

# 编译原理与技术 H7

PB18111697 王章瀚

•

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];
```

a.type 应为:

1. array(?,array(3,int)), 这里用? 表示是因为确实不知道大小, 比如在 C 中直接这样写会报错.  
(或者干脆写成 pointer(array(3,int)))
2. array(3,pointer(int))
3. pointer(array(3,int))
4. pointer(array(3,pointer(int)))
5. array(3,pointer(pointer(int)))

## 教材 5.6

下列文法定义字面常量表. 符号的解释和图 5.2 文法的那些相同, 增加类型 list, 它表示类型 T 的元素表.

$$\begin{aligned}P &\rightarrow D; E \\D &\rightarrow D; D | id : T \\T &\rightarrow list\ of\ T | char | integer \\E &\rightarrow (L) | literal | num | id | nil \\L &\rightarrow E, L | E\end{aligned}$$

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.

可以写成:

$P \rightarrow D; E$	
$D \rightarrow D; D$	
$D \rightarrow id : T$	$\{addtype(id.entry, T.type); \}$
$T \rightarrow list\ of\ T_1$	$\{T.type = list(T_1.type)\}$
$T \rightarrow char$	$\{T.type = char; \}$
$T \rightarrow integer$	$\{T.type = integer; \}$
$E \rightarrow (L)$	$\{E.type = list(T.type)\}$
$E \rightarrow literal$	$\{E.type = char\}$
$E \rightarrow num$	$\{E.type = integer; \}$
$E \rightarrow id$	$\{E.type = looup(id.entry); \}$
$E \rightarrow nil$	$\{E.type = list(?)\}$
$L \rightarrow E, L_1$	$\{L.type = if(L_1.type == E.type)L_1.type\ else\ type_{error}\}$
$L \rightarrow E$	$\{L.type = E.type\}$

## 教材 5.15

找出下列表式的最一般的合一代换:

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

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

如果 (b) 的  $\delta$  是  $\alpha$  呢?

(1). 原来的: 最一般的合一代换就是

$$S(\beta) = (pointer(\alpha))$$

$$S(\gamma) = (pointer(\alpha))$$

$$S(\delta) = (pointer(\alpha))$$

最后能代换成:

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

(2). 如果 (b) 的  $\delta$  是  $\alpha$ , 就变成

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

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

这时它不存在合一代换. 因为如果要合一, 就要有

$$S(\beta) = S(pointer(\alpha))$$

$$S(\beta) = S(\gamma)$$

$$S(\gamma) = S(\alpha)$$

从而需要  $S(\alpha) = S(pointer(\alpha))$ , 这要求一个类型的指针类型依然是这个类型, 显然不合理.

## 教材 5.17

效仿例 5.5, 推导下面 map 的多态类型:

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

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, 内部定义的标志符类型是:

$\text{null} : \forall \alpha. \text{list}(\alpha) \rightarrow \text{boolean};$   
 $\text{nil} : \forall \alpha. \text{list}(\alpha);$   
 $\text{cons} : \forall \alpha. (\alpha \times \text{list}(\alpha)) \rightarrow \text{list}(\alpha);$   
 $\text{hd} : \forall \alpha. \text{list}(\alpha) \rightarrow \alpha;$   
 $\text{tl} : \forall \alpha. \text{list}(\alpha) \rightarrow \text{list}(\alpha);$

答:

整个推导过程如下:

行	定型断言	代换	规则
1	$f : \gamma$		Exp Id
2	$l : \delta$		Exp Id
3	$\text{map} : \epsilon$		Exp Id
4	$\text{map}(f, l) : \zeta$	$\epsilon = \gamma \times \delta \rightarrow \zeta$	Exp FunCall
5	$\text{null} : \text{list}(\alpha_1) \rightarrow \text{boolean}$		Exp Id Fresh
6	$\text{null}(l) : \text{boolean}$	$\delta = \text{list}(\alpha_1)$	Exp FunCall
7	$\text{nil} : \text{list}(\alpha_2)$		Exp Id Fresh
8	$\text{hd} : \text{list}(\alpha_3) \rightarrow \alpha_3$		Exp Id Fresh
9	$\text{hd}(l) : \alpha_3$	$\alpha_3 = \alpha_1$	Exp FunCall
10	$f(\text{hd}(l)) : \alpha_4$	$\gamma = \alpha_1 \rightarrow \alpha_4$	Exp FunCall
11	$\text{tl} : \text{list}(\alpha_5) \rightarrow \text{list}(\alpha_5)$		Exp Id
12	$\text{tl}(l) : \text{list}(\alpha_5)$	$\alpha_5 = \alpha_1$	Exp Id
13	$\text{map}(f, \text{tl}(l)) : \zeta_1$	$\zeta_1 = \zeta$	Exp FunCall
14	$\text{cons} : (\alpha_6 \times \text{list}(\alpha_6)) \rightarrow \text{list}(\alpha_6)$		Exp Id Fresh
15	$\text{cons}(f(\text{hd}(l)), \text{map}(f, \text{tl}(l))) : \text{list}(\alpha_6)$	$\alpha_6 = \alpha_4$ $\zeta = \text{list}(\alpha_4)$	Exp FunCall
16	$\text{if } \text{boolean} \times \lambda \times \lambda \rightarrow \lambda$	$\lambda = \text{list}(\alpha_4)$	Exp Id Fresh
17	$\text{if}(\dots) : \text{list}(\alpha_4)$	$\alpha_4 = \alpha_2$	Exp FunCall
18	$\text{match} : \mu \times \mu \rightarrow \mu$		Exp Id Fresh
19	$\text{match}(\dots) : \text{list}(\alpha_2)$	$\alpha_8 = \alpha_2$	Exp FunCall

这样, 就推断出来了

$$f : \alpha_1 \rightarrow \alpha_4$$

$$l : list(\alpha_1)$$

$$map : ((\alpha_1 \rightarrow \alpha_4) \times list(\alpha_1)) \rightarrow list(\alpha_4)$$

从而有

$$map : \forall \alpha. \forall \beta. ((\alpha \rightarrow \beta) \times list(\alpha)) \rightarrow list(\beta)$$