Principe d'induction [MPLI]

I. Récurrence

A. Raisonnements par récurrences sur $\mathbb{N}$

Théorème 1 [MPLI Th 3.1] **Premier principe d'induction** Soit $P(n)$ un prédicat dépendant de l'entier n. Si les deux conditions suivantes sont vérifiées :

$$\begin{cases} P(0) \text{ est vrai} \\ \forall n \in \mathbb{N}(P(n) \Rightarrow P(n+1)) \end{cases} \Longrightarrow \forall n \in \mathbb{N}, P(n) \text{ est vrai.}$$

Remarque 2 On appelle la vérification de $P(0)$ l'*Initialisation*, et la vérification que $P(n) \Rightarrow P(n+1)$ l'*Hérédité*.

Remarque 3 On peut généraliser le théorème précédent avec un entier de départ n_0 quelconque, à la place de 0 :

$$\begin{cases} P(n_0) \text{ est vrai} \\ \forall n \geq n_0(P(n) \Rightarrow P(n+1)) \end{cases} \Longrightarrow \forall n \geq n_0, P(n) \text{ est vraie}$$

Exemple 4 [MPLI Ex 3.3] On peut montrer par récurrence $\Sigma_{i=1}^{n} i = \frac{n(n+1)}{2}$

Théorème 5 [MPLI Th 3.4] **Deuxième principe d'induction (Récurrence forte)** Soit $P(n)$ un prédicat dépendant de n, Si la proposition suivante est vérifiée :

$$\begin{cases} P(0) \text{ est vrai} \\ \forall n \in \mathbb{N}((\forall k < n, P(k)) \Rightarrow P(n)) \end{cases} \Longrightarrow \forall n \in \mathbb{N}, P(n) \text{ est vraie.}$$

Remarque 6 [MPLI Rem 3.5.1] Le premier point est redondant avec le second. En effet, pour $n = 0$, on a $\forall k < 0 P(k) \Rightarrow P(0)$, or comme il n'existe pas d'entiers strictement plus petit que 0, il faut prouver $P(0)$.

Remarque 7 [MPLI Rem 3.5.2] De même que pour le premier principe d'induction, on peut commencer à partir d'un entier n_0 quelconque.

Remarque 8 [MPLI Rem 3.5.3] Sur $\mathbb{N}$, les deux principes d'induction sont équivalent.

Remarque 9 **Invariants de boucle** Étant donnés un algorithme et une boucle de cet algorithme, un **invariant de boucle** est une propriété qui, quelles que soient les entrées valides fournies à l'algorithme :

- est valide avant le premier tour de boucle
- est préservée par chaque tour de boucle

Alors, l'invariant est vérifié à la fin de l'exécution de la boucle.

DEV

Exemple 10 [TOR] Il est possible de prouver la correction de l'algorithme de Dijktsra en utilisant une preuve par récurrence d'un invariant de boucle.

B. Relation d'ordre [MPLI 2]

Définition 11 [MPLI Def 2.1] **Relation d'ordre** Une relation d'ordre $\mathcal{R}$ sur un ensemble E est une relation binaire :

- réflexive : $\forall x \in E, x\mathcal{R}x$
- antisimétrique : $\forall x, y \in E, x\mathcal{R}y \wedge y\mathcal{R}x \Rightarrow x = y$
- transitive : $\forall x, y, z \in E, x\mathcal{R}y \wedge y\mathcal{R}z \Rightarrow x\mathcal{R}z$

On dit que la relation est totale si : $\forall x \neq y, x\mathcal{R}y \vee y\mathcal{R}x$

Définition 12 [MPLI Def 2.11] **Ensemble ordonné** Un ensemble ordonné $(E, \leq)$ est un ensemble muni d'une relation d'ordre $\leq$.

Exemple 13 [MPLI Ex 2.12] Les entiers naturels $\mathbb{N}$ peuvent être muni de l'ordre naturel $\leq$ mais aussi de l'ordre $|$ de divisibilité.

