Programmation fonctionnelle

L2 Info et Math-info, 2018–19

Marc Zeitoun

5 octobre 2018

Contrôle continu

Tutorat disponible

- 4 tuteurs,
- Tutorat libre service (entre 12h et 14h).

Plan

Types somme

Types Somme

- Définition de types contenant un ensemble fini de valeurs :

```
type figure = Roi | Dame | Cavalier | Valet
```

- Similaire, pour l'instant, aux `enum` de C.

Types Somme

- Définition de types contenant un ensemble fini de valeurs :

```
type figure = Roi | Dame | Cavalier | Valet
```

- Similaire, pour l'instant, aux `enum` de C.
- Le nom du type est un identificateur, commençant par une minuscule.
- Les valeurs sont des noms commençant par une majuscule.

Types Somme

- On veut parfois faire porter une information supplémentaire aux valeurs.

```
type entier_ou_infini = Entier of int | Infini
```

Types Somme

- On veut parfois faire porter une information supplémentaire aux valeurs.

```
type entier_ou_infini = Entier of int | Infini
```

- Des valeurs légales de ce type sont
  - `Entier(8)`
  - `Entier(-42)` ,
  - `Infini` .

Types Somme

- On veut parfois faire porter une information supplémentaire aux valeurs.

```
type entier_ou_infini = Entier of int | Infini
```

- Des valeurs légales de ce type sont
  - `Entier(8)`
  - `Entier(-42)` ,
  - `Infini` .

Exercice : définir un type pour représenter des cartes de tarot.

Types Produit Cartésien

- L'opérateur `*` permet de représenter un **produit Cartésien**.
- Le produit Cartésien de A et B est l'ensemble

$$A \times B = \{(a, b) \mid a \in A \text{ et } b \in B\}.$$

Un type pour représenter des listes

- On peut redéfinir un type **liste d'entiers** de la façon suivante.

```
type liste_entiers = ListeVide
                   | Noeud of int * liste_entiers
```

(Définition en fait inutile, les constructeurs de listes sont prédéfinis)

Un type pour représenter des listes

- On peut redéfinir un type **liste d'entiers** de la façon suivante.

```
type liste_entiers = ListeVide
                   | Noeud of int * liste_entiers
```

(Définition en fait inutile, les constructeurs de listes sont prédéfinis)

- Éléments légaux de ce type ?

Un type pour représenter des listes

- On peut redéfinir un type **liste d'entiers** de la façon suivante.

```
type liste_entiers = ListeVide
                   | Noeud of int * liste_entiers
```

(Définition en fait inutile, les constructeurs de listes sont prédéfinis)

- Éléments légaux de ce type ?
- Comment définir un type liste **générique** ?

Un type pour représenter des listes

- On peut redéfinir un type **liste d'entiers** de la façon suivante.

```
type liste_entiers = ListeVide
                   | Noeud of int * liste_entiers
```

(Définition en fait inutile, les constructeurs de listes sont prédéfinis)

- Éléments légaux de ce type ?
- Comment définir un type liste **générique** ?

```
type 'a liste = ListeVide
              | Noeud of 'a * 'a liste
```

La construction match

- Permet d'**inspecter** la forme d'une valeur.
- Calcule une expression.
- **Très utile** pour les types somme.

```
match x with
  | ListeVide   -> .....
  | Noeud(x, l) -> .....
```

La construction match

- Permet d'**inspecter** la forme d'une valeur.
- Calcule une expression.
- **Très utile** pour les types somme.

```
match x with
  | ListeVide   -> .....
  | Noeud(x, l) -> .....
```

- Ce qui est calculé par `match` est une expression des `.....`, celle qui correspond à la valeur de l'expression `x`.

La construction match

- Permet d'**inspecter** la forme d'une valeur.
- Calcule une expression.
- **Très utile** pour les types somme.

```
match x with
  | ListeVide    -> .....
  | Noeud(x, l) -> .....
```

- Ce qui est calculé par `match` est une expression des `.....`, celle qui correspond à la valeur de l'expression `x`.
- **Exemple** Calcul de la longueur d'une liste.