Programmation Fonctionnelle : des concepts aux applications web (NFP119)

Exceptions

Tristan Crolard

Laboratoire CEDRIC
Equipe « Systèmes Sûrs »

tristan.crolard@cnam.fr

cedric.cnam.fr/sys/crolard

La gestion des erreurs

Il existe deux sortes d'exceptions, celles qui correspondent à des erreurs dans le programme (et qui ne devraient donc pas se produire) et celles qui correspondent à des situations rares (mais qui peuvent se produire). En OCaml, les exceptions servent *a priori* à gérer les erreurs. La syntaxe est donnée dans ces exemples :

```
exception Empty

let hd l =
  match l with
   | (x::_) -> x
   | _ -> raise Empty
```

```
exception UnequalLengths

let rec zipEq (l1, l2) =
  match (l1, l2) with
  | (h1::t1, h2::t2) -> (h1, h2)::zipEq(t1, t2)
  | ([], []) -> []
  | (_, _) -> raise UnequalLengths
```

Remarque

Nous ne verrons pas comment attraper une exception (puisqu'elles ne doivent pas *a priori* être levée dans un programme correct).

Le type `'a option`

Le système de type nous permet de gérer les situations exceptionnelles. Une situation exceptionnelle fréquente corresond à l'absence de valeur à retourner. Le type `'a option` permet de gérer proprement cette situation.

Les constructeurs pour ce type sont :

```
None : 'a option
```

```
Some : 'a -> 'a option
```

Exemples

```
let head l =
  match l with
  | (h::_) -> Some h
  | [] -> None

let tail l =
  match l with
  | (_::t) -> Some t
  | [] -> None
```

Filtrage

Comme pour les listes, on utilise le filtrage pour récupérer la valeur éventuelle.

Exemples

```
let is_some opt =
  match opt with
  | Some _ -> true
  | None -> false

let get opt =
  match opt with
  | Some v -> v
  | None -> raise Not_found
```

(où `Not_found` est une exception prédéfinie de OCaml)

Les enregistrements (records)

Définition d'un type record (immutable)

```
type point = { x: int; y: int };;
```

Définition d'un record

```
let pt = { x = 3; y = 5 };;
let init (z: int): point = { x = z; y = z }
```

Consultation d'un record

```
let (px, py) = (pt.x, pt.y);;
match pt with { x = a; y = b } -> a + b;;
```

Mise à jour fonctionnelle d'un record

```
let pt2 = { pt with y = 9 };;
```