Ordres bien fondés et application à la terminaison d'algorithmes

Damien Mégy

20 décembre 2024

Table des matières

1 Motivation

Dans ce chapitre, on présente une notion théorique, celle d'ordre bien fondé, ainsi qu'un principe de récurrence généralisé, valable sur les ensembles bien fondés et non plus seulement $\mathbb{N}$.

Ensuite, on applique ceci à la rédaction de preuves de terminaison de certains algorithmes.

2 Prérequis

Chapitre de relations d'ordre de première année. Relations d'ordre, ordre total, plus petit élément, majorants et minorants, applications croissantes.

Sources possibles pour réviser :

1. Tout-en-un L1 Ramis-Warusfel à la BU.

2. Chapitre sur les relations d'équivalence et d'ordre sur Bibmath : https://www.bibmath.net/ressources/index.php?action=affiche&quoi=bde/logique/relations&type=fexo. Les derniers exercices correspondent au début de ce chapitre de compléments. (Mais les définitions prises ne sont pas forcément les mêmes.)
3. Mansuy, Tout-en-un MPSI, livre « anciens programmes » épuisé et mis à disposition gratuitement par l'auteur : http://www.rogermansuy.fr/pdf/MPSI2019.pdf
4. Document d'accompagnement de l'ancienne UE de L1 couvrant ce thème à Nancy : https://github.com/dmegy/decouverteDesMaths/blob/master/cours.pdf. Ce poly un peu expérimental est un bon moyen de réviser les fondamentaux de L1 quelques années après. Certains exercices sont assez difficiles. Tous ces exercices ont été ajoutés à la base exo7, qui en contenait déjà beaucoup d'autres plus classiques.
5. Exo 7 : http://exo7.emath.fr/bin/extract5.php, catégorie L1 Algèbre /100.05.

3 Introduction et rappels

Définition 3.1. *Soit $\leq_E$ un ordre sur un ensemble E. L'**ordre strict** associé à $\leq$ est la relation binaire $<_E$ sur E définie par*

$$x <_E y \iff x \leq_E y \text{ et } x \neq y.$$

Exemples et contre-exemples fondamentaux de relations d'ordre

1. L'ensemble $\mathbb{N}$ muni de la relation d'ordre usuelle $\leq$ est un ensemble ordonné.
2. L'ensemble $\mathbb{N}$ muni de la relation de divisibilité est un ensemble ordonné.
3. L'ensemble $\mathbb{Z}$ muni de la relation de divisibilité n'est **pas** un ensemble ordonné.
4. Si E est un ensemble, l'ensemble $\mathcal{P}(E)$ des parties de E est naturellement ordonné par la relation $\subseteq$.
5. Si $(E, \leq_E)$ et $(F, \leq_F)$ sont des ensembles ordonnés, l'ensemble produit $E \times F$ peut être muni :
   (a) de l'**ordre produit**, définie par :
   $$(e, f) \leq_{E\times F} (e', f') \iff e \leq_E e' \text{ et } f \leq_F f';$$
   (b) de l'**ordre lexicographique**, défini par
   $$(e, f) \leq_{lex} (e', f') \iff (e <_E e' \text{ ou } (e = e' \text{ et } f \leq_F f')).$$

Définition 3.2. *Un ordre $\leq$ sur E est dit total si tous les éléments sont comparables :*

$$\forall (x, y) \in E^2, x \leq y \text{ ou } y \leq x.$$

Exercice. *1. Montrer que l'ordre produit sur $\mathbb{N}^2$ n'est pas total.*

2. *Montrer que la divisibilité sur $\mathbb{N}$ n'est pas un ordre total.*
3. *Montrer que l'ordre lexicographique sur $\mathbb{N}^2$ est total.*
4. *Montrer que si E et F sont bien ordonnés, l'ordre lexicographique sur le produit $E \times F$ est un ordre total.*

Une suite dans un ensemble E est un élément de $E^{\mathbb{N}}$, c'est-à-dire une application de $\mathbb{N}$ dans E. On note en général les suites comme des familles indexées par $\mathbb{N}$, par exemple comme ceci : $u = (u_n)_{n\in\mathbb{N}}$. Si l'ensemble E est muni d'une relation d'ordre $\leq_E$, Une suite est (strictement) (dé)croissante si l'application en question est (strictement) (dé)croissante, vue comme application entre les ensembles ordonnés $(\mathbb{N}, \leq)$ (ordre usuel) et $(E, \leq_E)$.

