MPRI 2014-15 Course 2-7-1

Examination december 3^{trd} 2014

2 hours.

1 Warm-up

A fellow student claims to have written terms of the following types in type theory. For each case, tell whether this is possible.

p_1 : $\Pi n : nat.\Sigma m : nat.\Sigma p : nat.n = m + m + p \wedge (\Sigma q : nat.p + q = 1)$
what can we say about the normal form of $\pi_1(p_1\ 12)$?

p_2 : $\Pi n : nat.\Sigma m : nat.m + m = n$

2 λ-hacking

We recall that the type of Church numerals in System F is:

$$Nat \equiv \forall\alpha.(\alpha \to \alpha) \to \alpha \to \alpha$$

.

a) Define a function d (for double) of type $Nat \to Nat$ which computes the double of a Church numeral. *Try to make this definition as short and elegant as possible.*

b) Similarly, define a function $d_1 : Nat \to Nat$ which corresponds to the function $n \mapsto 2n + 1$.

With Church numerals, the natural number n is represented by a term of size $O(n)$. We want to use a more compact representation, corresponding to the binary encoding on the natural number. A fellow student proposes the following type :

$$Bin \equiv \forall\alpha.(\alpha \to \alpha) \to (\alpha \to \alpha) \to \alpha \to \alpha$$

.

c) Can you describe the closed, normal terms of that type ? Explain how they can be viewed as natural numbers.

d) Define the function $B2C : Bin \to Nat$ which converts the binary representation of a natural number to the Church one.

3 Let's iterate

We are working in the Calculus of Constructions with the two sorts `Prop` and `Kind` (with `Prop : Kind`).

a) We define the type Nat : `Prop` of natural numbers, and the objects 0 (zero) and S (successor). Recall what is the usual way to define these.

We are given the following "mystery" predicate :

```
Definition myst : Nat -> Prop :=
 fun x : Nat =>
   forall P : Nat -> Prop,
     (P O) ->
     (forall x, (P x) -> (P (S (S (S x))))) ->
     (P x).
```

b) What is the "meaning" of this property ? Are there objects `n` such that `(myst n)` holds ? If so, give two examples of such proofs (that is examples of objects of type `(myst n)` for two different `n`).

c) Suppose you have an object `t` of type `(myst 666)`. What can you say about the following object:
`(t (fun _:Nat => Nat) O (fun (_ :  Nat)(x:Nat) => (S x))).`

d) Change the definition of myst, so that when `t :  (myst 1024)`, the term above reduces to (the Church numeral) 10.

4 Un soupcon de cardinalité

Georg Cantor, the father of modern set theory gave a famous proof that there is no injection from $N \to N$ to N. Here is a reminder of the proof.

> Suppose there is a sequence $u_1, u_2, \ldots$ of sequences of natural numbers. That is each u_i is itself a sequence $u_I^1, u_i^2, \ldots$.
>
> Consider then the sequence defined by $d = (u_i^i + 1)_{i \in N}$. It is easy to see (by a diagonal argument) that there is no j such that $d = u_j$.

Let us see if we can reproduce this argument in Type Theory.

In this part, you are asked to convince me that you have a precise view of how things should be done to formalize this in Coq. (Actually, we are naturally in type theory; so you can use Coq style syntax or λ-style syntax, as you wish.) You do not have to give tactics sequences or write complete proof terms, but state precisely what is proved. Try to be precise, clear and concise.

a) A is a *retract* of a type B if there exists functions $f : A \to B$ and $g : B \to A$ such that $g \circ f = Id_A$. You can define it as :

```
Definition retr (A B : Prop) :=
    exists f : A -> B, exists g : B -> A, forall x : A, g (f x) = x.
```

How can you prove that `nat` is a retract of `nat -> nat` ?
How do you show that `nat` is a retract of itself ?

b) How you state that `nat -> nat` is *not* a retract of `nat` ? How do you prove it ?

c) What *very* simple statement can you deduce about `nat` and `nat -> nat` ?

d) Suppose some developper of Coq has gone insane and added a new reduction rule :

$$nat \to nat \quad \rhd_\beta \quad nat$$

Is the system coherent ? (can one prove false)

e) Is there something else remarquable happening ?