

Mathématiques pour l'Informatique

Induction

David Teller

13/03/2007

Introduction

Récurrence

Induction structurelle

Définition

Propriétés

Applications

Fonctions

Conclusions

Récurrence simple

Commençons par quelques rappels.

Théorème (Récurrence)

Soit $\{\mathcal{H}_n/n \in \mathbf{N}\}$ un ensemble de propositions logiques tel que

- ▶ \mathcal{H}_0 soit vrai
- ▶ $\forall n \in \mathbf{N}, \mathcal{H}_n \Rightarrow \mathcal{H}_{n+1}$.

Alors pour tout n de \mathbf{N} , \mathcal{H}_n est vrai.

Points-clés :

initialisation

héritage – vérifier que l'héritage est bien prouvé pour tout n !

Récurrence forte

Théorème (Récurrence forte)

Soit $\{\mathcal{H}_n/n \in \mathbf{N}\}$ un ensemble de propositions logiques tel que

- ▶ \mathcal{H}_0 soit vrai
- ▶ $\forall n \in \mathbf{N}, (\forall k \leq n, \mathcal{H}_k) \Rightarrow \mathcal{H}_{n+1}$.

Alors pour tout n de \mathbf{N} , \mathcal{H}_n est vrai.

À vous...

Proposition

Dans tout groupe de n personnes, tous les gens ont le même âge.

Preuve Notons, pour tout n , P_n la proposition "Dans tout groupe de n personnes, tous les gens ont le même âge"

- ▶ P_1 Trivial.
- ▶ $P_n \stackrel{?}{\Rightarrow} P_{n+1}$ Soit n tel que P_n soit vrai. Numérotons les individus $1, 2, \dots, n$. Soient alors G le groupe composé des individus $1, 2, \dots, n-1$ et H le groupe composé des individus $2, \dots, n$. Par hypothèse de récurrence, comme G est un groupe de n individus, tous les individus de G ont le même âge. Par hypothèse de récurrence, comme H est un groupe de n individus, tous les individus de H ont le même âge. Comme la personne 2 est à la fois dans G et dans H , tous les individus de H et tous les individus de G ont le même âge que cette personne 2. Ce qui prouve le cas.

Cqfd.

Hein ?



Programme électoral

Jusqu'à présent, nous avons utilisé la récurrence pour raisonner sur les entiers, les listes, les arbres...

Les différentes techniques d'induction généralisent la récurrence pour traiter plus confortablement des ensembles moins linéaires que les listes.

Introduction

Récurrence

Induction structurelle

- Définition
- Propriétés
- Applications
- Fonctions

Conclusions



Définition

Définition (Définition par induction structurelle)

Un sous-ensemble X de E est défini par induction structurelle à partir de l'ensemble de base B et des opérateurs d'induction

$I = \{\Phi_1 : E^{a_1} \rightarrow E, \Phi_2 : E^{a_2} \rightarrow E, \dots, \Phi_n : E^{a_n} \rightarrow E\}$ si X est le plus petit ensemble contenant B et stable par tous les Φ_i .

Notation :

$$\begin{array}{l}
 X ::= B_1 \\
 \quad | B_2 \\
 \quad | \vdots \\
 \quad | B_n \\
 \quad | \Phi_1(X, X \dots) \\
 \quad | \Phi_2(X, X \dots) \\
 \quad | \Phi_3(X, X \dots) \\
 \quad | \vdots \\
 \quad | \vdots
 \end{array}$$

Q Ça vous rappelle quelque chose ?



Exemples

- ▶ en OCaml, les listes d'entiers sont définies à partir du constructeur de base [] et des constructeurs de structure $\{l \mapsto i::l / i \in \text{int}\}$, d'arité
- ▶ les entiers naturels sont définis à partir du constructeur de base 0 et du constructeur de structure s (ou "suivant")
- ▶ les arbres binaires OCaml sont définis à partir des constructeurs de base $\{\text{Feuille}(i) / i \in \text{int}\}$

Parenthèses

Proposition

Il existe une définition par induction structurelle de l'ensemble des expressions bien parenthésées, dans le monoïde libre des parenthèses.

Base ϵ (le mot vide)

Induction
$$E^2 \longrightarrow E \quad E \longrightarrow E$$

$$e, f \mapsto e \cdot f \quad e \mapsto (\cdot e \cdot)$$

Nombres

Proposition (Nombres pairs)

Il existe une définition par induction structurelle de l'ensemble des nombres pairs.

