HW7

1. 请写出下面的变量a的类型表达式

```
1. int a[][3];
2. int *a[3];
3. int (*a)[3];
4. int *(*a)[3];
5. int **a[3];
6. int *(*a[3])[2];
```

HW7

(1) pointer(array(3, integer))

(2) array(3, pointer(integer))

(3) pointer(array(3, integer))

(4) pointer(array(3, pointer(integer)))

(5) array(3, pointer(pointer(integer)))

(6) array(3, pointer(array(2, pointer(integer))))

- 参考资料：http://unixwiz.net/techtips/reading-cdecl.html

HW7


```
1.  P --> D; E
2.  D --> D; D | id : T
3.  T --> list of T | char | integer
4.  E --> (L) | literal | num | id | nil
5.  L --> E, L | E
```

写一个类似5.3节中的翻译方案，以确定表达式(E)和表(L)的类型。

产生式	翻译方案
$P \rightarrow D; E$	
$D \rightarrow D; D$	
$D \rightarrow id : T$	{addtype(id.entry,T.type);}
$T \rightarrow list\ of\ T_1$	{T.type=list(T_1. *type*);}
$T \rightarrow char$	{T.type=char;}
$T \rightarrow integer$	{T.type=integer;}
$E \rightarrow id$	{E.type=lookup(id.entry);}
$E \rightarrow num$	{E.type=integer;}
$E \rightarrow nil$	{E.type=void;}
$E \rightarrow literal$	{E.type=char;}
$E \rightarrow (L)$	{E.type=list(L.type);}
$L \rightarrow E$	{L.type=E.type;}
$L \rightarrow E, L_1$	{if(E.type==L_1.type){L.type=E.type;}else{L.type=type_error;}};

HW8

教材5.15 找出下列表达式的最一般的合一代换:

(a) (pointer (α)) $\times$ ($\beta \rightarrow \gamma$)

(b) $\beta \times (\gamma \rightarrow \delta)$

如果(b)的δ是α呢?

HW8

$$\alpha \to \alpha$$

$$\beta \to pointer(\alpha)$$

$$\gamma \to pointer(\alpha)$$

$$\delta \to pointer(\alpha)$$

HW8

$$\alpha \rightarrow pointer(\alpha)$$

$$\beta \rightarrow pointer(\alpha)$$

$$\gamma \rightarrow pointer(\alpha)$$

$$\delta \rightarrow pointer(\alpha)$$

HW8

教材 5.17 效仿例5.5，推导下面map的多态类型：

map : $\forall\alpha.\forall\beta.((\alpha \to \beta) \times \text{list}(\alpha)) \to \text{list}(\beta)$

map的ML定义是

```
1. fun map (f, l ) =
2.    if null (l ) then nil
3.    else cons (f (hd (l)), map (f, tl (l ) ) );
```

在这个函数体中，内部定义的标识符的类型是：

null : $\forall\alpha.\text{list}(\alpha) \to \text{boolean}$;

nil : $\forall\alpha.\text{list}(\alpha)$;

cons : $\forall\alpha.(\alpha \times \text{list}(\alpha)) \to \text{list}(\alpha)$;

hd : $\forall\alpha.\ \text{list}(\alpha) \to \alpha$;

tl : $\forall\alpha.\ \text{list}(\alpha) \to \text{list}(\alpha)$;

HW8

序号	定型断言	代换	规则
1	f : α		Exp Id
2	l : β		Exp Id
3	map : γ		Exp Id
4	map(f, l) : δ	γ = (α × β) → δ	Exp FunCall
5	null : list(α_0) → boolean		Exp Id Fresh
6	null(l) : boolean	β = list(α_0)	Exp FunCall + (2)
7	nil : list(α_1)		Exp Id Fresh
8	l : list(α_0)		(2)
9	hd : list(α_2) → α_2		Exp Id Fresh
10	hd(l) : α_0	$\alpha_2 = \alpha_0$	Exp FunCall
11	f(hd(l)) : α_3	α = α_0 → α_3	Exp Id
12	f : α_0 → α_3		(1)
12	f : α_0 → α_3		(1)
13	tl : list(α_4) → list(α_4)		Exp Id Fresh
14	tl(l) : list(α_0)	$\alpha_4 = \alpha_0$	Exp FunCall
15	map : ((α_0 → α_3) × list(α_0)) → δ		(3)
16	map(f, tl(l)) : δ		Exp FunCall
17	cons : α_5 × list(α_5) → list(α_5)		Exp Id Fresh
18	cons : list(α_3)	$\alpha_5 = \alpha_3$, δ = list(α_3)	Exp FunCall
19	if : boolean × list(α_6) × list(α_6) → list(α_6)		Exp Id Fresh
20	if : list(α_1)		Exp FunCall
21	match : α_7 × α_7 → α_7		Exp Id Fresh
22	match : list(α_1)	α_7 = list(α_1)	Exp FunCall

