Les quatre concepts de l'informatique

Gilles Dowek

▶ **To cite this version:**

Gilles Dowek. Les quatre concepts de l'informatique. 2011. hal-04048556

HAL Id: hal-04048556
https://inria.hal.science/hal-04048556

Preprint submitted on 28 Mar 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Les quatre concepts de l'informatique

Gilles Dowek

INRIA

23 avenue d'Italie, CS 81321, 75214 Paris Cedex 13, France
gilles.dowek@inria.fr

À la question « Qu'est ce que l'informatique ? », nous sommes souvent tentés de donner une réponse de la forme « L'informatique est ... ». Ainsi, l'informatique est souvent définie comme la science des algorithmes, la technique du traitement de l'information, la partie constructive des mathématiques, une branche de l'électronique, une physique du signe, ... Toutes ces réponses sont imparfaites, car partielles. Au lieu de tenter une réponse de la forme « L'informatique est ... », il semble plus sage de tenter des réponses de la forme « En informatique, il y a ... ». Plusieurs réponses peuvent alors contribuer à énumérer les différents constituants de l'informatique, jusqu'à ce que suffisamment de constituants soient énumérés pour couvrir l'extension courante du mot « informatique ». Il est alors possible, dans une seconde étape, de chercher les relations entre ces différents constituants, qui forment la structure de ce champ de la connaissance.

Ces constituants peuvent être des branches de l'informatique. Ainsi, on peut dire qu'en informatique il y a la théorie des langages de programmation, la théorie des bases de données, ... mais il semble plus pertinent d'énumérer, non les sous-disciplines, mais les concepts utilisés dans ce domaine de la connaissance, qui ont sans doute une stabilité temporelle plus grande que les diverses sous-disciplines qui se recomposent sans cesse. Les relations entre ces différents concepts sont également sans doute plus révélatrices de la structure profonde de l'informatique, que les relations entre ces diverses sous-disciplines.

Quatre concepts – algorithme, machine, langage et information – semblent suffisants pour couvrir l'ensemble de ce que nous appelons « informatique ».

Le concept d'algorithme

Le concept d'algorithme est le plus ancien, puisque 2500 ans avant notre ère, les comptables utilisaient déjà des algorithmes pour effectuer les quatre opérations, calculer des prêts, des héritages, ... et les arpenteurs pour calculer l'aire de surfaces agricoles.

Ce concept d'algorithme n'est pas propre à l'informatique, puisqu'il est également utilisé en mathématiques, où existent de nombreux algorithmes pour résoudre des équations de manière exacte ou approchée, dériver et intégrer des expressions fonctionnelles, calculer la probabilité de certains événements, … Si on fait démarrer l'informatique dans les années 1930, ce concept est donc bien antérieur à l'informatique, mais si on fait démarrer les mathématiques au V^{e} siècle avant notre ère, il est également antérieur aux mathématiques.

Un algorithme est une recette qui permet de résoudre un certain problème de manière systématique.

Un exemple paradigmatique est la recette de la tarte aux pommes, qui permet de résoudre un problème : faire une tarte aux pommes. Toutefois, dans cet exemple, il est important de distinguer la recette en tant que texte de la recette en tant que pratique. Une recette peut être exécutée et même être transmise de génération en génération, sans être écrite, ni même verbalisée. C'est cette seconde notion qui correspond à la notion d'algorithme. Dès que la recette est écrite ou verbalisée, elle doit être comparée, non à la notion d'algorithme, mais à celle de programme.

L'égalité entre deux algorithmes

La notion d'algorithme est plus grossière que la notion de programme : plusieurs programmes peuvent correspondre au même algorithme, mais elle est moins grossière que celle de problème : un même problème – par exemple, trier une liste – peut-être résolu par plusieurs algorithmes.

Ces deux remarques mènent à une question qui n'est pas encore résolue aujourd'hui : quand peut-on dire que deux algorithmes sont identiques ? Clairement, s'il sont exprimés par le même texte, ils sont identiques, s'il ne résolvent pas le même problème, il sont différents. Mais, entre ces deux extrêmes, il règne un certain flou. On s'accorde en général à penser que si deux algorithmes n'ont pas la même complexité – s'ils demandent un temps différent pour résoudre le même problème – ils sont différents, mais cette caractérisation est encore trop grossière : divers algorithmes de tri, quoique de même complexité, peuvent être différents.