Base 0

Induction $n \mapsto n + 2$

L'ensemble des nombres pairs peut se définir dans \mathbf{N} par induction à partir de la base $\{0\}$ et des opérateurs d'induction $\{\text{plusdeux} : n \mapsto n + 2\}$.

Attention, l'ensemble des nombres pairs n'est pas le seul sous-ensemble de \mathbf{N} à contenir 0 et stable par *plusdeux* !

Plus généralement

Il arrive fréquemment qu'on construise un ensemble par induction à partir d'autres ensembles construits par induction, etc.

Avec quelques efforts, on peut construire par induction :

- ▶ Expressions arithmétiques, logiques...
- ▶ Langages de programmation.
- ▶ Le français parlé par une vache espagnole.

Dénombrements

Théorème

Si X est un ensemble défini par induction structurelle, tout élément e de X peut s'obtenir en appliquant un nombre fini d'opérateurs d'induction aux bases.

Preuve Définissons la suite $(X_i)_{i \in \mathbf{N}}$ de sous-ensembles de E par

- ▶ $X_0 = \mathcal{B}$;
- ▶ $\forall n \in \mathbf{N}$,
 $X_{n+1} = X_n \cup \{\Phi(x_1, \dots, x_k) / \Phi : E^k \longrightarrow E \in \mathcal{I}, x_1, \dots, x_k \in X_n\}$.

Schématiquement, on va poser $X_\omega = \bigcup_{n \in \mathbf{N}} X_n$ et chercher à prouver que

- ▶ $\forall n, X_n \subseteq X$, si bien que $X_\omega \subseteq X$;
- ▶ X_ω contient \mathcal{B} et est bien stable par les opérateurs d'induction donc contient bien X .



Sur les entiers

Proposition

Soit $\{P_n / n \in \mathbf{N}\}$ un ensemble de propositions tel que

- ▶ P_0 est vrai
- ▶ $\forall n \in \mathbf{N}, P_n \Rightarrow P_{n+1}$.

Alors pour tout n de \mathbf{N} , P_n est vrai.

Cette proposition ne devrait pas vous surprendre.



Raisonnement par induction structurelle

Théorème (Induction structurelle)

Soit X un ensemble défini inductivement. Soit $\{P(x) / x \in X\}$ un ensemble de propositions tel que

- ▶ $\forall x \in \mathcal{B}, P(x)$ est vrai
- ▶ $\forall \Phi : E^k \longrightarrow E \in \mathcal{I}, \forall x_1, \dots, x_k \in X$,
 $(\forall i, P(x_i)) \Rightarrow P(\Phi(x_1, \dots, x_k))$.

Alors pour tout x de X , $P(x)$ est vrai.

Ce raisonnement généralise la récurrence à des ensembles un peu plus complexes.



Sur les arbres

Soit A l'ensemble des arbres binaires, défini comme

$$\begin{array}{lcl}
 A & ::= & \text{Feuille}(0) \\
 & & | \text{Feuille}(1) \\
 & & \vdots \\
 & & \vdots \\
 & & | \text{Feuille}(n) \\
 & & \vdots \\
 & & \vdots \\
 & & | \text{Noeud}(A, A)
 \end{array}$$

Proposition

Soit $\{P(x) / x \in A\}$ un ensemble de propositions tel que

- ▶ $\forall n \in \mathbf{N}, P(\text{Feuille}(n))$ est vrai
- ▶ $\forall a, b \in A, P(a) \wedge P(b) \Rightarrow P(\text{Noeud}(a, b))$

Alors pour tout x de A , $P(x)$ est vrai.



Preuve

Théorème (Induction structurelle)

Soit X un ensemble défini inductivement. Soit $\{P(x)/x \in X\}$ un ensemble de propositions tel que

- ▶ $\forall x \in \mathcal{B}, P(x)$ est vrai
- ▶ $\forall \Phi : E^k \rightarrow E \in \mathcal{I}, \forall x_1, \dots, x_k \in X,$
 $(\forall i, P(x_i)) \Rightarrow P(\Phi(x_1, \dots, x_k))$.

Alors pour tout x de X , $P(x)$ est vrai.

Preuve Soit Y l'ensemble défini comme $\{x \in X/P(x)\}$. Alors Y contient \mathcal{B} et Y est stable par tous les opérateurs de \mathcal{I} . Donc Y contient X . Comme $Y \subseteq X$, nous avons $Y = X$. Nous pouvons nous servir de cela pour prouver le principe de récurrence, sans utiliser le fait que \mathbf{N} est bien ordonné !

