Lambda-Calculus and Type Theory
ISR 2024
Obergurgl, Austria
Herman Geuvers & Niels van der Weide
Radboud University Nijmegen NL
Exercises Day 3

Lecture 6. Polymorphic types

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	$\quad y : \bot$	
3	$\quad x : \bot \to \bot$	app, 1
4	$\quad x\, y : \bot$	app, 3, 2
5	$\quad x\, y : \top \to \bot$	app, 4
6	$\quad x\, y\, x : \bot$	app, 4, 1
7	$\quad\quad z : \bot$	
8	$\quad\quad z : \top \to \bot$	app, 7
9	$\quad\quad z\, x : \bot$	app, 8, 1
10	$\quad\quad z\, x : \bot \to \bot$	app, 9
11	$\quad\quad z\, x\, z : \bot$	app, 10, 1
12	$\quad \lambda z.z\, x\, z : \bot \to \bot$	λ-rule, 7, 11
13	$\quad x\, y\, x : (\bot \to \bot) \to \bot$	app, 6
14	$\quad x\, y\, x\, (\lambda z.z\, x\, z) : \bot$	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 $\lambda 2$ à la Curry. Also give the typing derivation of your result.

3. Define:

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

(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 $\mathrm{leaf} : B \to \mathrm{Tree}_{A,B}$ and $\mathrm{join} : \mathrm{Tree}_{A,B} \to \mathrm{Tree}_{A,B} \to A \to \mathrm{Tree}_{A,B}$.
Answer: .....
$\mathrm{leaf} := \lambda b : B.\lambda\alpha.\,\lambda f : B \to \alpha.\,\lambda h : A \to \alpha \to \alpha \to \alpha.\,f\,b$ and join is defined as follows:

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

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

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

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

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

Lecture 7. Higher order logic in the Calculus of constructions and in Coq

NB. The exercises can also be made with Coq. Please look at the web site for the `HOL_inductivetypes.v` file.

1. Definition in CC (for $t, q : A$):

$$t =_A q := \Pi P : A \to *.\,(Pt \to Pq)$$

(a) (basic) Prove that this equality is reflexive by giving a term of type $\Pi x : A.\, x =_A x$.
Answer: .....
Without typing derivation:
$\lambda x : A.\, \lambda P : A \to *.\, \lambda h : P\, x.\, h : \Pi x : A.\, x =_A x$
End Answer .....

(b) (basic) Prove that this equality is transitive by giving a term of type $\Pi x, y, z : A.\, x =_A y \to y =_A z \to x =_A z$.
Answer: .....
Without typing derivation:
$\lambda x, y, z : A.\, \lambda f : x =_A y.\, \lambda g : y =_A z.\, \lambda P : A \to *.\, \lambda h : P\, x.\, g\, P (f\, P\, h) :$

$\Pi x, y, z : A.\, x =_A y \to y =_A z \to x =_A z$
End Answer .....

(c) (advanced) Prove that this equality is symmetric by giving a term of the type $\Pi x, y : A.\, x =_A y \to y =_A x$.
Answer: .....
Without typing derivation:
$\lambda x, y : A.\, \lambda e : x =_A y.\, \lambda P : A \to *.\, \lambda h : P\, y.\, e\, (\lambda z : A.\, P\, z \to P\, x)\, (\lambda u : P\, x.\, u)\, h : \Pi x, y : A.\, x =_A y \to y =_A x$
With typing derivation. At line 5 we need a smart choice for P' so that e can help us to construct a term of type $P\, x$ from $h : P\, y$. The trick is to take $\lambda z : A.\, P\, z \to P\, x$ for P'.

1		$x, y : A$
2		$e : \Pi P' : A \to *.\, P'\, x \to P\, y$
3		$P : A \to *$
4		$h : P\, y$
5		$e\, (\lambda z : A.\, P\, z \to P\, x) : (P\, x \to P\, x) \to (P\, y \to P\, x)$
6		$u : P\, x$
7		$u : P\, x$
8		$\lambda u : P\, x.\, u : P\, x \to P\, x$
9		$e\, (\lambda z : A.\, P\, z \to P\, x)\, (\lambda u : P\, x.\, u) : P\, y \to P\, x$
10		$e\, (\lambda z : A.\, P\, z \to P\, x)\, (\lambda u : P\, x.\, u)\, h : P\, x$
11		$\lambda h : P\, y.\, e\, (\lambda z : A.\, P\, z \to P\, x)\, (\lambda u : P\, x.\, u)\, h : P\, y \to P\, x$
12		$\lambda P : A \to *.\, \lambda h : P\, y.\, e\, (\lambda z : A.\, P\, z \to P\, x)\, (\lambda u : P\, x.\, u)\, h : y =_A x$
13		$\lambda e : x =_A y.\, \lambda P : A \to *.\, \lambda h : P\, y.\, e\, (\lambda z : A.\, P\, z \to P\, x)\, (\lambda u : P\, x.\, u)\, h : x =_A y \to y =_A x$
14		$\lambda x, y : A.\, \lambda e : x =_A y.\, \lambda P : A \to *.\, \lambda h : P\, y.\, e\, (\lambda z : A.\, P\, z \to P\, x)\, (\lambda u : P\, x.\, u)\, h :$
15		$\Pi x, y : A.\, x =_A y \to y =_A x$

End Answer .....

