## Exercises of the class of Herman Geuvers

## Exercises 1a: Simple Type Theory

1. Find inhabitants (i.e. *closed* terms) of the following types (in STT)
   (a) $(\alpha\to\beta)\to(\beta\to\gamma)\to\alpha\to\gamma$
   (b) $\alpha\to\beta\to(\alpha\to\beta\to\gamma)\to\gamma$
   (c) $((\alpha\to\beta\to\alpha)\to\alpha)\to\alpha$
   (d) $\beta\to((\alpha\to\beta)\to\gamma)\to\gamma$

2. The type $\alpha\to(\alpha\to\alpha)\to\alpha$ is also called $\mathsf{nat}$.
   (a) Show that there are infinitely many closed terms (inhabitants) of type $\mathsf{nat}$.
   (b) Describe a term $0 : \mathsf{nat}$ and the succesor $\mathsf{succ} : \mathsf{nat}\to\mathsf{nat}$.
   (c) Describe the derivations that the (infinitely many) terms under (a) correspond to.
   (d) Construct a derivation of $((\alpha\to\beta)\to\gamma)\to\beta\to\gamma$ and the associated typed $\lambda$-term.

3. Add *product types* to $\lambda\to$, that is add $\sigma\times\tau$ to the types and
   (a) Add the appropriate term constructors to extend the term language of $\lambda\to$.
   (b) Give typing rules for terms of type $\sigma\times\tau$, by giving an *elimination rule* and an *introduction rule*. (A term of type $\sigma\times\tau$ should be built up from a term of type $\sigma$ and a term of type $\tau$.)
   (c) Give a reduction rule for the new term constructors. Try to give an "$\beta$-like" rule and an "$\eta$-like" rule.

4. Prove the claim made in the proof of the Weak Normalization theorem (page 16 of the slides of lesson 2): If we reduce in $P$ a redex of maximum height (height $h(P)$) that does not contain any other redex of height $h(P)$, obtaining the term $Q$, then $m(Q) <_l m(P)$.

5. Fill the three gaps in the proof of Strong Normalization (page 17 of the slides of lesson 2). That is, prove
   (a) $[\![\sigma]\!] \subseteq \mathsf{SN}$ (by induction on $\sigma$)
   (b) If $M[N/x]\vec{P} \in [\![\tau]\!]$, $N \in [\![\sigma]\!]$, then $(\lambda x.M)N\vec{P} \in [\![\tau]\!]$ (by induction on $\sigma$)
   (c) (By induction on the derivation of $\Gamma \vdash M : \sigma$).

$$\left.\begin{array}{c} x_1{:}\tau_1,\ldots,x_n{:}\tau_n \vdash M : \sigma \\ N_1 \in [\![\tau_1]\!],\ldots,N_n \in [\![\tau_n]\!] \end{array}\right\} \Rightarrow M[N_1/x_1,\ldots N_n/x_n] \in [\![\sigma]\!]$$

6. Prove the Substitution Lemma (by induction on the derivation of $\Gamma, x : \tau, \Delta \vdash M : \sigma$). (That is, prove that if $\Gamma, x : \tau, \Delta \vdash M : \sigma$ and $\Gamma \vdash P : \tau$, then $\Gamma, \Delta \vdash M[P/x] : \sigma$.)

## Exercises 1b: Polymorphic Lambda Calculus

1. $\bot$ is the type $\forall\alpha.\alpha$. Give the typing derivations of the following typing. $\lambda x{:}\bot.\lambda\alpha.x(\alpha{\to}\alpha)(x\alpha)$.

2. Find terms of the following types in $\lambda 2$. (See the slides for the definitions.)

   (a) $\sigma{\to}\sigma \vee \tau$. Now make this term polymorphic in $\sigma$ and $\tau$.

   (b) $\sigma{\to}\tau{\to}\sigma \wedge \tau$

   (c) $\forall\beta.\sigma{\to}\exists\alpha.\sigma[\alpha/\beta]$. Which logical rule does this term correspond to?

