编译原理

第十章 优化

第十章 优化

- 优化概述
- 局部优化
- 循环优化

第十章 优化

- 优化概述
- 局部优化
- 循环优化

10.2 局部优化

- 局限于基本块范围内的优化称为基本块内的优化，或称局部优化
- 在一个基本块内通常可以实行下面的优化
  - 删除公共子表达式
  - 删除无用赋值
  - 合并已知量
  - 临时变量改名
  - 交换语句的位置
  - 代数变换

优化措施

- 合并已知量

T_1:=2

…

T_2:=4*T_1

变换成

T_2:=8

- 临时变量改名

T:=b+c

其中 T 是一个临时变量名。把这个语句改成:

S:=b+c

优化措施

- 交换语句的位置

T_1:=b+c

T_2:=x+y

- 代数变换

x:=x+0

或　x:=x*1

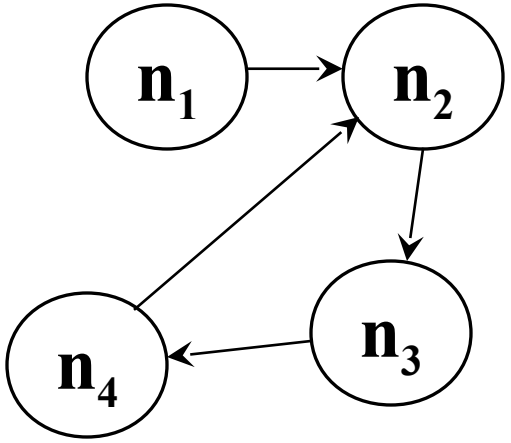
x:=y**2

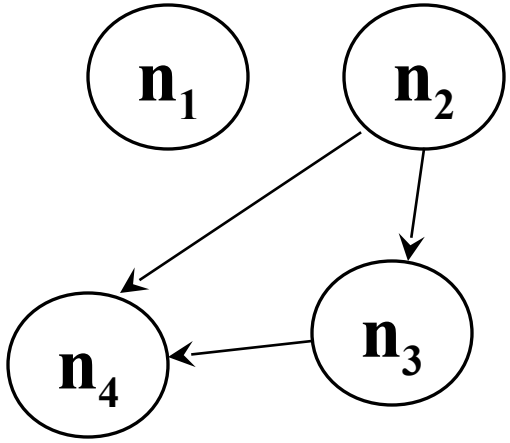
变换成

x:=y*y

10.2.2 基本块的 DAG 表示及其应用

- 有向图
- 有向边：$n_i \rightarrow n_j$
- 前驱： n_i 是 n_j 的前驱
- 后继： n_j 是 n_i 的后继
- 通路：$n_1 \rightarrow n_2$，$n_2 \rightarrow n_3$，…，$n_{k-1} \rightarrow n_k$
- 环路： $n_1 = n_k$
- DAG： 无环路有向图 (Directed Acyclic Graph)