2. The transitive closure of a binary relation R on A has been defined as follows.

$$\begin{array}{lll} \mathsf{trclos}\, R & := & \lambda x, y : A. \\ & & (\forall Q : A \to A \to *. (\mathsf{trans}\, Q \to (R \subseteq Q) \to (Q\, x\, y))). \end{array}$$

(a) (basic) Prove – by giving a proof-term – that the transitive closure of R contains R.

(b) (medium) Prove – by giving a proof-term – that the transitive closure is transitive.
Answer: .....
Without typing derivation:
$\lambda x, y : A. \lambda e : x =_A y. \lambda P : A \to *. \lambda h : P\, y. e\, (\lambda z : A. P\, z \to P\, x)\, (\lambda u : P\, x. u)\, h : \Pi x, y : A. x =_A y \to y =_A x$
With typing derivation. At line 5 we need a smart choice for P' so that e can help us to construct a term of type $P\, x$ from $h : P\, y$. The trick is to take $\lambda z : A. P\, z \to P\, x$ for P'.

1	$x, y, z : A$
2	$h : \mathsf{trclos}\, R\, x\, y$
3	$g : \mathsf{trclos}\, R\, y\, z$
4	$Q : A \to A \to *$
5	$t : \mathsf{trans}\, Q$
6	$s : R \subseteq Q$
7	$g\, Q\, t\, s : Q\, y\, z$
8	$h\, Q\, t\, s : Q\, x\, y$
9	$t\, x\, y\, z : Q\, x\, y \to Q\, y\, z \to Q\, x\, z$
10	$t\, x\, y\, z\, (h\, Q\, t\, s)\, (g\, Q\, t\, s) : Q\, x\, z$
11	$\lambda s : R \subseteq Q. t\, x\, y\, z\, (h\, Q\, t\, s)\, (g\, Q\, t\, s) : R \subseteq Q \to Q\, x\, z$
12	$\lambda t : \mathsf{trans}\, Q. \lambda s : R \subseteq Q. t\, x\, y\, z\, (h\, Q\, t\, s)\, (g\, Q\, t\, s) : \mathsf{trans}\, Q \to R \subseteq Q \to Q\, x\, z$
13	$\lambda Q : A \to A \to *. \lambda t : \mathsf{trans}\, Q. \lambda s : R \subseteq Q. t\, x\, y\, z\, (h\, Q\, t\, s)\, (g\, Q\, t\, s) : \mathsf{trclos}\, R\, x\, z$
14	$\lambda h : \mathsf{trclos}\, R\, x\, y. \lambda g : \mathsf{trclos}\, R\, y\, z. \lambda Q : A \to A \to *. \lambda t : \mathsf{trans}\, Q. \lambda s : R \subseteq Q. t\, x\, y\, z\, (h\, Q\, t\, s)$
15	$: \mathsf{trclos}\, R\, x\, y \to \mathsf{trclos}\, R\, y\, z \to \mathsf{trclos}\, R\, x\, z$
16	$\lambda x, y, z : A. \lambda h : \mathsf{trclos}\, R\, x\, y. \lambda g : \mathsf{trclos}\, R\, y\, z. \lambda Q : A \to A \to *. \lambda t : \mathsf{trans}\, Q. \lambda s : R \subseteq Q.$
17	$t\, x\, y\, z\, (h\, Q\, t\, s)\, (g\, Q\, t\, s) : \mathsf{trans}(\mathsf{trclos}\, R)$

End Answer .....

(c) (basic) Prove – by giving a proof-term – that, if P is transitive and P contains R, then P contains $\mathsf{trclos}\, R$.

3. The existential quantifier has been defined by

$$\exists x : \sigma.\, \phi := \forall \alpha : *.\, (\forall x : \sigma.\, \phi \to \alpha) \to \alpha$$

   (a) (medium) Given $t : \sigma$ and $q : P\, t$, give a term M such that $M : \exists x : \sigma.\, P\, x$

   (b) (medium) Given $q : \exists x : \sigma.\, P\, x$ and $h : \forall y : \sigma.\, P\, y \to C$ with $y \notin \mathrm{FV}(C)$, give a term N of type C.

4. For $D : *$, $A, B : D \to *$, we define $A \subseteq B$ as $\forall x : D.\, A\, x \to B\, x$. We now define

$$\begin{array}{lcl} A \cap B & := & \lambda x : D.\, \forall P : D \to *.\, (\forall y : D.\, A\, y \to B\, y \to P\, y) \to P\, x \\ A \cup B & := & \lambda x : D.\, \forall P : D \to *.\, A \subseteq P \to B \subseteq P \to P\, x \end{array}$$

   Prove the following, by giving a (proof) term of the type. Remember that $X \vee Y$ is defined as $\forall \alpha : *.\, (X \to \alpha) \to (Y \to \alpha) \to \alpha$.

   (a) $A \subseteq A \cup B$.

   (b) (This is a hard question) $\forall x : D.\, (A \cup B)\, x \to A\, x \vee B\, x$.

   (c) $A \cap B \subseteq A$.

   (d) $\forall x : D.\, A\, x \to B\, x \to (A \cap B)\, x$.