Intermède :

Induction

Induction

Bibliographie

- A. Arnold et I. Guessarian, *Mathématiques pour l'informatique*

L'induction

Un outil formel élégant très utile en informatique

- elle permet de faire des définitions « récursives »
  - d'ensembles
  - de fonctions
- elle propose une technique de preuve souvent plus élégante que la récursion sur les entiers
  - mais la récursion sur les entiers est toujours possible

Définitions inductives

La définition inductive d'une partie X d'un ensemble consiste

- en la donnée explicite de certains éléments de X (*bases*),
- en la donnée de moyens de construire de nouveaux éléments de X à partir d'éléments déjà connus (*étapes inductives*).

Définition inductives

Définition

Soit E un ensemble. Une *définition inductive* d'une partie X de E consiste en la donnée

- d'un sous ensemble B de E,
- d'un ensemble K de fonctions (partielles) $\Phi : E^{a(\Phi)} \longrightarrow E$, où $a(\Phi) \in \mathbb{N}$ est *l'arité* de Φ.

X est défini comme étant **le plus petit** ensemble vérifiant les assertions (B) et (I) suivantes

(B) $B \subseteq X$

(I) $\forall \Phi \in K, \forall x_1, \dots, x_{a(\Phi)} \in X, \ \Phi(x_1, \dots, x_{a(\Phi)}) \in X.$

Définition inductives

L'ensemble ainsi défini est donc

$$X = \bigcap_{Y \in \mathcal{F}} Y$$

où $\mathcal{F} = \{Y \subseteq E \mid B \subseteq Y \text{ et } Y \text{ vérifie } (I)\}$.

- Cela justifie le terme « **le plus petit** ensemble ».

Notation : nous pourrons noter une définition inductive sous la forme

(B) $x \in X \quad (\forall x \in B)$

(I) $x_1, \ldots, x_{a(\Phi)} \in X \Longrightarrow \Phi(x_1, \ldots, x_{a(\Phi)}) \in X \quad (\forall \Phi \in K)$.

Exemples

L'ensemble $P \subseteq \mathbb{N}$ des entiers pairs

(B) $0 \in P$

(I) $n \in P \Longrightarrow n + 2 \in P$

L'ensemble $\mathrm{AB} \subseteq (A \cup \{\emptyset, (\,, \,)\,, \,;\,\})^*$ des arbres binaires sur un alphabet A

(B) $\emptyset \in \mathrm{AB}$

(I) $\forall a \in A,\ g, d \in \mathrm{AB} \Longrightarrow (a; g; d) \in \mathrm{AB}$

Définition explicite

Théorème

Si X est défini inductivement par les conditions (B) et (I), tout élément de X peut s'obtenir à partir de la base en appliquant un nombre fini d'étapes inductives.

$$X = \bigcup_{n \in \mathbb{N}} X_n$$

où

$$\begin{array}{rcl} X_0 & = & B \\ X_{n+1} & = & X_n \cup \{\Phi(x_1, \ldots, x_{a(\Phi)}) \mid x_1, \ldots, x_{a(\Phi)} \in X_n \text{ et } \Phi \in K\} \end{array}$$

Remarque : tout élément de X peut être représenté graphiquement par une structure arborescente.

Les types récursifs cachent une définition inductive

En Caml

```
type list =
 | Nil
 | Cons of int*list
```

En Coq

```
Inductive list : Set :=
 | Nil
 | Cons (n:nat) (l:list).
```

ou

```
Inductive list : Set :=
 | Nil: list
 | Cons: nat→list→list.
```

Preuve par induction

Théorème

Soit X un ensemble défini inductivement par les conditions (B) et (I), et soit $\mathcal{P}(x)$ un prédicat exprimant une propriété de l'élément x de X. Si les conditions suivantes sont vérifiées :

- *$\mathcal{P}(x)$ est vraie pour chaque $x \in B$,*
- *pour tout $x_1, \ldots, x_{a(\Phi)} \in X$, si $\mathcal{P}(x_1), \ldots, \mathcal{P}(x_{a(\Phi)})$ sont vraies alors $\mathcal{P}(\Phi(x_1, \ldots, x_{a(\Phi)}))$ est vraie, pour tout $\Phi \in K$,*

alors $\mathcal{P}(x)$ est vraie pour tout $x \in X$.

Coq : génération d'un principe d'induction

```
Check list_ind.
>list_ind : ∀ P : list → Prop,
       P Nil →
       (∀ (n : nat) (l : list), P l → P (Cons n l)) →
       ∀ l : list, P l
```

Exercice

On définit sur les arbres binaires AB

- le *nombre de feuilles* $f(x)$ d'un arbre x comme le nombre d'occurrences du symbole $\emptyset$,
- le *nombre de nœuds* $n(x)$ d'un arbre x comme le nombre d'occurrences des symboles de A.

Montrer que pour tout $x \in \mathrm{AB}$

$$n(x) \leqslant 2f(x) - 1$$

Définition non-ambiguë

Définition

La définition inductive d'un ensemble X par les conditions (B) et (I) est dite *non-ambiguë* si

- pour tout $x_1, \ldots, x_{a(\Phi)} \in X$ et $\Phi \in K$, $\Phi(x_1, \ldots, x_{a(\Phi)}) \notin B$,
- et pour tout $x_1, \ldots, x_{a(\Phi)}, x'_1, \ldots, x'_{a(\Phi')} \in X$ et $\Phi, \Phi' \in K$, $\Phi(x_1, \ldots, x_{a(\Phi)}) = \Phi'(x'_1, \ldots, x'_{a(\Phi')})$ implique $\Phi = \Phi'$ et $x_1 = x'_1, \ldots, x_{a(\Phi)} = x'_{a(\Phi)}$.

Exercice : donner des exemples de

- définition non-ambiguë
- définition ambiguë

Fonctions définies inductivement

Théorème

Soit $X \subseteq E$ un ensemble défini inductivement par les conditions (B) et (I) tel que X est non-ambiguë, soit F un ensemble quelconque, soit f_B une fonction de $B \longrightarrow F$ et une famille de fonctions $f_\Phi \in E^{a(\Phi)} \times F^{a(\Phi)} \longrightarrow F$, pour tout $\Phi \in K$.
Il existe une unique fonction $f \in X \longrightarrow F$ telle que

- *pour tout $x \in B$,*
$$f(x) = f_B(x)$$
- *pour tout $x_1, \ldots, x_{a(\Phi)} \in X$ et $\Phi \in K$,*
$$f(\Phi(x_1, \ldots, x_{a(\Phi)})) = f_\Phi(x_1, \ldots, x_{a(\Phi)}, f(x_1), \ldots, f(x_{a(\Phi)}))$$

Fonctions définies par induction Caml/Coq

En Caml

```
let rec length : list → int = function
 | Nil → 0
 | Cons (x,l) → 1 + length l
```

En Coq

```
Fixpoint length (l:list) : nat :=
 match l with
   | Nil ⇒ 0
   | Cons x q ⇒ 1 + length q
 end.
```