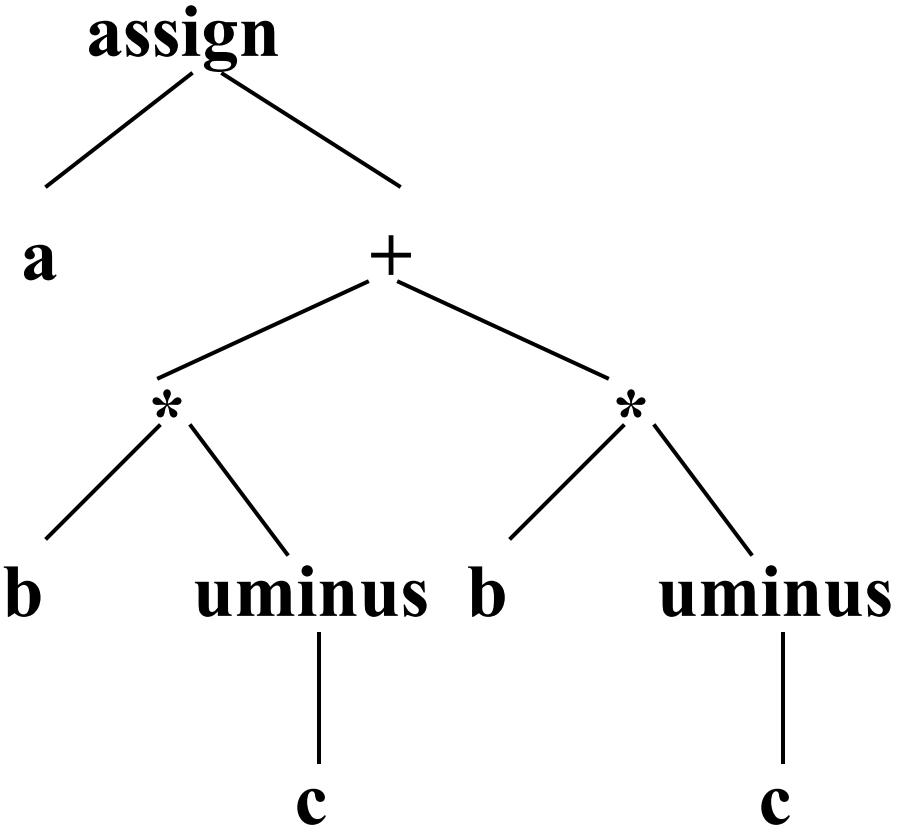




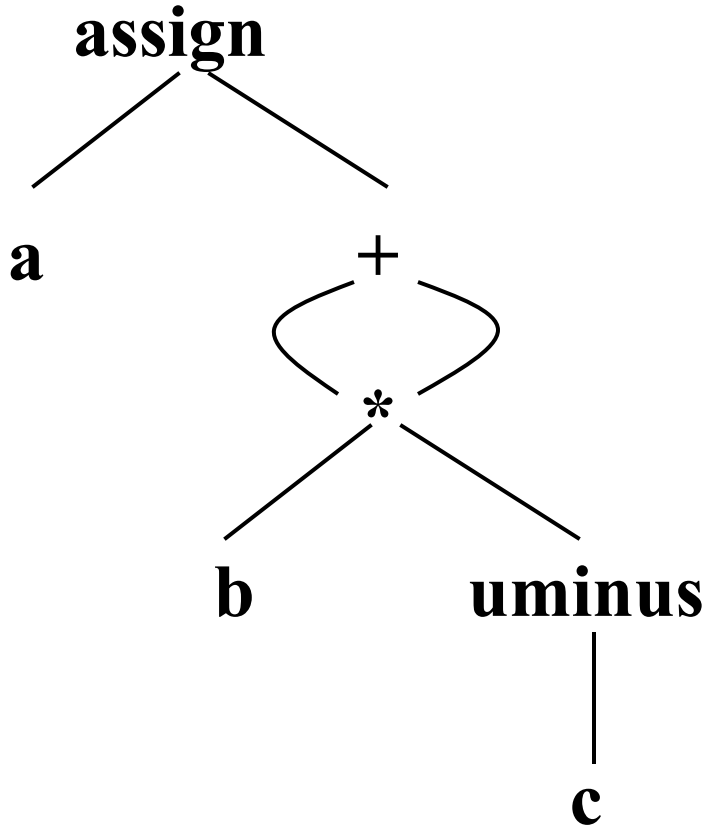
7.1.2 图表示法

- 图表示法
  - DAG
  - 抽象语法树
- 无循环有向图 (Directed Acyclic Graph，简称 DAG)
  - 对表达式中的每个子表达式，DAG 中都有一个结点
  - 一个内部结点代表一个操作符，它的孩子代表操作数
  - 在一个 DAG 中代表公共子表达式的结点具有多个父结点

