MA122 - Récurrence et induction structurelle

Frédéric Herbreteau
frederic.herbreteau@bordeaux-inp.fr

10 octobre 2024

1. Preuve par récurrence

Preuve par récurrence

Théorème

Soit $P(n)$ un prédicat sur les entiers naturels. Si :

- $P(0)$ *est vrai*
- *et* $P(k)$ *implique* $P(k+1)$ *pour tout* $k \in \mathbb{N}$

Alors, $P(n)$ *est vrai pour tout entier naturel* n

$$\frac{P(0) \qquad \forall k \in \mathbb{N}, (P(k) \textbf{ implique } P(k+1))}{\forall n \in \mathbb{N}, P(n)}$$

Exemple

$P(n)$ défini par "$1 + 2 + \cdots + n = \frac{n(n+1)}{2}$"

Méthodologie (preuve par récurrence)

1. **Indiquer** que vous procédez par induction

2. **Définir** le prédicat $P(n)$ à prouver
   - $P(n)$ est appelé "**hypothèse d'induction**"
   - souvent la **partie cruciale** de la preuve

3. **Montrer** "$P(0)$" **vrai** ("cas de base")

4. **Montrer** "**si** $P(k)$ **alors** $P(k+1)$" **pour** $k \in \mathbb{N}$ ("induction")
   - Prouve les implications : $P(0) \implies P(1)$, $P(1) \implies P(2)$, ...

5. **Conclure** par le principe d'induction que $P(n)$ est vrai pour tout $n \in \mathbb{N}$

Exemple de preuve par récurrence

Exemple

Montrer que le prédicat $P(n)$ ci-dessous est vrai pour tout $n \in \mathbb{N}$

$$P(n) \triangleq \text{“}1 + 2 + \cdots + n = \frac{n(n+1)}{2}\text{”}$$

On prouve P par récurrence.

Base : on prouve $P(0)$ vraie

Induction : soit k un entier naturel quelconque. **On suppose $P(k)$ vraie (hypothèse d'induction)**, et on montre $P(k+1)$

Alors, on en déduit que $P(n)$ est vraie pour tout $n \in \mathbb{N}$.

Récurrence forte

Théorème

Soit $P(n)$ un prédicat sur les entiers naturels. Si :

- $P(0)$ *est vrai*
- *et* $(\forall i \leq k, P(i))$ *implique* $P(k+1)$ *pour tout* $k \in \mathbb{N}$

Alors, $P(n)$ est vrai pour tout entier naturel n

$$\frac{P(0) \qquad \forall k \in \mathbb{N}, ((\forall i \leq k, P(i)) \textbf{ implique } P(k+1))}{\forall n \in \mathbb{N}, P(n)}$$

Remarque : On utilise $(\forall i \leq k, P(i))$ pour démontrer $P(k+1)$ au lieu de $P(k)$ seulement

Une preuve où la récurrence forte est utile

Exemple

Montrer par récurrence sur $n \in \mathbb{N}$:
"tout entier $n \geq 2$ se décompose en un produit de nombres premiers"

- tentative avec la récurrence faible
- preuve en récurrence forte

Récurrence faible vs. forte

Théorème

Tout prédicat $P(n)$ qui peut être montré vrai par induction forte peut également l'être par induction faible

- Supposons $P(0)$ et $\forall k \in \mathbb{N}, ((\forall i \leq k, P(i)) \implies P(k))$ sont vraies
- Alors, on peut montrer par **induction faible** que le prédicat $Q(n)$: "$\forall k \leq n, P(k)$" est vrai pour tout n naturel
- Et on en déduit que $P(n)$ est vrai pour tout n naturel

Remarque : par **simplicité**, on préfère utiliser l'induction faible quand c'est possible

2. Induction structurelle

2.1 Structures de données récursives

Définition récursive d'un ensemble

Définition

Un **type de données récursif** T est défini par :

- un ensemble B d'**éléments de base**
- un ensemble de fonctions K qui permettent de **construire** de nouveaux éléments de T à partir de ceux déjà connus

Le type T défini l'**ensemble X_T d'éléments** construits à partir des éléments de B en leur appliquant un **nombre fini** de fois les constructeurs de K.

