课时6 语义分析和中间代码的生成

高数帮

考点	重要程度	占分	题型
1.中间语言	★★★★★	5～10	问答题
2.布尔表达式的翻译	★★★★	5～10	问答题
3.控制语句的翻译	★★★★	5～10	问答题

6.1 中间语言

常用的中间语言：

1.后缀式

2.逆波兰表示

3.图表示： DAG、抽象语法树

4.三地址代码

三元式

四元式

间接三元式

高数帮

一、后缀式

一个表达式E的后缀形式可以如下定义：

1. 如果E是一个变量或常量，则E的后缀式是E自身。
2. 如果E是E_1 op E_2形式的表达式，其中op是任何二元操作符，则E的后缀式为E_1' E_2' op，其中E_1'和E_2'分别为E_1和E_2的后缀式。
3. 如果E是(E_1)形式的表达式，则E_1的后缀式就是E的后缀式。

高数帮

二、逆波兰表示法

不用括号。只要知道每个算符的目数，对于后缀式，不论从哪一端进行扫描，都能对它进行唯一分解。

三、后缀式的计算

用一个栈实现。

一般的计算过程是：自左至右扫描后缀式，每碰到运算量就把它推进栈。每碰到k目运算符就把它作用于栈顶的k个项，并用运算结果代替这k个项。

把表达式翻译成后缀式的语义规则描述

产生式	语义动作
$E \rightarrow E^{(1)} op\ E^{(2)}$	$E.code := E^{(1)}.code \parallel E^{(2)}.code \parallel op$
$E \rightarrow (E^{(1)})$	$E.code := E^{(1)}.code$
$E \rightarrow id$	$E.code := id$

E.code表示E后缀形式

op表示任意二元操作符

“||”表示后缀形式的连接。

E→E(1)op E(2)	E.code:= E(1).code || E(2).code ||op
E→ (E(1))	E.code:= E(1).code
E→id	E.code:=id

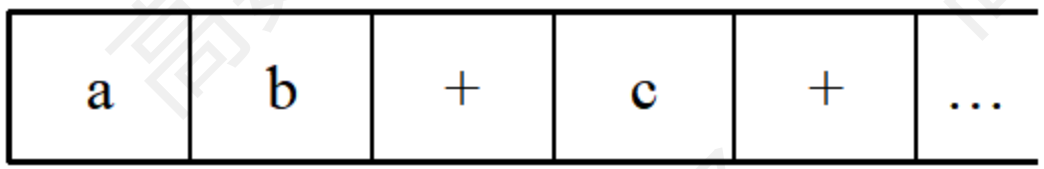
数组POST存放后缀式：k为下标，初值为1

上述语义动作可实现为：

产生式	程序段
$E \rightarrow E^{(1)} op\ E^{(2)}$	{POST[k]:=op;k:=k+1}
$E \rightarrow (E^{(1)})$	{}
E→i	{POST[k]:=i;k:=k+1}

如：输入串a+b+c的分析和翻译

POST：　1　2　3　4　5



四、图表示法

1.图表示法：DAG、抽象语法树

2.无循环有向图(Directed Acyclic Graph，简称DAG)