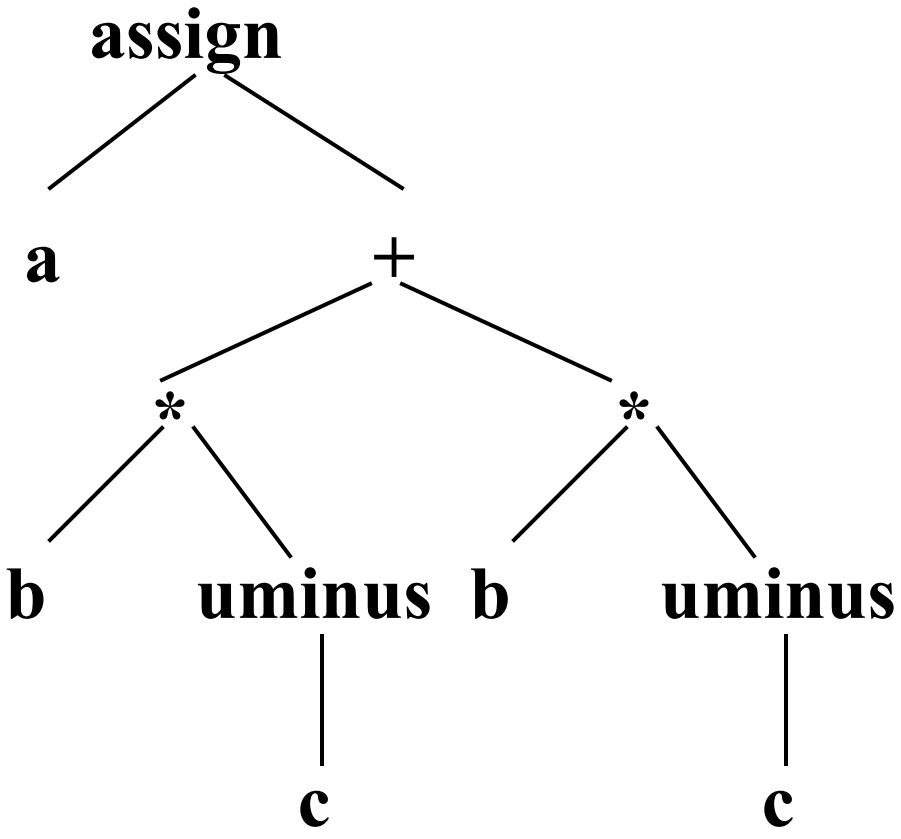
a:=b*(-c)+b*(-c) 的图表示法



抽象语法树



DAG



抽象语法树

抽象语法树对应的代码

T_1:=-c

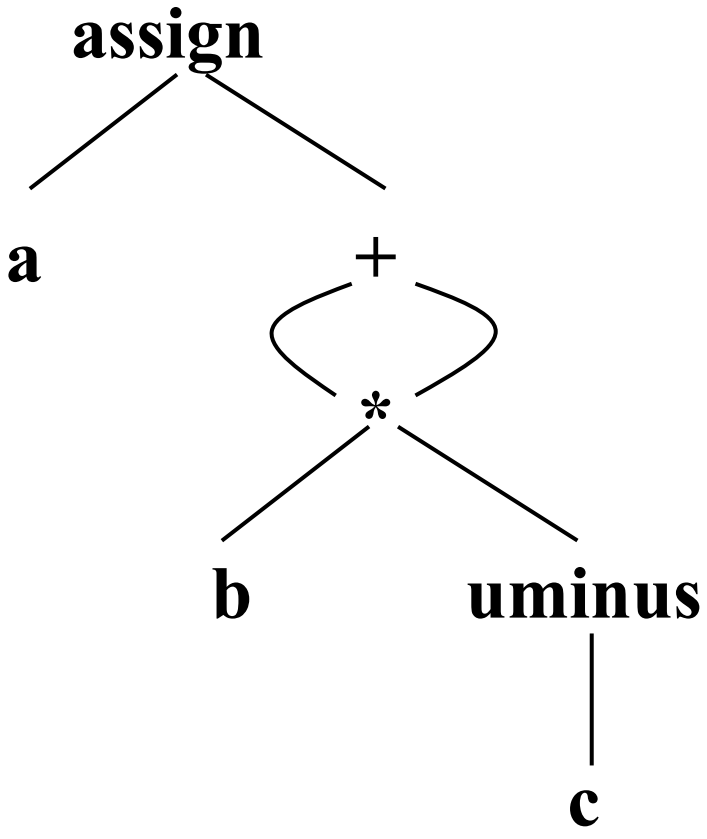
T_2:=b*T_1

T_3:=-c

T_4:=b*T_3

T_5:=T_2+T_4

a:=T_5



DAG

抽象语法树对应的代码：

T_1:=-c

T_2:=b*T_1

T_3:=-c

T_4:=b*T_3

T_5:=T_2+T_4

a:=T_5

DAG 对应的代码：

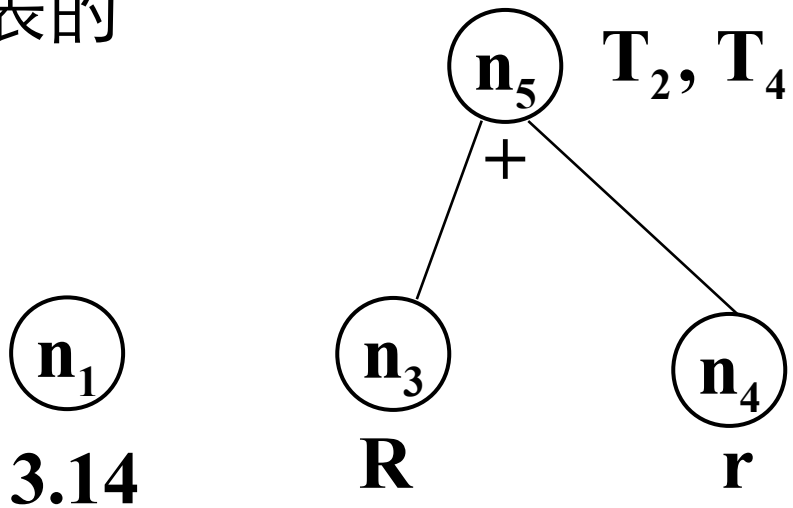
T_1:=-c

T_2:=b*T_1

T_5:=T_2+T_2

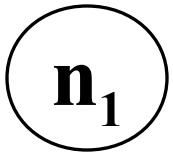
a:=T_5

- 描述计算过程的 DAG 是一种带有下述标记或附加信息的 DAG:
  - 图的叶结点以一标识符或常数作为标记，表示该结点代表该变量或常数的值
  - 图的内部结点以一运算符作为标记，表示该结点代表应用该运算符对其后继结点所代表的值进行运算的结果