Exemple

Définition récursive de $\mathbb{N}$:

- 0 est un entier naturel
- si n est un entier naturel, alors $n+1$ est un entier naturel

$0 =$ $\quad$ $1 =$ $\quad$ $5 =$

D'autres types récursifs

- **Entiers pairs**

  Base : 0 est un entier pair

  Constructeurs : pour tout entier pair n, $(n+2)$ d'une part, et $(-n)$ d'autre part, sont des entiers pairs

- **Listes d'entiers**

  Base : $\langle\rangle$ est la liste vide

  Constructeur : pour toute liste l et tout entier n, $\langle n, l\rangle$ est une liste

- **Arbres binaires étiquetés**

  Base : `Empty` est un arbre binaire (l'arbre vide)

  Constructeur : pour tous arbres binaires l et r et tout entier n, $\texttt{Node}(n, l, r)$ est un arbre binaire

Définitions ambigües

Définition

Une définition récursive est **non-ambigüe** si tout élément est construit de manière unique.

Exemple

On défini l'ensemble des paires d'entiers naturels par :

Base : $(0, 0)$ est une paire d'entiers

Constructeur : si (m, n) est une paire d'entiers, alors $(m + 1, n)$, et $(m, n + 1)$ sont des paires d'entiers

- Comment peut-on construire $(1, 1)$?
- Peut-on énumérer les paires (m, n) à partir de cette définition ?

2.2 Fonctions récursives

Fonctions récursives sur $\mathbb{N}$

Méthodologie :

- définir $f(0)$ (et si nécessaire $f(1)$, $f(2)$, ...)
- définir $f(n)$ en fonction de $f(n-1)$, $f(n-2)$, ...

Exemple

- **Factorielle :**

  Base : $fact(0) = 1$

  Récursion : $fact(n) = n \times fact(n-1)$

- **Fibonacci :**

  Base : $F(0) = 0,\ F(1) = 1$

  Récursion : $F(n) = F(n-1) + F(n-2)$

Définitions mal formées

Exemple

- Absence de cas de base : $f_1(n) = 1 + f_1(n-1)$
- Récursion divergente : $f_2(n) = \begin{cases} 0 & \text{si } n = 0 \\ f_2(n+1) & \text{sinon} \end{cases}$
- Conjecture de Collatz : $f_3(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ f_3(n/2) & \text{si } n \text{ est pair} \\ f_3(3n+1) & \text{si } n \text{ est impair} \end{cases}$

Fonction récursive sur un type récursif

Définition

Une **fonction récursive** f sur un type récursif **non-ambigü** $T = (B, K)$ est définie par :

- la valeur de $f(b)$ pour tout élément $b \in B$
- la valeur de $f(k(x_1, \ldots, x_n))$ à partir de $f(x_1)$, …, $f(x_n)$ et x_1, …, x_n pour tout constructeur $k \in K$

Exemple

- Longueur d'une liste :

  Base : $|\langle\rangle| =$

  Récursion : $|\langle n, l\rangle| =$

- Concaténation de listes :

  Base : $\langle\rangle \cdot l =$

  Récursion : $\langle n, l_1\rangle \cdot l_2 =$

Applications et définitions ambigües

Attention : les fonctions récursives sont définies uniquement pour les **types non-ambigüs**

Exemple

Rappel : définition récursive des paires d'entiers naturels

Base : $(0, 0)$ est une paire d'entiers

Constructeur : si (m, n) est une paire d'entiers, alors $(m + 1, n)$, et $(m, n + 1)$ sont des paires d'entiers

"Définition" de f :

$$
\begin{aligned}
f(0,0) &= 0 \\
f(m,n) &= \max(m,n) \text{ si } m \neq n \\
f(m+1,n) &= 2 * f(m,n) \text{ si } m+1 = n \\
f(m,n+1) &= 3 * f(m,n) \text{ si } m = n+1
\end{aligned}
$$

Valeurs de $f(1,1)$?

