Ensembles inductifs & induction structurelle

Pierre Karpman

Lycée Champollion MP2I
https://membres-ljk.imag.fr/Pierre.Karpman/CPGE/2025/

Table des matières

Objectifs

On veut :

- Formaliser le principe de construction d'*ensembles inductifs*
- Introduire une méthode de preuve sur les objets de ces ensembles qui généralise les preuves par récurrence
- Déterminer dans quels cas il est possible de (naturellement) définir des fonctions, un ordre sur les éléments de ces ensembles

Table des matières

Exemples introductifs : entiers naturels & listes

Entiers « à la Peano »

On peut définir l'ensemble des entiers naturels $\mathbb{N}$ comme :

- $O \in \mathbb{N}$
- Si $n \in \mathbb{N}$, alors $Sn \in \mathbb{N}$

Alors par exemple,
$SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSO \in \mathbb{N}$

Listes

Soit un ensemble $\mathcal{V}$, on peut définir l'ensemble $L(\mathcal{V})$ des listes à éléments dans $\mathcal{V}$ comme :

- $N \in L(\mathcal{V})$
- Si $t \in L(\mathcal{V})$, $x \in \mathcal{V}$, alors $C(x, t) \in L(\mathcal{V})$

Alors par exemple pour $\mathcal{V} = \{a, b, c\}$, $C(a, N)$, $C(a, C(b, N))$, $C(b, C(c, N)) \in L(\mathcal{V})$

Ensemble défini inductivement

Soit E un ensemble ambiant (généralement implicite et pas très intéressant)

Définition inductive

Une *définition inductive sur E* est donnée par :

- Un sous-ensemble $B \subset E$ de *règles de bases* (ou *assertions*) (ou constructeurs de base)
- Un ensemble I de fonctions (éventuellement partielles) $\varphi : E^{a(\varphi)} \to E$ d'arité $a(\varphi)$: les *règles d'inférence* (ou constructeurs)

Ensemble défini inductivement

L'ensemble *défini inductivement* sur E par B & I est le **plus petit** sous-ensemble $X \subseteq E$ tel que :

- $B \subseteq X$
- $\forall \varphi \in I, \forall (x_1, \ldots, x_{a(\varphi)}) \in X^{a(\varphi)}$, si $\varphi(x_1, \ldots, x_{a(\varphi)})$ est définie et vaut x, alors $x \in X$

Autrement dit, c'est le plus petit sous-ensemble de E qui contient B et est stable par application des éléments de I

Existence

On montre que X défini inductivement sur E par B & I existe bien (non constructivement :(

- On considère $A \in 2^E$ l'ensemble des parties de E qui contiennent B et sont stables par application de I
- $A \ni E$ qui est donc non vide
- On a $X = \bigcap_{Y \in A} Y$

Exemple : mots bien parenthésés

Mots bien parenthésés

L'ensemble des *mots bien parenthésés* sur l'alphabet $\Sigma := \{[,]\}$ est défini inductivement sur Σ^* par :

- $B = \{\varepsilon\}$
- $I = \{\varphi : x \mapsto [\cdot x \cdot], \quad \psi : x, y \mapsto x \cdot y\}$

Autrement dit le mot vide est bien parenthésé, et un mot est bien parenthésé ss.'il peut s'obtenir en entourant un mot bien parenthésé par [et] ou en concaténant deux mots bien parenthésés (rem. en passant : la concaténation $\cdot$ est associative)

- Un sens évident par définition de I, l'autre par le fait que l'on considère le **plus petit sous-ensemble** de Σ^* etc.

