Preuves assistées par ordinateur

Document 2 - Entraînement à l'écriture de fonctions

10 février 2025

I

Savoir écrire correctement une fonction

Un problème typique

En informatique, beaucoup de problèmes consistent à répondre des questions comme celle-ci :

« Écrire une fonction qui prend en entrée un ou plusieurs objets `objet_1`, `objet_2`, ..., et qui renvoie en sortie un objet « `objet_final` » dont la définition est `<DEFINITION A SATISFAIRE>` ».

Exemple du TP1

Par exemple dans le TP1, on pose la question :
« Écrire une fonction LEAN nommée `plus_deux` qui prend en entrée une fonction $f : \mathbb{N} \to \mathbb{N}$, et qui renvoie en sortie la fonction définie par $x \in \mathbb{N} \mapsto f(x) + 2 \in \mathbb{N}$ ».

Quatre questions à se poser

Avant de taper la réponse, il faut réfléchir et répondre aux quatre questions suivantes :

1. Combien d'objets la fonction à écrire doit-elle prendre en entrée ?
2. Quels sont les types de ces objets ?
3. Quel est le type de l'objet que la fonction doit produire en sortie ?
4. Ainsi, quel est le **type** de la fonction à écrire ?

Dans l'exemple du TP1, les réponses à ces questions sont les suivantes :

1. La fonction `plus_deux` prend en entrée **un seul objet** : une fonction $f : \mathbb{N} \to \mathbb{N}$.
2. Le type de l'objet en entrée est donc `Nat→Nat`.
3. La fonction `plus_deux` doit renvoyer la *fonction* $x \in \mathbb{N} \mapsto x + 2 \in \mathbb{N}$. Le type de l'objet à renvoyer est donc `Nat→Nat`.
4. La fonction `plus_deux` prend en entrée un objet de type `Nat→Nat`, et renvoie un objet de type `Nat→Nat`. Elle est donc de type

   `(Nat→Nat) → (Nat→Nat)`

Une fois cela fait, on peut écrire le début de la définition, sous la forme

```
def nom_fonction : type_fonction :=
    fun variable_1 variable_2... variable_n => expression_sortie
```

Bien sûr, il faut remplacer les noms `variable_1 variable_2... variable_n` et l'expression `expression_sortie` en fonction du contexte.

Pour les noms des variables, il est bien d'utiliser des noms qui indiquent de quels objets il s'agit : par exemple `n`, `m`, `p`... si ce sont des entiers, `x`, `y`, `z`... si ce sont des nombres réels, ou encore `f`, `g`, `h`... si ce sont des fonctions.

Exemple du TP1

• Dans l'exemple du TP, on sait que le type de la fonction à écrire est `(Nat→Nat)→(Nat→Nat)`.

Il y a une seule variable en entrée : la fonction $f : \mathbb{N} \to \mathbb{N}$. Appelons la « `f` », comme dans l'énoncé de l'exercice.

On peut donc déjà écrire le début de la définition de la façon suivante :

```
def plus_deux : (Nat→Nat)→(Nat→Nat):=
    fun f => expression_sortie
```

• Pour déterminer l'expression « `expression_sortie` », il faut se demander : « que doit renvoyer la fonction `plus_deux` ? La réponse est dans l'énoncé : la fonction renvoie la fonction $x \mapsto f(x) + 2$.

Il faut donc remplacer « `expression_sortie` » par « `fun x => (f x) + 2` » :

Réponse à la question du TP1

```
def plus_deux : (Nat→Nat)→(Nat→Nat):=
    fun f => ( fun x => (f x) + 2)
```

II

Un autre exemple

Exercice

Écrire en LEAN une fonction `decale` qui prend en entrée une fonction $f : \mathbb{N} \to \mathbb{N}$, puis un entier $n \in \mathbb{N}$, et qui renvoie la valeur $f(n) + n$.

Comme indiqué plus haut, il y a quatre questions à se poser.

1. Combien d'objets la fonction à écrire doit-elle prendre en entrée ?
   (Deux objets : une fonction $f : \mathbb{N} \to \mathbb{N}$ et un entier $n \in \mathbb{N}$).

2. Quels sont les types de ces objets ?
   (La fonction est du type `Nat→Nat` et l'entier du type `Nat`)

3. Quel est le type de l'objet que la fonction doit produire en sortie ?
   (La fonction renvoie un entier, donc un objet de type `Nat`)

4. Ainsi, quel est le **type** de la fonction à écrire ?
   (La fonction `evalue` prend en entrée un objet de type `Nat→Nat`, puis un objet de type `Nat`, puis renvoie un objet de type `Nat`). Elle est donc de type
   `(Nat → Nat) → Nat → Nat )`

On peut donc écrire le début de la définition, en donnant aux variables les noms « f » et « n » donnés dans l'énoncé :

```
def decale : (Nat→Nat)→Nat→Nat:=
    fun f n => expression_sortie
```

Que doit renvoyer la fonction ? La réponse est dans l'énoncé : la fonction renvoie la valeur « $f(n) + n$ ».

Si on écrit cela avec la syntaxe de LEAN, cela donne la réponse à la question.

Solution de l'exercice

```
def decale : (Nat→Nat)→Nat→Nat:=
    fun f n => (f n) + n
```

II

Quatre exercices

Voici quatre questions similaires. Les réponses sont sur la diapositive suivante.

1. Écrire une fonction `met_au_carre` qui prend en entrée une fonction $f : \mathbb{Z} \to \mathbb{Z}$, et qui renvoie la fonction $x \in \mathbb{Z} \mapsto f(x)^2 \in \mathbb{Z}$.

2. Écrire une fonction `compose` qui prend en entrée deux fonctions $f : \mathbb{Z} \to \mathbb{Z}$ et $g : \mathbb{Z} \to \mathbb{Z}$, et qui renvoie la fonction $f \circ g : x \in \mathbb{Z} \mapsto f(g(x)) \in \mathbb{Z}$.

3. Écrire une fonction `constante` qui prend en entrée un entier $n \in \mathbb{N}$ et qui renvoie en sortie la fonction constante $x \in \mathbb{N} \mapsto n \in \mathbb{N}$.

4. Écrire une fonction `trois_fonctions` qui prend en entrée trois fonctions $f, g, h : \mathbb{N} \to \mathbb{N}$ et qui renvoie en sortie la fonction $x \in \mathbb{N} \mapsto f(g(x)) - h(x) \in \mathbb{N}$.

Réponses

1. ```
   def met_au_carre : (Nat →Nat) → (Nat→Nat) :=
       fun f => (fun x => (f x)^2)
   ```

2. ```
   def compose : (Nat →Nat) → (Nat→Nat)→ (Nat →Nat):=
       fun f g => (fun x => f (g x))
   ```

3. ```
   def constante : Nat→ (Nat→Nat):=
       fun  n => (fun x => n)
   ```

4. ```
   def trois_fonctions :
   (Nat →Nat) → (Nat→Nat)→ (Nat →Nat)→ (Nat →Nat):=
       fun f g h => (fun x => f (g x) - (h x))
   ```