DM3: Omniscience

simon.castellan@ens-lyon.fr

October 21, 2014

Due for the 25/11/2014

The goal of this assignment is to study the notion of omniscience in the type theory underlying Coq. Informally a type X is omniscient when for every predicate $p : X \to$ `bool`, one can decide whether p is constantly true or if it is false at some point $x_0 \in X$.

It is easy to see that classically every type is omniscient as the omniscience is just the excluded middle for the formula $\exists x, p(x) =$ `true`. In this assignment, we show that there exists infinite sets that are omniscient, even in an intuitionistic setting.

The Coq questions marked with **(*)** might be easier to solve with intermediate lemmas.

1 Warm-up

1.1 Extensionality

In Coq, equality on functional types is not the point-wise equality we usually have in mathematics. It is a more intentional equality that also compares the implementation. However, here we want to work with extensional equality. To do so, there are two usual ways:

- assume an axiom $\forall f g, (\forall x, f\ x = g\ x) \Rightarrow f = g$, however we lose the computational content of the proofs,
- quotient the functional types by the point-wise equality, the problem being that quotients are hard to do in type theory.

In this assignment, we use the second solution by using *setoids*. A setoid is a pair $(X, =_X)$ where X is a type and $=_X$: $X \to X \to$ `Prop` is an equivalence relation over X. The idea is that a setoid $(X, =_X)$ represents X modulo $=_X$, however the quotient is never actually made. Here is how we can define that in Coq:

```
Record equivalence {X : Set} (R : X → X → Prop) : Set :=
  mkEq {
    refl: forall x, R x x;
    symm: forall x y, R x y -> R y x;
    trans: (forall x y z, R x y -> R y z -> R x z)
}.

Record setoid : Type :=
  mkSetoid {
    set : Set;
    R : set → set → Prop;
    R_eq : equivalence R
}.
```

As an exemple of setoids, any type with Coq's equality can be seen as a setoid. Note that the refine tactic allow you to input a term with holes (underscores) and refine generates a goal for each hole it cannot infer. Here we have to give explicity the `set` and the `R` component because Coq cannot infer it.

```
Definition setoid_of_set (X : Set) : setoid.
  refine (mkSetoid (set:=X) (R := fun x y => x = y) _).
  apply mkEq; [auto | auto | apply eq_trans].
Defined.
Definition bool_setoid := setoid_of_set bool.
Definition nat_setoid := setoid_of_set nat.
Notation "'ℕ'" := (nat_setoid).
```

Question 1

If $(X, =_X)$ and $(Y, =_Y)$ are setoids, define the setoid $(X \Rightarrow Y, =_{X \Rightarrow Y})$ of the extensional functions from X to Y (extensional meaning that they send two elements related by $=_X$ to two elements related by $=_Y$), with the point-wise equality:

```
Definition extensional {X Y : setoid} (f : set X → set Y) :=
  forall x y, R X x y -> R Y (f x) (f y).
Hint Unfold extensional.
Definition arrow_setoid (X : setoid) (Y : setoid) : setoid.
refine (mkSetoid (set := { f : set X → set Y | extensional f })
                 (R := (fun f g => (* to do *)))  (* to do *)
                 _).
(* to do *)
Defined.
Notation "X ⇒ Y" := (arrow_setoid X Y) (at level 80).
```

The notation $\{ f : A$ `| P(f) }` is called a sigma type. Its elements are pairs (f, p) where f has type A and p is a proof of $P(f)$ with $P : A \to$ `Prop`. In Coq the pair is written `exist` $P\ f\ p$. The projection are `proj1_sig` and `proj2_sig`. We can now define for a setoid X what it means to be omniscient.

```
Definition omniscient (X : setoid) :=
  forall p : set (X ⇒ bool_setoid),
    (exists x, proj1_sig p x = false) \/ (forall x, proj1_sig p x = true).
```

1.2 Omniscience and selection function

We give a more concrete definition of omniscience that we will use in the next questions. We say a setoid X is searchable if there exists a selection function ε from the setoid $X \Rightarrow$ `bool` to X such that whenever $p(\varepsilon(p))$ is true, then p is constantly true.

Question 2

Define the predicate "searchable".

```
Definition searchable (X : setoid) := (* to do *).
```

Question 3
Prove that if X is searchable then it is omniscient.
(Optional) Does the converse hold in our formalization? Why?

