Proof assistants

TD 2- Specifics of the Calculus of Constructions

1 Inductive Predicates

A- Give an inductive definition `even : nat -> Prop` for the predicate " to be even ".
B- Caracterize with an inductive definition a relation `exp : nat -> nat -> nat -> Prop` with two constructors corresponding to the graph of the function $n^p = q$ on natural numbers.

2 Recursive types

A- Propose in Coq an inductive definition with parameter corresponding to the ML type of polymorphic lists:

```
type 'a list = nil | cons of 'a * 'a list
```

B- Coq library defines the binary product, the unit type and the type of natural numbers:

```
Inductive prod (A B : Type) : Type := pair : A → B → prod A B.
Inductive unit : Type := tt : unit.
Inductive nat : Type := O : nat | S : nat → nat.
```

Construct an expression `prodn` in CCI of type `Type` → `nat` → `Type` which builds the n-ary product of a given type A: (i.e. `prodn` A n is $A \times \ldots \times A$ (n times)). The definition will be by recursion on n.

Give an expression `length` of type $\forall A.$ `list` $A \to$ `nat` which computes the length of a list.

Give an expression `embed` of type $\forall A. \forall l :$ `list` $A.$ `prodn` A (`length` l) which translates a list into a n-uple.

3 Termination of fixpoints

Are the following fixpoints well-founded in CCI ? explain why ?

```
Fixpoint leq (n p: nat) {struct n} : bool :=
  match n with
  | O ⇒ true
  | S n' ⇒ match p with O ⇒ false | S p' ⇒ leq n' p' end
  end.
Definition exp (p:nat) :=
 (fix f (n:nat) : nat :=
 match leq p n with | true ⇒ S 0 | false ⇒ f (S n) + f (S n)  end)
 0.
Definition ackermann := fix f (n:nat) : nat → nat := match n with
  | O ⇒ S
  | S n' ⇒ fix g (m:nat) : nat := match m with
                                   | O ⇒ f n' (S O)
                                   | S m' ⇒ f n' (g m')
                                   end
  end.
```

4 Strong elimination

Let t_1 and t_2 be two arbitrary terms of type T_1 and T_2. Is the following function typable ?

```
Definition g (b:bool) := match b with true ⇒ t1 | false ⇒ t2 end.
```

If yes, give the corresponding return clause.

5 Restrictions on sorts in eliminations

A- We introduce the following definition of the true proposition:

```
Inductive True : Prop := I : True.
```

Write a function from `unit` to `True` which is one-to-one.
B- We now introduce

```
Inductive BOOL : Prop := TRUE : BOOL | FALSE : BOOL.
```

Can we show the equivalence between `bool` and `BOOL` ? Show that such an equivalence gives a proof of the negation of the principle of ¡¡ proof-irrelevance ¿¿ in `Prop`, i. e. $\forall P$: `Prop` $\forall p\, q : P.\, p = q)$.

6 The type W of well-founded trees (exam 2008)

The type W of well-founded trees is parameterised by a type A and a family of types $B : A \to$ `Type`. It has only one constructor and is defined by :

```
Inductive W (A:Type) (B:A → Type) : Type :=
    node : forall (a:A), (B a →W A B) →W A B.
```

The type A is used to parameterised the nodes and the type $B\,a$ give the arity of the node parameterised by a.

1. Give the type of dependent elimination for type W on sort `Type`.
2. In order to encode the type `nat` of natural numbers with `O` and `S`, we need two types of nodes. We take $A =$ `bool`. The constructor `O` corresponds to $a =$ `false`, it does not expect any argument so we take B `false` $=$ `empty`. The constructor `S` corresponds to $a =$ `true`, it takes one argument, we define B `true` $=$ `unit`.
   Using this encoding, give the terms corresponding to `nat`, `O` et `S`.
3. Propose an encoding using W for the type `tree` of binary trees parameterised by a type of values V, which means that we have a constructor `leaf` of type (`tree` V) and a constructor `bin` of type `tree` $V \to V \to$ `tree` $V \to$ `tree` V. Define the type and its constructors using this encoding.
4. Given a variable n of type `nat`, build two functions f_1 and f_2 of type `unit` $\to$ `nat` such that $\forall x :$ `unit`, $f_i\, x = n$ is provable but such that f_1 and f_2 are not convertible.
5. Which consequence does it have on the encoding of `nat` using W? Propose an equality on the type W which solves this problem.