4 Ordres bien fondés

Définition 4.1. *Soit $(E, \leq)$ un ensemble ordonné, $X \subseteq E$ une partie de E et $a \in X$. On dit que :*

1. *a est un plus petit élément de X si $\forall x \in X, a \leq x$.*
2. *a est un élément minimal de X s'il n'existe pas de $y \in X$ avec $x < a$.*

(On définit de même la notion de plus grand élément et d'élément maximal.)

Exemples 4.2.
1. *Dans $\mathbb{N} \setminus \{0, 1\}$ muni de la divisibilité, 5 est minimal mais n'est pas un plus petit élément.*
2. *Un plus petit élément est unique, un élément minimal pas forcément.*
3. *Dans un ensemble totalement ordonné, un élément minimal est un (le) plus petit élément.*

Définition 4.3. *Soit $(E, \leq)$ un ensemble ordonné. On dit que :*

1. *$(E, \leq)$ est <u>bien ordonné</u> si toute partie non vide admet un plus petit élément.*
2. *$(E, \leq)$ est <u>bien fondé</u> si toute partie non vide admet un élément minimal.*

Exemples 4.4.
1. *Un bon ordre est toujours bien fondé.*
2. *L'ordre usuel $\leq$ sur $\mathbb{Z}$ n'est pas un bon ordre, ni bien fondé : les parties $\mathbb{Z}$, ou $2\mathbb{Z}$, ou $\mathbb{Z}_-$, n'ont pas de plus petit élément, ni d'élément minimal.*
3. *L'ordre usuel sur $\mathbb{N}$ est un bon ordre.*
4. *La divisibilité sur $\mathbb{N}$ est bien fondée, mais n'est pas un bon ordre : la partie $\{2, 3\}$ n'a pas de plus petit élément pour la divisibilité.*
5. *l'ensemble $\mathcal{P}(\mathbb{N})$ avec l'ordre usuel induit par l'inclusion de parties n'est pas bien fondé : la partie formée par les parties $[n, \infty[$ n'a pas d'élément minimal.*

Exercice. *Montrer qu'un ordre est bon ssi il est bien fondé et total.*

Théorème 4.5. *Un ensemble ordonné E est bien fondé si et seulement s'il n'existe pas de suite strictement décroissante dans E.*

Démonstration.
1. Supposons qu'il existe une suite $(u_n)_{n\in\mathbb{N}}$ d'éléments de E, strictement décroissante. Alors, l'image de la suite $X = \{u_n, n \in \mathbb{N}\}$ est une partie non vide qui n'a pas d'élément minimal. En effet, si $x \in X$, alors par définition de X il existe $n \in \mathbb{N}$ tel que $x = u_n$. Définissons alors $y = u_{n+1}$. On a $y \in X$ et $y < x$ car la suite est strictement décroissante.
2. Réciproquement, s'il existe une partie non vide $X \subseteq E$ sans élément minimal, on peut construire par récurrence une suite $(u_n)_{n\in\mathbb{N}}$ strictement décroissante en prenant $u_0 \in X$ puisque X est non vide, puis comme X n'a pas d'élément minimal, u_0 n'est pas minimal donc il existe $u_1 \in X$ avec $u_1 < u_0$, mais u_1 n'est pas non plus minimal etc.

□

Exercice. *Un ensemble est bien fondé ssi toute suite décroissante est stationnaire.*

Exercice. *Soient E et F des ensembles ordonnés. Si E et F sont bien fondés, le produit $E \times F$ muni de l'ordre lexicographique est bien fondé.*

Théorème 4.6 (Principe de récurrence généralisée / récurrence transfinie / récurrence bien fondée)**.** *Soit $(E, \leq)$ un ensemble bien fondé et soir $<$ l'ordre strict associé. Soit $\mathcal{P}(x)$ un prédicat sur E. (Une assertion à paramètre $x \in E$, autrement dit une fonction de E dans les booléens.)*

Supposons que :

$$\forall x, (\forall y < x, P(y)) \implies P(x)$$

Alors, $\forall x \in E, P(x)$.