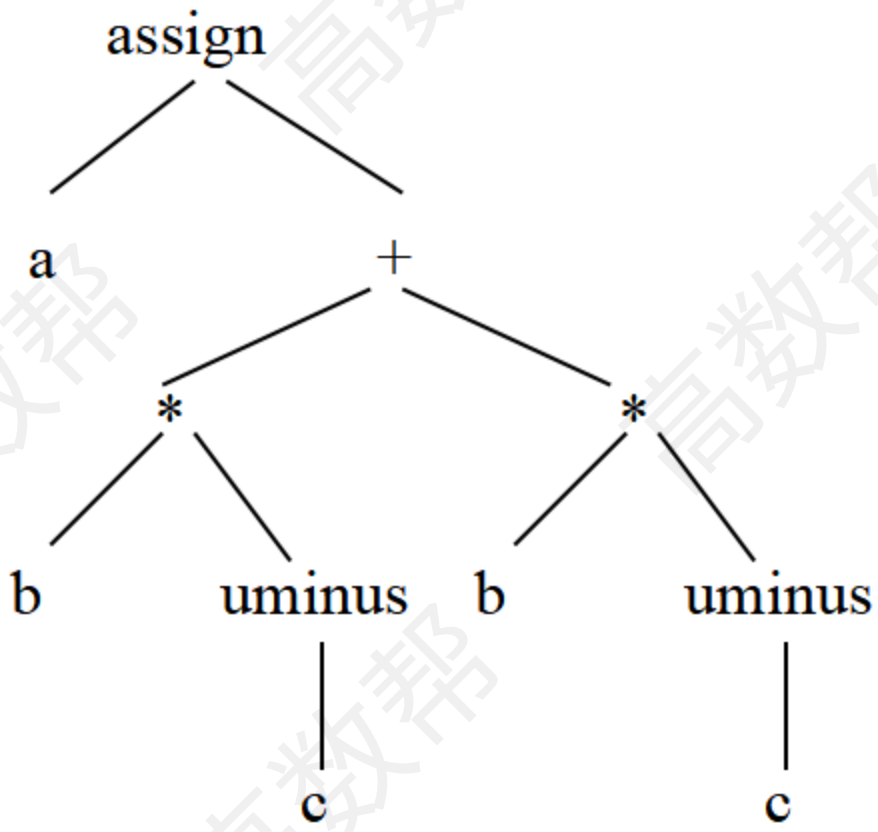
对表达式中的每个子表达式，DAG中都有一个结点

一个内部结点代表一个操作符，它的孩子代表操作数

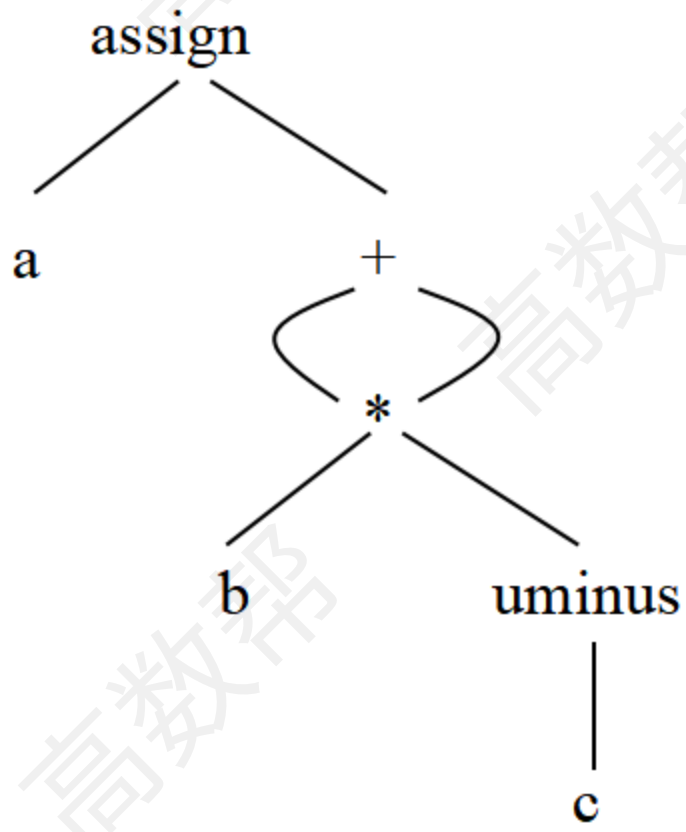
在一个DAG中代表公共子表达式的结点具有多个父结点

高数帮

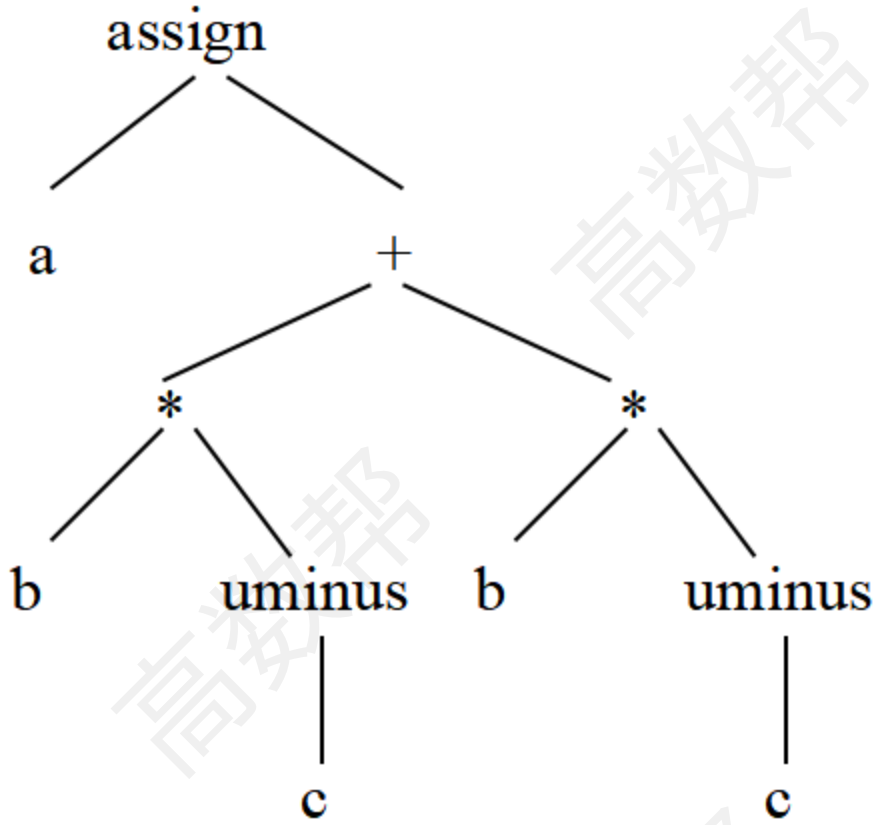
a:=b*(-c)+b*(-c)的图表示法



抽象语法树



DAG



抽象语法树

抽象语法树对应的代码：

T_1:=-c

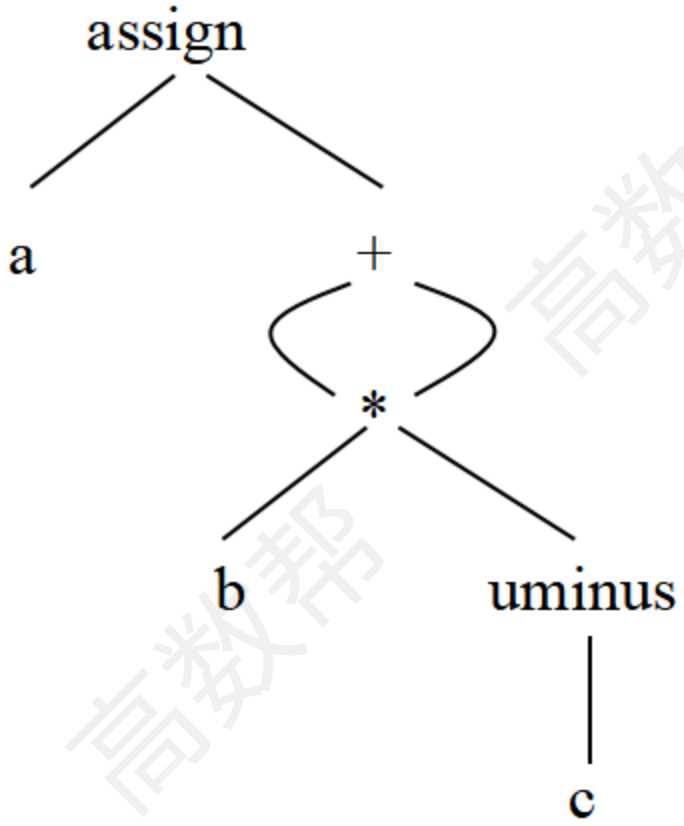
T_2:=b*T_1

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

五、三地址代码

三地址代码

x:=y op z

三地址代码可以看成是抽象语法树或DAG的一种线性表示

视频讲解更清晰

高数帮

1.三元式

通过计算临时变量值的语句的位置来引用这个临时变量

三个域：op、arg1和arg2

	op	arg1	arg2
(0)	uminus	c	
(1)	*	b	(0)
(2)	uminus	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	assign	a	(4)

