Week 8. Inductive types

In-class problems

1. Let be given the inductive type of natural numbers $\mathsf{nat} : *$ with constructors $\mathsf{0} : \mathsf{nat}$ and $\mathsf{suc} : \mathsf{nat} \to \mathsf{nat}$. Using its recursor, define the function $\mathsf{sub} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ of *truncated* subtraction (i.e., if the difference would be negative, then the function has value zero.)

2. One version of the *Ackermann function* is recursively defined by the equations:

$$\begin{array}{rcl} A(0,y) & = & y+1 \\ A(x+1,0) & = & A(x,1) \\ A(x+1,y+1) & = & A(x,A(x+1,y)) \end{array}$$

   Define this function as a term $A : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ of Gödel's system T. Also, to get an impression of this function, give explicit formulas for $A(1,y)$, $A(2,y)$ and $A(3,y)$ and calculate the value of $A(4,2)$.

3. Give the typing (formation, constructors, recursor) and reduction rules for the type of lists of natural numbers. Give both types for a dependent and a non-dependent recursor.

4. Give the typing (formation, constructors, recursor) and reduction rules for the type of polymorphic vectors. Give both types for a dependent and a non-dependent recursor.

5. Show how to define an inductive predicate $\mathsf{even} : \mathsf{nat} \to \mathsf{nat}$ that says whether its argument is even. Give the typing (formation, constructors, recursor) and reduction rules. Give both types for a dependent and a non-dependent recursor. (Hint: it is generally easier to first determine the type of the dependent recursor, as described in the lecture.)

6. Give the typing (formation, constructors, recursor) and reduction rules for Leibniz equality. Give both types for a dependent and a non-dependent recursor.

Take-home problems

1. Give the type of the dependent recursor for the product type $A \times B$, and from that derive the type of a non-dependent recursor. Show how the functions π_1 and π_2 from MLW can be defined from this second recursor. Also, show how this recursor can be defined in terms of π_1 and π_2.

2. Define the lists over a given type A as a W-type.

3. Give the typing (formation, constructors, recursor) and reduction rules of the W-trees. Give both types for a dependent and a non-dependent recursor.

4. Let $\bot$ be the empty inductive type, use the abbreviation $\neg A := A \to \bot$, and let $=$ be the inductively defined Leibniz equality. Give a proof term for
$$\neg(0 = \mathsf{suc}\ 0)$$
You are allowed to use the 'large' recursor on the natural numbers which defines a function in $\mathsf{nat} \to *$. (In MLW there only is a 'large' recursor for the Booleans.)

5. Given the predicate even from exercise 5 on the front of this sheet, give a proof term for
$$\neg\, \mathsf{even}\, (\mathsf{suc}\ 0)$$

6. Give an example of terms $A : *$, $P : A \to *$ and $f : * \to *$ such that:
$$\vdash \forall x : A.\, f(Px))$$
$$\vdash \neg f(\forall x : A.\, Px)$$
(Hint: take $A := \mathsf{bool}$ and $P := \lambda x : A.\, \mathsf{bool}$.)