故 `map : ((α₀ → α₁) × list(α₀)) → list(α₁)`

即 `map : ∀α.∀β.((α→β)×list(α)) → list (β)`

HW8

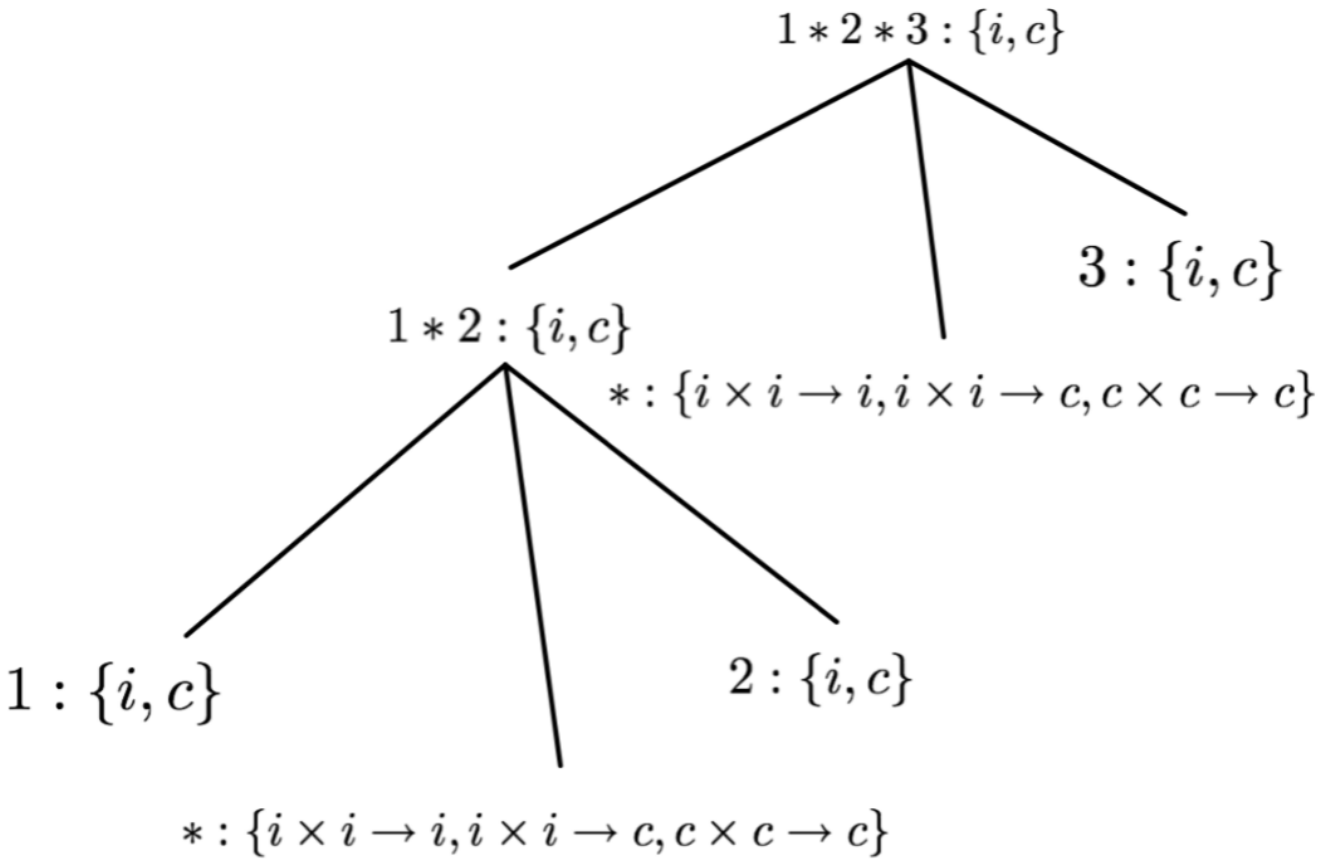
习题 5.21 使用例5.9的规则，确定下列哪些表达式有唯一类型（假定z是复数）：