Bis

On prouvera (en « exercice ») que prendre $I = \{\varphi' : x, y \mapsto [\cdot x \cdot] \cdot y\}$ définit le même ensemble

Exemple : arbres

Toute définition inductive est moralement un cas particulier d'une définition d'arbre

Arbres binaires stricts

L'ensemble des arbres binaires stricts (non étiquetés) est défini inductivement (sur un ensemble E implicite) par :

- $B = \{\mathrm{F}\}$
- $I = \{x, y \mapsto \mathrm{N}(x, y)\}$

Arbres quelconques

L'ensemble des arbres ∞-aires (non étiquetés) est (informellement) défini inductivement (sur un ensemble E implicite) par :

- $B = \{\mathrm{F}\}$
- $I = \{x \mapsto \mathrm{N1}(x),\ (x, y) \mapsto \mathrm{N2}(x, y),\ (x, y, z) \mapsto \mathrm{N3}(x, y, z), \dots\}$

Informatisation

Approche syntaxique : type récursif

Une définition inductive s'implémente (par ex. en OCaml) souvent **syntaxiquement** par un type récursif t :

- Les éléments de B sont définis comme des constructeurs zéroaires
- Les éléments de I sont définis comme des constructeurs (1+)-aires

On confond alors X et l'ensemble $\mathcal{T}$ des habitants du type t

- C'est le type (la **syntaxe**) qui donne l'appartenance à l'ensemble inductif

Cette approche marche souvent **mais pas toujours**, par exemple :

- Elle force les règles d'inférence à être totales
- Pas/mal adapté à B, I infinis

Exercice

Donnez un exemple de définition inductive vue au cours de l'année qui ne peut pas s'implémenter syntaxiquement par un type récursif OCaml usuel (dans le sens où l'on a une définition « naturelle » de t mais $X \subset \mathcal{T}$)

Informatisation *bis* : quand la syntaxe (de base) ne suffit pas

Prédicats

On peut représenter X par un type inductif t t.q. $X \subseteq \mathcal{T}$ et un prédicat (une fonction booléenne) $\varphi : t \to \{\top, \bot\}$ telle que $\varphi(x) = \top \Leftrightarrow x \in X$

Constructeurs intelligents

On peut implémenter les règles d'inférence par des fonctions (plus forcément totales), qui n'utiliseront les constructeurs du type que quand ils construisent un élément de X

Meilleure syntaxe 🙈

On peut enrichir le système de type, par exemple avec des *types algébriques généralisés* (GADTs), pour être plus précis dans la création de $\mathcal{T}$

(Si vous n'avez pas encore de sujet de TIPE…)

Existence *bis*

Observations

- La preuve d'existence de X donnée ci-dessus est non-constructive
  - Elle se base aussi sur un E explicite
- L'informatisation des définitions décrite ci-dessus est (forcément…) constructive, et ne définit pas E

Construction de l'ensemble inductif

Soit B, I une définition inductive, on définit :

- $I(S)$ pour un ensemble S comme $\{\varphi(s) \mid \varphi \in I, s \in S^{a(\varphi)}\}$
- Et la suite d'ensembles (Y_n) récursivement nothing surprising here par :

$$Y_0 = B,\ Y_{n+1} = Y_n \cup I(Y_n)$$

On peut alors définir $X = \bigcup_{n\in\mathbb{N}} Y_n$

Hauteur

La *hauteur* de $x \in X$ est définie par $\min_n x \in Y_n$ pour la suite ci-dessus

Table des matières

Rappel : récurrence

Principe de récurrence

La définition inductive des entiers « à la Peano » est associée à un principe de récurrence : si un prédicat $\mathcal{P}$ est tel que :

- $\mathcal{P}(O)$ (est vrai)
- $\forall n \in \mathbb{N}.\ \mathcal{P}(n) \Rightarrow \mathcal{P}(Sn)$

alors $\mathcal{P}$ est vrai pour tout $n \in \mathbb{N}$

Fonctions récursives

On peut définir une fonction F récursivement sur tous les entiers en définissant :

- $F(O)$
- $\forall n \in \mathbb{N}.\ F(Sn)$ en fonction de n et $F(n)$

Induction structurelle

En programmation, on utilise des fonctions récursives pas uniquement sur des entiers naturels (mais aussi sur des listes, des arbres…). Peut on aussi généraliser le principe de preuve par récurrence ?

