Manipulation des types sommes

laurent.jospin.59@free.fr, http://jospin.lstl.fr

Lycée Saint-Louis, Paris

1 Un type pour Rien ou Quelque chose

Le type `'a option` défini par : **`type`** `'a option = None | Some` **`of`** `'a` permet de définir des valeurs qui peuvent être soit `None`, soit d'un type donné. Ce type est prédéfini en Caml.

1. Ecrire une fonction qui ajoute deux `int option` et renvoie un `int option`. On renverra `None` dès que l'un des deux vaut `None`.

2. Ecrire une fonction qui prend en argument une fonction `'a -> 'b` et une valeur de type `'a option` et qui renvoie un `'b option` correspondant à l'image si la valeur n'est pas `None` et `None` sinon. Qu'obtient-on avec une application partielle ?

Remarque. Le module `Option` contient tout un tas de fonctions utiles (à partir de 4.08).

2 Des Entiers façon Péano

Un type somme récursif permet d'exprimer, dans ce type, des valeurs contenant un nombre non borné de constructeurs.

Exemple.

```
1 type entierPeano = Zero | Suivant of entierPeano
```

3. Ecrire des fonctions de somme, produit et différence sur de tels entiers.

4. Ecrire des fonctions mutuellement récursives testant la parité et l'imparité d'un tel entier.

5. Ecrire une version sans récursivité mutuelle d'une des deux fonctions précédentes.

6. Ecrire une fonction donnant le reste (sous la forme d'un entier de Péano) de la division par 3 d'un entier de Péano.

3 Une liste de valeurs

Si, de plus, certains constructeurs portent avec eux au moins une donnée, on peut maintenant construire des structures de données.

Le type `'a list` que l'on pourrait définir par : **`type`** `'a list = Nil | Cons` **`of`** `'a * 'a list` permet de représenter une liste de valeurs.

Exemple.

— `Nil` est une liste vide,
— `Cons(1, Nil)` est une liste contenant 1,
— `Cons(2, (Cons 1, Nil))` est une liste contenant 2 en tête, puis 1.

La version intégrée à Caml utilise `[]` à la place de `Nil` et une version infixe du constructeur `Cons` notée `::` . Ainsi `[]` est une liste vide, et `4::3::2::1::[]` une liste à quatre éléments qu'on peut aussi noter plus commodément `[4; 3; 2; 1]`. Une liste de valeur est donc soit vide, soit un couple constitué d'une première valeur appelée la tête et d'une liste appelée la queue.

On travaillera directement sur le type `list` prédéfini.

7. Ecrire une fonction à valeur dans `'a option` donnant, si elle existe, la valeur d'indice k dans la liste en indiçant à 0 à partir de la tête.

8. Ecrire une fonction qui détermine la somme des valeurs d'une liste d'entiers.

9. Ecrire une fonction qui généralise le principe de la fonction précédente pour une opération quelconque fournie en argument.

10. Ecrire une fonction qui détermine le maximum d'une liste (non vide).

4 Un type pour représenter des expressions arithmétiques

Si un constructeur porte avec lui plusieurs valers du type récursif, on peut alors construire une structure arborescente.

Exemple.

```
type 'a expression =
    | Oppose of 'a expression
    | Somme of 'a expression * 'a expression
    | Produit of 'a expression * 'a expression
    | Constante of 'a
```

11. Ecrire une fonction récursive évaluant (*ie* donnant le résultat) une telle expression dans le cas où elle porte sur des entiers.