  - 各个结点上可能附加一个或多个标识符（称附加标识符）表示这些变量具有该结点所代表的值



10.2.2 基本块的 DAG 表示及应用

- 一个基本块，可用一个 DAG 来表示
- 与各四元式相对应的 DAG 结点形式：

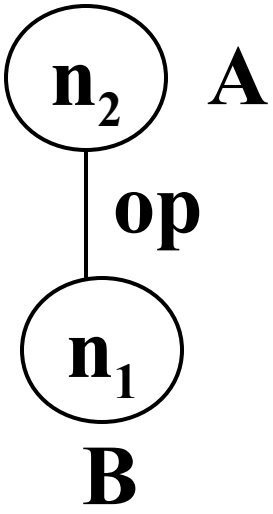
四元式	DAG 图
(0) 0 型： A:=B	 **A**
(:= ， B ， - ， A)	**B**

四元式

DAG 图

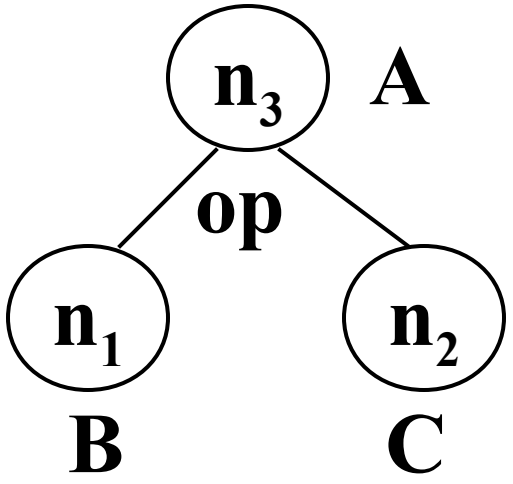
(1) 1 型 : A:=op B

(op , B , - , A)



(2) 2 型 : A:=B op C

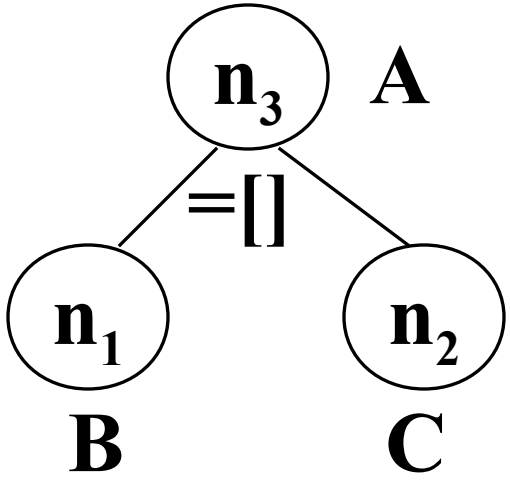
(op , B , C , A)



四元式　　DAG 图

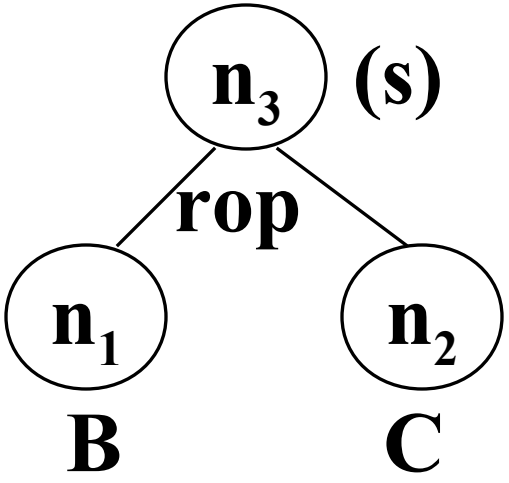
(3) 2 型 : A:=B[C]

(=[] ， B ， C ， A)



(4) 2 型 : if B rop C goto (s)

(jrop ， B ， C ， (s))