Démonstration. Supposons $\forall x, (\forall y < x, P(y)) \implies P(x)$, assertion que l'on notera (H). Montrons $\forall x \in E, P(x)$. Supposons par l'absurde que $\forall x \in E, P(x)$ soit fausse. La partie $A := \{x \in E, non(P(x))\}$ est alors non vide. Elle possède donc un élément minimal, que l'on note m. Par définition, $\forall y \in E, y < m \implies y \notin A$, c'est-à-dire $\forall y \in E, y < m \implies P(m)$. D'après H, on en déduit $P(m)$, contradiction. □

Remarque 4.7. *L'hypothèse $\forall x, (\forall y < x, P(y)) \implies P(x)$ ressemble à une hypothèse d'hérédité de type récurrence forte, mais elle contient aussi l'initialisation (ou : les initialisations), car elle implique que pour tout x minimal dans E, on a $P(x)$. En effet, il n'y a alors aucun élément $y < x$ donc la prémisse $\forall y < x, P(y)$ est vraie, donc par implication, $P(x)$ est vraie.*

5 Application : terminaisons d'algorithmes

Rappel : la terminaison d'un algorithme n'est pas toujours facile à déterminer. L'exemple le plus célèbre est la conjecture de Syracuse, ou problème de Collatz : il n'est pas connu si la fonction suivante termine pour toute entrée $n \in \mathbb{N}$:

```
def Syracuse(n):
  while n != 1:
    if n % 2 == 0:
      n = n / 2
    else:
      n = 3*n+1
  return 0
```

Même lorsque l'on sait que l'algorithme termine, rédiger une preuve de la terminaison peut être plus ou moins long.

Dans les cas simples, comme le calcul récursif d'une factorielle, l'entrée est un entier positif qui diminue à chaque appel récursif, ce qui implique qu'il n'y a qu'un nombre fini d'appels successifs.

Mais parfois, il n'y a pas de choix clair d'entier positif qui diminue strictement, comme par exemple dans la fonction suivante qui prend en entrée un couple d'entiers strictement positifs :

```
def pgcd(a: int, b: int) -> int:
  # prend des entiers strictement positifs en entree
  if a == b:
    return a
  if a > b:
    return pgcd(a-b,b)
  else:
    return pgcd(a,b-a)
```

À chaque appel récursif, a ne diminue pas forcément strictement, ni b.

Dans ce genre de situation, une technique de rédaction possible consiste à utiliser la notion d'ensemble ordonné bien fondé.

Dans le cas itératif, on va identifier un variant de boucle à valeur dans un ensemble bien fondé, qui décroit strictement à chaque tour de boucle. Ceci implique que la boucle ne tourne qu'un nombre fini de fois.

Dans le cas récursif, on va montrer que l'entrée appartient à un ensemble bien fondé et qu'à chaque appel récursif, l'entrée diminue strictement. La profondeur des appels récursifs est donc finie.

Dans l'exemple de calcul récursif de pgcd, le couple (a, b) est un couple d'entiers strictement positifs et s'il y a un appel récursif, il se fait sur un couple (a', b') d'entiers strictement positifs qui est strictement inférieur à (a, b) pour l'ordre lexicographique. Comme cet ordre est bien fondé, il n'existe donc pas de suites infinies strictement décroissantes pour cet ordre et donc l'algorithme termine.

Exercice. *On suppose que l'on a une fonction* `isPrime(n)` *prenant un entier et renvoyant un booléen, et une fonction* `getNonTrivialDivisor(n)` *prenant un nombre* > 2 *composé et renvoyant un diviseur non trivial de ce nombre.*

Rédiger la preuve de terminaison de la fonction suivante, prenant en entrée un entier strictement positif.

```
def mystere(n):
```

```
# input : un entier >=2
if isPrime(n):
  return 1
a = getNonTrivialDivisor(n)
return mystere(a) + mystere(n/a)
```

6 Ouverture

Il existe beaucoup de littérature sur les préordres. Voir par exemple :

1. https://en.wikipedia.org/wiki/Well-quasi-ordering
2. Well quasi-orders for algorithms, (cours de L3 info) https://wikimpri.dptinfo.ens-cachan.fr/lib/exe/fetch.php?media=cours:upload:poly-2-9-1v02oct2017.pdf