Démonstration et algorithme deux façons de penser, deux dialectes du langage mathématique

Emmanuel Beffara

Institut de Mathématiques de Marseille
Université d'Aix-Marseille

Forum Mathématiques vivantes
Mathématiques et langages
Lille – 18 mars 2017

Le langage mathématique

Les mathématiques se font en langue naturelle :

- Le but premier est de se faire comprendre de son lecteur ou interlocuteur.
- Comme dans toute activité technique, il y a du jargon :
  - un vocabulaire pour désigner des notions spécifiques au domaine,
  - une grammaire pour structurer le discours de façon adaptée au propos.
- Le niveau de détail employé dépend du niveau d'expertise attendue de la part de l'interlocuteur.

Démonstration et algorithme

Dans un cas comme dans l'autre, l'objet du discours est formel.

Démonstration / raisonnement

On veut convaincre l'interlocuteur de la validité d'une conséquence logique (ou de la véracité d'un énoncé).

Algorithme / calcul

On veut décrire à l'interlocuteur un procédé permettant d'obtenir un résultat à partir d'une donnée quelconque.

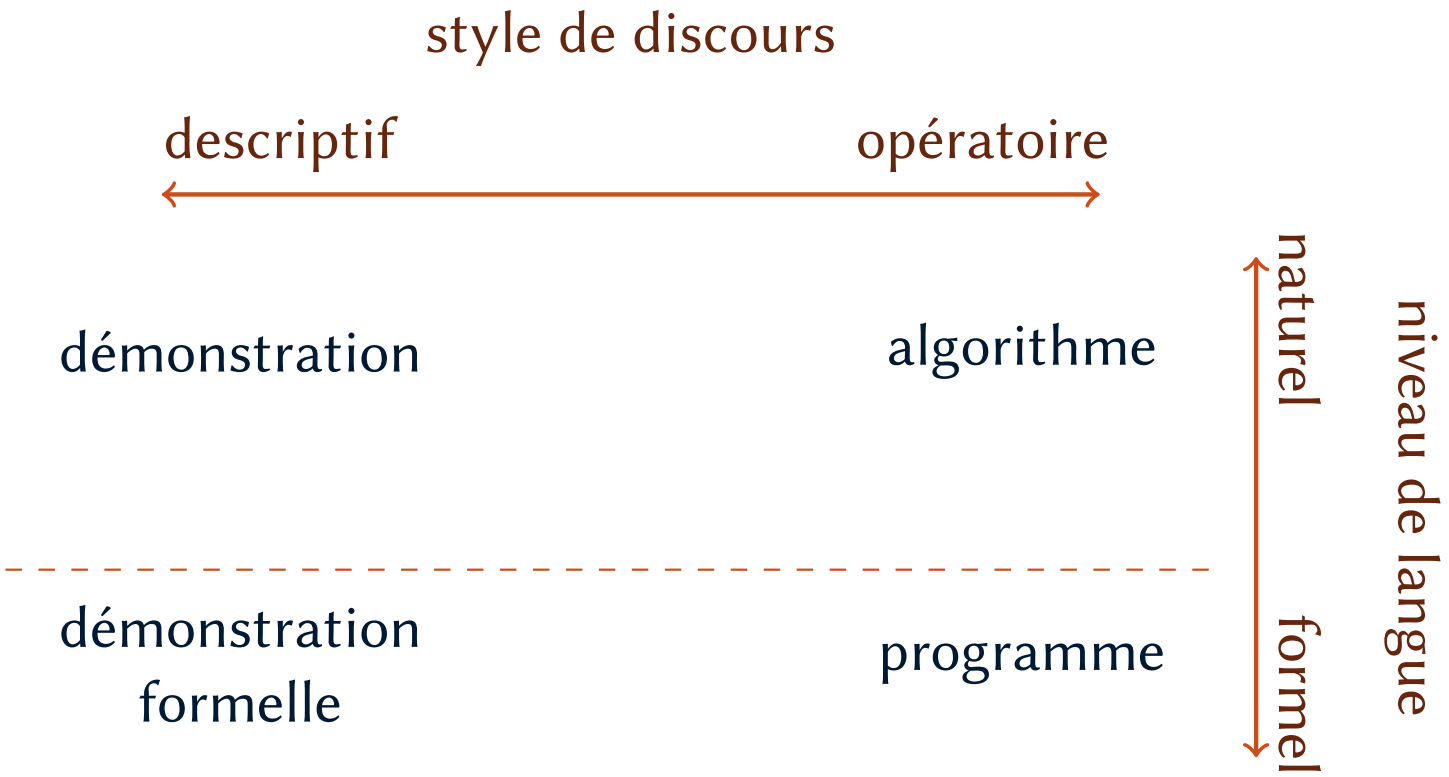
Dans les deux cas, on emploie des arguments

justifications logiques / étapes de calcul

que l'on pourrait en principe préciser jusqu'à atteindre des principes premiers

postulats et règles logiques / opérations élémentaires.

