Proving with Computer Assistance, 2IMF15

Herman Geuvers, TUE

Exercises on Polymorphic type Theory

1. Recall: $\bot := \forall\alpha.\,\alpha$, $\top := \forall\alpha.\,\alpha{\to}\alpha$.

   (a) Verify that in Church $\lambda 2$: $\lambda x{:}\top.\,x\top x : \top{\to}\top$.
   Answer: ....

1	| $x : \forall\alpha.\alpha{\to}\alpha$	
2	| $x\,\top : \top \to \top$	app, 1
3	| $x\,\top\,x : \top$	app, 2
4	$\lambda x{:}\top.x\,\top\,x : \top \to \top$	λ-rule, 1, 3

   End Answer ....

   (b) Verify that in Curry $\lambda 2$: $\lambda x.\,x\,x : \top{\to}\top$
   Answer: ....

1	| $x : \forall\alpha.\alpha{\to}\alpha$	
2	| $x : \top \to \top$	app, 1
3	| $x\,x : \top$	app, 2
4	$\lambda x.x\,x : \top \to \top$	λ-rule, 1, 3

   End Answer ....

   (c) Find a type in Curry $\lambda 2$ for $\lambda x.\,x\,x\,x$
   Answer: ....

1	| $x : \forall\alpha.\alpha{\to}\alpha$	
2	| $x : \top \to \top$	app, 1
3	| $x\,x : \top$	app, 2
4	| $x\,x : \top \to \top$	app, 3
5	| $x\,x\,x : \top$	app, 4
6	$\lambda x.x\,x\,x : \top \to \top$	λ-rule, 1, 5

   OR:

1	| $x : \bot$	
2	| $x : \bot \to \bot \to \bot$	app, 1
3	| $x\,x : \bot \to \bot$	app, 2, 1
4	| $x\,x\,x : \bot$	app, 3, 1
5	$\lambda x.x\,x\,x : \bot \to \bot$	λ-rule, 1, 4

End Answer . . . .

(d) Find a type in Curry $\lambda 2$ for $\lambda x.\,(x\,x)(x\,x)$

Answer: . . . .

1	$x : \bot$	
2	$x : \bot \to \bot$	app, 1
3	$x\,x : \bot$	app, 2, 1
4	$x\,x : \bot \to \bot$	app, 3
5	$(x\,x)\,(x\,x) : \bot$	app, 4, 3
6	$\lambda x.(x\,x)(x\,x) : \bot \to \bot$	λ-rule, 1, 5

End Answer . . . .

(e) Find a type in Curry $\lambda 2$ for $\lambda z.\,z(\lambda x.\,x\,x)$

2. Let $x : \top$ and remember that $\top := \forall \alpha{:}\,*\,.\,\alpha{\to}\alpha$.

(a) Give a type to the term

$$\lambda y.\,x\,y\;x(\lambda z.\,z\,x\,z)$$

in $\lambda 2$ à la Curry and give the typing derivation of your result.

Answer: . . . .

1	$x : \top$	
2	$y : \bot$	
3	$x : \bot \to \bot$	app, 1
4	$x\,y : \bot$	app, 3, 2
5	$x\,y : \top \to \bot$	app, 4
6	$x\,y\,x : \bot$	app, 4, 1
7	$z : \bot$	
8	$z : \top \to \bot$	app, 7
9	$z\,x : \bot$	app, 8, 1
10	$z\,x : \bot \to \bot$	app, 9
11	$z\,x\,z : \bot$	app, 10, 1
12	$\lambda z.z\,x\,z : \bot \to \bot$	λ-rule, 7, 11
13	$x\,y\,x : (\bot \to \bot) \to \bot$	app, 6
14	$x\,y\,x\,(\lambda z.z\,x\,z) : \bot$	app, 13, 12
15	$\lambda y.x\,y\,x\,(\lambda z.z\,x\,z) : \bot \to \bot$	λ-rule, 2, 14

End Answer . . . .

(b) Give a type to the term

$$\lambda y.\, x\, y\, (x(\lambda z.\, z\, z))$$

in λ2 à la Curry. Also give the typing derivation of your result.

