DS 1

Option informatique, première année

Julien Reichert

Toutes les fonctions de ce DS sont à écrire en Caml.

Exercice 1 : Réécrire une version de la fonction (déjà existante, pour info) de signature `rev : 'a list -> 'a list` telle que `rev l` retourne le miroir de `l`.

Exercice 2 : Réécrire de même une version des fonctions `for_all` et `exists` qui déterminent si tous les éléments (resp. au moins un élément) de la liste en deuxième argument satisfont (resp. satisfait) un prédicat [1] en premier argument. On donnera d'abord la signature des fonctions.

Exercice 3 : Prouver que la fonction `mccarthy` définie ci-dessous retourne `91` quel que soit l'entier inférieur à `102` en argument.

```
let rec mccarthy = function
| n when n > 100 -> n - 10
| n -> mccarthy (mccarthy n + 11);;
```

Exercice 4 : Soit une fonction de signature `f : int list -> int list` telle que `f l` existe dès que `l` n'est pas la liste vide et `hd l >= 0`, et telle que `f l` soit une liste dont le premier élément est positif et soit ce premier élément est strictement inférieur au premier élément de `l`, soit la taille de `f l` est strictement inférieure à la taille de `l`. [2] On considère la fonction suivante : `let rec boucle = function [0] -> false | l -> boucle (f l);;`.

1) Prouver que la terminaison de `boucle` n'est pas garantie en donnant un exemple de fonction `f` respectant ces propriétés (avec son code) et une liste initiale provoquant une boucle infinie.

2) Prouver que la terminaison de `boucle` est garantie quelle que soit la fonction `f` telle que pour toute liste `l` le premier élément de `f l` soit de plus toujours inférieur ou égal au premier élément de `l`.

Exercice 5 : Démontrer qu'il n'existe pas de fonction `termine : (int -> int) -> int -> bool` telle que `termine f n` détermine si le calcul de `f n` termine.

Remarque : Ce résultat se généralise et il s'agit d'une preuve fondamentale en calculabilité : il est indécidable de créer une machine de Turing qui détermine si une machine de Turing s'arrête sur son entrée.

Indication : Raisonner par l'absurde et écrire une fonction récursive `f` impliquant le résultat de `termine` sur `f`. En pratique, on pourra démontrer qu'il n'existe pas de fonction `termine : int -> int -> bool` dont le premier argument est le numéro d'une fonction de signature `(int -> int)` (ces fonctions étant dénombrables [3]).

1. c'est-à-dire une fonction prenant en argument des objets du type commun des éléments de la liste et retournant un booléen
2. En particulier, cela impose que dans ce cas `f [0]` soit la liste vide (`f []` provoque une erreur quoi qu'il arrive).
3. Conséquence : puisque le nombre de fonctions mathématiques de $\mathbb{Z}$ dans $\mathbb{Z}$ est indénombrable, il y en a qui ne peuvent pas être représentées par des fonctions écrites sous forme de programmes.