Différentes formes de discours



PGCD : théorème d'existence

Théorème

Pour tous entiers naturels a et b il existe un unique entier naturel d qui divise a et b tel que tout diviseur commun de a et b divise d.

On procède par récurrence forte sur a. Si $a = 0$ alors b est le PGCD de a et b. Sinon posons la division euclidienne $b = aq + r$ de sorte que $0 \leq r < a$. Par hypothèse de récurrence sur r on déduit qu'il existe un entier d qui est PGCD de r et a. Comme d divise r et a il divise aussi $aq + r$ donc b. De plus tout diviseur commun de b et a doit aussi diviser $b - aq = r$, donc diviser d par hypothèse. Donc d est le PGCD de a et b.

PGCD : algorithme itératif

Algorithme $PGCD(a, b)$

Entrées a, b : deux entiers naturels

Sortie le PGCD de a et b

Traitement Tant que $a \neq 0$, faire :

$r \leftarrow b \bmod a$,

$b \leftarrow a$,

$a \leftarrow r$.

Renvoyer b.

On justifie la correction de l'algorithme par deux ingrédients :

- invariant : le PGCD de a et b est le même à chaque tour de boucle
- terminaison : a décroît strictement (et est à valeur entière)

PGCD : de la démonstration à l'algorithme

Si on interprète directement la démonstration, on obtient :

Algorithme $PGCD(a, b)$

Entrées a, b : deux entiers naturels

Sortie le PGCD de a et b

Traitement
Si $a = 0$, alors
    renvoyer b.
Sinon
    renvoyer $PGCD(b \bmod a, a)$.

- L'algorithme obtenu à partir de la démonstration est naturellement *récursif* et *sans affectation de variables*.
- Le passage à une boucle « tant que ... » avec affectation aux variables a et b est une transformation classique.

PGCD : dans un manuel scolaire

Théorème — a et b sont des entiers naturels non nuls

Si $a \geqslant b$ alors PGCD(a ; b) = PGCD(b ; r) où r est le reste de la division euclidienne de a par b.

Remarque : Dans cet algorithme, appelé aussi « algorithme d'Euclide », le PGCD est le dernier reste non nul.

Exemple 2 : Trouve le PGCD de 782 et de 136 par la **méthode des divisions successives**.

- On effectue la division euclidienne de 782 par 136 : $782 = 136 \times 5 + 102$.

Donc PGCD(782 ; 136) = PGCD(136 ; 102).

- On cherche maintenant PGCD(136 ; 102) : on applique à nouveau la propriété.

On effectue la division euclidienne de 136 par 102 : $136 = 102 \times 1 + 34$.
Donc PGCD(136 ; 102) = PGCD(102 ; 34).

- On continue avec PGCD(102 ; 34).

On effectue la division euclidienne de 102 par 34 : $102 = 34 \times 3$.

Le reste est égal à **0** donc **PGCD(782 ; 136) = 34** qui est le dernier reste non nul.

782	136
102	5

136	102
34	1

102	34
0	3

(extrait du manuel *Sésamath* de 3ème)

Comparaison des dialectes

Les discours sont de formes différentes :

Démonstrations style *descriptif*, on énonce des faits (indicatif présent, verbes d'état) et des relations de causalité

Algorithmes style *opératoire*, on prescrit des actions (impératif ou infinitif à valeur impérative, verbes d'action)

Une partie du vocabulaire est partagée, mais avec des différences de signification :

- variables
- tournures conditionnelles « si ... alors ... »
- quantification « pour tout ... »

Variables

Caractéristiques communes entre les dialectes :

- un nom (une lettre) qui désigne une valeur,
- libre (parlant) ou lié (muet) avec une portée.

Différents types d'usages :

usage	démonstration	algorithme
définition explicite	Posons $p = a^2 + 2ab$.	$p \leftarrow a^2 + 2ab$
supposé donné	Par hypothèse, $a \neq 0$.	Algorithme PGCD(a,b)
choisi de façon générique	Soit a un entier positif.	
redéfini explicitement		$a \leftarrow a + 1$

Remarques sur l'égalité et l'affectation

L'égalité construit des propositions (style déclaratif),
l'affectation modifie une variable (style opératoire).