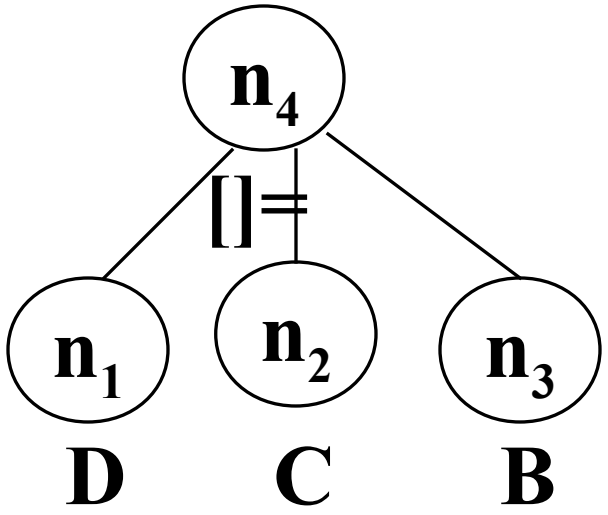


四元式

DAG 图

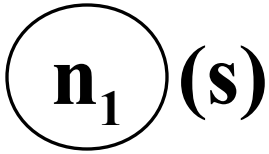
(5) 3 型 : D[C]:=B

([]= , B , D , C)



(6) 0 型 : goto (s)

(j , - , - , (s))



- 假设 DAG 各结点信息将用某种适当的数据结构存放 (如链表) 。另设置一个标识符与结点的对应函数：

$$Node(A)=\begin{cases} n & \text{如果存在一个结点}n\text{，}A\text{是其上的标记或附加标识符} \\ null & \text{否则} \end{cases}$$

- 0，1，2 型四元式的基本块的 DAG 构造算法

对基本块中每一四元式，依次执行以下步骤：

1. 准备操作数的结点
2. 合并已知量
3. 删除公共子表达式
4. 删除无用赋值

1. 准备操作数的结点
2. 合并已知量
3. 删除公共子表达式
4. 删除无用赋值

1. 准备操作数的结点

- 如果 NODE(B) 无定义，则构造一标记为 B 的叶结点并定义 NODE(B) 为这个结点；
  - 如果当前四元式是 0 型，则记 NODE(B) 的值为 n，转 4。
  - 如果当前四元式是 1 型，则转 2(1)
  - 如果当前四元式是 2 型，则 (i) 如果 NODE(C) 无定义，则构造一标记为 C 的叶结点并定义 NODE(C) 为这个结点； (ii) 转 2(2) 。

A:= op B

A:=B op C

1. 准备操作数的结点
2. 合并已知量
3. 删除公共子表达式
4. 删除无用赋值

2. 合并已知量

(1) 如果 NODE(B) 是标记为常数的叶结点，则转 2(3)；否则，转 3(1)

A:= op B

(2) 如果 NODE(B) 和 NODE(C) 都是标记为常数的叶结点，则转 2(4)；否则，转 3(2)

A:=B op C

(3) 执行 op B (即合并已知量) 。令得到的新常数为 P 。如果 NODE(B) 是处理当前四元式时新构造出来的结点，则删除它。如果 NODE(P) 无定义，则构造一用 P 作标记的叶结点 n 。置 NODE(P)=n ，转 4

(4) 执行 B op C (即合并已知量) 。令得到的新常数为 P 。如果 NODE(B) 或 NODE(C) 是处理当前四元式时新构造出来的结点，则删除它。如果 NODE(P) 无定义，则构造一用 P 作标记的叶结点 n 。置 NODE(P)=n ，转 4

1. 准备操作数的结点
2. 合并已知量
3. 删除公共子表达式
4. 删除无用赋值

3. 删除公共子表达式

A:= op B

(1) 检查 DAG 中是否已有一结点，其唯一后继为 NODE(B) 且标记为 op(即公共子表达式) 。如果没有，则构造该结点 n ，否则，把已有的结点作为它的结点并设该结点为 n 。转 4 。

(2) 检查 DAG 中是否已有一结点，其左后继为 NODE(B) ，右后继为 NODE(C) ，且标记为 op(即公共子表达式) 。如果没有，则构造该结点 n ，否则，把已有的结点作为它的结点并设该结点为 n 。转 4 。

A:=B op C

1. 准备操作数的结点
2. 合并已知量
3. 删除公共子表达式
4. 删除无用赋值

4. 删除无用赋值

如果 NODE(A) 无定义，则把 A 附加在结点 n 上并令 NODE(A)=n; 否则，先把 A 从 NODE(A) 结点上的附加标识符集中删除 (注意，如果 NODE(A) 是叶结点，则其 A 标记部不删除) 。把 A 附加到新结点 n 上并置 NODE(A)=n 。转处理下一四元式。

