2.2.4 Caml

La définition de la fonction Σ de Caml présente quelques différences avec la définition de la fonction Σ de Java. Tout d'abord, en Caml, tous les arguments formels sont finaux, on ne crée donc jamais de nouvelles références au moment de l'appel d'une fonction.

Ensuite, il y a, en Caml, un seul espace de noms pour les fonctions et les variables. En Java, le programme

```
class Prog {

 static int f (final int x) {
  return x + 1;}

 static int f = 4;

 public static void main (String [] args) {
  System.out.println(f(f));}}
```

est correct et dans l'expression `f(f)`*, la première occurrence de* `f` *désigne la fonction* `f` *et la seconde la variable* `f`*. En Caml, en revanche, le programme*

```
let f x = x + 1 in let f = 4 in print_int(f f)
```

est incorrect. En effet, les deux occurrences de `f` *désignent la variable entière* `f`*, la fonction étant cachée par la variable entière. Il n'y a donc pas d'environnement global : les variables globales sont déclarées et les fonctions sont définies dans l'environnement, comme les variables. Au moment de l'appel d'une fonction* `f`*, il est donc impossible de fabriquer l'environnement dans lequel on doit évaluer le corps de la fonction en utilisant l'environnement global. Dans l'environnement, on doit donc associer au nom* `f`*, non seulement la liste des arguments formels et le corps de la fonction, mais aussi l'environnement à étendre avec les arguments pour exécuter le corps de la fonction. Cet environnement est l'environnement dans lequel la fonction est définie.*

Ainsi, alors que le programme Java

```
class Prog {

 static int f () {return x;}

 static int x = 4;

 public static void main (String [] args) {
  System.out.println(f ());}}
```