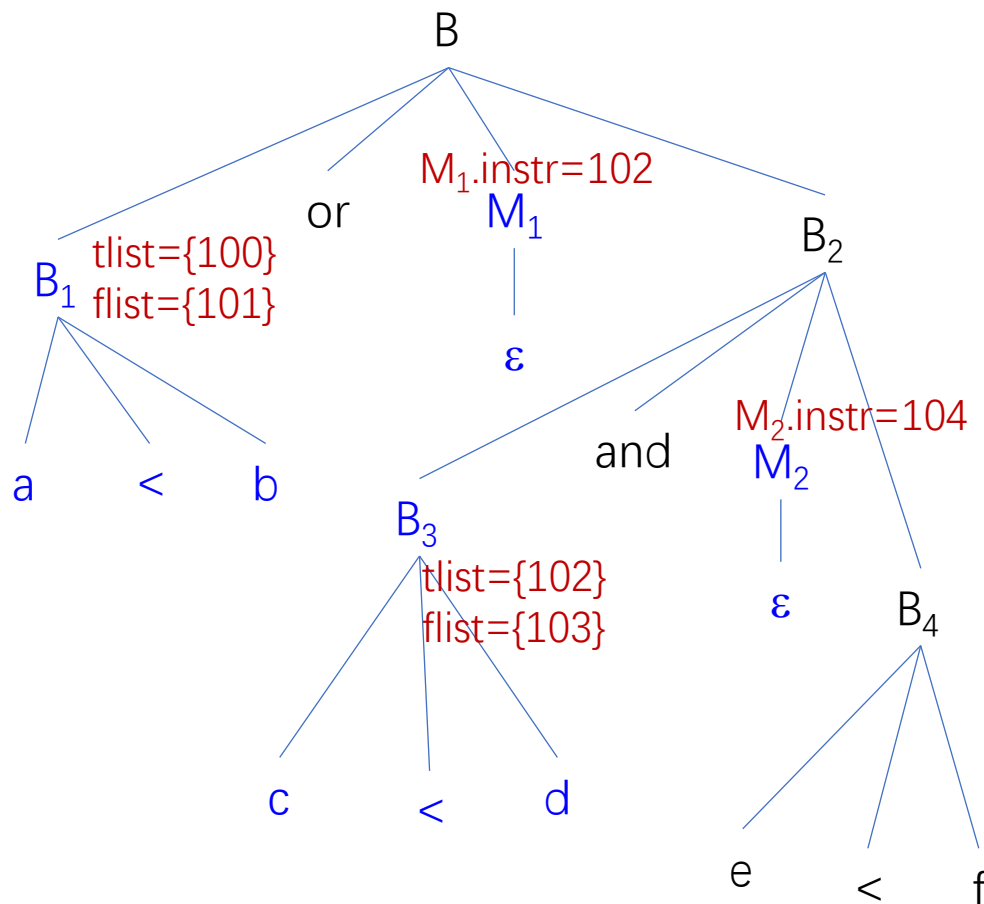
布尔表达式的翻译



$a < b$ or $c < d$ and $e < f$

假设 $\text{nextinstr} = 100$

(100) if $a < b$ goto -
(101) goto -
(102) if $c < d$ goto -
(103) goto -





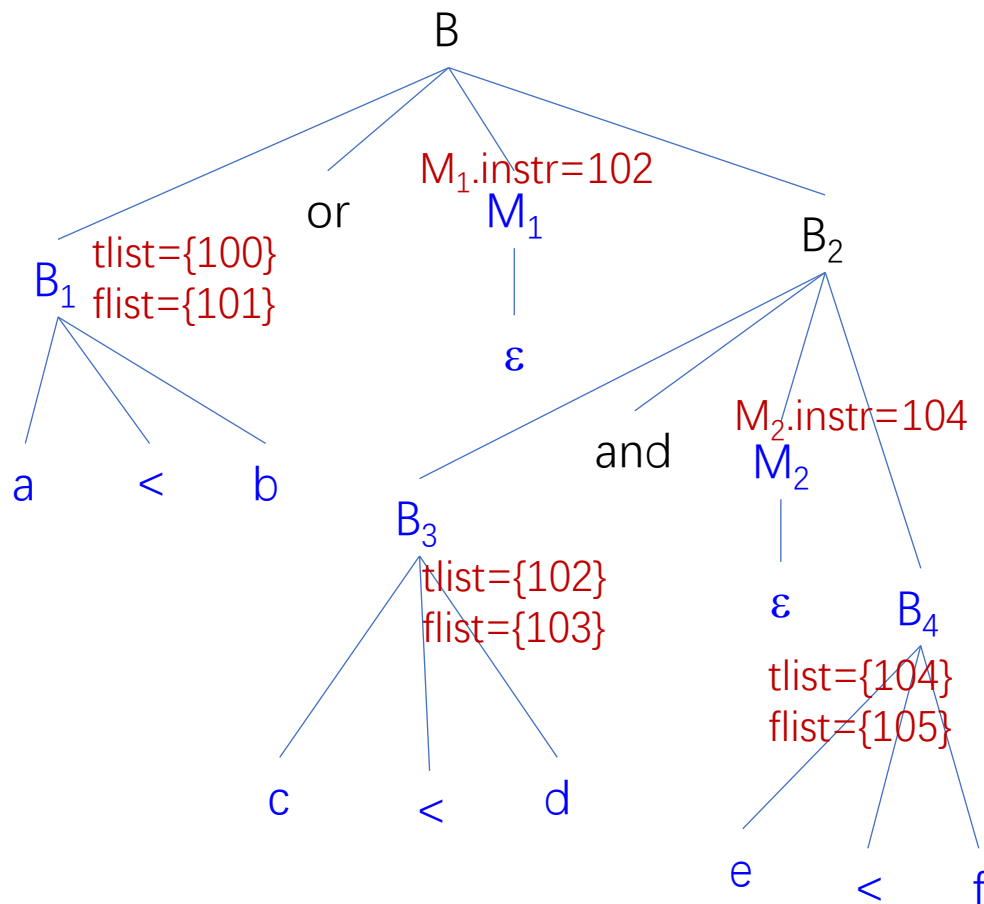
布尔表达式的翻译



$a < b$ or $c < d$ and $e < f$

假设 $\text{nextinstr} = 100$

(100) if $a < b$ goto -
