Exercices on Typed Lambda-Calculus

Thierry Coquand
Chalmers University

September 2008

Exercice 1: We can define like in untyped lambda calculus the notion of term in normal form. Show that a term in normal form can be written $\lambda x_1 : T_1 \dots \lambda x_k : T_k\ x\ M_1 \dots M_l$ where we can have $k = 0$ or $l = 0$ and $M_1, \dots, M_l$ are in normal form.

Use this to enumerate the closed terms of the following types (ι is a ground type)

1. $\iota \to \iota$,
2. $\iota \to \iota \to \iota$,
3. $(\iota \to \iota) \to \iota \to \iota$,
4. $\iota \to (\iota \to \iota) \to \iota$,
5. $(\iota \to \iota) \to \iota$.
6. $((\iota \to \iota) \to \iota) \to \iota$.
7. $((\iota \to \iota) \to \iota) \to \iota$.
8. $((\alpha \to \beta) \to \gamma) \to \beta \to \gamma$.

Exercice 2: Cantor's theorem can be expressed as follows in higher-order logic: there is no surjective function $H : \iota \to (\iota \to \mathbf{o})$. It is enough for this to deduce a contradiction from the existence of a spliting function $A : (\iota \to \mathbf{o}) \to \iota$, that is a function which satisfies

$$(\forall f)(\forall x)\ [f\ x \iff H\ (A\ f)\ x].$$

For this, it is enough to find a function $f : \iota \to \mathbf{o}$ and an individual $x : \iota$ such that

$$f\ x = \neg\ (H\ (A\ f)\ x).$$

A typical unification problem is to find a suitable f and x as terms of simply typed lambda-calculus. (Cantor's theorem has been proved automatically in an implementation of logic based on simply typed lambda-calculus.) For solving this unification problem, enumerate all terms of type ι and of type $\iota \to \mathbf{o}$ that are built from the constant

- $\neg : \mathbf{o} \to \mathbf{o}$,
- $H : \iota \to (\iota \to \mathbf{o})$,
- $A : (\iota \to \mathbf{o}) \to \iota$.

These two sets of terms are best described by a mutual inductive definition.

Exercice 3: Show that there exists no term $t : B \to B$ which may use the constant $0, 1 : B$ such that $t\ 0 = 1$ and $t\ 1 = 0$.

Exercice 4: (Difficult) This problem illustrates the importance of explicitely typing the variable. An alternative notation would be to use the same term as in untyped lamba-calculus (untyped variable). For instance $\lambda x\ x$ would be both of type $\iota \to \iota$ and $(\iota \to \iota) \to \iota \to \iota$. One can consider the typed variable as a pair of one untyped variable and a type. It is then possible to define a map $t \longmapsto |t|$ that to a typed term associate its "stripped" version where we forget about type annotations. In the previous example, if we give both terms and types, it is possible to reconstruct in an unique way a typed term t such that $|t| = \lambda x\ x$. Show that this is general for any untyped term in normal form u that there is at most on typed term t of a given type such that $|t| = u$.

Show that this may not the case for an arbitrary term. However, show that if $|t_1| = |t_2|$ and t_1, t_2 are of the same type then t_1 and t_2 are convertible.

Show that this is always the case for a λI term, by showing that if u reduces by head reduction to u' and the u has a unique typed decoration then u has a unique typed decoration, and using the normalisation theorem for typed lambda calculus.