- Certains langages de programmation créent de la confusion :

langages	affectation	égalité
C et dérivés, Python ...	=	==
Pascal, Caml ...	:=	=

- L'emploi d'égalités à valeur de définition perturbe la franche séparation entre les deux notions :

  Soit $\Delta = b^2 - 4ac$, ...

  à cause du statut particulier de l'égalité par rapport aux autres prédicats, qui permet d'écrire

  Soit Δ *tel que* $\Delta = b^2 - 4ac$, ...

Structure conditionnelle

La tournure « Si ... alors ... » est employée dans tous les contextes que l'on considère, avec des significations différentes :

Proposition « Si n est pair alors n^2 est pair. »
Énonce une implication.

Démonstration « Si $a \neq 0$ alors posons $b = aq + r$ ... »
Fait partie d'une disjonction de cas.

Algorithme « Si $a = 0$ alors renvoyer b. »
Décrit un branchement conditionnel.

On n'emploie jamais « Si ⟨action⟩ alors ... » qui a une signification encore différente (aspect temporel).

Quantification et itération

De même, les tournures de la forme « Pour tout ... » on un sens différent selon la nature (déclarative ou opératoire) de l'énoncé où elles apparaissent.

Proposition « Pour tout x, on a $x^2 + 2x + 1 > 0$. »
Énonce un propriété universelle des éléments d'un ensemble.

Algorithme « Pour chaque x dans L, faire $s \leftarrow s + x$. »
Décrit la répétition d'une opération sur tous les éléments d'un collection donnée.

Le langage des démonstrations n'emploie pas le « pour tout » mais des variables génériques pour démontrer les énoncés universels.

Comment faire le lien ?

Des mots et des tournures sont utilisés dans les raisonnements et dans les algorithmes mais avec un sens différent. Comment va-t-on s'y retrouver ?

Comment faire le lien ?

Des mots et des tournures sont utilisés dans les raisonnements et dans les algorithmes mais avec un sens différent. Comment va-t-on s'y retrouver ?

Si les langages se ressemblent, ce n'est pas par hasard !
On peut faire le lien de plusieurs façons :

- démontrer la correction d'un algorithme
- extraire un algorithme d'une démonstration
- comprendre un énoncé logique par son aspect opératoire

Correction des algorithmes

Démontrer qu'un algorithme est correct consiste à

- donner une spécification
- établir des propriétés des variables à chaque étape
- établir que le calcul termine

Ce qui revient à montrer l'existence d'un résultat ayant les propriétés voulues pour des données correspondant aux conditions initiales.

- la structure de l'algorithme guide celle de la démonstration,
- l'argument de terminaison justifie que les raisonnements sont valides (récurrences bien formées, pas d'arguments circulaires).

Réciproquement, une démonstration contient un procédé permettant de calculer un témoin du résultat démontré.

Un exemple en analyse

Théorème

La suite $\left(1+\frac{a}{n}\right)^n$ *tend vers* e^a.

Une démonstration va en fin de compte se ramener à établir que

pour toute distance ε
il existe un rang N
à partir duquel on a toujours $\left|\left(1+\frac{a}{n}\right)^n - e^a\right| \leq \varepsilon$.

c'est la spécification d'un algorithme qui sert à calculer des approximations de e^a.

Correspondance de Curry-Howard

La proximité entre algorithme et démonstration est justifiée d'un point de vue théorique par un correspondance formelle : l'isomorphisme de Curry-Howard

logique	calcul
démonstration	programme
axiome	opération élémentaire
règle de déduction	structure de contrôle
récurrence	itération
normalisation	évaluation
lemme	sous-programme
théorie	interface de programmation

Initialement entre logique intuitionniste et calcul fonctionnel, largement étendue depuis et avec de nombreuses retombées théoriques et pratiques.

Tentative de conclusion

Démonstration et algorithme peuvent être vus comme deux visions complémentaires d'une même construction abstraite.

- Pensée descriptive : s'attacher aux propriétés des objets, aux conséquences logiques. La construction d'une démonstration réintroduit une dynamique de nature algorithmique dans le discours.
- Pensée opératoire : s'attacher à la méthode de construction d'une réponse à une question. La justification de la méthode réintroduit un aspect descriptif qui permet le raisonnement générique.

La proximité des dialectes n'est pas fortuite, même s'il s'agit de dialectes distincts du langage mathématique.