Définition 14 [MPLI 2.2.3] **Produit d'ensemble ordonnés** Soient $(E_1, \leq_1)$ et $(E_2, \leq_2)$ deux ensembles ordonnés. Le produit direct de ces deux ensembles est $(E_1 \times E_2, \leq_{\text{dir}})$ avec la relation $\leq$ définie par $(x_1, x_2) \leq_{\text{dir}} (y_1, y_2) \Leftrightarrow x_1 \leq_1 y_1$ et $x_2 \leq_2 y_2$.
Il est aussi possible de munir $E \times F$ de l'ordre lexicographique $\leq_{\text{lex}}$ définie par $(x_1, x_2) \leq_{\text{lex}} (y_1, y_2) \Leftrightarrow x_1 \leq_1 y_1$ ou $(x_1 = x_2$ et $x_2 \leq_2 y_2)$.

Définition 15 [MPLI 2.2.3] **Ordre léxicographique général** On peut généraliser l'ordre léxicographique à des séquences de taille arbitraires. Si $(\Sigma, \leq)$ est un alphabet ordonné, on définit l'ordre $\leq_{\text{lex}}$ par $u \leq_{\text{lex}} v$ si et seulement si :

- soit u est un préfixe de v
- soit $\exists\, k, u_1 ... u_k = v_1 ... v_k$ et $u_{k+1} \leq v_{k+1}$

Définition 16 [MPLI Prop 2.21] Un élément x est dit **minimal** si

$$\forall y \in E, y \leq x \Rightarrow y = x$$

[TOR]

Définition 17 **Ensemble bien fondé** Une relation d'ordre $\leq$ sur un ensemble E est bien fondée s'il n'y a pas de suite infinie strictement décroissante d'élément de E.

Exemple 18 $\mathbb{N}$ munit de l'ordre naturel est un ensemble bien fondé. $\mathbb{Z}$ ne l'est pas.

Exemple 19 Avec un alphabet Σ contenant au moins deux lettres $a < b$, l'ordre lexicographique sur Σ^* n'est pas bien fondé. La suite $(a^n b)_{\mathbb{N}}$ est infinie décroissante.

Proposition 20 Un ensemble ordonné $(E, \leq)$ est bien fondé si et seulement si toute partie non vide de E admet au moins un élément minimal.

Proposition 21 Si $(E, \leq_E)$ et $(F, \leq_F)$ sont des ensembles bien fondé alors l'ordre lexicographique $\leq_{\text{lex}}$ est bien fondé sur $E \times F$.

C. Induction bien fondée

Théorème 22 Soit $(E, \leq)$ un ensemble bien fondé et P une proposition dépendant d'un élément x de E. Si on a :

$$\forall x \in E, ((\forall y < x, P(y)) \Rightarrow P(x))$$

Remarque 23 Ce théorème appliqué à $(\mathbb{N}, \leq)$ donne le principe de récurrence forte.

Proposition 24 Si E et F sont des ensembles bien fondés, on déduit de la proposition 19 et du théorème 22 un principe d'induction sur $E \times F$.

II. Définition Inductive

Idée 25 **Constructeurs** [TOR 6.4.3] Pour définir formellement un ensemble d'objets inductifs, on décrit chaque objets de base et chaque manière de construire un nouvel objet à partir d'objets plus petits. Ce sont des *constructeurs*. Tout objet est alors construit par une combinaison d'applications explicites de ces *constructeurs*.

A. Définition Inductive [MPLI 3.2]

Définiton 26 **Ensemble définis inductivement** Soit E un ensemble. une définition inductive d'une partie X de E consiste en la donnée d'un ensemble K d'opérations $\Phi : E^{a(\Phi)} \to E$, où $a(\Phi) \in \mathbb{N}$ est l'arité de Φ.
X est alors défini comme étant le plus petit ensemble vérifiant :

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

Remarque 27 [TOR] Cette définition ne sépare pas explicitement les cas de base des cas de combinaison, qui sont tous de la même façon associés à des constructeurs. Ces deux cas se distinguent par l'arité des constructeurs associés.

Exemple 28 **Arbres Binaires** On peut définir inductivement l'ensemble AB des arbres binaires étiquetés, comme sous ensemble de l'ensemble des graphes enracinés :

- $E \in \mathrm{AB}$
- $\forall g, d \in \mathrm{AB}, r \in \mathbb{R}, N(g, r, d) \in \mathrm{AB}$

Implémentation 29 [TOR Code 7.22]
On peut facilement implémenter une définition inductive en langage OCaml, en construisant un type :

```
type 'a tree =
  | E
  | N of 'a tree * 'a
       * 'a tree
```

Exemple 30 **Définition inductive de $\mathbb{N}$** On peut définir inductivement l'ensemble des entiers positifs :

- Un cas de base, l'entier zero, noté Z
- Une règle de construction : si $n \in \mathbb{N}$, alors son successeur, noté $S(n)$, est encore dans $\mathbb{N}$.