Les algorithmes non déterministes

Une idée largement répandue, mais fausse, est qu'un algorithme est nécessairement déterministe : il ne laisse aucune place à l'aléa. Au contraire, de nombreux algorithmes ont besoin de l'aléa pour fonctionner.

Par exemple, si deux utilisateurs du téléphone s'appellent au même instant, chacun sera dirigé vers la messagerie de l'autre. S'ils réitèrent leur appel après un temps fixe, le même phénomène se reproduira. En revanche, s'ils réitèrent leur appel après un temps aléatoire, ils réussiront presque sûrement à se parler.

Ce sur quoi les algorithmes opèrent

Un algorithme opère sur quelque chose : par exemple la recette de la tarte aux pommes opère sur des ingrédients : les œufs, la farine, ... Les algorithmes mathématiques opèrent toujours sur des objets mathématiques : des nombres, des polynômes, des matrices, des équations, ... Mais la notion d'algorithme est plus vaste, puisque les algorithmes peuvent opérer sur des données beaucoup plus variées : des arbres, des graphes, d'autres algorithmes, des images, des sons, des textes, ...

Un texte est une suite finie de symboles appartenant à un alphabet fini. Une telle donnée peut être qualifiée de « symbolique ». Une matrice, un polynôme, ... peuvent être représentés de manière symbolique. Une idée largement répandue, mais fausse, est qu'un algorithme opère toujours sur des données symboliques. C'est, encore une fois, une vision trop restrictive : de nombreux algorithmes, par exemple la recette de la tarte aux pommes, opèrent sur des données qui ne sont pas symboliques : des œufs, de la farine, …

Le concept de machine

Pendant 4500 ans, nous avons conçu des algorithmes, pour les exécuter « à la main », mais depuis quelques décennies, nous utilisons pour cela des outils. Une telle utilisation d'outils semble naturelle, puisqu'un algorithme est fait pour être exécuté « sans réfléchir » et c'est ainsi que, depuis l'Antiquité, nous avons utilisé des abaques – baguettes à calculer, bouliers, échiquiers, ... – et des machines mécaniques, avant que nous ayons les connaissances techniques nécessaires pour construire les outils les plus courants aujourd'hui : les ordinateurs.

La diversité des machines

Une machine est un outil, c'est-à-dire un système matériel, qui obéit donc aux lois de la physique. Les machines comprennent les ordinateurs, des appareils plus spécialisés comme les appareils photos ou les téléphones, et plus généralement tous les systèmes physiques, pour lequel nous avons défini un protocole d'interaction, qui nous permet d'échanger des données.

Ainsi, nous pouvons imaginer une machine dans laquelle nous laissons tomber une balle dans le vide pendant un temps t et mesurons la distance parcourue par la balle. Cette machine effectue un calcul particulier : elle élève le nombre t au carré. Le même système physique, muni d'un autre protocole, effectue un autre calcul : si nous mesurons la durée nécessaire à la balle pour parcourir une distance d, nous obtenons une machine qui calcule la racine carrée du nombre d.

Aujourd'hui, de nombreuses machines exploitent les propriétés physiques des semi-conducteurs, mais rien n'impose que cela soit toujours le cas et la recherche en informatique explore de nombreuses alternatives, en tentant, en particulier, d'exploiter plus systématiquement les possibilités de la physique quantique, ou de s'inspirer des processus du vivant.

La puissance de calcul et les principes de la physique

Le fait que les machines obéissent aux lois de la physique limite leur puissance : les machines pourraient calculer beaucoup plus de choses si l'information pouvait voyager instantanément ou s'il était possible de stocker une quantité infinie d'information dans un volume fini.

De manière moins spéculative, la physique du calcul établit un lien entre la réversibilité d'un calcul, c'est-à-dire la possibilité de retrouver les valeurs d'entrée de l'algorithme à partir de son résultat et la réversibilité de l'évolution du système physique qui effectue ce calcul : quand une machine effectue un calcul irréversible, le processus physique lui-même est irréversible, et il dissipe donc de la chaleur. Le lien entre l'énergie dissipée sous forme de chaleur et l'information perdue peut même être quantifié : effacer un bit d'information dissipe, sous forme de chaleur, au moins une énergie $k\,T\,\ln(2)$, où k est la constante de Boltzmann, et augmente donc l'entropie au moins de $k\ln(2)$.

