Induction sur les entiers

Utilisations

– Pour définir des fonctions sur les entiers
– Pour définir des fonctions sur d'autres objects

Induction sur les entiers pour la multiplication

$$\begin{aligned} \texttt{mul}(0, m) &= 0 \\ \texttt{mul}(n+1, m) &= \texttt{mul}(n, m) + m \end{aligned}$$

Complexité de `mul`

Quel est le nombre d'appels à la fonction `mul` ?

$$\begin{aligned} &\texttt{mul}(4,5) \rightarrow \\ &\texttt{mul}(3,5)+5 \rightarrow \\ &\texttt{mul}(2,5)+5+5 \rightarrow \\ &\texttt{mul}(1,5)+5+5+5 \rightarrow \\ &\texttt{mul}(0,5)+5+5+5+5 \rightarrow \\ &0+5+5+5+5=20 \end{aligned}$$

Induction sur les entiers pour l'exponentiation

On veut définir la fonction

$$\texttt{exp}(m, r) = \underbrace{m \times \ldots \times m}_{r \text{ fois}}$$

$$\texttt{exp}(m, r) = \underbrace{m \times \ldots \times m}_{r-1 \text{ fois}} \times m$$

Ceci donne l'équation

$$\texttt{exp}(m, r) = \texttt{exp}(m, r-1) \times m$$

Quelles sont les bonnes valeurs de r pour que cette équation soit bien définie ? $r > 0$.

Ceci donne deux équations :

$$\begin{aligned} \texttt{exp}(m, 0) &= 1 \\ \texttt{exp}(m, n+1) &= \texttt{exp}(m, n) * m \end{aligned}$$

Complexité de `exp` (I)

Quel est le nombre d'appels à la fonction `exp` ?

$$\begin{aligned} &\texttt{exp}(3,6) \rightarrow \\ &\texttt{exp}(3,5)*3 \rightarrow \\ &\texttt{exp}(3,4)*3*3 \rightarrow \\ &\texttt{exp}(3,3)*3*3*3 \rightarrow \\ &\texttt{exp}(3,2)*3*3*3*3 \rightarrow \\ &\texttt{exp}(3,1)*3*3*3*3*3 \rightarrow \\ &\texttt{exp}(3,0)*3*3*3*3*3*3 \rightarrow \\ &1*3*3*3*3*3*3=729 \end{aligned}$$

Est-ce qu'on peut faire mieux ?

r est pair :	m^{2*n}	$=$	$(m^2)^n$
r est impair :	m^{2*n+1}	$=$	$(m^2)^n * m$

Quelles sont les bonnes valeurs de n pour que ces équations soient bien définies ? $n > 0$ et $n \geq 0$. Ceci donne

$$\begin{aligned} \texttt{exp}(m, 0) &= 1 \\ \texttt{exp}(m, 2*(n+1)) &= \texttt{exp}(m*m, n+1) \\ \texttt{exp}(m, (2*n)+1) &= \texttt{exp}(m*m, n)*m \end{aligned}$$

Complexité de `exp` (II)

Quel est le nombre d'appels à la fonction $\texttt{exp}(n, m)$?

$$\begin{aligned} &\texttt{exp}(3,6) \rightarrow \\ &\texttt{exp}(9,3) \rightarrow \\ &\texttt{exp}(81,1)*9 \rightarrow \\ &\texttt{exp}(6561,0)*81*9 \rightarrow \\ &1*81*9=729 \end{aligned}$$

$$\begin{aligned} &\texttt{exp}(3,8) \rightarrow \\ &\texttt{exp}(9,4) \rightarrow \\ &\texttt{exp}(81,2) \rightarrow \\ &\texttt{exp}(6561,1) \rightarrow \\ &\texttt{exp}(6561*6561,0)*6561 = 1*6561 = 6561 \end{aligned}$$

$2 + log_2 n = log_2(2n)$ pour $n > 0$

Induction sur les entiers pour compter (I)

Définition : Une permutation d'un ensemble fini A est une bijection de A dans A.

Exercice : Soit $n \geq 0$. Le nombre de permutations d'un ensemble de n éléments (noté P_n) est $n!$.

Induction sur les entiers pour compter (II)

Définition : Soit $n \geq 0$. Un arrangement d'ordre $k \leq n$ d'un ensemble A de n éléments est un sous-ensemble totalement ordonné de A ayant k éléments.

Exercice : Le nombre d'arrangements d'ordre k d'un ensemble A de n éléments (noté A_n^k) est $\frac{n!}{(n-k)!}$.

Induction sur les entiers pour compter (III)

Définition : Soit $n \geq 0$. Une combinaison d'ordre $k \leq n$ d'un ensemble A de n éléments est un sous-ensemble de A ayant k éléments.

Exercice : Le nombre de combinaisons d'ordre k d'un ensemble A de n éléments (noté C_n^k) est $\frac{n!}{(n-k)!k!}$.

Fonction entières : la suite de Fibonacci

On définit par recurrence les entiers de Fibonacci comme la suite ayant les propriétés suivantes :

$fib(0) = 1$
$fib(1) = 1$
$fib(n) = fib(n-1) + fib(n-2)$ pour $n \geq 2$

On peut montrer par induction que, pour $n > 0$, on a

$$fib(n) = \frac{\sqrt{5}+1}{2\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n + \frac{\sqrt{5}-1}{2\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n$$

C'est donc une fonction exponentielle !

Fibonacci (II)

En Ocaml :

```
# let rec fib =
      function
   0 -> 1
      | 1 -> 1
      | n -> fib(n-1)+fib(n-2);;
val fib : int -> int = <fun>
```

Le nombre d'appels recursifs a la fonction fib suit la meme lois que fib elle même !