Implémentation 31
En OCaml:

```
type int =
  | Z
  | S of
int
```

Théorème 32 [MPLI Thm 3.10] Si X admet une définition inductive, alors tout élément de X peut s'obtenir en appliquant un nombre fini d'étapes inductives.

B. Preuve par induction structurelle

Proposition 33 [MPLI Prop 3.11] **Preuve par induction** Soit X un ensemble défini inductivement, et soit $P(x)$ un prédicat exprimant une propriété de l'élément x de X. Si la condition

$$\left(P(x_1), ..., P\left(i_{a(\Phi)}\right)\right) \Rightarrow P\left(\Phi\left(x_1, ..., x_{a(\Phi)}\right)\right)$$

pour chaque $\Phi \in K$, alors $P(x)$ est vraie pour tout $x \in X$

Remarque 34 [MPLI Rem 3.12] Si l'on considère la définition inductive de $\mathbb{N}$, le premier principe d'induction sur les entiers correspond à la définition ci dessus. Toutes les preuves par récurrences sont donc des exemples de preuves par induction.

Exemple 35 [MPLI Ex 3.16] Soit h, n les fonctions donnant respectivement la hauteur et le nombre de nœud d'un arbre, on peut montrer par induction que $\forall x \in \mathrm{AB}, n(x) \leq 2^{h(x)-1}$

Exemple 36 **Transformation de Tseitsin** La transformation de Tseitsin utilise une preuve par induction structurelle sur une formule de la logique propositionnelle.

III. Fonctions définies par induction

A. Définition inductive non-ambiguë

Définition 37 [MPLI Def 3.18] Une définition inductive d'un ensemble X est dite non ambiguë si pour tout $x \in X$, il existe des unique $\Phi \in K$ et $x_1, ..., x_{a(\Phi)} \in X$ tels que $x = \Phi\left(x_1, ..., x_{a(\Phi)}\right)$.

Remarque 38 Plus intuitivement, cela signifie qu'il n'existe qu'une seule façon de construire un élément x de X.

Exemple 39 [MPLI Ex 3.19]

Définition ambiguë de $\mathbb{N}^2$

- $(0,0) \in \mathbb{N}^2$
- $(n,m) \in \mathbb{N}^2 \Rightarrow (n+1,m) \in \mathbb{N}^2$
- $(n,m) \in \mathbb{N}^2 \Rightarrow (n,m+1) \in \mathbb{N}^2$

Définition non ambiguë de $\mathbb{N}^2$:

- $(0,n), (n,0)$ pour $n \in \mathbb{N}$
- $(n,m) \in \mathbb{N}^2 \Rightarrow (n+1,m+1) \in \mathbb{N}^2$

B. Fonctions définies par induction

Définition 40 [MPLI Def 3.20] Soit $X \subseteq E$ un ensemble défini inductivement de façon non ambiguë, et soit F un ensemble quelconque. La définition inductive d'une application ψ de X dans F consiste en

- L'expression de $\varphi\left(\Phi\left(x_1, ..., x_{a(\Phi)}\right)\right)$ à partir des $x_1, ..., x_{a(\Phi)}$ et des $\Phi(x_1), ..., \Phi\left(x_{a(\Phi)}\right)$ pour chaque $\Phi \in K$. On écrira
$$\psi\left(\Phi\left(x_1, ..., x_{a(\Phi)}\right)\right) = \psi_\Phi\left(\psi(x_1), ..., \psi\left(x_{a(\Phi)}\right)\right)$$

où ψ_Φ est une application de $E^{a(\Phi)} \times F^{a(\Phi)}$ dans F.

Remarque 41 La non ambiguïté de E assure la bonne définition de ψ.

Exemple 42 [MPLI Ex 3.24] La fonction taille d'un arbre binaire sur AB se définit inductivement par :

$$\begin{cases} t(E) = 0 \\ \forall g, d \in \mathrm{AB}, r \in \mathbb{R}, t(g,r,d) = 1 + t(g) + t(d) \end{cases}$$

Implémentation 43 [TOR Programme 7.22] Cette définition inductive se traduit très bien en langage OCaml, à l'aide d'un `match with` sur le type défini inductivement :

```
let rec size (t: 'a tree) : int = match t with
  | E -> 0
  | N(g,_,d) -> 1 + size g + size d
```

DEV

Exemple 44 **fonctions Premiers, Suivants et Annulables sur les expressions régulières.** [CAR] [DRAG] [TIGER] On peut définir inductivement l'ensemble des expression régulière sur un alphabet.