高数帮

【题1】

1.x[i]:=y

	op	arg1	arg2
(0)	[ ] =	x	i
(1)	assign	(0)	y

2.x:=y[i]

	op	arg1	arg2
(0)	= [ ]	y	i
(1)	assign	x	(0)

高数帮

2.四元式

一个带有四个域的记录结构，这四个域分别称为op, arg1, arg2及result

	op	arg1	arg2	result
(0)	uminus	c		T_1
(1)	*	b	T_1	T_2
(2)	uminus	c		T_3
(3)	*	b	T_3	T_4
(4)	+	T_2	T_4	T_5
(5)	:=	T_5		a

3.间接三元式

为了便于优化，用 三元式表+间接码表 表示中间代码

间接码表:一张指示器表，按运算的先后次序列出有关三元式在三元式表中的位置。

优点: 方便优化，节省空间

如，语句

a:=b*(-c)+b*(-c)

的间接三元式表示如表所示。

	op	arg1	arg2
(0)	uminus	c	
(1)	*	b	(0)
(2)	+	(1)	(1)
(3)	assign	a	(2)

间接码表(0) (1) (0) (1) (2) (3)

6.2 布尔表达式的翻译

1.布尔表达式的两个基本作用:

1)用于逻辑演算,计算逻辑值;

2)用于控制语句的条件式.

2.产生布尔表达式的文法:

$$E \rightarrow E \text{ or } E \mid E \text{ and } E \mid \neg E \mid (E) \mid i \text{ rop } i \mid i$$

视频讲解更清晰

高数帮

一、计算布尔表达式通常采用两种方法:

1. 如同计算算术表达式一样,一步步算

1 or (not 0 and 0) or 0

=1 or (1 and 0) or 0

=1 or 0 or 0

=1 or 0

=1

2. 采用某种优化措施

把A or B解释成　　if A then true else B

把A and B解释成　　if A then B else false

把¬ A解释成　　if A then false else true

高数帮

二、两种不同的翻译方法

第一种翻译法：

A or B and C=D翻译成

(1) (=, C, D, T_1)

(2) (and, B, T_1, T_2)

(3) (or, A, T_2, T_3)

第二种翻译法适合于作为条件表达式的布尔表达式使用.

高数帮

(1) 数值表示法

a or b and not c 翻译成

T_1:=not c

T_2:=b and T_1

T_3:=a or T_1

a<b的关系表达式可等价地写成

if a<b then 1 else 0，翻译成

```
100:    if a<b goto 103
101:    T:=0
102:    goto 104
103:    T:=1
104:
```