- 例 10.4 试构造以下基本块 G 的 DAG

(1) T_0:=3.14
(2) T_1:=2*T_0
(3) T_2:=R+r
(4) A:=T_1*T_2
(5) B:=A
(6) T_3:=2*T_0
(7) T_4:=R+r
(8) T_5:=T_3*T_4
(9) T_6:=R-r
(10) B:=T_5*T_6

(1) T_0:=3.14

(2) T_1:=2*T_0

(3) T_2:=R+r

(4) A:=T_1*T_2

(5) B:=A

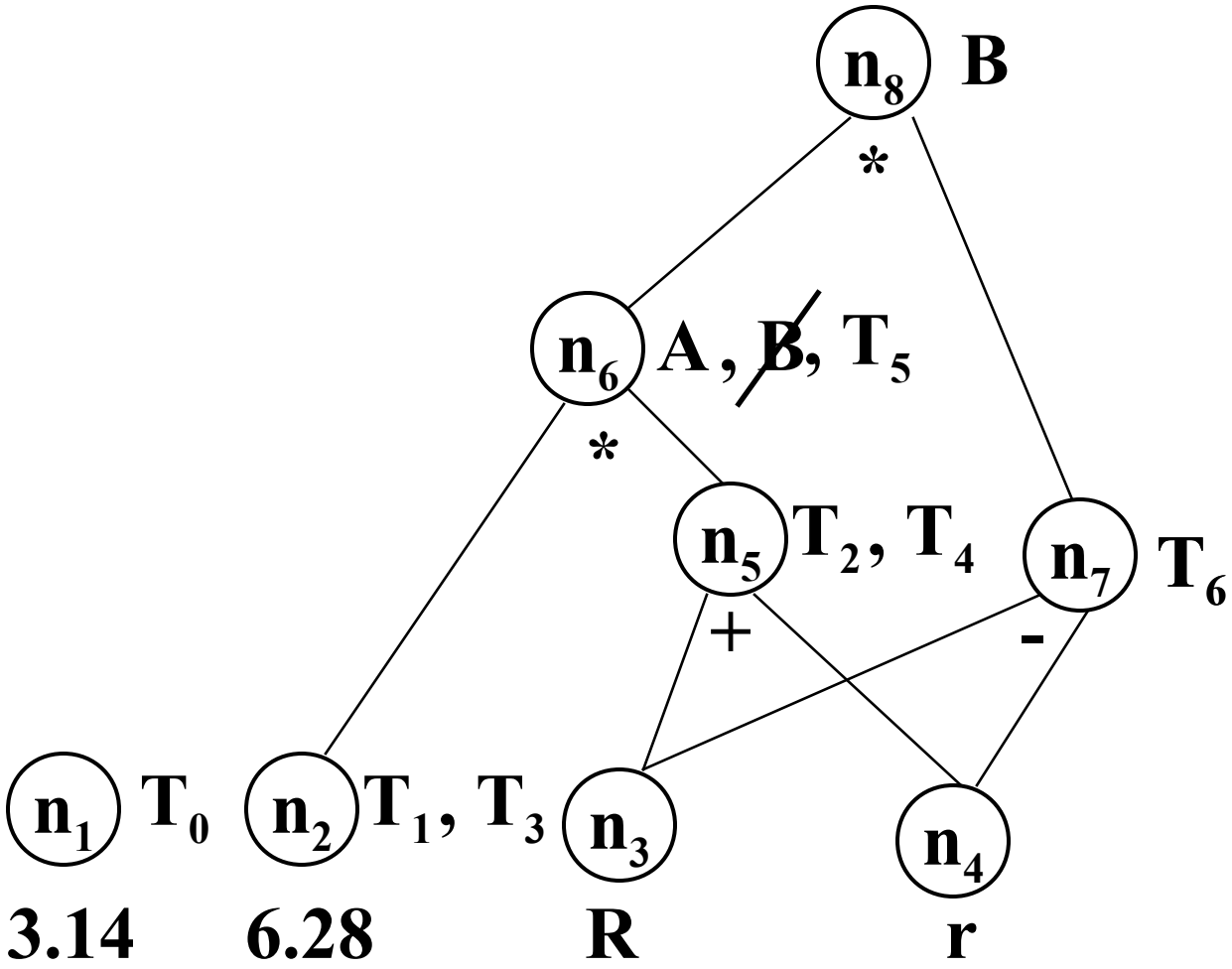
(6) T_3:=2*T_0

(7) T_4:=R+r

(8) T_5:=T_3*T_4

(9) T_6:=R-r