L'espace et les réseaux

Comme tout système physique, une machine peut avoir une extension plus ou moins grande dans l'espace et, à l'ordinateur de bureau, s'est peu à peu substitué une autre machine, beaucoup plus étendue, constituée de milliards d'ordinateurs interconnectés en réseau. Après l'ordinateur, le réseau est donc une deuxième instance de ce concept de machine, qui donne à cette question d'extension dans l'espace une place essentielle. On peut ici parler de « géométrie du calcul » quand on se pose la

question de la manière de répartir un calcul sur différents ordinateurs situés aux quatre coins du globe ou sur la manière de transmettre un message d'un point à un autre.

Les ordinateurs

Nous avons évoqué deux machines, qui, pour la première, calcule le carré d'un nombre et, pour la seconde, calcule sa racine carrée. Chacune de ces machines ne permet de résoudre qu'un seul problème : calculer un carré, calculer une racine carrée. Certaines machines polyvalentes, comme les calculettes de poche, permettent de résoudre plusieurs problèmes : calculer une somme, une différence, un produit, un quotient. Néanmoins ce type de machines ne peut résoudre qu'un nombre fini de problèmes : même un couteau de l'armée suisse n'a qu'un nombre fini de lames.

D'où une idée, ancienne, de concevoir des machines paramétrables, comme les orgues de barbarie, dont on peut changer la mélodie, ou les métiers à tisser, dont on peut changer le motif, en changeant de cartes perforées.

Cette évolution a abouti à des machines universelles qui peuvent exécuter n'importe quel algorithme opérant sur des données symboliques, pourvu qu'on les munisse des bonnes « cartes perforées », du bon programme : les ordinateurs.

Les machines parallèles et spécialisées

La construction de machines universelles n'a cependant pas achevé l'histoire de la construction des machines, puisque de nombreuses améliorations peuvent encore leur être apportées. En particulier, il est possibles de grouper ensemble plusieurs machines, ou bien dispersées en divers endroits et donc en réseau, ou bien situées à quelques mètres les unes des autres, en grille, ou bien situées sur un même circuit intégré. Quand plusieurs machines sont ainsi reliées, elles peuvent l'être de manière synchrone, ce qui signifie que leurs horloges battent au même rythme ou de manière asynchrone, auquel cas l'exécution de l'ensemble des machines est nécessairement non déterministe, l'aléa étant introduit par la dérive des horloges, qui est inconnue.

D'autres machines, plus spécialisées, sont utilisées dans de nombreux appareils – téléphones, appareils photos, avions, ... – on parle alors de « systèmes embarqués ».

Le concept de langage

Une recette de cuisine, un algorithme pour traverser la rue, peuvent s'exécuter et même se transmettre sans être écrits ou verbalisés. Ainsi, l'algorithme que nos cellules utilisent pour synthétiser une protéine à partir d'un brin d'ARN messager n'a – pour autant que nous sachions – pas été écrit, ni verbalisé, avant d'être exécuté dans les premières cellules, puis transmis de génération en génération.

Cependant, nous pouvons aussi écrire ces algorithmes. Et les écrire une étape nécessaire pour les communiquer à une machine paramétrable. Pour cela nous avons besoin d'un langage. Un pâtissier n'a pas besoin de connaître les verbes « éplucher », « mélanger », « étaler », ... pour exécuter la recette de la tarte aux pommes, mais il a besoin de créer, ou d'apprendre, ces mots, s'il veut la verbaliser et l'écrire.

Les langages de programmation

Pour écrire les algorithmes, nous pouvons utiliser une langue naturelle comme le français ou alors utiliser un langage plus simple et plus précis : un langage de programmation. Décrire un algorithme dans un langage de programmation est aujourd'hui indispensable pour que cet algorithme soit exécuté par une machine paramétrable, car nous ne savons pas encore fabriquer de machines qui exécutent des algorithmes exprimés dans une langue naturelle.

Toutefois, il est important de remarquer que les premiers langages de programmation universels – le lambda-calcul, le langage des machines de Turing – sont antérieurs de quelques années à la construction des premières machines universelles, les premiers ordinateurs. En effet, écrire les algorithmes dans un langage de programmation est aussi un moyen de l'exprimer clairement et de le communiquer à d'autres. C'est aussi un moyen de lui donner une forme symbolique et de permettre à d'autres algorithmes, qui opèrent sur des données symboliques, d'opérer sur lui.

