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

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

Complexité de `mul`

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

$$\begin{aligned} &\texttt{mul}(4,89) \rightarrow \\ &\texttt{mul}(3,89)+89 \rightarrow \\ &\texttt{mul}(2,89)+89+89 \rightarrow \\ &\texttt{mul}(1,89)+89+89+89 \rightarrow \\ &\texttt{mul}(0,89)+89+89+89+89 \rightarrow \\ &0+89+89+89+89=356 \end{aligned}$$

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 :

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

Complexité de exp (I)

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

$\texttt{exp}(3,6) \rightarrow$

$\texttt{exp}(3,5) * 3 \rightarrow$

$\texttt{exp}(3,4) * 3 * 3 \rightarrow$

$\texttt{exp}(3,3) * 3 * 3 * 3 \rightarrow$

$\texttt{exp}(3,2) * 3 * 3 * 3 * 3 \rightarrow$

$\texttt{exp}(3,1) * 3 * 3 * 3 * 3 * 3 \rightarrow$

$\texttt{exp}(3,0) * 3 * 3 * 3 * 3 * 3 * 3 \rightarrow$

$1 * 3 * 3 * 3 * 3 * 3 * 3 = 729$

Est-ce qu'on peut faire mieux ?

$$\begin{array}{llcl} r \text{ est pair :} & m^{2*n} & = & (m^2)^n \\ r \text{ est impair :} & m^{(2*n)+1} & = & (m^2)^n * m \end{array}$$

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

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

Complexité de exp (II)

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

$\texttt{exp}(3, 6) \rightarrow$

$\texttt{exp}(9, 3) \rightarrow$

$\texttt{exp}(81, 1) * 9 \rightarrow$

$\texttt{exp}(6561, 0) * 81 * 9 \rightarrow$

$1 * 81 * 9 = 729$

$\texttt{exp}(3, 8) \rightarrow$

$\texttt{exp}(9, 4) \rightarrow$

$\texttt{exp}(81, 2) \rightarrow$

$\texttt{exp}(6561, 1) \rightarrow$

$\texttt{exp}(6561 * 6561, 0) * 6561 = 1 * 6561 = 6561$

$2 + log_2 m$

Induction sur les entiers pour raisonner sur les mots

Soient L_1 et L_2 deux langages sur l'alphabet A. On définit la concatenation de L_1 et L_2 comme :

$$L_1 \circ L_2 = \{concat(u, v) \mid u \in L_1, v \in L_2\}$$

On définit la puissance d'un langage L comme :

$$L^0 = \{\epsilon\} \text{ et } L^{n+1} = L \circ L^n$$

On définit l'étoile d'un langage L comme :

$$L^* = \bigcup_{n \in \mathbb{N}} L^n$$

Exercice : Soient L et M deux langages sur l'alphabet A t.q. $\epsilon \notin L$. Montrer que

$$L^* \circ M = L \circ (L^* \circ M) \cup M$$

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

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

Induction sur les entiers pour compter (III)

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 approcher une fonction

Problème original :

Étant donné $x \in \mathbb{R}$ t.q. $x \geq 0$ trouver la racine carrée y de x, c'est-à-dire y vérifiant $y^2 = x$.

$$\texttt{racine} : \mathbb{R}^* \rightarrow \mathbb{R}^* \quad (\texttt{racine}(x) \text{ calcule } \sqrt{x})$$

Problème reformulé :