布尔表达式a<b or c<d and e<f的翻译结果

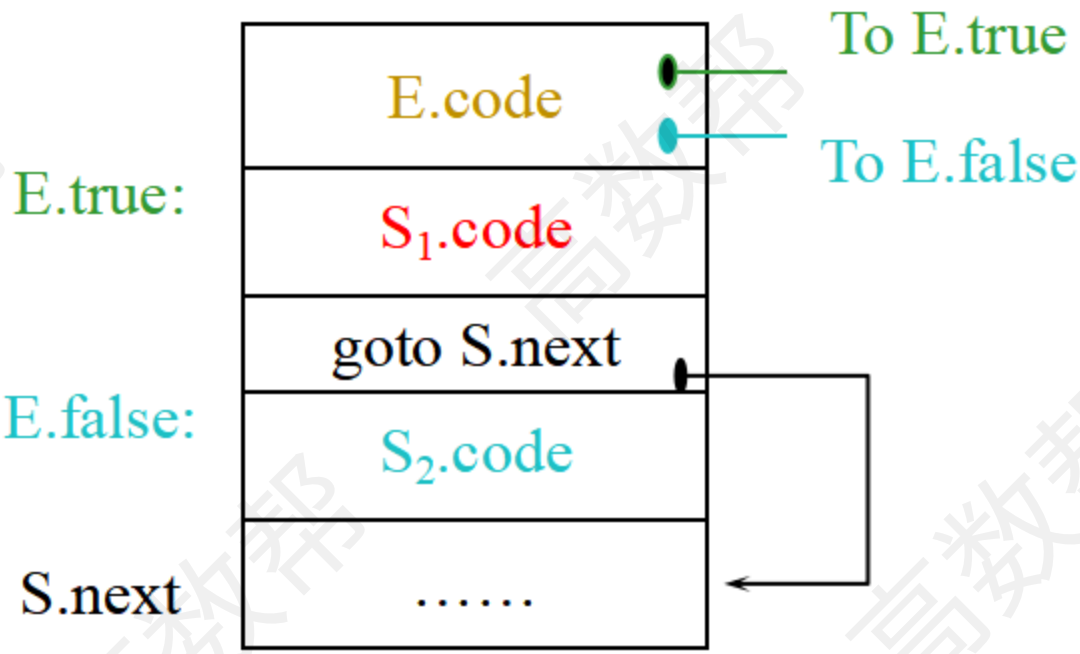
100: if a<b goto 103
101: T_1:=0
102: goto 104
103: T_1:=1
104: if c<d goto 107
105: T_2:=0
106: goto 108
107: T_2:=1
108: if e<f goto 111
109: T_3:=0
110: goto 112
111: T_3:=1
112: T_4:=T_2 and T_3
113: T_5:=T_1 or T_4

高数帮

(2) 作为条件控制的布尔式翻译

条件语句 if E then S_1 else S_2

赋予 E 两种出口:一真一假



【题1】：把语句：if a>c or b <d then S_1 else S_2翻译成如下的一串三地址代码

	if a>c goto L2 “真”出口
	goto L1
L1:	if b<d goto L2 “真”出口
	goto L3 “假”出口
L2:	(关于S1的三地址代码序列)
	goto Lnext
L3:	(关于S2的三地址代码序列)
Lnext:	

高数帮

考虑如下表达式：

a<b or c<d and e<f

假定整个表达式的真假出口已分别置为Ltrue和Lfalse，则按定义将生成如下的代码：

```
        if a<b goto Ltrue
        goto L1
L1:     if c<d goto L2
        goto Lfalse
L2:     if e<f goto Ltrue
        goto Lfalse
```