Langue naturelle et langage formel

La démarcation entre en une langue naturelle et un langage formel est difficile à définir. Toutefois, nous pouvons l'esquisser en remarquant que la plupart des usagers d'une langue naturelle n'en ont appris la grammaire qu'après avoir appris la langue elle-même : un enfant sait utiliser l'imparfait avant de connaître le mot « imparfait », autrement dit, il connaît l'algorithme de la conjugaison, avant de savoir le verbaliser. À l'inverse, on apprend un langage formel en apprenant d'abord sa grammaire. De même, la grammaire d'une langue naturelle ne semble jamais complètement connue, comme en témoigne la difficulté de faire des programmes de correction orthographique ou de traduction automatique. La grammaire d'un langage formel est, en revanche, explicite. Enfin, et cela explique sans doute les deux remarques ci-avant, la grammaire d'un langage formel est souvent beaucoup plus simple que celle d'une langue naturelle.

Bien entendu ces remarques établissent une démarcation un peu floue, puisqu'elle concernent davantage la manière dont nous connaissons ces langages que ces langages eux-mêmes. De ce fait, un certain nombre d'exemples peuvent être à la frontière. Par exemple, la manière dont on apprend une langue étrangère, et *a fortiori* une langue morte, se rapproche parfois davantage de la manière dont on apprend un langage formel, que de la manière dont on apprend sa langue maternelle.

Au delà des langages de programmation

À coté des langages de programmation, il y a de nombreux autres langages formels. Par exemple, des langages pour spécifier des programmes, des langages pour raisonner sur des programmes, des langages pour exprimer des requêtes dans une base de donnée, ... Et un certain nombre d'algorithmes sont communs à tous ces langages. Par exemple, décider si une phrase est bien formée ou non dans un langage demande d'utiliser un algorithme qui est indépendant du fait que ce langage est un langage de programmation, de spécification, de requête, ...

Si bien que l'informatique introduit une notion très générale de langage, dont les langues naturelles, les langage de programmation, mais aussi le langage mathématique ou les langages de la logique, sont des cas particuliers. En tant que science des langages, l'informatique prolonge les travaux plus anciens de la logique formelle.

Le concept d'information

Le concept de langage est apparu en informatique, essentiellement parce que, pour communiquer des algorithmes à des machines, nous devons utiliser un langage formel. Ainsi, les algorithmes sont devenus des programmes. De même, alors qu'un algorithme peut opérer avec des données variées : des images, des sons, des œufs, de la farine, ... un algorithme, exécuté par une machine, opère souvent avec des données représentées de manière symbolique.

Représenter les données de manière symbolique

Ainsi, pour être traitée par un ordinateur, une image est souvent pixellisée. Pour pixelliser une image en noir et blanc, par exemple, on la superpose avec une grille faite de quelques millions de petits carrés et chaque carré est décrété noir ou blanc selon que le noir ou le blanc domine dans cette petite partie de l'image. On obtient ainsi une approximation de l'image : par exemple, de minuscules détails plus petit que la taille d'un carré sont perdus. Mais cette image peut être représentée de manière symbolique : il suffit d'écrire la valeur de chacun des carrés, en commençant en haut à gauche et en terminant en bas à droite : blanc, blanc, noir, noir, blanc, noir, blanc, blanc, ...

On parle souvent de « numérisation » pour désigner cette opération, car en notant 0 pour un carré noir et 1 pour un carré blanc, on peut écrire cette suite 11001011... et donc, finalement, comme un nombre entier représenté en base deux. Cependant, ce n'est pas le fait que l'information soit exprimée sous forme d'un entier qui est importante, c'est le fait qu'elle soit exprimée de manière symbolique.

Compresser, chiffrer

Représenter des sons, des images, ... en utilisant le même alphabet – typiquement les symboles 0 et 1 – permet d'utiliser des techniques similaires pour compresser ou chiffrer l'information. Par exemple, une suite formée de mille deux cent quatre vingt seize fois le symbole 0 se note de manière plus brève (1296)0 que 000 ... De même, en groupant dans une suite de 0 et de 1, les symboles par deux et en remplaçant le groupe 00 par *a*, le groupe 01 par *b*, le groupe 10 par *c* et le groupe 11 par *d*, on obtient une suite deux fois plus courte mais dans un alphabet à quatre lettres. Si les *c* sont fréquents dans cette suite, mais que les *b* et les *d* sont rares, on peut décider de recoder *c* par la suite 0 et *a*, *b* et *d* par les suites 10, 110 et 111 on obtient alors une suite plus courte que la suite initiale, à partir de laquelle on peut cependant la reconstruire.

Ultimement, on peut se demander quelle est la manière optimale de coder une longue suite, et c'est souvent sous la forme d'un programme qui engendre cette suite quand on l'exécute. La taille du plus court programme qui engendre un message est une manière de définir la *quantité d'information* contenue dans ce message. Cette théorie de la compression aboutit donc à une théorie quantitative de l'information.