Fibonacci (III)

Peut-on faire mieux ?

Solution 1 : utiliser la forme close, mais cela depend de la precision des flottants.

Solution 2 : travailler avec des couples

$$\begin{aligned} (fib(n), fib(n-1)) &= (fib(n-1) + fib(n-2), fib(n-1)) \\ &= (fun(x,y) -> (x+y, x))(fib(n-1), fib(n-2)) \end{aligned}$$

```
let fibetape (x,y) = x+y,x;;
val fibetape : int * int -> int * int = <fun>

let rec iter f a = function 0 -> a | n -> f(iter f a (n-1));;
val iter : ('a -> 'a) -> 'a -> int -> 'a = <fun>

let fibcouple n = iter fibetape (1,1) n;;
```

```
val fibcouple : int -> int * int = <fun>

let fastfib n = snd (fibcouple n);;
val fib : int -> int = <fun>
```

Exercice : prouver par induction que `fibcouple n = (fib n+1, fib n)`.
En déduire que `fastfib n = fib n`.

Recherche d'équations récursives à un argument

Pour définir une fonction récursive f, on peut chercher à exprimer f sous la forme d'une équation de la forme :

$$f(x) = g(x, f(h(x)))$$

- h représente la façon dont on passe de la valeur x à une valeur "plus petite" $h(x)$ sur laquelle on procède à l'appel récursif $f(h(x))$. Cette fonction doit vérifier $h(x) < x$ pour un ordre $<$ bien fondé.
- g traduit la façon dont on utilise le résultat de l'appel récursif pour décrire la valeur de $f(x)$.

Exemple de la fonction `Sigma`

$\texttt{Sigma} : \mathbb{N} \to \mathbb{N}$ est définie par :

$$\texttt{Sigma}(n) = n + (n-1) + (n-2) + \ldots + 1 + 0$$

L'équation récursive est

$$\texttt{Sigma}(x) = x + \texttt{Sigma}(x-1)$$

On a donc $h(x) = x - 1$ et $g(n, m) = n + m$.

Cas particuliers dans les équations récursives

Il faut identifier les cas pour lesquels les équations posent problème :
- soit parce qu'elles donnent des résultats faux,
- soit parce qu'elles ne correspondent pas à des expressions bien formées ($h(x)$ n'est pas dans le domaine de f).

On traite ces cas à part, ce sont les cas de base.

Exemple de la fonction `Sigma`

La valeur $h(0)$ n'est pas dans le domaine de la fonction `Sigma`. On pose donc

$$\begin{array}{lcl} \texttt{Sigma}(0) & = & 0 \\ \texttt{Sigma}(x) & = & x + \texttt{Sigma}(x-1), \text{ pour } x > 0 \end{array}$$

Un autre exemple

$\texttt{Pi} : \mathbb{N}^+ \to \mathbb{N}^+$ est définie par :

$$\texttt{Pi}(n) = \begin{cases} n^2 \times (n-2)^2 \times (n-4)^2 \times \ldots \times 2, & \text{si } n \text{ pair} \\ n^2 \times (n-2)^2 \times (n-4)^2 \times \ldots \times 1, & \text{si } n \text{ impair} \end{cases}$$

L'équation récursive est

$$\texttt{Pi}(x) = x^2 \times \texttt{Pi}(x-2)$$

On a donc $h(x) = x - 2$ et $g(n, m) = n \times m$.

Les valeurs $h(2)$ et $h(1)$ ne sont pas dans le domaine de la fonction. On pose donc

$$\begin{array}{lcl} \texttt{Pi}(1) & = & 1 \\ \texttt{Pi}(2) & = & 4 \\ \texttt{Pi}(x) & = & x^2 \times \texttt{Pi}(x-2), \text{ pour } x > 2 \end{array}$$

Équations récursives à plusieurs arguments

On generalise le schéma récursif :

$$f(y, x) = g(y, x, f(i(y), h(x)))$$

- $h(x)$ produit une valeur "plus petite" que x. C'est l'argument sur lequel on fait la récurrence.
- $i(y)$ produit une valeur différente de y (pas forcement plus petite).
- g traduit la façon dont on utilise le résultat de l'appel récursif pour décrire la valeur de $f(y, x)$.

Exemple de la fonction `exp` (première version)

Mettre l'équation $exp(y, x) = y \times exp(y, x-1)$ sous la forme

$$exp(y, x) = g(y, x, exp(i(y), h(x)))$$

on obtient :

$$h(x) = x - 1 \quad i(y) = y \quad g(y, x, z) = y \times z$$

ayant comme cas particulier :

$$exp(y, 0) = 1$$

Exemple de la fonction `exp` (deuxième version)

Mettre

$$\begin{array}{lcl} \texttt{exp}(m, 0) & = & 1 \\ \texttt{exp}(m, 2*(n+1)) & = & \texttt{exp}(m*m, n+1) \\ \texttt{exp}(m, (2*n)+1) & = & \texttt{exp}(m*m, n)*m \end{array}$$

sous la forme

$$exp(y, x) = g(y, x, exp(i(y), h(x)))$$

On obtient

$$h(x) = x \; div \; 2$$
$$i(y) = y \times y$$
$$g(y, x, z) = \begin{cases} z & \text{si } x \text{ pair} \\ y \times z & \text{si } x \text{ impair} \end{cases}$$

ayant comme cas particulier

$$exp(y, 0) = 1$$

car $h(0) \not< 0$.