   (d) Given $M : \exists\alpha.\sigma$ and $F : \forall\alpha.\sigma{\to}\tau$, with $\alpha \notin \mathrm{FV}(\tau)$, construct a term of type $\tau$. Which logical rule does this term correspond to?

3. Define the type of booleans $\mathsf{bool}$ in $\lambda 2$ as $\mathsf{bool} := \forall\alpha.\alpha{\to}\alpha{\to}\alpha$

   (a) Define $\mathsf{true} : \mathsf{bool}$ and $\mathsf{false} : \mathsf{bool}$.

   (b) Define conjunction and disjunction over the booleans

4. Recall the natural numbers in $\lambda 2$.

   (a) Define exponentiation $\mathsf{exp} : \mathsf{nat}{\to}\mathsf{nat}{\to}\mathsf{nat}$ on the natural numbers in $\lambda 2$. (Use the iterator and already defined functions.)

   (b) Define the function $\mathsf{Z?} : \mathsf{nat}{\to}\mathsf{bool}$ such that $\mathsf{Z?}0 =_\beta \mathsf{true}$ and $\mathsf{Z?}(Sx) =_\beta \mathsf{false}$.

5. The type of lists over $A$ is defined by $\mathsf{list}_A := \forall\alpha.\alpha{\to}(A{\to}\alpha{\to}\alpha){\to}\alpha$.

   (a) Define the "head" function over $\mathsf{list}_A$. This function requires a "default value" for the case of the nil-list:

   $$\mathsf{head} : A{\to}\mathsf{list}_A{\to}A.$$

   NB. The tail function is not so easy to define. It can't be defined directly by iteration.

   (b) Define the function $\mathsf{suclist} : \mathsf{list}_{\mathsf{nat}}{\to}\mathsf{nat}$ that adds 1 to each element in a list of natural numbers. (See the "map" function on the slides.)

6. Consider the type of Binary trees with nodes in $A$ and leaves in $B$, as given in the lecture:

   $$\mathsf{tree}_{A,B} := \forall\alpha.(B{\to}\alpha){\to}(A{\to}\alpha{\to}\alpha{\to}\alpha){\to}\alpha$$

   (a) Define the functions $\mathsf{leaf} : B{\to}\mathsf{tree}_{A,B}$ and $\mathsf{join} : A{\to}\mathsf{tree}_{A,B}{\to}\mathsf{tree}_{A,B}{\to}\mathsf{tree}_{A,B}$.

   (b) Define the *iterator* for $\mathsf{tree}$:

   $$\mathsf{it} : \forall\gamma.(B{\to}\gamma){\to}(A{\to}\gamma{\to}\gamma{\to}\gamma){\to}\mathsf{tree}_{A,B}{\to}\gamma.$$

   (Given a type $\gamma$ and functions $f : (\gamma{\to}B)$ and $g : (\gamma{\to}A{\to}A)$, $\mathsf{it}$ should produce a function from $\mathsf{tree}_{A,B}$ to $\gamma$.)

(c) Take $B :=$ nat and write a function tsuml that computes the sum of all leaves.

(d) Take $A :=$ nat and write a function tsumln that computes the sum of all leaves and nodes.

(e) Take $A :=$ bool and write a function tend that computes the leave (a term of type $B$) that is found by going "left" if the boolean in the node is true and "right" if it's false.

(f) Take $A, B :=$ bool and write a function tpath that computes the path (as a term of type $\mathrm{List}_{\mathsf{bool}}$ to the leaf by going "left" if the boolean in the node is true and "right" if it's false.

