1 Définitions

Une fonction est un objet mathématique qui associe à une donnée un objet.

Un algorithme est une suite finie d'opérations élémentaires prises dans un catalogue fini d'opérations possibles, à appliquer dans un ordre déterminé à un nombre fini de données finies, pour arriver en un nombre fini d'étapes, à un certain résultat et cela pour toutes valeurs des données (d'après Encyclopedia Universalis). Un algorithme est donc une suite d'opérations nécessaires au calcul explicite d'une fonction à partir d'opérations élémentaires fixées par avance. Une fonction est calculable s'il existe un algorithme qui permet de calculer l'image de toute donnée.

Un programme est la description dans un langage donné d'un algorithme, c'est donc une suite finie de symboles (lettres, chiffres, etc).

Un algorithme est destiné à être exécuté par un opérateur (pour nous, un ordinateur). On appelle processus la mise en œuvre d'un algorithme par un opérateur sur un jeu de données (on dit aussi calcul plutôt que processus et calculateur plutôt que opérateur). L'opérateur doit donc connaître la description de l'algorithme (un programme) et être en mesure de l'appliquer : on suppose donc que l'opérateur est capable d'effectuer les opérations élémentaires de l'algorithme.

Diverses théories des algorithmes, des opérations élémentaires et des calculateurs ont été proposées depuis les années 30 (λ-calcul, fonctions récursives, modèle de Turing, etc). A chaque fois, il a été montré que si une fonction était calculable selon une certaine théorie, elle l'était selon toute autre. Cela laisse penser qu'il existe une notion universelle du calculable, ce qu'a énoncé Church dans sa Thèse.

2 Calculabilité

Un programme décrit comment en pratique calculer l'image d'une donnée par une fonction.

Notons $\mathbb{P}$ l'ensemble des programmes et $\mathscr{F}$ l'ensemble des fonctions. On dispose donc d'une application Sem, appelée sémantique, de $\mathbb{P}$ dans $\mathscr{F}$ qui à un programme associe une fonction (cette application dépend du jeu d'opérations élémentaires disponibles).

Cette application n'est pas injective (deux programmes peuvent définir la même fonction).

Elle n'est pas non plus surjective, car $\mathbb{P}$ est beaucoup « plus petit » que $\mathscr{F}$.

On ordonne le langage dans lequel on écrit les programmes (ordre lexicographique, par exemple) : on peut alors numéroter chaque programme, on dit que $\mathbb{P}$ est dénombrable, c'est-à-dire qu'il existe une bijection de $\mathbb{N}$ dans $\mathbb{P}$. Donc l'ensemble des fonctions calculables (*i.e.* l'image de Sem) est aussi dénombrable.

Or on peut montrer que l'ensemble des fonctions n'est pas dénombrable (voir annexe). Il existe donc des fonctions non calculables.

3 Problème de l'arrêt

La plupart des fonctions qui seraient grandement utiles à l'étude des algorithmes sont malheureusement non calculables.

L'exemple le plus représentatif est la fonction d'arrêt. Le problème est le suivant : un texte (un pseudo-programme) étant donné, est-il un vrai programme ? Autrement dit, si on applique le pseudo-algorithme décrit par le texte à une certaine donnée, est-on sûr que le processus ainsi amorcé va s'arrêter et donner un résultat ? On dit que l'algorithme termine dans ce cas.

Pour montrer que la fonction d'arrêt n'est pas calculable, on le prouve par l'absurde. On suppose donc qu'il existe un programme Arret tel que Sem(Arret) est la fonction d'arrêt. Cette fonction d'arrêt prend en paramètre un texte et donne comme résultat "vrai" si le pseudo-algorithme décrit par le programme termine pour toute valeur des données et "faux" sinon. La fonction d'arrêt est donc décrite par un programme exécutable par un opérateur *ad hoc*.

On crée alors le pseudo-programme suivant (écrit en pseudo-code) :

```
absurde() :
  stop := Arret(texte de la fonction "absurde()");
  tant que stop = vrai faire
    ()
  finfaire;
  sortie : 0;
```

On fait exécuter le pseudo-programme "absurde". La première étape de l'exécution consiste à calculer la valeur de la fonction d'arrêt sur le (pseudo-)programme "absurbe".

Si ce processus termine, alors la variable "stop" vaut vrai. Donc la boucle suivante (qui ne fait rien) ne s'arrête jamais, car sa condition est toujours vraie, donc le programme "absurde" ne termine pas.

Si ce processus ne termine pas, alors la variable "stop" vaut faux. Donc la boucle suivante n'est même pas effectuée, donc le programme "absurde" termine.

Dans les deux cas, on a une contradiction.

