Compile Principle assignment

Author Haihan Gao

(1)pointer(array(3, int))

第一题

```
(2)pointer(3,pointer(int))
(3)pointer(array(3,int))
(4)pointer(array(3,pointer(int)))
(5)pointer(3,pointer(pointer(int)))
教材5.6
P->D:E
D->D<sub>1</sub>;D<sub>2</sub>
D->id:T {addtype(id.entry,T.type);}
T->list of T_1 {T.type=T_1.type;}
T->char {T.type=char;}
T->integer {T.type=integer;}
E->(L) {E.type=L.type;};
E->literal {E.type=literal;}
E->num {E.type=num;}
E->id {E.type=lookup(id.entry);}
E->nil {E.type=nil}
L->E,L_1 {L.type=L_1.type;}
L->E {L.type=E.type;}
教材5.15
\beta = (pointer(\alpha)) \ \gamma = pointer(\alpha) \ \delta = pointer(\alpha)
```

教材5.17

将 β 修改为 α

```
这里面函数参数为f和l,假设f和l的类型是\alpha \beta Step1 列出类型声明 f:\alpha l:\beta
```

```
\begin{split} null : \forall \; \alpha. \, list(\alpha) -> boolean \\ nil : \forall \; \alpha. \, list(\alpha) \\ cons : \forall \alpha. \, (\alpha \times list(\alpha)) -> list(\alpha) \\ hd : \forall \alpha. \, list(\alpha) -> \alpha \\ \forall \alpha. \, list(\alpha) -> list(\alpha) \\ if : \forall boollean \times list(\alpha) \times list(\alpha) -> (\forall \alpha. \, list(\alpha) -> boolean) \\ \text{match} : \forall \alpha \forall \beta -> list(\alpha) \end{split}
\begin{array}{l} \text{match} (\\ \text{match} (f, 1); \\ \text{if null(1) then nil} \\ \text{else cons} (f(\text{hd}(1)), \text{map}(f, \text{tl}(1))); \\ ) \end{array}
```

Step2 代换推导

为规则编号

- (1)Exp id
- (2)Exp Funcall
- (3)Exp id Fresh

类型断言	代换	规则
f: lpha		(1)
l:eta		(1)
$map:\gamma$		(1)
$map(f.l):\delta$	$\gamma = (\alpha \times \beta) - > \delta$	(2)
null: list(lpha i) - > boolean		(3)
null(l):boolean	eta=list(lpha i)	(2)
nil: list(lpha i+1)		(3)
l: list(lpha i)		
hd: list(lpha i+2) -> lpha i+2		(3)
hd: list(lpha i+2) -> lpha i+2		(3)
hd(l): lpha i	lpha i + 2 = lpha i	(2)
f(hd(l)): lpha i + 3	lpha=lpha i->lpha i+3	(1)
f: lpha - > lpha i + 3		
tl: list(lpha i+4) - > list(lpha i+4)		(3)
tl(l): list(lpha i)	lpha i + 4 - > lpha i	(2)
$map: ((lpha 0 - > lpha i + 3) imes list(lpha i)) - > \delta$		
$map(f,tl(l)):\delta$		(2)
cons: lpha i + 5 imes list(lpha i + 5) - > list(lpha i + 5)		(3)
cons(): list(lpha i+3)	$lpha i + 5 = lpha i + 3, \delta = list(lpha i + 3)$	(2)
if:boolean imes (lpha i+6) imes list(lpha i+6) -> list(lpha i+6)		(3)
if(): list(lpha i+1)	lpha i + 6 = lpha i + 1, lpha i + 3 = lpha i + 1	(2)
match: lpha i + 7 imes lpha i + 7 - > lpha i + 7		(3)
match(): list(lpha i+1)	lpha i + 7 = list(lpha i + 1)	(2)