Question 4
(*) Show that the types of the shape $\{x : \texttt{nat} \mid x \leq k\}$ are omniscient (with the usual equality).

```
Definition finite_setoid (k: nat) : setoid.
refine (mkSetoid (set := { x | x ≤ k}) (R := (fun x y => proj1_sig x = proj1_sig y)) _).
split; [auto | auto | intros; apply eq_trans with (y := proj1_sig y); auto].
Defined.

Lemma finites_are_omniscient : forall k, omniscient (finite_setoid k).
Proof.
(* to do *)
Qed.
```

1.3 The minimum function

In the next part, we will make use of a function that computes the minimum of a function `nat` $\to$ `bool` on a finite segment: `min` f n returns `true` if and only if f is constantly true on $\{0, \ldots, n-1\}$.

Question 5
Complete the definition:

```
Fixpoint min (f : nat → bool) (n:nat) :=
  (* to do *)
```

Question 6
(*) Show the following lemma

```
Lemma compute_minimum :
  forall f n, min f n = false -> exists p, f p = false ∧ (forall k, k < p -> f k = true).
```

2 The type $\mathbb{N}_\infty$

We are now looking to build an infinite type that is omniscient as we saw $\mathbb{N}$ is not. For that, we are interested in the type of decreasing infinite boolean sequences represented by functions `nat` $\to$ `bool`.

Question 7
Define the *setoid* $\mathbb{N}_\infty$:

```
Definition Decreasing (α : nat -> bool) :=
  forall i k, i ≤ k -> α i = false -> α k = false.
Definition N_infty : setoid.
refine (mkSetoid
```

```
        (set := { α : nat -> bool | Decreasing α })
        (R := fun α β => (* to do *))
        _).
(* to do *)
Defined.
Notation "ℕ∞" := N_infty.
Notation "x ≡ y" := (R N_infty x y) (at level 80). (* ≡ représente l'égalité sur ℕ∞ *)
```

We have a natural element of the type $\mathbb{N}_\infty$, the constant function equal to `true`:

Question 8

Define ω the member of $\mathbb{N}_\infty$ constantly equal to `true`.

```
Definition ω : set ℕ∞.
refine (exist _ (fun x => true) _).
(* to do *)
Defined.
```

Question 9

For each natural number $k \in \mathbb{N}$, define an element of $\mathbb{N}_\infty$ so that if α is ultimely equal to `false`, then it is in the image of `of_nat`.

```
Definition of_nat (k : nat) : set ℕ∞.
(* to do *)
Defined.
```

Question 10

(On paper) Show that, classically $\mathbb{N}_\infty =$ `of_nat`$(\mathbb{N}) \cup \{\omega\}$. What classical principle did you invoke?

Question 11

Show the following lemma:

```
Lemma LPO_equiv : omniscient ℕ <-> forall x : set ℕ∞, x ≡ ω \/ exists k, x ≡ of_nat k.
Proof.
(* to do *)
Qed.
```

Thus the existence of a bijection between $\mathbb{N}$ and $\mathbb{N}_\infty$ is equivalent to the omniscience of $\mathbb{N}$.

Question 12

(On paper) Justify informally why the omniscience of $\mathbb{N}$ cannot be derived in a logic that satisfies the disjunction theorem (if $\vdash A \lor B$ then either $\vdash A$ or $\vdash B$, constructively).

(Hint: Use the indecidability of the halting problem.)

However, there are in a sense, the only observable elements of this type:

Question 13
Show the following lemma:

```
Lemma density :
  forall p : set (ℕ∞ ⇒ bool_setoid),
    proj1_sig p ω = true ->
    (forall k, proj1_sig p (of_nat k) = true) ->
    forall x, proj1_sig p x = true.
Proof.
(* to do *)
Qed.
```

3 $\mathbb{N}_\infty$ is omniscient

We have now the tools to show that ℕ∞ is omniscient.

Question 14
Complete the definition:

```
Definition ε (p : set (ℕ∞ ⇒ bool_setoid)) : set ℕ∞.
refine (exist _ (fun n => min (fun m => proj1_sig p (of_nat m)) n) _).
(* to do *)
Defined.
```

Question 15
(*) Show that ε is a selection function:

```
Lemma ε_correct : forall p, p (ε p) = true <-> forall x, p x = true.
Proof.
(* to do *)
Qed.
```

(Hint: use the lemma `density`)

Question 16
Deduce from that the omniscience of $\mathbb{N}_\infty$.

Question 17
(*) Show that for all predicate $p : \mathbb{N}_\infty \to$ `bool`, it is either constantly true or it is false on a finite element:

```
Lemma finite_falsification :
  forall p : set (ℕ∞ ⇒ bool_setoid),
    (exists x, (¬ (x ≡ ω) /\ proj1_sig p x = false)) \/ (forall n, proj1_sig p (of_nat n) = true).
Proof.
(* to do *)
Qed.
```