Martin-Löf Type Theory: Example proof

Carl Factora

March 3, 2016

Preliminary definitions

```
open import Data.Nat using (; suc)
open import Relation.Binary.PropositionalEquality
  using (__; refl; cong; sym; trans)
open import Data.Product
open import Data.Empty

-- Natural Number Recursion and Induction principle
rec : (C : Set)  C  (  C  C)    C
rec C c f 0 = c
rec C c f (suc n) = f n (rec C c f n)

ind : (C :   Set)  C 0  ((n : )  C n  C (suc n))
       (n : )  C n
ind C c f zero = c
ind C c f (suc n) = f n (ind C c f n)

-- Addition
add :
add = rec (  ) ( n  n) ( m r n  suc (r n))
```

We also define the following propositions:

```
 : Set  Set
 A = A
```

```
__ : (i j : )  Set
i  j = [ k   ] (add i k  j)

_<_ : (i j : )  Set
i < j = (i  j)   (i  j)

0si : (i : )   (0  suc i)
0si i =  ()
```

Here, we define and `<` as:

- `i  j` is provable if and only if there exists a `k` such that `i + k = j`.
- `i < j` is provable if and only if we can provide a proof that `i  j` and that `i  j`.

Thus, both propositions are proven by constructing by pairs. The proof for `i  j` must be a dependent pair since its corresponding proposition uses a specific natural number (viz. `k`).

`0si` is a proof that 0 is never equal to the successor of a natural number. Here, we write `()` since the second assumption of the proof, `0  suc i`, allows us to simply appeal to the fact that it is absurd to say that 0 is equal to any natural number `+ 1`, by definition of the natural numbers and the fact that Agda assumes that terms that are constructed differently (i.e., 0 and `suc` for natural numbers) can never be equal.

The Proof

Using the above, we can prove the following:

```
0< : (i : )  (0 < suc i)
0< = ?
```

The proof of this proposition requires that we prove that 0 is less than all natural numbers. We do this using induction. More specifically, this proof requires us to perform induction on a natural number, `i`. To do this, we use the `ind` principle defined above. For each step of our proof, we also provide the corresponding proof environment (viz. `Proof Environment`) that represents the given proof obligation for each hole (viz. { }).

```
0< : (i : )  (0 < suc i)
0< = ind { }0 { }1 { }2

{---Proof Environment
  ?0 :   Set
  ?1 : 0 < suc 0
  ?2 : (n : )  0 < suc n  0 < suc (suc n)
-}
```

The first of these obligations (i.e., `?0`) requires us to declare the proposition we are wanting to prove. In general, this simply mirrors the proofs type definition.

```
0< : (i : )  (0 < suc i)
0< = ind ( i  0 < suc i) { }1 { }2

{---Proof Environment
  ?1 : 0 < suc 0
  ?2 : (n : )  0 < suc n  0 < suc (suc n)
-}
```

Our next obligation requires us to prove the proposition in the case that `i = 0` (i.e., we must show that `0 < suc 0`). Given our definition of `<`, the proof this by constructing a pair, containing proofs of `0  suc 0` and ` (0  suc 0)`.

```
0< : (i : )  (0 < suc i)
0< = ind ( i  0 < suc i)
          ({ }0 , { }1)
          { }2

{---Proof Environment
  ?0 : 0  suc 0
  ?1 :  (0  suc 0)
  ?2 : (n : )  0 < suc n  0 < suc (suc n)
-}
```

To prove `?0`, we must construct a dependent pair that contains a natural number `k`, such that `0 + k = 1`. Thus, we construct a pair containing the natural number 1 and as well as the proof that `1 = 1` (e.g. `refl`). To prove `?1`, we use our proof of `0si` on 0 to prove that ` (0  suc 0)`.

```
0< : (i : )  (0 < suc i)
0< = ind ( i  0 < suc i)
          ((1 , refl) , 0si 0)
          { }2

{---Proof Environment
  ?2 : (n : )  0 < suc n  0 < suc (suc n)
-}
```

To prove the final proposition, we must prove that for all natural numbers, `n`, with the assumption that 0 is less than `suc n`, we can show that 0 is less than `suc (suc n)` (i.e., `n + 2`).

```
0< : (i : )  (0 < suc i)
0< = ind ( i  0 < suc i)
          ((1 , refl) , 0si 0)
          ( n ih  { }0)

{---Proof Environment
  ?0 : 0 < suc (suc n)
-}
```

In this case, our inductive hypothesis, `ih`, is actually not necessary, since we can simply employ the same strategy as in the earlier proof for the base-case.

```
  0< : (i : )  (0 < suc i)
  0< = ind ( i  0 < suc i)
            ((1 , refl) , 0si 0)
            ( n _  ((suc (suc n)) , refl) , 0si (suc n))
```