MPSI, TP d'informatique: familiarisation avec Caml Light

Alexis Saurin

21 janvier 2003

1 Expressions de base en Caml Light

Question 8: Définitions

```
let x = 2 + 5;;
2*x-1;;
y;;
let test = 1<2;;
if test then x else 2*x;;
```

Question 9: Définitions locales

```
let z=1 in z+x;;
z;;
let test = 3 in test + 4;;
test;;
```

Question 10: Les fonctions
Caml Light est ce qu'on appelle un langage fonctionnel, c'est-à-dire que les fonctions sont des expressions comme les autres : elles peuvent être passées en arguments d'autres fonctions, renvoyées en résultat... on peut manipuler une fonction en tant que telle et pas seulement via l'application à un argument.

```
fun x -> x;; (* une fonction est donc
une expression comme une autre... *)
let id = fun x -> x;;  (* ... que l'on
peut nommer grâce à la construction let,
comme toute autre expression de Caml light*)
id 1;;    (* la notation de type 'a -> 'a *)
id 2.99;; (* pour la fonction id indique *)
id true;; (* qu'on peut l'appliquer à *)
id id;;   (* n'importe quoi... *)
fun x -> x+1;;
let incremente = fun x -> x+1;;
let incremente_bis x = x+1;;
let incremente_ter = function x -> x+1;;
incremente 2;;
incremente 4.5;;
if true then id else x -> x+1;;
```

2 Premiers programmes en Caml Light

On va maintenant écrire quelques programmes en Caml Light utilisant les expressions que l'on a rencontré jusque là.

2.1 Un peu de calcul

Question 11:
Construire la fonction qui ajoute 3 à un entier.
Construire la fonction qui rend 0 si son entrée vaut 0 et 1 si c'est n'importe quel autre entier.

Question 12: Élévations au carré
Programmer en Caml Light la fonction qui élève au carré, dans le cas d'entiers (`carre_int`), dans le cas de flottants (`carre_float`).
Testez `carre_int` sur quelques valeurs que vous choisirez, puis sur les nombres 262144 et 16777216. Comment vous expliquez-vous ces résultats ?

On dispose des fonctions `int_of_float` et `float_of_int` qui permettent de faire la conversion entre entiers et flottants. Essayez-les sur quelques exemples significatifs pour voir leur comportement.
Que pensez-vous de la définition suivante de `carre_int` :

```
let carre_int x = int_of_float (carre_float
                  (float_of_int x));;
```

? En quoi cette définition est-elle moins satisfaisante que la précédente ? en quoi peut-elle poser problème (pensez au fait que sur un ordinateur, `int` n'est pas vraiment égal à $\mathbb{N}$ et que `float` n'est pas vraiment $\mathbb{R}$...) ? Pose-t-elle vraiment problème ici ?

Question 13: Des calculs erronés
On a vu à la question précédente que `carre_int` avait sur certains nombres un comportement que l'on n'avait pas prévu au départ. Cela est dû aux limites physiques de l'ordinateur et à la représentation interne des entiers (ce n'est pas un trait propre à Caml

Light, tous les langages de programmation ont des limitations de ce genre). On va voir maintenant un autre cas où cette limitation apparaît (mais pour les flottants cette fois) :

Construire la fonction `constante_1` qui prend un couple de flottants (a,b) et qui retourne `1.` si a est nul et $\frac{a+b-b}{a}$ sinon. Testez cette fonction sur différentes valeurs du couple (a, b), a restant petit et b devenant grand (notamment : `1000000000000.`, `100000000000000.`, `100000000000000000.` pour b, a valant `1.`).

Question 14: Des définitions par récurrences ?
Les fonctions numériques qu'on a défini jusqu'ici ne sont pas très intéressantes. Cela vient en grande partie du fait que la plupart des fonctions calculables intéressantes de $\mathbb{N}$ dans $\mathbb{N}$ sont définies par récurrence (ou au moins contiennent quelque part dans leur définition une définition par récurrence), et qu'on ne sait pas pour l'instant définir en Caml Light une fonction par récurrence.

Essayer le texte suivant dans l'interpréteur Caml Light :

```
let frec x = if x=0 then 1 else frec (x-1);;
```

puis :

```
let frec n = 2*n;;
let frec x = if x=0 then 1 else frec (x-1);;
f 4;;
```

Comprenez-vous ce qui se passe ?

On voudrait tout de même définir la fonction factorielle en Caml Light. Écrire un programme calculant la factorielle sans utiliser de définition par récurrence, mais en utilisant les éléments de programmation impérative déjà vus en cours.

2.2 Manipulation de fonctions

Puisque pour l'instant on dispose de peu de choses pour définir des fonctions numériques, on va plutôt s'intéresser à la notion de fonctionnalité en elle-même : on va s'intéresser à des fonctions manipulant d'autres fonctions.

Question 15: Fonctions d'ordre supérieur

1. Construire une fonction `applique` qui prend un couple constitué d'une fonction f et d'un élément x en argument et qui retourne f(x). Observez bien le type de la fonction `applique`.
2. Construire la fonction `compose` qui prend une fonction f et une fonction g en arguments et qui retourne la fonction composée : fog.

Question 16: Fonctions à plusieurs arguments, curryfication et décurryfication

1. Définir une fonction `somme1` qui prend en argument un entier n et retourne la fonction qui ajoute n ;
2. Définir une fonction `somme2` qui prend en argument une paire d'entiers et retourne leur somme.

Dans les deux cas précédents, on a en fait cherché à construire une fonction de plusieurs arguments de deux manières différentes :
- la première manière : `let f x y = ...` est appelée le style curryfié ;
- la seconde manière : `let f (x,y) = ...` est appelée le style décurryfié ;

On voit bien que l'on peut exprimer exactement les mêmes fonctions dans chacun des deux styles[1], même si syntaxiquement, les deux styles sont différents (il suffit de regarder les types de `somme1` et `somme2`...).

On veut rendre automatique la traduction d'un style dans l'autre. Écrire une fonction `curryfier` et une fonction `decurryfier` qui font cette traduction.

2.3 Un nouveau type

Question 17: Enregistrements
Il existe d'autres types en Caml Light que ceux déjà rencontrés, et on peut en définir de nouveaux, en particulier une famille de types appellés types enregistrements :

```
type pave = {L : float; l: float; h: float};;
let p={L=1.5; l=1.; h=3.8};;
p.L;;
```

Écrire un programme calculant le volume d'un parallélépipède rectangle.

Question 18: Nombres complexes
Définir un type pour les nombres complexes et les opérations principales sur les complexes : partie réelle, partie imaginaire, addition, multiplication, conjugaison, module.

[1] cela vient en fait de l'isomorphisme entre les espaces de fonctions C^{A*B} et $(C^B)^A$