三、布尔表达式的翻译

1.两遍扫描

为给定的输入串构造一棵语法树；

对语法树进行深度优先遍历，进行语义规则中规定的翻译。

2.一遍扫描

视频讲解更清晰

高数帮

2.一遍扫描实现布尔表达式的翻译

1）采用四元式形式

2）把四元式存入一个数组中，数组下标就代表四元式的标号

3）约定

四元式(jnz, a, -, p)　　表示　if a goto p

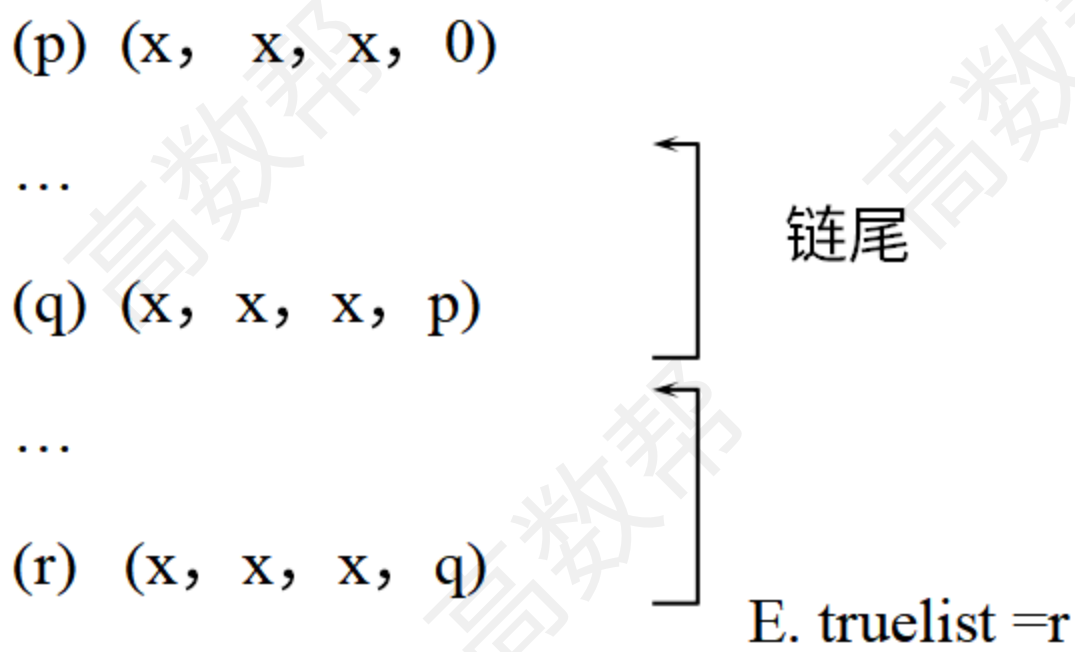
四元式(jrop, x, y, p)　　表示　if x rop y goto p

四元式(j, -, -, p)　　表示　goto p

有时,四元式转移地址无法立即知道,我们只好把这个未完成的四元式地址作为E的语义值保存,待机"回填"。

为非终结符E赋予两个综合属性E.truelist和E.falselist。它们分别记录布尔表达式E所应的四元式中需回填“真”、“假”出口的四元式的标号所构成的链表

比如:假定E的四元式中需要回填"真"出口的p，q，r三个四元式，则E.truelist为下列链:



a<b or c<d and e<f

100(j<, a, b, 0)

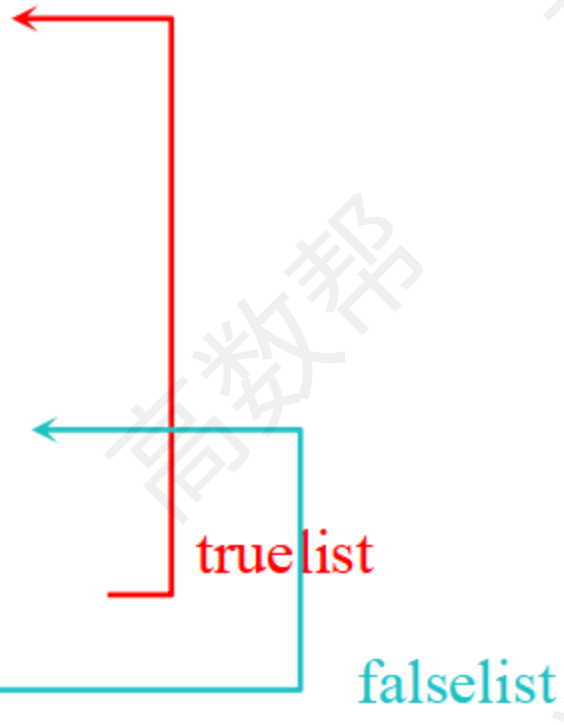
101(j, -, -, 102)

102(j<, c, d, 104)

103(j, -, -, 0)

104(j<, e, f, 100)

105(j, -, -, 103)



高数帮

6.3 控制语句的翻译

考虑如下语句：

```
while a<b do
        if c<d then    x:=y+z
               else    x:=y-z
```

生成下列代码：

```
L1:     if a<b goto L2
        goto Lnext
L2:     if c<d goto L3
        goto L4
L3:     T1:=y+z
        x:=T1
        goto L1
L4:     T2:=y-z
        x:=T2
        goto L1
Lnext:
```

翻译语句

```
while (a<b) do
        if (c<d) then x:=y+z;
```

100(j<, a, b, 102)

101(j, -, -, 107)

102(j<, c, d, 104)

103(j, -, -, 100)

104(+, y, z, T)

105(:=, T, -, x)

106 (j, -, -, 100)

107