Lien avec la programmation (caml)

- **Définition du type "liste" :**

  Base : $\langle\rangle$ est la liste vide

  Constructeur : pour toute liste l et tout entier n, $\langle n, l\rangle$ est une liste

```
type list = [] | (::) int * list
```

- **Longueur d'une liste :**

  Base : $|\langle\rangle| = 0$

  Récursion : $|\langle n, l\rangle| = 1 + |l|$

```
let rec length l =
    match l with
      [] -> 0
    | n :: ll -> 1 + length ll
```

2.3 Induction structurelle

Induction structurelle faible

Soit $T = (B, K)$ la définition récursive d'un ensemble X_T.

Théorème

Soit P un prédicat défini sur le type inductif T. Si :

Base : $P(b)$ *est vrai pour tout élément* $b \in B$

Induction : $P(k(x_1, \ldots, x_n))$ *est vrai sous hypothèse* $P(x_1)$, ..., $P(x_n)$, *pour tout constructeur* $k \in K$

alors, $P(x)$ *est vrai pour tout élément* x *de* X_T.

$$\frac{\forall b \in B, P(b) \qquad \forall k \in K, (P(x_1) \wedge \cdots \wedge P(x_n) \implies P(k(x_1, \ldots, x_n)))}{\forall x \in X_T, P(x)}$$

Méthodologie (preuve par induction structurelle)

Soit $T = (B, K)$ un type inductif qui définit un ensemble X_T d'éléments.

1. **Indiquer** que vous procédez par induction structurelle
2. **Définir** le prédicat $P(x)$ à prouver
   - $P(x)$ est appelé "**hypothèse d'induction**"
   - souvent la **partie cruciale** de la preuve
3. **Montrer** "$P(b)$" **vrai pour tout** $b \in B$ ("cas de base")
4. **Montrer "si** $P(x_1)$**, ...,** $P(x_n)$ **alors** $P(k(x_1, \ldots, x_n))$**" pour tout constructeur** $k \in K$ ("induction")
5. **Conclure** par le principe d'induction structurelle que $P(x)$ est vrai pour tout $x \in X_T$

Exemple de preuve par induction structurelle

Exemple

"Pour toutes listes l_1 et l_2, $|l_1 \cdot l_2| = |l_1| + |l_2|$" (*)

On prouve par induction sur l_1. Soit $P(l_1) \triangleq \forall l_2, |l_1 \cdot l_2| = |l_1| + |l_2|$

Base : on prouve $P(\langle\rangle)$ vrai

Induction : soit l_1 une liste quelconque et n un entier. **On suppose $P(l_1)$ vrai (hypothèse d'induction), et on montre $P(\langle n, l_1 \rangle)$**

Alors, on en déduit que $P(l_1)$ est vrai pour toute liste l_1, donc que (*) est vraie

Induction structurelle forte

Soit $T = (B, K)$ la définition récursive d'un ensemble X_T.

Définition

Soit x_1 et x_2 deux éléments de X_T, on définit $x_1 \leq_T x_2$ si x_1 apparaît dans la construction de x_2

Exemple

$\langle\rangle \leq_{liste} \langle 1, \langle\rangle\rangle \leq_{liste} \langle 0, \langle 1, \langle\rangle\rangle\rangle$ mais $\langle 0, \langle\rangle\rangle \not\leq_{liste} \langle 0, \langle 1, \langle\rangle\rangle\rangle$

Théorème

Soit P un prédicat défini sur le type inductif T. Si :

Base : $P(b)$ *est vrai pour tout élément* $b \in B$

Induction : $P(k(x_1, \ldots, x_n))$ *est vrai sous hypothèse "pour tout* $y_1 \leq_T x_1$, ..., $y_n \leq_T x_n$, $P(y_1)$, ..., $P(y_n)$*", et pour tout constructeur* $k \in K$

alors, $P(x)$ *est vrai pour tout élément* x *de* X_T.

Conclusion sur l'induction

- Les techniques d'induction ont de **larges applications** en informatique : (programmation : types, fonctions récursives, ainsi que dénombrement, réseaux, etc)
- La technique de **preuve par induction** est puissante et s'applique à de nombreux contextes (mathématiques, preuve de programmes, etc.)
- On peut ramener une preuve par induction structurelle à une preuve par récurrence (sur $\mathbb{N}$) ...
- ... mais les preuves par induction structurelle sont **plus claires et plus simples**