À partir de cette définition, on peut alors construire par induction les fonctions Premiers, Suivants et Annulables.

Remarque 45 [MPLI] La non ambiguïté de la définition de AB justifie le filtrage par motif sur ses éléments.

Principe d'induction [MPLI]

I. Récurrence

A. Raisonnements par récurrences sur ℕ

1 **Thm** [MPLI Th 3.1] Premier principe d'induction

2 **Rem**

3 **Rem**

4 **Ex** [MPLI Ex 3.3]

5 **Thm** [MPLI Th 3.4] Deuxième principe d'induction (Récurrence forte)

6 **Rem** [MPLI Rem 3.5.1]

7 **Rem** [MPLI Rem 3.5.2]

8 **Rem** [MPLI Rem 3.5.3]

9 **Rem** Invariants de boucle

10 **Ex** [TOR]

B. Relation d'ordre [MPLI 2]

11 **Def** [MPLI Def 2.1] Relation d'ordre

12 **Def** [MPLI Def 2.11] Ensemble ordonné

13 **Ex** [MPLI Ex 2.12]

14 **Def** [MPLI 2.2.3]Produit d'ensemble ordonnés

15 **Def** [MPLI 2.2.3]Ordre léxicographique général

16 **Def** [MPLI Prop 2.21]

17 **Def** Ensemble bien fondé

18 **Ex**

19 **Ex**

20 **Prop**

21 **Prop**

C. Induction bien fondée

22 **Thm**

23 **Rem**

24 **Prop**

II. Définition Inductive

25 **Idée** Constructeurs [TOR 6.4.3]

A. Définition Inductive[MPLI 3.2]

26 **Définiton** Ensemble définis inductivement

27 **Rem** [TOR]

28 **Ex** Arbres Binaires

29 **Implem** [TOR Code 7.22]

30 **Ex** Définition inductive de ℕ

31 **Implem**

32 **Thm** [MPLI Thm 3.10]

B. Preuve par induction structurelle

33 **Prop** [MPLI Prop 3.11] Preuve par induction

34 **Rem** [MPLI Rem 3.12]

35 **Ex** [MPLI Ex 3.16]

36 **Ex** Transformation de Tseitsin

III. Fonctions définies par induction

A. Définition inductive non-ambiguë

37 **Def** [MPLI Def 3.18]

38 **Rem**

39 **Ex** [MPLI Ex 3.19]

B. Fonctions définies par induction

40 **Def** [MPLI Def 3.20]

41 **Rem**

42 **Ex** [MPLI Ex 3.24]

43 **Implem** [TOR Programme 7.22]

44 **Ex** fonctions Premiers, Suivants et Annulables sur les expressions régulières.[CAR] [DRAG] [TIGER]

45 **Rem** [MPLI]

- Cette leçon s'appuie surtout sur [MPLI] pour la structure du plan et le formalisme mathématiques, avec des exemples de code OCaml venant de [TOR] .
- [MPLI] Défini les ensembles inductivement en dissociant les cas de base des autres constructeurs, là où [TOR] dis que les cas de base sont les constructeurs d'arités 0. Ici, on a fais le choix d'adapter les définitions et propriétés de [MPLI] pour utiliser la même définition que [TOR] . On a préféré ce formalisme car plus simple pour écrire les propriétés, et plus proche du code OCaml sur les type inductifs et les match case.

Bibliographie

[MPLI] A. Arnold & I. Guessarian, *Mathématiques pour l'informatique Avec 309 exercices corrigés.*

[TOR] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen & L. Sartre, *MP2I MPI, Informatique Cours et exercices corrigés.*

[CAR] O. Carton & G. G. Gagnees, *Langages formels: Calculabilité et complexité.*

[DRAG] A. Aho & M. Lam & R. Sethi & J. Ullman, *Compilateurs, Principes, techniques et outils.*

[TIGER] A. Appel, *Modern compiler implementation.*

Benjamin Voisin - Axel Stengel