7. Prove Strong Normalization for $\lambda 2$ by proving the following by induction on the derivation.
**Proposition**

$$x_1 : \tau_1, \ldots, x_n : \tau_n \vdash M : \sigma \Rightarrow M[P_1/x_1, \ldots, P_n/x_n] \in [\![\sigma]\!]_\rho$$

for all valuations $\rho$ and $P_1 \in [\![\tau_1]\!]_\rho, \ldots, P_n \in [\![\tau_n]\!]_\rho$
See the slides or the Handbook article by Barendregt (Def 4.1.7, page 50) for the derivation rules of $\lambda 2$. See the slides for how this fits in the proof of SN.

## Exercises of the class of Herman Geuvers

## Exercises 2a: Higher Order Logic

1. Define the Leibniz equality on $A$ as $t =_A q := \forall P{:}A{\to}\mathsf{Prop}.(P\,t){\to}(P\,q)$. Prove the following by finding terms of the associated types.
   (a) reflexivity of $=_A$: $\forall x{:}A.x =_A x$.
   (b) transitivity of $=_A$: $\forall x, y, z{:}A.x =_A y \to y =_A z \to x =_A z$.
   (c) symmetry of $=_A$: $\forall x, y{:}A.x =_A y \to y =_A x$.

2. The transitive closure of a relation $R$ is defined as follows.

   $$\mathsf{trclos} := \lambda R{:}A{\to}A{\to}\mathsf{Prop}.\lambda x, y{:}A.(\forall Q{:}A{\to}A{\to}\mathsf{Prop}.(\mathsf{trans}(Q){\to}(R \subseteq Q){\to}(Q\,x\,y))).$$

   So $\mathsf{trclos}$ is of type $(A{\to}A{\to}\mathsf{Prop}){\to}(A{\to}A{\to}\mathsf{Prop})$
   (a) Define the notions $\mathsf{trans}$ and $\subseteq$ in the definition of $\mathsf{trclos}$.
   (b) Prove that the transitive closure is transitive. (Find a term of type $\mathsf{trans}(\mathsf{trclos}\,R)$).
   (c) Prove that the transitive closure of $R$ contains $R$. (Find a term of type $R \subseteq (\mathsf{trclos}\,R)$).

3. In this exercises we will prove in higher order logic a variant of the Knaster-Tarski fixed-point theorem.
   Given a domain $A$, we identify $A{\to}\mathsf{Prop}$ with the collection of subsets of $A$. In this exercise we consider maps $\Phi : (A{\to}\mathsf{Prop}){\to}(A{\to}\mathsf{Prop})$, mapping subsets of $A$ to subsets of $A$.
   $\Phi$ is asumed to be *monotone*: $\forall P, Q{:}A{\to}\mathsf{Prop}.P \subseteq Q \to (\Phi\,P) \subseteq (\Phi\,Q)$.
   ($P \subseteq Q$ is an abbreviation for $\forall x{:}A.(P\,x){\to}(Q\,x)$, which is also gives away the answer to the exercise above.)
   $P : A{\to}\mathsf{Prop}$ is called $\Phi$-*closed* if $(\Phi\,P) \subseteq P$.
   (a) Define (formally) $X : A{\to}\mathsf{Prop}$ as the *smallest* $\Phi$-*closed* subset of $A$.
   (b) Prove (for arbitrary $P : A{\to}\mathsf{Prop}$): if $P$ is $\Phi$-closed, then $X \subseteq P$.
   (Find a term of type $\forall P{:}A{\to}\mathsf{Prop}.(\Phi\,P) \subseteq P \to X \subseteq P$.)
   (c) Prove $(\Phi\,X) \subseteq X$.
   (d) Prove $X \subseteq (\Phi\,X)$.
   (e) Conclude that $X$ is the *least fixed point of* $\Phi$:
      i. $X \approx (\Phi\,X)$,
      ii. $P \approx (\Phi\,P) \to X \subseteq P$.

   where we take the equality $\approx$ to be defined in the set-theoretical way as
   $P \approx Q := P \subseteq Q \wedge Q \subseteq P$.