Parenthésage

Considérons l'ensemble F des parenthèses bien formées, défini par

$$F ::= \epsilon$$

$$| F \cdot F$$

$$| (\cdot F \cdot)$$

Proposition

Tout élément de F contient autant de parenthèses gauches que de parenthèses droites.

Preuve par induction structurelle sur la structure de F .

Hypothèse Notre hypothèse d'induction $P(a)$ est "a contient autant de parenthèses gauches que de parenthèses droites".

Base Comme ϵ ne contient aucune parenthèse, $P(\epsilon)$ est vrai.

Induction 1 Soient f et g deux éléments de F contenant autant de parenthèses gauches que de parenthèses droites. Alors $f \cdot g$ contient autant de parenthèses gauches que de parenthèses droites, ce qui prouve le cas.

Induction 2 Soit f un élément de F contenant autant de parenthèses gauches que de parenthèses droites. Alors $(\cdot f \cdot)$ contient

Arbres

Soit A l'ensemble des arbres binaires, défini comme

$$A ::= \text{Feuille}(0)$$

$$| \text{Feuille}(1)$$

$$\vdots$$

$$\vdots$$

$$| \text{Feuille}(n)$$

$$\vdots$$

$$\vdots$$

$$| \text{Noeud}(A, A)$$

Soient h , n et f les fonctions de A vers \mathbf{N} donnant respectivement

- ▶ la hauteur de l'arbre
- ▶ le nombre de nœuds de l'arbre
- ▶ le nombre de feuilles de l'arbre.

Proposition

Pour tout arbre a de A , nous avons $n(a) \leq 2^{h(a)} - 1$.

Une preuve par induction

Proposition

Pour tout arbre a de A , nous avons $n(a) \leq 2^{h(a)} - 1$.

Preuve

Prouvons ceci par induction sur la structure de A . Considérons comme hypothèse $\mathcal{H}(a) = \{n(a) \leq 2^{h(a)} - 1\}$.

Bases Pour un arbre réduit à une feuille, nous avons $n(a) = 0$ et $h(a) = 0$. Comme $0 \leq 0$, le cas est prouvé.

Induction Considérons un arbre $c = \text{Feuille}(a, b)$ tel que $\mathcal{H}(a)$ et $\mathcal{H}(b)$ soient vraies. Alors nous avons $n(c) = n(a) + n(b) + 1$ et $h(c) = \max(h(a), h(b)) + 1$. Par conséquent, d'après $\mathcal{H}(a)$ et $\mathcal{H}(b)$, nous avons $n(c) \leq 2^{h(c)} - 1 + 2^{h(c)} - 1 + 1 = 2^{h(c)+1} - 1$, ce qui prouve le cas.

Par induction, nous venons donc de prouver que $\mathcal{H}(a)$ est vraie pour tout arbre a .

Fonction définie par induction

Il est possible de définir des fonctions *par induction*, c'est-à-dire à partir de la structure d'un ensemble.

Q Des exemples ?

À peu près la moitié des fonctions que nous définissons en OCaml.

Définition non-ambiguë

Définition (Définition non-ambiguë)

Un ensemble X est défini par induction *de manière non-ambiguë* si chaque élément de X n'admit qu'une seule écriture possible à partir des bases et des opérateurs d'induction qui ont servi à définir X .

Exemples

Si $d : \mathbf{N} \rightarrow \mathbf{N}$ est définie par $\forall n \in \mathbf{N}, d(n) = n + 2$ et $t : \mathbf{N} \rightarrow \mathbf{N}$ est définie par $\forall n \in \mathbf{N}, t(n) = n + 3$,

$$\begin{array}{l}
 X ::= 0 \\
 \quad | \\
 \quad | 1 \\
 \quad | 2 \\
 \quad | d(X)
 \end{array}$$

et

$$\begin{array}{l}
 Y ::= 0 \\
 \quad | d(Y) \\
 \quad | t(Y)
 \end{array}$$

En particulier

En OCaml, on emploie des ensembles de *termes*, c'est-à-dire des ensembles où les opérateurs d'induction sont juste des *symboles*. Ces ensembles sont toujours non-ambigus.