4 Opérations élémentaires

- l'affectation : on peut donner des valeurs à des symboles (appelés en général variables) et modifier leurs valeurs, on note en général := l'opérateur d'affectation (ou <−)
- les opérations arithmétiques sur les entiers relatifs
- les évaluations de valeurs logiques (vrai/faux) et les opérateurs associés (et, ou, non, <, ...)
- la fonction conditionnelle (si ... alors ... sinon ...)
- les boucles indexées et conditionnelles (pour ... variant de ... à ... faire ... / tant que ... faire ...)

Pour décrire un algorithme, on utilisera du pseudo-code :

```
algo:=fonction(x : entier)
  y:=x^2;
  si y < x + 100 alors retourner x
  sinon retourner x−100
  finsi;
```

On peut montrer que cet ensemble d'opérations est calculatoirement complet, c'est-à-dire que toute fonction calculable selon une certaine méthode l'est aussi grâce à ces opérations, à un codage près des objets.

En général, les ordinateurs munis d'un langage classique sont capables d'en faire plus : ils peuvent manipuler des nombres à virgules, des tableaux, des chaînes de caractères, etc, toutes choses pouvant être codés par un nombre fini d'entiers. Le langage enseigné cette année est le langage CAML dans sa version « caml-light », développé par l'INRIA.

Voici en langage CAML le programme décrivant le même algorithme

```
let algo x =
  let y = x^2 in
  if y < x + 100 then x
  else x−100
```

A Dénombrabilité

A.1 Définitions

Un ensemble E est dit dénombrable si et seulement si il existe une bijection de E dans $\mathbb{N}$.

Proposition 1 *Si E, F sont deux ensembles, si E est dénombrable, et s'il existe une bijection de E dans F, alors F est dénombrable.*

Démonstration. Une composée de bijections est une bijection. •

A.2 Exemples fondamentaux

- ▷ $\mathbb{Z}$ est dénombrable.
- ▷ $\mathbb{N}^2$ est dénombrable.
- ▷ Plus généralement, un produit cartésien fini d'ensembles dénombrables est dénombrable.

A.3 Propriétés des ensembles dénombrables

Proposition 2 *Toute partie infinie de $\mathbb{N}$ est dénombrable.*

Démonstration. Soit A une partie infinie de $\mathbb{N}$. Puisque A n'est pas vide, d'après la propriété fondamentale de $\mathbb{N}$, A a un minimum a_0.

Comme A est infinie, $A - \{a_0\}$ est encore une partie non vide de $\mathbb{N}$, donc elle admet un minimum a_1.

Plus généralement, on suppose avoir construit $a_0, .., a_n$ éléments de A tels que pour tout $i \in \{0, ..., n\}$, $a_i = \min A - \{a_0, ..., a_{i-1}\}$.

Comme A est infinie, $A - \{a_0, ..., a_n\}$ est une partie non vide de $\mathbb{N}$, donc elle admet un minimum a_{n+1}.

On a alors construit $a_0, .., a_n, a_{n+1}$ éléments de A tels que pour tout $i \in \{0, ..., n+1\}$, $a_i = \min A - \{a_0, ..., a_{i-1}\}$.

D'après le principe de récurrence, on peut donc construire une suite (a_n) d'éléments de A tels que pour tout $i \in \mathbb{N}$, $a_i = \min A - \{a_0, ..., a_{i-1}\}$.

Par construction, cette suite est strictement croissante, donc injective. De plus, pour tout $b \in A$, si on note $m = \text{card}\{a \in A / a < b\}$, alors $b = a_m$: autrement dit, cette suite est surjective. On a ainsi créé une bijection de $\mathbb{N}$ dans A, donc A est dénombrable. •

Proposition 3 *Soit E un ensemble. S'il existe une injection de E dans $\mathbb{N}$, alors E est fini ou dénombrable. Plus généralement, si E, F sont deux ensembles, si F est dénombrable et s'il existe une injection de E dans F, alors E est fini ou dénombrable.*

Démonstration. Une injection de E dans $\mathbb{N}$ induit une bijection de E sur une partie de $\mathbb{N}$. Si cette partie est finie, alors E l'est aussi, sinon E est en bijection avec une partie infinie de $\mathbb{N}$, donc est dénombrable.

Le deuxième point en découle, car une composée d'injections est une injection. •

Proposition 4 *Soit E un ensemble. S'il existe une surjection de $\mathbb{N}$ dans E, alors E est fini ou dénombrable. Plus généralement, si E, F sont deux ensembles, si E est dénombrable et s'il existe une surjection de E dans F, alors F est fini ou dénombrable.*

Démonstration. Soit f une surjection de $\mathbb{N}$ dans E. Pour tout $x \in E$, $f^{-1}(x)$ est une partie non vide de $\mathbb{N}$, donc admet un minimum $\varphi(x)$. De cette façon, on crée une application φ de E dans $\mathbb{N}$. Pour tout $x \in E$, $f(\varphi(x)) = x$ donc φ est injective. D'après la proposition précédente, E est donc fini ou dénombrable.

