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

```
int a[][3];
int *a[3];
int (*a)[3];
int *(*a)[3];
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的表. 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.

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

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:

$$
\begin{array}{ll}
P \to D; E & \\
D \to D; D & \\
D \to id : T & \{addtype(id.entry, T.type);\} \\
T \to list\ of\ T_1 & \{T.type = list(T_1.type)\} \\
T \to char & \{T.type = char;\} \\
T \to integer & \{T.type = integer;\} \\
E \to (L) & \{E.type = list(T.type)\} \\
E \to literal & \{E.type = char\} \\
E \to num & \{E.type = integer;\} \\
E \to id & \{E.type = looup(id.entry);\} \\
E \to nil & \{E.type = list(?)\} \\
L \to E, L_1 & \{L.type = if(L_1.type == E.type)L_1.type\ else\ type_error\} \\
L \to E & \{L.type = E.type\}
\end{array}
$$

教材5.15

找出下列表达式的最一般的合一代换:

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

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

如果(b)的δ是α呢?

(1). 原来的: 最一般的合一代换就是

$$
\begin{aligned}
S(\beta) &= (pointer(\alpha)) \\
S(\gamma) &= (pointer(\alpha)) \\
S(\delta) &= (pointer(\alpha))
\end{aligned}
$$

最后能代换成:

$$(pointer(\alpha)) \times (pointer(\alpha) \to pointer(\alpha))$$

(2). 如果(b)的δ是α, 就变成

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

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

这时它不存在合一代换. 因为如果要合一, 就要有

$$
\begin{aligned}
S(\beta) &= S(pointer(\alpha)) \\
S(\beta) &= S(\gamma) \\
S(\gamma) &= S(\alpha)
\end{aligned}
$$

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:

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

map的ML定义是:

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

在这个函数体中, 内部定义的标志符类型是:

$$null : \forall\alpha.list(\alpha) \to boolean;$$
$$nil : \forall\alpha.list(\alpha);$$
$$cons : \forall\alpha.(\alpha \times list(\alpha)) \to list(\alpha);$$
$$hd : \forall\alpha.list(\alpha) \to \alpha;$$
$$tl : \forall\alpha.list(\alpha) \to list(\alpha);$$

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	$map : \epsilon$		Exp Id
4	$map(f,l) : \zeta$	$\epsilon = \gamma \times \delta \to \zeta$	Exp FunCall
5	$null : list(\alpha_1) \to boolean$		Exp Id Fresh
6	$null(l) : boolean$	$\delta = list(\alpha_1)$	Exp FunCall
7	$nil : list(\alpha_2)$		Exp Id Fresh
8	$hd : list(\alpha_3) \to \alpha_3$		Exp Id Fresh
9	$hd(l) : \alpha_3$	$\alpha_3 = \alpha_1$	Exp FunCall
10	$f(hd(l)) : \alpha_4$	$\gamma = \alpha_1 \to \alpha_4$	Exp FunCall
11	$tl : list(\alpha_5) \to list(\alpha_5)$		Exp Id
12	$tl(l) : list(\alpha_5)$	$\alpha_5 = \alpha_1$	Exp Id
13	$map(f, tl(l)) : \zeta_1$	$\zeta_1 = \zeta$	Exp FunCall
14	$cons : (\alpha_6 \times list(\alpha_6)) \to list(\alpha_6)$		Exp Id Fresh
15	$cons(f(hd(l)), map(f, tl(l))) : list(\alpha_6)$	$\alpha_6 = \alpha_4$ $\zeta = list(\alpha_4)$	Exp FunCall
16	$if\ boolean \times \lambda \times \lambda \to \lambda$	$\lambda = list(\alpha_4)$	Exp Id Fresh
17	$if(...) : list(\alpha_4)$	$\alpha_4 = \alpha_2$	Exp FunCall
18	$match : \mu \times \mu \to \mu$		Exp Id Fresh
19	$match(...) : list(\alpha_2)$	$\alpha_8 = \alpha_2$	Exp FunCall

这样, 就推断出来了

$$
\begin{aligned}
&f : \alpha_1 \to \alpha_4 \\
&l : list(\alpha_1) \\
&map : ((\alpha_1 \to \alpha_4) \times list(\alpha_1)) \to list(\alpha_4)
\end{aligned}
$$

从而有

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