(a) 1 * 2 * 3

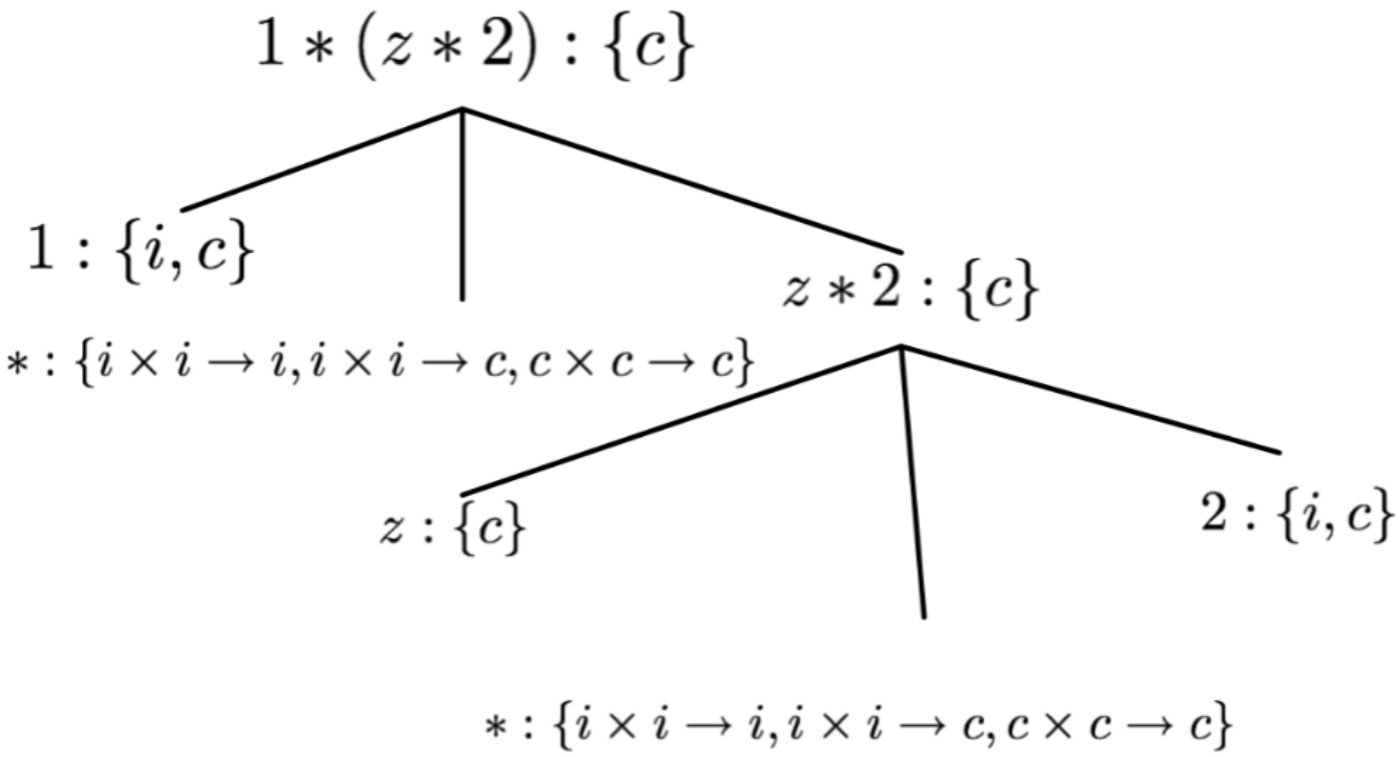
(b) 1 * (z * 2)