Principe d'induction structurelle

Soit X un ensemble défini inductivement par B & I, $\mathcal{P}$ un prédicat sur X. Si

- $\forall x \in B.\ \mathcal{P}(x)$
- $\forall \varphi \in I.\ \forall x_1, \ldots, x_{a(\varphi)} \in X.\ \bigwedge_{i=1}^{a(\varphi)} \mathcal{P}(x_i) \Rightarrow \mathcal{P}(\varphi(x_1, \ldots, x_{a(\varphi)}))$

alors $\mathcal{P}$ est vrai pour tout $x \in X$

Remarque

On peut faire un parallèle entre le principe d'induction structurelle (en tant que méthode générale de preuve) et le *fold* (à droite) sur un type récursif (en tant que méthode générale de construction de fonctions récursives), *cf.* plus bas

Induction structurelle *bis*

Preuve du principe (esquisse)

Par récurrence forte sur la hauteur de x

- Cas de base okay par hypothèse sur $\mathcal{P}$
- Cas récursif : supp. $\mathcal{P}(x')$ pour tout x' de hauteur $\leq n$, alors tout x de hauteur $n+1$ est égal à $\varphi(x_1, \ldots, x_{a(\varphi)})$ pour un certain φ et certains x_i de hauteur $\leq n$, donc $\mathcal{P}(x_1), \ldots$ par hypothèse de récurrence forte, et $\mathcal{P}(x)$ par hypothèse sur $\mathcal{P}$

Remarques

- On peut en principe toujours remplacer une preuve par induction structurelle par une preuve par récurrence forte sur la hauteur (explicite)
- Une version par induction structurelle peut être plus claire (ou pas)
- On peut rédiger les cas comme des motifs de filtrage (ce qui peut aider à s'assurer que l'on n'en oublie pas !)
- Comme pour une preuve par récurrence, il faut toujours **préciser ce sur quoi on effectue l'induction**

Exemple : longueur de $x@y$

```
let rec len = function [] -> 0 | _::xs -> 1 + (len xs)
let rec (@) x y = match x with [] -> y | x'::xs -> x'::(xs @ y)
len (x @ y) = (len x) + (len y)
```

Prouvons $\mathcal{P}(x) : \forall y.$ `len (x @ y) = (len x) + (len y)` par induction structurelle sur x :

- Cas de base : `x = []` :
  - Par calcul : `x @ y = y`; `len (x @ y) = len (y)`; `len x = 0`
- Cas inductif : `x = x'::xs` :
  - Par calcul, `x @ y = x'::(xs @ y)` ;
  - et `len (x @ y) = 1 + (len (xs @ y))`
  - et `len x = 1 + len xs`.
  - Par hypothèse d'induction la propriété est vraie pour `xs`, `y` :
    `len (xs @ y) = (len xs) + (len y)`
  - Donc `len (x @ y) = 1 + (len xs) + (len y)`
  - Ce qui est égal à `(len x) + (len y)`

Par le principe d'induction structurelle, $\mathcal{P}(x)$ est vraie pour tout x, et donc `len (x @ y) = (len x) + (len y)` est vrai pour tout x, pour tout y

Exercice : mots bien parenthésés

Soit X défini inductivement par :

- $B = \{\varepsilon\}$
- $I = \{\varphi : x \mapsto [\cdot x \cdot], \quad \psi : x, y \mapsto x \cdot y\}$

X' défini inductivement par :

- $B' = \{\varepsilon\}$
- $I' = \{\varphi' : x, y \mapsto [\cdot x \cdot] \cdot y\}$

Utilisez le principe d'induction structurelle pour montrer que $X = X'$

Exercice : proposition de correction

$X' \subseteq X$

On montre $x \in X' \Rightarrow x \in X$ par induction structurelle sur x :

- Cas de base : $x = \varepsilon$: immédiat par $\varepsilon \in B$
- Cas inductif : $x = \varphi'(x', y')$
  - Par hypothèse d'induction, $x', y' \in X$
  - Par définition de φ', $x = [\cdot x' \cdot] \cdot y'$
  - Par définition de φ, ψ, $x = \psi(\varphi(x'), y') \in X$

Et la propriété est vraie par le principe d'induction structurelle

$X \subseteq X'$

On montre $x \in X \Rightarrow x \in X'$ par induction structurelle sur x (IH1) :

- Cas de base : $x = \varepsilon$: immédiat par $\varepsilon \in B$
- Premier cas inductif : $x = \varphi(x')$
  - Par définition de φ, $x = [\cdot x' \cdot]$
  - Par hypothèse d'induction, $x' \in X'$
  - Par définition de φ', $x = \varphi'(x', \varepsilon) \in X'$
- Second cas inductif : $x = \psi(x', y')$: ???