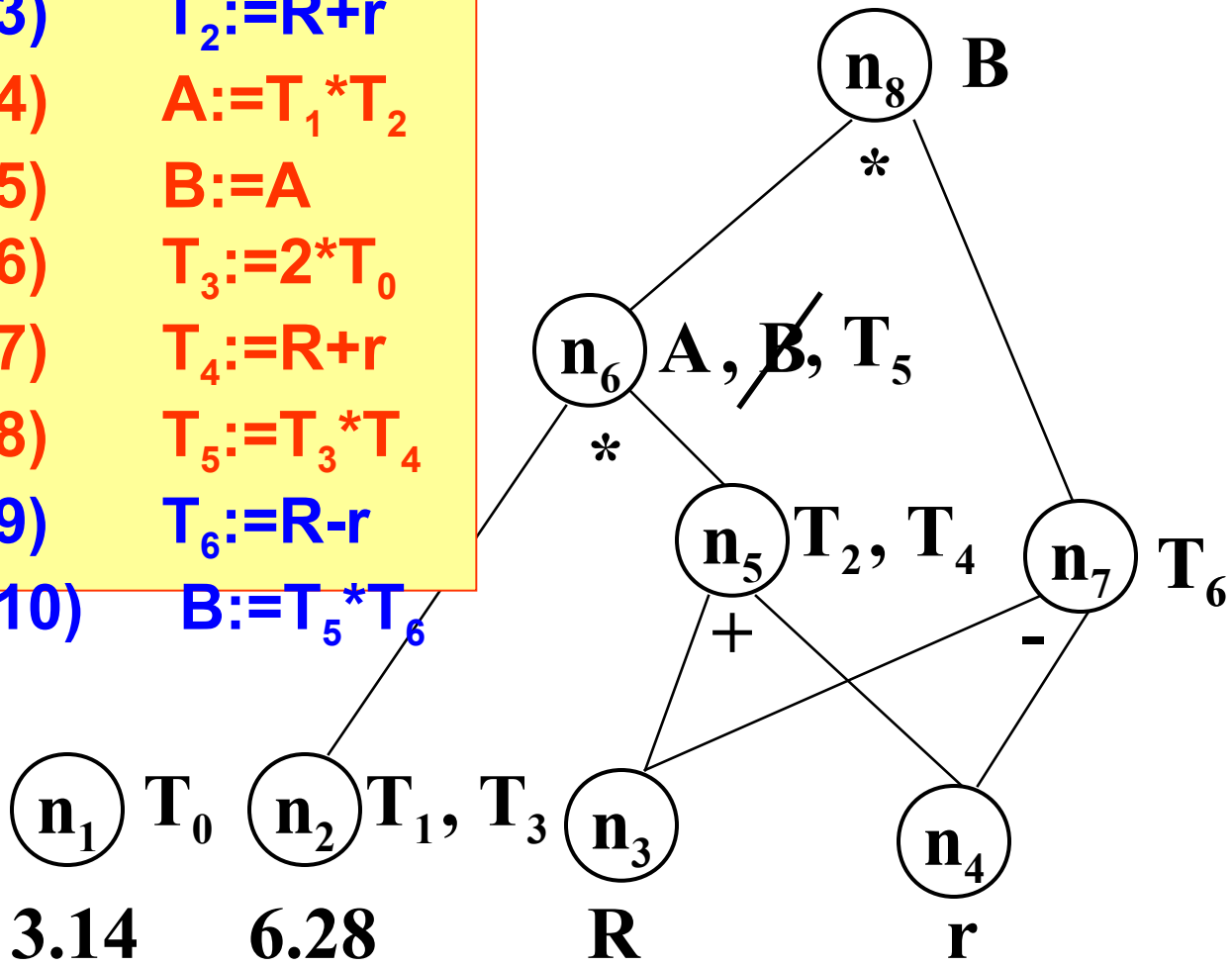
(10) B:=T_5*T_6



- 优化后的四元

(1) T_0:=3.14
(2) T_1:=6.28
(3) T_3:=6.28
(4) T_2:=R+r
(5) T_4:=T_2
(6) A:=6.28*T_2
(7) T_5:=A
(8) T_6:=R-r
(9) B:=A*T_6

(1) T_0:=3.14
(2) T_1:=2*T_0
(3) T_2:=R+r
(4) A:=T_1*T_2
(5) B:=A
(6) T_3:=2*T_0
(7) T_4:=R+r
(8) T_5:=T_3*T_4
(9) T_6:=R-r
(10) B:=T_5*T_6