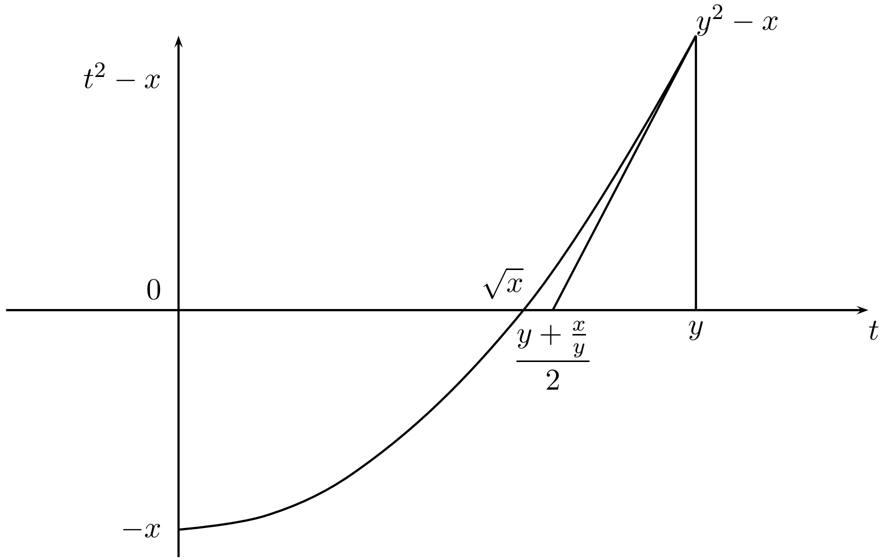
On cherche une valeur approchée y de la racine carrée d'un nombre réel, c'est-à-dire y vérifiant $|y^2 - x| < \epsilon$ pour un ϵ arbitraire.

$$\texttt{racine} : \mathbb{R}^* \times \mathbb{R}^* \rightarrow \mathbb{R}^* \quad (|\texttt{racine}(x, \epsilon)^2 - x| < \epsilon)$$

Méthode de Newton

- S'applique à une fonction f sur les réels
- Permet de construire une suite $y_o, y_1, y_2, \ldots$ qui converge vers un point y_n tel que $f(y_n) = 0$.

Dans notre exemple, pour calculer la racine de x il suffit de considérer la fonction $f(t) = t^2 - x$.



Procédure d'approximation

1. Choisir une valeur initiale pour la suite, par exemple 1,
2. Affiner l'estimation tant qu'elle n'est pas assez précise,

   - L'estimation y est assez précise si $|y^2 - x| < \epsilon$
   - Pour affiner l'estimation y, calculer $\dfrac{y + \frac{x}{y}}{2}$

Éléments à programmer

`racine` : $\mathbb{R}^* \times \mathbb{R}^* \rightarrow \mathbb{R}^*$ $\quad |\texttt{racine}(x, \epsilon)^2 - x| < \epsilon$

`assez_precis` : $\mathbb{R}^* \times \mathbb{R}^* \times \mathbb{R}^* \rightarrow$ `boolen`

`assez_precis`$(x, \epsilon, y) = \mathit{vrai}$ ssi $|y^2 - x| < \epsilon$

`affine_estime` : $\mathbb{R}^* \times \mathbb{R}^* \rightarrow \mathbb{R}^*$

`affine_estime`(x, y) calcule $\frac{y + \frac{x}{y}}{2}$

`cherche_estime` : $\mathbb{R}^* \times \mathbb{R}^* \times \mathbb{R}^* \rightarrow \mathbb{R}^*$

`cherche_estime`(x, ϵ, e) cherche y t.q. $|y^2 - x| < \epsilon$

à partir d'une estimation e de y

Codage en CAML

```
#let carre_float(x) = x *. x;;
val carre_float : float -> float = <fun>
#let affine_estim(x,y) = (y +. (x /. y)) /. 2.;;
val affine_estim : float * float -> float = <fun>
#let assez_precis(x,epsi,y) =
     abs_float(x -. carre_float(y)) < epsi;;
val assez_precis : float * float * float -> bool = <fun>
#let rec cherche_estim(x,epsi,y) =
  if assez_precis(x,epsi,y) then y
  else cherche_estim(x,epsi,affine_estim(x,y));;
#let racine(x,epsi) = cherche_estim(x,epsi,1.);;
val racine : float * float -> float = <fun>
```


Test

```
#let epsi = 0.0000001;;
val epsi : float = 1e-07
# racine(4.,epsi);;
- : float = 2
# racine (2.,epsi);;
- : float = 1.41421356237
```


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 : la fonction `Sigma`

`Sigma` : $\mathbb{N} \rightarrow \mathbb{N}$ 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 (par exemple si $h(x)$ n'est pas dans le domaine de f).

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

Exemple : la fonction `Sigma`

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

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

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 $g(n, m) = n^2 \times m$.

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

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

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