Exercice : proposition de correction *bis*

$x = \psi(x', y')$

Par IH1, $x', y' \in X'$, et l'on doit montrer $\psi(x', y') \in X'$

On raisonne alors par induction structurelle sur x' (IH2) **en tant qu'élément de** X' (on fait un `match` imbriqué) pour prouver que pour tout $z \in X'$, $\psi(z, y') \in X'$

- Cas de base : $z = \varepsilon$: par substitution et neutralité de ε, $\psi(\varepsilon, y') = y' \in X'$
- Cas inductif : $z = \varphi'(x'', y'')$
  - avec $x'', y'' \in X'$
  - Par IH2, $\psi(x'', y') \in X'$, $\psi(y'', y') \in X'$
  - Par définition de φ' : $z = [\cdot x'' \cdot] \cdot y''$
  - Par définition de ψ et substitution : $\psi(z, y') = [\cdot x'' \cdot] \cdot y'' \cdot y' = \varphi'(x'', y'' \cdot y')$ par associativité de $\cdot$
  - Or $y'' \cdot y' = \psi(y'', y') \in X'$ (par IH2)
  - Donc $\varphi'(x'', y'' \cdot y') = \psi(z, y') \in X'$ (par définition de X')

Et la propriété est vraie par le principe d'induction structurelle

Exercice : proposition de correction *ter*

Formalisée comme précédemment, la preuve par induction structurelle précédente est assez longue. On pourrait être tenté de la résumer par exemple par :

$X' \subseteq X$

- $\varphi'(x, y) = \psi(\varphi(x), y)$

$X \subseteq X'$

- $\varphi(x) = \varphi'(x, \varepsilon)$
- $\psi(\varepsilon, y) = y$
- $\psi(\psi(\varphi(x), y), z) = \varphi'(x, y \cdot z)$

Mais toute la difficulté est alors de s'assurer que l'on a bien traité tous les cas…

Remarque

Il peut arriver que l'on rédige une preuve par induction structurelle sans en fait utiliser l'hypothèse d'induction : l'induction structurelle sert alors uniquement à « déconstruire » la définition. De la même façon qu'une fonction sur un type récursif n'est pas nécessairement elle-même récursive…

On pourrait tout à fait formaliser à part entière une telle règle de raisonnement par déconstruction

Table des matières

Ambiguïté

Il est naturel de vouloir définir des fonctions récursives sur un ensemble défini inductivement. Il faut cependant (parfois) faire attention au fait qu'elles ne sont pas forcément bien définies

Ambiguïté

Une définition inductive par B & I est *non ambiguë* ssi. :

- $\forall \varphi \in I. \forall x_1, \ldots, x_{a(\varphi)} \in X. \varphi(x_1, \ldots, x_{a(\varphi)}) \notin B$
- $\forall \varphi, \psi \in I. \forall x_1, \ldots, x_{a(\varphi)}, y_1, \ldots, y_{a(\psi)} \in X.$

$$\varphi(x_1, \ldots, x_{a(\varphi)}) = \psi(y_1, \ldots, y_{a(\psi)}) \Rightarrow \left(\varphi = \psi \wedge \bigwedge_{i=1}^{a(\varphi)} x_i = y_i \right)$$

et *ambiguë* sinon

Exemples

Expressions arithmétiques

L'ensemble des expressions arithmétiques définies inductivement sur $\{0, 1, 2, 3, 4, +, \times\}^*$ par :

- $B = \{0, 1, 2, 3, 4\}$
- $I = \{\varphi : x, y \mapsto x + y,\ \psi : x, y \mapsto x \times y\}$

est ambiguë : on peut construire $1 + 2 \times 3$ comme $\varphi(1, \psi(2, 3))$ ou $\psi(\varphi(1, 2), 3)$

