Ingénierie de la preuve - CM4

19 février 2018

1 Polymorphisme et ordre supéreieur

1.1 polymorphisme

Quand une fonction sur un type peut être rendue générique et utilisable sur un autre type.

```
Definition : id_nat : (n : nat) := n.
Definition : id_bool : (b : bool) := b.
Definition id_gauss (g : gauss) := g.
```

1.2 Fonction polymorphe

```
Definition id (A : Set) (x : A) := x.
```

Évidement :

$$\forall n, id\_nat\ n = id\_nat\ n.$$
$$\forall b, id\_bool\ b = id\_bool\ b.$$
$$\forall g, id\_gauss\ g = id\_gauss\ g.$$

On pourrait redéfinir les fonctions :

$$Definition\ id\_nat := id\ nat.$$
$$Checkid\_nat.$$
$$id\_nat : nat \to nat$$
$$Definition\ id\_bool := id\ bool.$$

1.3 Ordre supérieur

Quand les fonctions prennent/peuvent prendre des fonctions en arguments

$$Definition\ id\ (A : Set)(x : A) := x.$$
$$Definition\ id\_fun := id(nat \to nat).$$
$$Check\ id\_fun.$$
$$(nat \to nat) \to (nat \to nat)$$

Une fonction polymorphe et d'ordre supérieur : exemple la fonction iter.

$$f^n = fof^{n-1}$$

$$f^n = f^{n-1}of$$

$$f^n(x) = f(f^{n-1}(x))$$

On aimerait avoir une fonction iter du type suivant :

$$\forall A : Set, (A \to A) \to nat \to A \to A$$

$$\begin{aligned}
&Fixpoint\ iter(A : Set)(f : A \to A)\\
&(n : nat)struct\ n : A \to A :\\
&matchnwith :\\
&0 => (fun(n : A) => n)\\
&|Sp => (fun(m : A) =>\\
&iter\ Afp(fm)\\
&end
\end{aligned}$$