3. Define:

$$\begin{array}{rcl} \sigma \times \tau & := & \forall \alpha.\, (\sigma \to \tau \to \alpha) \to \alpha, \\ \sigma + \tau & := & \forall \alpha.\, (\sigma \to \alpha) \to (\tau \to \alpha) \to \alpha \end{array}$$

(a) Define $\mathrm{inl} : \sigma \to \sigma + \tau$.

(b) Define pairing : $[-,-] : \sigma \to \tau \to \sigma \times \tau$

Answer: ....

$$\lambda x : \sigma.\, \lambda y : \tau.\, \lambda \alpha.\, \lambda h : \sigma \to \tau \to \alpha.\, h\, x\, y$$

End Answer ....

(c) Define the first projection : $\pi_1 : \sigma \times \tau \to \sigma$ and show that $\pi_1[x, y] =_\beta x$.

Answer: ....

$\pi_1 := \lambda z : \sigma \times \tau.\, z\, \sigma\, (\lambda x : \sigma.\, \lambda y : \tau.\, x)$

End Answer ....

4. Define the type of binary tress with leaves in B and node labels in A:

$$\mathrm{Tree}_{A,B} \quad := \quad \forall \alpha. (B \to \alpha) \to (A \to \alpha \to \alpha \to \alpha) \to \alpha.$$

(a) Define leaf : $B \to \mathrm{Tree}_{A,B}$ and join : $\mathrm{Tree}_{A,B} \to \mathrm{Tree}_{A,B} \to A \to \mathrm{Tree}_{A,B}$.

Answer: ....

leaf $:= \lambda b : B. \lambda \alpha.\, \lambda f : B \to \alpha.\, \lambda h : A \to \alpha \to \alpha \to \alpha.\, f\, b$ and join is defined as follows:

$$\begin{array}{rcl} \mathrm{join} & := & \lambda t_1 : \mathrm{Tree}_{A,B}.\, \lambda t_2 : \mathrm{Tree}_{A,B}.\, \lambda a : A. \\ & & \lambda \alpha.\, \lambda f : B \to \alpha.\, \lambda h : A \to \alpha \to \alpha \to \alpha.\, ha(t_1 \alpha f h)(t_2 \alpha f h) \end{array}$$

End Answer ....

(b) Give the Tree-iteration scheme for $\mathrm{Tree}_{A,B}$ and define $h : \mathrm{Tree}_{A,B} \to \mathrm{Nat}$ that counts the number of leaves of a tree.

Answer: ....

The Tree iteration scheme is: given a type D and $f : B \to D$, $g : A \to D \to D \to D$, there is a term $k : \mathrm{Tree}_{A,B} \to D$ satisfying

$$\begin{array}{rcl} k\, (\mathrm{leaf}\, b) & = & f\, b \\ k\, (\mathrm{join}\, a\, t_1\, t_2) & = & g\, a\, (k\, t_1)\, (k\, t_2) \end{array}$$

as a matter of fact k is just $\lambda t : \mathrm{Tree}_{A,B}.t\, D\, f\, g$.

The function h that counts the number of leaves satisfies

$$\begin{aligned} h\,(\mathrm{leaf}\,b) &= S\,0 \\ h\,(\mathrm{join}\,a\,t_1\,t_2) &= \mathrm{Plus}\,(h\,t_1)\,(h\,t_2) \end{aligned}$$

so we can take $h := \lambda t : \mathrm{Tree}_{A,B}.t\,\mathrm{Nat}\,(\lambda b : B.S\,0)\,(\lambda a : A, \lambda n_1, n_2 : \mathrm{Nat}.\mathrm{Plus}\,n_1\,n_2)$.

End Answer . . . .

(c) Define $g : \mathrm{Tree}_{A,B} \to B$ that computes the left-most leaf of a tree.

Answer: . . . .

Just the final answer: $g := \lambda t : \mathrm{Tree}_{A,B}.t\,\mathrm{Nat}\,(\lambda b : B.b)\,(\lambda a : A, \lambda b_1, b_2 : B.b_1)$.

End Answer . . . .