Proof Assistants – TP. 4

Bruno Barras

Jan 12, 2017

1 Proof by structural induction

Consider the definition of lists (already in the prelude):

```
Require Import List.
Inductive list (A : Type) : Type :=
    nil : list A | cons : A → list A → list A
```

1- Implement a function `belast : nat -> list nat -> list nat` such that:

- `belast x nil = nil`
- `belast x (cons y l) = cons x (belast y l)`

2- Show the following statement:

```
Lemma length_belast (x : nat) (s : list nat) : length (belast x s) = length s.
```

3- Implement a function `skip : list nat -> list nat` such that:

- `skip nil = nil`
- `skip cons x nil = nil`
- `skip cons x (cons y nil) = skip (cons y nil)`

4- Show the following statement:

```
Lemma length_skip l :
  2 * length (skip l) ≤ length l.
```

2 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.
```

3 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.

4 The type W of well-founded trees

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.