(101) goto -
(102) if $c < d$ goto -
(103) goto -
(104) if $e < f$ goto -
(105) goto -





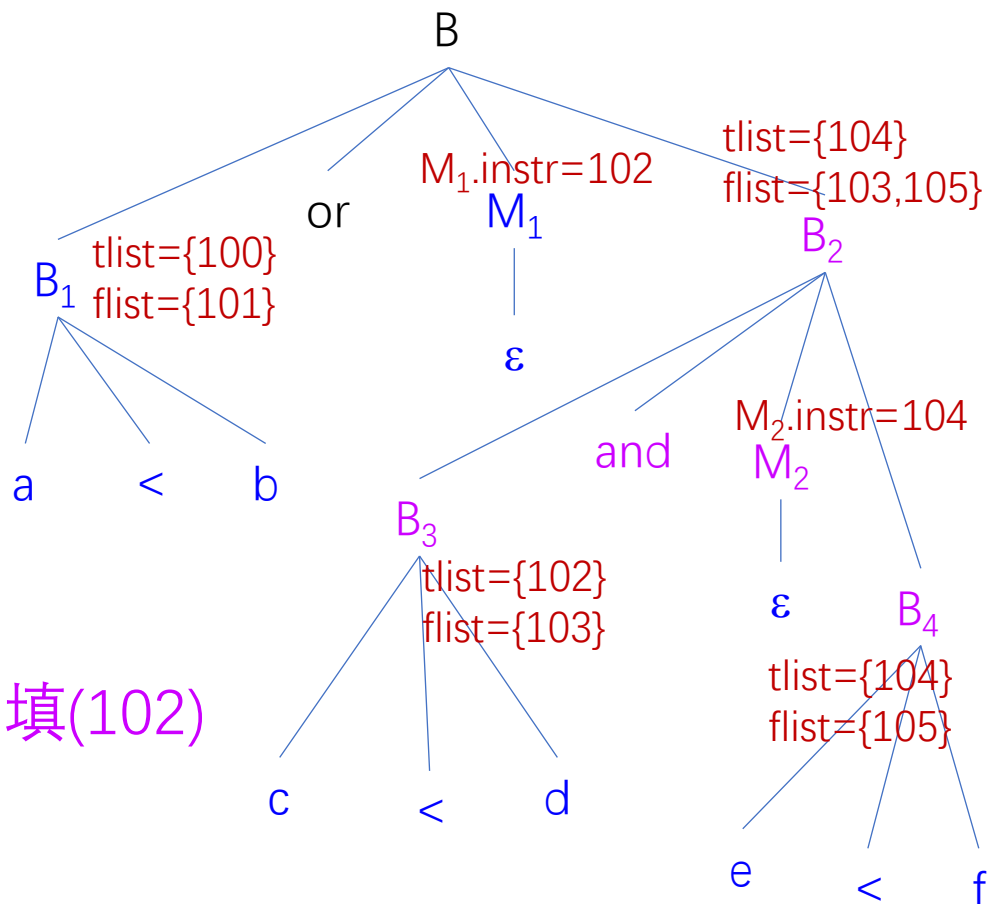
布尔表达式的翻译



$a < b$ or $c < d$ and $e < f$

假设 $\text{nextinstr} = 100$

(100) if $a < b$ goto -
(101) goto -
(102) if $c < d$ goto 104 // 用104回填(102)
(103) goto -
(104) if $e < f$ goto -
(105) goto -