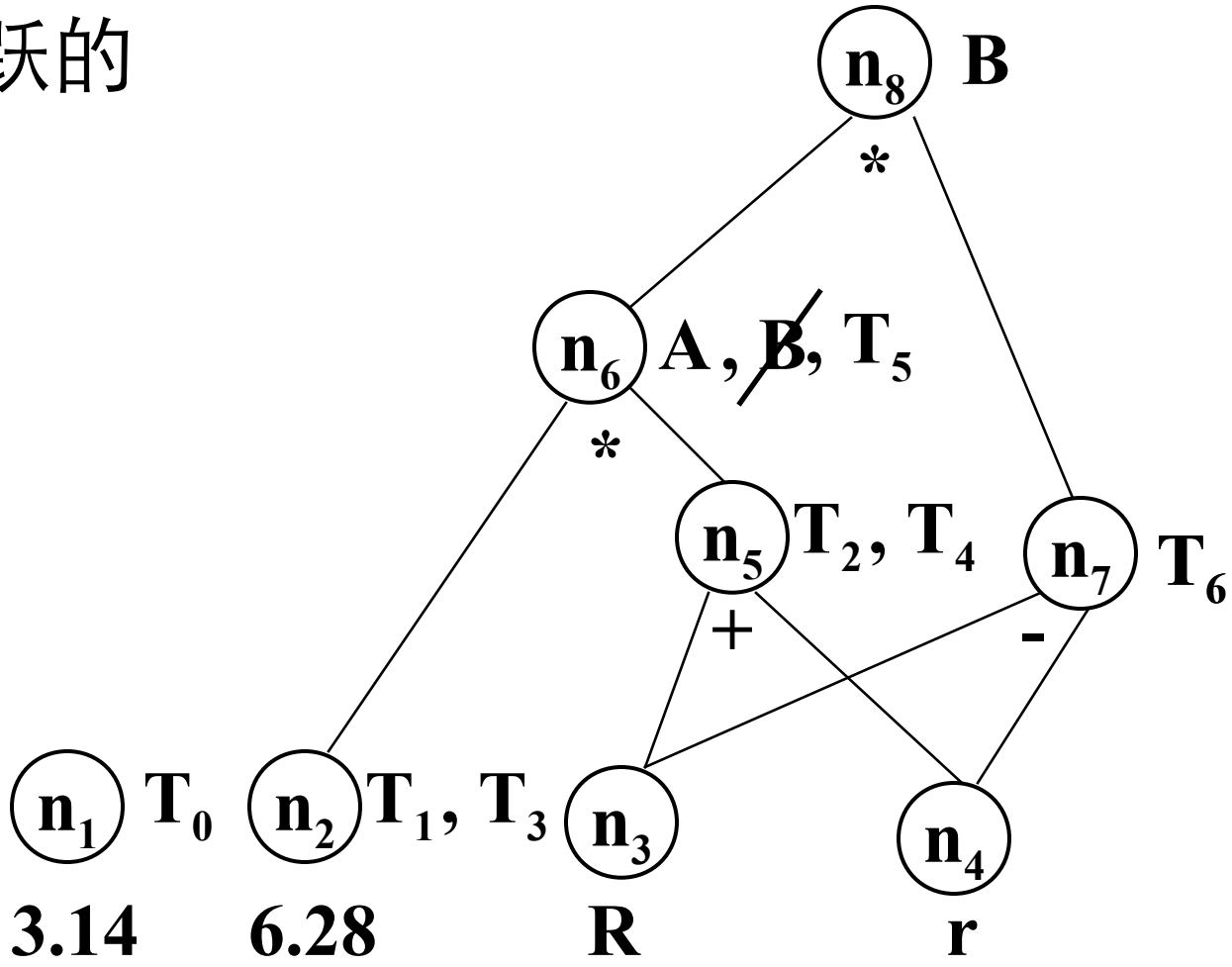


- 优化后的四元式——若只有 A 和 B 是出基本块之后活跃的

(1) T_2:=R+r
(2) A:=6.28*T_2
(3) T_6:=R-r
(4) B:=A*T_6

↓

(1) S_1:=R+r
(2) A:=6.28*S_1
(3) S_2:=R-r



从 DAG 中得到的优化信息

- 在基本块外被定值并在基本块内被引用的所有标识符，就是作为叶子结点上标记的那些标识符
- 在基本块内被定值并且该值在基本块后面可以被引用的所有标识符，就是 DAG 各结点上的那些附加标识符

小结

- 基本块的 DAG 表示及其应用
- 基本块的 DAG 构造算法

作业

- P306-3