4. Recall the induction principle over natural numbers as a higher order formula. Given a domain $N$ and $0 : N, S : N \to N$, $\mathrm{Ind}_N$ is

$$\forall P{:}N{\to}\mathsf{Prop}.(P\,0) \to (\forall x{:}N.(P\,x){\to}(P(S\,x))) \to \forall x{:}N.(P\,x)$$

   (a) Consider a datatype of lists over a base domain $A$. So we have two base domains $A$ and $L$ and we let $\mathrm{Nil} : L, \mathrm{Cons} : A{\to}L{\to}L$. Define the induction principle over lists.

   (b) Consider a datatype of binary trees with leaves in base type $A$ and node labels in base type $B$. So So we have three base domains $A, B$ and $T : \mathsf{Set}$ and we let $\mathrm{Leaf} : A{\to}T, \mathrm{Join} : B{\to}T{\to}T{\to}T$. Define the induction principle over these trees.

## Exercises 2b: Extensions of $\lambda$HOL; the $\lambda$ cube; PTSs

1. (a) Explain for every $\to$ and $\Pi$ in the following judgment which $\Pi$-rule (of $\lambda$HOL) is needed to make it a valid construction.

$$A : \mathsf{Type}, R : A{\to}A{\to}\mathsf{Prop} \vdash \Pi x{:}A.\Pi Q{:}(A{\to}\mathsf{Prop}){\to}\mathsf{Prop}.Q(R\,x){\to}R\,x\,x : \mathsf{Prop}$$

   (b) Do the same for the following judgment in CC.

$$A : \mathsf{Prop} \vdash \Pi F{:}(\Pi\alpha{:}\mathsf{Prop}.\Pi Q{:}\alpha{\to}\mathsf{Prop}.\Pi y{:}\alpha Q\,y{\to}Q\,y).F\,A \to \mathsf{Prop} : \mathsf{Type}$$

2. Give a context $\Gamma$ and a term $M$ of the type

$$(\Pi x{:}A.(R\,x\,a{\to}R\,a\,(f\,x))){\to}R\,a\,a{\to}R\,a\,(f\,a)$$

   in this context.
   What is the simplest system of the $\lambda$ cube in which this typing is valid?

3. (a) Recall the polymorphic type of *lists over* $A$, $\mathrm{List}_A$ and define it in $\lambda 2$. (So $A : \mathsf{Prop} \vdash \mathrm{List}_A : \mathsf{Prop}$; verify that this is indeed possible in the $\lambda$-cube system $\lambda 2$.)

   (b) Define *induction over lists* as a proposition in $\lambda$P2. (So $A : \mathsf{Prop} \vdash \mathsf{ind}_{\mathrm{List}} : \mathsf{Prop}$; verify that this is indeed possible in the $\lambda$-cube system $\lambda$P2.)

4. Define in CC, $\varphi := \forall x{:}A.x = a$, $\psi := \forall x{:}B.\exists y{:}B.x \neq y$ (with $A, B : \mathsf{Prop}$) and define

$$\mathrm{EXT} := \forall\alpha,\beta{:}\mathsf{Prop}.(\alpha \Leftrightarrow \beta) \Rightarrow (\alpha =_{\mathsf{Prop}} \beta).$$

   Give a term of type $\perp$ in CC in the following context

$$e : \mathrm{EXT}, A, B : \mathsf{Prop}, h_1 : \varphi, h_2 : \psi$$

   Alternatively you may try to find this term in Coq, see the file `coq_ex7.v8`.

5. Prove the following basic property for any Pure Type System $(\mathcal{S}, \mathcal{A}, \mathcal{R})$. (By induction on the derivation.)
   (Variable Lemma)
   If $\Gamma \vdash M : A$, then $\Gamma \vdash x : B$ for all $x : B \in \Gamma$.

6. Prove the Substitution Lemma for PTSs. (By induction on the derivation; do the cases for the last rule being (weak) or ($\lambda$).