programme sur cette donnée.

L'ajout des autres types de données dynamiques ne permet pas de calculer davantage de fonctions. Cela est dû au fait qu'en utilisant une bijection entre $\mathbb{N}$ et $\mathbb{N}^2$, il est, en théorie, possible de coder n'importe quelle donnée dynamique comme un grand entier. Autrement dit, avoir un type des grands entiers ou tous les types de données dynamiques est équivalent du point de vue de la calculabilité.

5.4 Caml

On peut définir en Caml un type enregistrement récursif, par exemple

```
type liste = {hd : int; tl : liste;}
```

Mais, comme il n'y a pas de valeur `null`*, la plus petite solution de cette définition est vide. Autrement dit, ce type ne contient que des valeurs infinies.*

Il faut donc expliciter l'union disjointe entre le singleton `{vide}` *et le produit cartésien* `int` $\times$ `liste`*. Une solution est d'utiliser un champ sélecteur*

```
type liste = {select : bool; hd : int; tl : liste;}
```

mais il faut une valeur par défaut de type `liste` *pour le cas où la liste est vide et, encore une fois, il n'y a pas de valeur* `null` *en Caml. La seule solution serait d'utiliser une valeur infinie comme valeur par défaut, ce qui, bien que possible, n'est pas très élégant.*

Pour résoudre ce problème, Caml a une construction primitive qui permet de construire des unions disjointes de produits cartésiens. L'union disjointe des types $\mathtt{T}^1_1 \times \ldots \times \mathtt{T}^1_{n_1}, \ldots, \mathtt{T}^p_1 \times \ldots \times \mathtt{T}^p_{n_p}$ *se définit par*

```
type T = C1 of T11 * ... * T1n1 | ... | Cp of Tp1 * ... * Tpnp
```

où $\mathtt{C}_1, \ldots, \mathtt{C}_p$ *sont des constantes qui remplacent les valeurs du champ sélecteur et que l'on appelle* constructeurs*, bien qu'elles aient peu à voir avec les constructeurs de Java.*

Par exemple, l'union disjointe d'un singleton — produit cartésien vide — et du produit cartésien `int` $\times$ `liste` *se définit ainsi*

```
type liste = Nil | Cons of int * liste

let l = Cons (5,Cons (4,Nil))
in print_list l
```