(c) (1 * z) * z

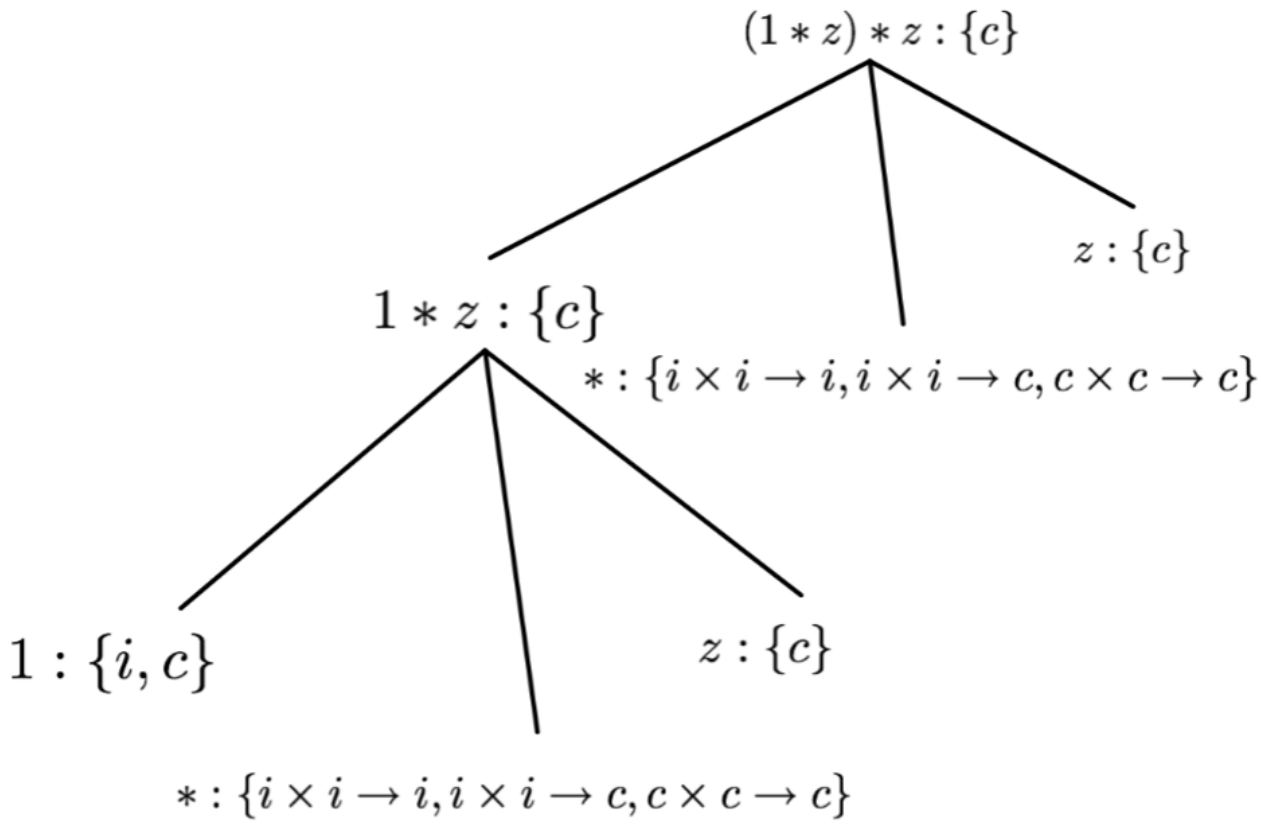
HW8


$1*2*3:\{i,c\}$
$3:\{i,c\}$
$1*2:\{i,c\}$
$*:\{i\times i\to i, i\times i\to c, c\times c\to c\}$
$1:\{i,c\}$
$2:\{i,c\}$
$*:\{i\times i\to i, i\times i\to c, c\times c\to c\}$

HW8


$1 * (z * 2) : \{c\}$
$1 : \{i, c\}$
$* : \{i \times i \to i, i \times i \to c, c \times c \to c\}$
$z * 2 : \{c\}$
$z : \{c\}$
$2 : \{i, c\}$
$* : \{i \times i \to i, i \times i \to c, c \times c \to c\}$

HW8


$(1 * z) * z : \{c\}$
$1 * z : \{c\}$
$z : \{c\}$
$* : \{i \times i \to i, i \times i \to c, c \times c \to c\}$
$1 : \{i, c\}$
$z : \{c\}$
$* : \{i \times i \to i, i \times i \to c, c \times c \to c\}$

HW9-1

- 7.4 修改计算声明名字的类型和相对地址的翻译方案或者树访问代码，允许名字表而不是单个名字出现在形式为 D->id:T 的声明中。

HW9-1


```
D->L:T {
    for(i in L.list){
        enter(i,T.type,offset);
        offset+=T.width;
    }
}

L->L1,id {L.list=L1.list;L.list.push(id.lexeme);}

L->id {L.list=new list();L.list.push(id.lexeme);}
```

HW9-1

$D \rightarrow L:T$	{ L.type = T.type ; L.width = T.width }
$L \rightarrow id,L1$	{ enter (id.lexeme , L.type , offset) ;
	offset = offset + L.width ;
	L1.type = L.type ;L1.width = L.width }
$L \rightarrow id$	{ enter (id.lexeme , L.type , offset) ;
	offset = offset + L.width }