Programmation fonctionnelle (Le langage CAML)

Dr Mouhamadou GAYE

UFR Sciences Et Technologies
Département d'Informatique
Licence 3 en Informatique Option Génie Logiciel

28 septembre 2020

Chapitre 2 : Fonctions

Fonctions

- **Fonctions à un argument**

  let identifiant argument = expression ; ;

  Exemple :

  ```
  let carre x = x * x ; ;
  val carre : int → int = <fun>
  ```

  Appel de la fonction :

  ```
  carre 2 ; ;
  - : int = 4
  carre -1 ; ;
  Error
  carre (-1) ; ;
  - : int = 1
  ```

Fonctions

- **Fonctions à plusieurs arguments**

  let identifiant argument_1 argument_2 ... argument_n = expression ; ;

Exemple :

```
let perimetre x y = 2 * (x + y) ; ;
val perimetre : int → int → int = <fun>
```

Appel de la fonction :

```
perimetre 5 4 ; ;
- : int = 18
```

NB : La fonction f x y n'équivaut pas à la fonction f (x, y)

Fonctions

- **Définition locale de fonction dans une expression**

Exemple :

```
let cube x = x * x * x in carre 3 - carre 2 ; ;
- : int = 19
cube 3 ; ;
- : Error
```

Fonctions

- **Définition locale de fonction dans une définition globale de fonction**

Exemple :

```
let cube x = let carre y = y * y in x * carre x ; ;
val cube : int → int = <fun>
cube 3 ; ;
- : int = 27
```

Fonctions

- **Fonction anonyme**

  let identifiant = function argument$_1$ $\rightarrow$ function argument$_2$ $\rightarrow$ ...
  function argument$_n$ $\rightarrow$ expression ; ;

  Exemple :

  ```
  let suivant = function x -> x + 1 ; ;
  val suivant : int -> int = <fun>
  suivant 2 ; ;
  - : int = 3
  ```

Fonctions

- **Fonction récursive**

  let rec identifiant arguments = expression ; ;

Exemple :

```
let rec fact n =
        if n = 0 then
                1
        else
                n * fact (n - 1) ; ;
val fact : int → int = <fun>
fact 5 ; ;
- : int = 120
```