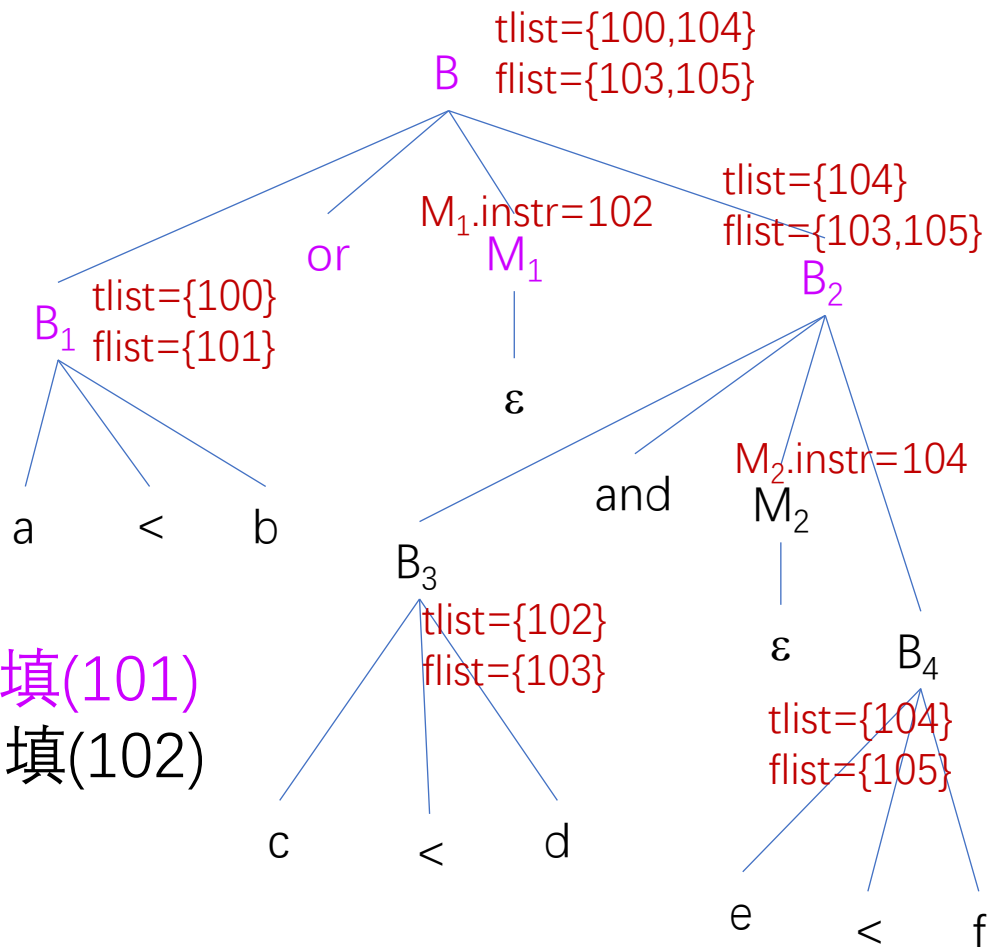




$a < b$ or $c < d$ and $e < f$

假设 $\text{nextinstr} = 100$

(100) if $a < b$ goto -
(101) goto 102 //用102回填(101)
(102) if $c < d$ goto 104 //用104回填(102)
(103) goto -
(104) if $e < f$ goto -
(105) goto -



其他部分的回填要依赖与其他语句的翻译



• 考虑布尔表达式

$a > b$ or true and not $c < f$

- 参考本ppt中布尔表达式短路计算、标号回填等翻译技术，生成对应的三地址代码。
- 假设nextinstr = 200
- 除了三地址代码外，画出LR分析方法对应的注释分析树（如slide 35），标注出属性和属性值。
- 结合LR分析方法指出回填的具体细节
 - 在使用哪一个产生式归约时候进行的回填
 - 用哪一个标号，回填了哪一个不完整的goto指令



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10月18/30日