Des données aux bases de données

L'émergence de cette notion d'information nous a aussi permis de prendre conscience du fait que, dans bien des cas, nous utilisons les machines, non pour transformer de l'information en lui appliquant des algorithmes, mais uniquement pour l'archiver.

La réflexion sur la manière d'archiver l'information et surtout de la restituer a donné naissance à la théorie des bases de données, théorie qui s'est grandement renouvelée avec l'apparition du web, le web étant lui-même une base de données.

L'unité de l'informatique

Ces quatre concepts, loin de définir quatre branches de l'informatique, sont souvent utilisés ensemble. C'est parce que nous voulons faire exécuter des algorithmes par des machines paramétrables que nous avons eu besoin d'exprimer ces algorithmes dans des langages formels et les données sur lesquelles ces algorithmes calculent comme des informations.

Les notions d'algorithme et de machine sont unies par l'idée d'exécution « sans réfléchir », qui a trouvé ses premières applications avec les abaques de l'Antiquité.

À la frontière des notions d'algorithme et de langage apparaît la notion de programme.

À la frontière des notions de langage et de machine apparaît la notion de langage machine. Ces langages sont des langages de programmation si simples que l'on arrive à concevoir des machines électroniques capables d'en exécuter directement les programmes et vers lequel on arrive à traduire – compiler –, souvent en plusieurs passes, les programmes écrits dans les langages de plus haut niveau, c'est-à-dire plus ergonomiques.

Enfin les notions de langage et d'information sont liées car ce sont deux instances d'une même idée : exprimer symboliquement des algorithmes, pour l'une, des données, pour l'autre.

La pensée informatique

Ces quatre concepts définissent également les différents modes d'interaction entre l'informatique et les autres sciences.

L'informatique propose de décrire de nombreux objets, en mathématiques et dans les sciences de la nature, comme des algorithmes. Et elle propose des langages pour le faire. Le concept d'information éclaire d'une lumière nouvelle des questions de physique ou de biologie, par exemple quand nous disons que l'information ne peut pas voyager plus vite que la vitesse de la lumière pour exprimer qu'un événement ne peut pas être la cause d'un autre s'ils ne sont pas séparés par une durée supérieure à leur distance divisée par la vitesse de la lumière ou quand nous disons que l'information nécessaire pour synthétiser les protéines est codée dans l'ARN. La notion de machine enfin apparaît dans le renouvellement de l'instrumentation scientifique.

Ce soucis du caractère algorithmique de la description des objets, du langage dans lequel ces descriptions sont exprimées, des flux d'information et des instruments sont plus généralement caractéristiques d'une « pensée informatique ». Cette pensée a bien entendu de nombreux précurseurs dans l'histoire, comme Galilée qui a à la fois introduit l'utilisation d'instruments – la lunette – et proposé d'utiliser un nouveau langage – le langage mathématique – dans les sciences de la nature, mais ce qui semble nouveau, en revanche, est de prêter attention simultanément, dans un même geste, à ces quatre questions.

La singularité de l'informatique

Imaginons qu'au début du XX^{e} siècle, un mathématicien spécialiste de calcul des probabilités, un comptable, un physicien spécialiste des évolutions irréversibles des systèmes, un ingénieur concepteur de métiers à tisser, un grammairien, un traducteur de textes anciens, un agent des services du chiffre et un archiviste se trouvent coincés dans un ascenseur en panne. Qu'auraient-ils eu à se dire ?

Sans doute pas grand chose, tant le concepts d'algorithme – spécialité du le mathématicien et du comptable – de machine – spécialité du physicien et de l'ingénieur – de langage – spécialité du grammairien et du traducteur – et d'information – spécialité de l'archiviste et de l'agent du chiffre – appartenaient à des mondes différents, non seulement à des sciences différentes : les mathématiques pour la notion d'algorithme, la physique pour la notion de machine, la linguistique pour la notion de langage, mais aussi à des domaines techniques – la comptabilité, le génie mécanique, le chiffre, ... – qui étaient perçus comme extérieurs aux sciences.

L'événement qu'a constitué notre prise de conscience du fait que ces quatre concepts formaient quatre pièces d'un puzzle qui s'assemblent parfaitement, et la transformation de l'architectonique des sciences qu'elle a impliqué, était un événement *a priori* très peu probable, un événement comme il y en a peu dans l'histoire.