C'est-à-dire

Définition (Fonction définie par induction)

Considérons un ensemble X défini de manière non ambiguë à l'aide des bases $\mathcal{B} = \{b_1 \dots b_m\}$ et des opérateurs d'induction $\mathcal{I} = \{\Phi_1, \Phi_2 \dots \Phi_n\}$.

Une fonction f peut être *définie par induction structurelle* sur X s'il existe

- ▶ $c_1 \dots c_m$ tels que $\forall i, f(b_i) = c_i$
- ▶ $f_1 \dots f_m$ tels que $\forall j, \forall x_1 \dots x_k, f(\Phi_j(x_1, \dots, x_k)) = f_j = (x_1, \dots, x_k)$.

Exemple Considérons l'ensemble des listes

$$\begin{array}{l}
 L ::= Nil \\
 \quad | Cons(0, L) \\
 \quad | Cons(1, L) \\
 \quad | Cons(2, L) \\
 \quad \vdots \\
 \quad \vdots
 \end{array}$$

La longueur l d'une liste se définit par induction comme

- ▶ $l(Nil) = 0$
- ▶ $\forall h \in \mathbf{N}, \forall t \in L, l(Cons(h, t)) = l(Cons(t)) + 1$.

À vous

Q Et en OCaml, ça donne quoi ?

```
let rec longueur = fonction
| [] -> 0
| h::t -> (longueur t)+1
```

Q Comment définir la fonction *comparaison* : $\mathbf{P} \times \mathbf{B}$ (où \mathbf{P} est l'ensemble des entiers de Peano et \mathbf{B} l'ensemble des booléens) ?

```
let rec comparaison = fonction
| Zero.Zero -> true
| Zero._ -> false
| _, Zero -> false
| Succ m, Succ n -> comparaison (m, n)
```

Q Comment définir la fonction *addition* : $\mathbf{P} \times \mathbf{P}$?

```
let rec addition = fonction
| Zero.n -> n
| Succ(p).n -> addition (p, (Succ n))
```

Q Comment définir la multiplication sur les entiers de Peano ?

```
let rec multiplication = fonction
```

Attention

Proposition

La définition inductive suivante sur \mathbf{N}^2 ne produit pas une fonction :

- ▶ $f(0,0) = 1$
- ▶ $f(n+1, m) = f(n, m)^2$
- ▶ $f(n, m+1) = 3 \cdot f(n, m)$

Preuve Nous avons $f(1,1) = f(0,1)^2 = (3 \cdot f(0,0))^2 = 9$. Nous avons aussi $f(1,1) = 3 \cdot f(1,0) = 3 \cdot f(0,0)^2 = 3$!

Moralité Faites attention à la *confluence* de vos définitions.

Note La définition de \mathbf{N} utilisée est ambiguë.

Q Pourquoi ce genre de choses ne pose-t-il pas de problèmes en OCaml ?

```
let rec f = fonction
| 0, 0 -> 1
| n, m when n > 0 -> let x = f (n-1,m) in x**x
| n, m when m > 0 -> 3 * (f(n,m-1))
| _ -> assert (false) ;;
```

Des fonctions aux ensembles

Considérons la définition (incomplète) de *mod* sur $\mathbf{N} \times \mathbf{N}^*$ par

$$\text{mod}(n, m) = \begin{cases} n & \text{si } n < m \\ \text{mod}(n-m, m) & \end{cases}$$

Exercice

En vous inspirant, construisez une définition par induction non ambiguë de $\mathbf{N} \times \mathbf{N}^*$.

Ça vous rappelle quelque chose ?

Introduction

Récurrence

Induction structurelle

- Définition
- Propriétés
- Applications
- Fonctions

Conclusions

Ce que nous avons vu

Nous avons présenté :

- ▶ la récurrence
- ▶ la récurrence forte
- ▶ l'induction sur des ensembles bien ordonnés
- ▶ les ensembles construits par induction structurelle
- ▶ le raisonnement par induction structurelle
- ▶ les fonctions définies par induction structurelle
- ▶ et quelques liens entre tout ça et la programmation.

Applications

Les inductions s'utilisent très fréquemment en programmation :

- ▶ pour la définition d'ensembles en OCaml, en particulier les listes et les arbres de syntaxe
- ▶ pour la preuve d'algorithmes
- ▶ pour les calculs de complexité algorithmique.

Au prochain épisode

Dans le chapitre prochain, nous parlerons de logique.
Avec un peu de chance, il y aura encore de l'induction et des ensembles...

En attendant...

Y a-t-il des questions ?