$1 + 2 \times 3$ a plusieurs *arbres de dérivation* possibles

Types récursifs usuels

Les ensembles définis inductivement par les types récursifs usuels OCaml sont non-ambigus par construction : chaque élément est littéralement défini par une suite ordonnée d'application de constructeurs

Fonctions sur un ensemble défini inductivement

Définition récursive

On peut définir une fonction F sur un ensemble X défini inductivement par une définition non-ambiguë B & I en définissant :

- $F(x)$ pour tout $x \in B$
- $F(\varphi(x_1, \ldots, x_{a(\varphi)}))$ comme $f_\varphi(x_1, \ldots, x_{a(\varphi)}, F(x_1), \ldots, F(x_{a(\varphi)}))$ avec une certaine fonction f_φ pour tout $\varphi \in I$

Autrement dit, on définit F explicitement sur les cas de base de la définition, et récursivement sur les cas « inductifs »

La fonction est bien définie sur tout élément de X, puisque tout $x \in X$ s'écrit **d'une unique façon** comme une suite **finie** (de longueur égale à sa hauteur) d'application des constructeurs

Fold

Pour un ensemble inductif défini par un type récursif fonctionnel usuel, la « méta » fonction qui définit F en fonction de ses cas de base, f_φ, etc. est le *fold* (à droite) du type

Exercice

Définissez la fonction factorielle sur l'ensemble des entiers naturels définis inductivement « à la Peano » (par exemple informatisés avec le `type nat = O | S of nat`)

Exercice : proposition de correction

C'est plus simple avec quelques fonctions intermédiaires…

```
type nat = O | S of nat

let rec add x y = match x with
    | O -> y
    | S x' -> S (add x' y)

let rec mul x y = match x with
    | O -> O
    | S O -> y
    | S x' -> add y (mul x' y)

let rec fact x = match x with
    | O -> S O
    | S x' -> mul x (fact x')
```

Exercice : autre proposition de correction

C'est bien plus simple en abstrayant…

```
let rec fold fo fsx = function
    | O -> fo
    | S x' as x -> fsx x (fold fo fsx x')

let rec fold' fo fso fsx = function
    | O -> fo
    | S O -> fso
    | S x' as x -> fsx x (fold' fo fso fsx x')

let add' x y = fold y (fun _ x' -> S x') x

let mul' x y = fold' O y (fun _ x' -> add' y x') x

let fact' x = fold (S O) (fun x x' -> mul' x x') x
```

Table des matières

Ordre induit par une définition inductive

Intuitivement, on veut dire que pour un ensemble défini inductivement de façon non-ambiguë, si $x = \varphi(x')$ alors x' est plus petit que x, etc.

Ordre induit

Soit B, I une définition inductive **non-ambiguë** d'un ensemble X, on définit la relation d'ordre partiel $<_1 X$ par :

$$\forall \varphi \in I.\ \forall x_1, \ldots, x_{a(\varphi)}.\ \forall i \in [\![a(\varphi)]\!].\ x_i <_1 \varphi(x_1, \ldots, x_{a(\varphi)})$$

et l'*ordre induit* par B, I par la fermeture réflexive-transitive de $<_1$

Remarques

- Si l'on note $\preceq$ l'ordre induit et $|\cdot|$ la hauteur, alors $x \preceq y \Rightarrow |x| \leq |y|$, mais le sens inverse ne tient pas forcément :
- L'ordre induit est généralement un ordre partiel

L'ordre induit est bien fondé

Ordre bien fondé

Une relation d'ordre $\leq$ sur un ensemble X est dite *bien fondée* ss.'il n'existe pas de suite infinie (x_n) d'éléments distincts de X satisfaisant $x_n > x_{n+1}$ pour tout n pour la relation d'ordre stricte $>$ associée (X ne possède pas de suite infinie strictement décroissante)

L'ordre induit par une définition inductive est bien fondé

Assez immédiat si l'on a prouvé la remarque précédente liant $\preceq$ et $\leq$ sur la hauteur

Vocabulaire

- Les éléments de B sont des *éléments minimaux* pour $\preceq$
- Si $x \prec y$, on dit que x est un *prédécesseur* de y, et y un *successeur* de x