Le deuxième point en découle, car une composée de surjections est une surjection. •

Corollaire 5 *Si E est un ensemble dénombrable et $f : E \to F$ une application, alors $f(E)$ est finie ou dénombrable.*

Démonstration. f induit une surjection de E dans $f(E)$. •

Exemples

a) $\mathbb{Q}$ est dénombrable, car on peut construire une injection de $\mathbb{Q}$ dans $\mathbb{Z} \times \mathbb{N}^*$: à tout rationnel r, on associe le couple $(p, q) \in \mathbb{Z} \times \mathbb{N}^*$ tel que $\frac{p}{q}$ soit le représentant irréductible de r.

b) L'ensemble des racines d'entiers $\sqrt[p]{n}$ est dénombrable, car l'application $(n, p) \mapsto \sqrt[p]{n}$ a pour ensemble de départ $\mathbb{N}^2$ qui est dénombrable.

A.4 Réunion dénombrable d'ensembles finis

Proposition 6 *Soit (E_n) une suite d'ensembles finis non vides deux à deux disjoints. Alors la réunion $\bigcup_{n\in\mathbb{N}} E_n$ est dénombrable.*

Démonstration. On note $E = \bigcup_{n\in\mathbb{N}} E_n$: pour $x \in E$, on note $n(x)$ l'unique entier n tel que $x \in E_n$.

Pour tout $n \in \mathbb{N}$, on pose f_n une bijection de E_n avec l'ensemble $\{1, 2, ..., \operatorname{card} E_n\}$.

À tout élément x de E, on associe le couple $\varphi(x) = (n(x), f_{n(x)}(x)) \in \mathbb{N} \times \mathbb{N}$. On définit ainsi une application φ de E dans $\mathbb{N} \times \mathbb{N}$ et on vérifie aisément qu'elle est injective.

On a donc une injection de E avec $\mathbb{N}^2$, qui est dénombrable, donc E est dénombrable (car il est évidemment infini). •

A.5 Un ensemble non dénombrable

Proposition 7 *L'ensemble des applications de $\mathbb{N}$ dans $\mathbb{N}$ n'est pas dénombrable.*

Démonstration. Par l'absurde, on suppose que cet ensemble $\mathscr{F}$ est dénombrable (il est manifestement infini, puisqu'il contient les applications constantes). Soit F une bijection de $\mathbb{N}$ dans $\mathscr{F}$: pour chaque $n \in \mathbb{N}$, $F(n)$ est une application de $\mathbb{N}$ dans $\mathbb{N}$.

On va construire une application de $\mathbb{N}$ dans $\mathbb{N}$ pour qu'elle soit différente de toutes les applications $F(n)$ par un procédé appelé « procédé diagonal de Cantor », inventé par le mathématicien allemand Cantor qui le premier a découvert que les ensembles infinis ne sont pas tous aussi gros les uns que les autres, et qu'il y a donc plusieurs niveaux d'infinis.

On définit une application g de $\mathbb{N}$ dans $\mathbb{N}$ de la façon suivante : pour $n \in \mathbb{N}$, on pose $g(n) = F(n)(n) + 1$.

Comme g appartient à $\mathscr{F}$, elle a un antécédent par F : soit donc p l'entier tel que $F(p) = g$.

Alors pour tout $i \in \mathbb{N}$, $F(p)(i) = g(i)$. Donc en particulier, on a $F(p)(p) = g(p)$, c'est-à-dire $F(p)(p) = F(p)(p) + 1$: contradiction.

Donc $\mathscr{F}$ est infini non dénombrable. •

Remarque. Il existe d'autres ensembles infinis non dénombrables. Par exemple, on peut montrer que $\mathbb{R}$ ou que tout intervalle de $\mathbb{R}$ non réduit à un point est non dénombrable.

B Application à la calculabilité

On considère un alphabet A (*i.e.* un ensemble fini de caractères autorisés) de cardinal a et un langage de programmation écrit avec cet alphabet. Un programme est un texte syntaxiquement correct écrit dans ce langage : c'est donc un ensemble fini de caractères.

Soit n un entier naturel, on note $\mathbb{P}_n$ l'ensemble des programmes constitué de n caractères et $\mathbb{P}$ l'ensemble de tous les programmes : $\mathbb{P} = \bigcup_{n\in\mathbb{N}} \mathbb{P}_n$.

Pour tout $n \in \mathbb{N}$, $\mathbb{P}_n$ est un ensemble fini, dont le cardinal est au plus a^n. Donc d'après une proposition précédente, $\mathbb{P}$ est un ensemble dénombrable : on peut donc numéroter les programmes.

L'image de $\mathbb{P}$ par la fonction sémantique (*i.e.* l'ensemble des fonctions calculables) est donc dénombrable, inclus dans un "ensemble" non dénombrable ("l'ensemble" des fonctions), donc il existe des fonctions non calculables. Et même mieux : il existe infiniment plus de fonctions non calculables que de fonctions calculables.