Raisonner avec les machines

Henri Stéphanou

Raisonner avec les machines

Philosophie de l'ordinateur

puf

ISBN 978-2-13-088946-5

Dépôt légal – 1re édition : 2026, avril

170 bis, boulevard du Montparnasse, 75014 Paris

Introduction

Polyvalence des ordinateurs

Voici comment l'informaticien Anatol Holt se remémore sa première rencontre avec les ordinateurs en 1952 :

> J'ai été frappé par une énigme. Comme tant d'autres technologies, l'ordinateur était un enfant de la guerre. Dans ce contexte, il était censé aider à effectuer des calculs extrêmement fastidieux, qui conduisaient à des nombres, et par là à une plus grande puissance de feu contre l'ennemi. Il était censé calculer (*to compute* en anglais), c'est pourquoi on l'appelait un ordinateur (*computer*). Cependant, très peu de clients potentiels de cette nouvelle machine avaient des problèmes de ce type. Ils l'utilisaient plutôt pour trier des données, contrôler le trafic ferroviaire, émettre des chèques de paie, etc. Dans mon jeune esprit, cela soulevait une question brûlante : si l'ordinateur ne sert pas vraiment – ou principalement – à calculer, alors, à quoi sert-il[1] ?

L'énigme de Holt est aussi pertinente aujourd'hui qu'elle l'était il y a soixante-dix ans. Depuis, les ordinateurs ont réussi à se rendre utiles, voire indispensables, à un éventail

de pratiques humaines bien plus large que les affaires militaires et commerciales que Holt mentionne. Les ordinateurs sont partout, derrière chaque écran, dans la poche de chacun, intégrés dans chaque machine et même dans les objets eux-mêmes, à tel point que les historiens de l'informatique Haigh et Ceruzzi les qualifient de « technologie dissolvante[2] ». Et ils ne se contentent pas de rendre nos pratiques plus efficaces ou plus économiques. Ils sont en train de *transformer* les affaires, les sciences, la culture, les relations sociales, voire la politique et le droit.

La singularité de l'ordinateur est manifeste lorsqu'on le compare aux autres grandes inventions des cent cinquante dernières années, l'électricité, le téléphone, la voiture, par exemple. S'il est aisé de déterminer le domaine d'utilité de chacune – l'énergie, la communication, le transport –, que dire de celui de l'ordinateur ? Son nom américain (*computer*) se réfère au calcul, les termes français *ordinateur* et *informatique* évoquent l'ordre et l'information. Ces notions n'ont pas de rapport évident entre elles. La référence à la notion d'information rapproche l'informatique des télécommunications, mais l'ordinateur fait apparemment bien plus de choses que transmettre des messages. Il semble opérer sur un autre plan que toutes ces inventions.

On peut avancer que le calcul est, de manière invisible, sous-jacent à de nombreuses activités humaines. On évoque souvent, à ce titre, la quantification effrénée de toutes choses qui a saisi l'Occident depuis le XVIII^e^ siècle, avec l'essor notamment des statistiques. Les ouvrages sur la question ont des titres évocateurs : *Quand le monde s'est fait nombre, La Gouvernance par les nombres, La Politique des grands nombres*[3]. La quantification des phénomènes naturels et

sociaux s'étant généralisée, et les ordinateurs étant de grands calculateurs, il n'est pas étonnant qu'ils deviennent des instruments privilégiés de la prise de décision.

La liaison entre quantification et informatique n'est néanmoins pas si évidente que cela. Une fabrication industrielle fait appel, certes, à des nombres qui représentent les dimensions de l'objet, la quantité de matière à injecter, la température d'injection, mais elle ne se laisse pas réduire à cela. Une entreprise a certes besoin d'une comptabilité exacte, qui détaille la moindre de ses opérations, mais ces chiffres ne représentent que le fantôme de son activité. Ces données n'expliquent pas comment il est possible d'automatiser la fabrication de produits ou des processus administratifs.

On peut proposer d'élargir ce qu'il faut entendre par quantification : il ne s'agit pas seulement de *mesurer* les phénomènes ainsi que leurs occurrences et leurs probabilités, mais également de les *énumérer* de manière exhaustive, de les mettre en listes et en tables. Ici le nombre n'est qu'un instrument de *codage de l'information*, et c'est donc en tant qu'ils sont des machines de *traitement de l'information* que les ordinateurs interviennent dans le monde. Cette interprétation éclaire mieux les exemples donnés par Holt : tri de données, émission de chèques, régulation ferroviaire. Elle met en avant la dimension formelle de ces activités. Elles sont bien des calculs, si l'on entend ce terme en un sens plus large, comme manipulation de symboles quelconques selon des règles rigoureuses, qui exigent une application sans faille et sans initiative.

Cependant, comment expliquer que tant de pratiques humaines, aussi variées, aussi variables, puissent être l'objet

d'une telle description par des règles rigoureuses ? Est-ce parce qu'elles ont essentiellement la forme de plans d'action, de procédures, que les informaticiens ne feraient que retranscrire dans un langage compréhensible par l'ordinateur ? Y aurait-il ainsi un langage générique capable de décrire toute action ?

Si l'on appelle *programmation* cette transcription, aussi problématique soit-elle à ce stade, un progrès est néanmoins accompli. On voit que les ordinateurs se distinguent des autres machines en ce qu'ils sont *programmés* pour répondre aux différents besoins qui les sollicitent. Ils sont des artefacts essentiellement *incomplets* sans les logiciels ou applications qui déterminent leur fonctionnement particulier, dans tel ou tel contexte. C'est cette incomplétude qui explique leur polyvalence (nous évitons pour l'instant le terme *universalité* car nous ne présupposons pas que les ordinateurs peuvent s'adapter à *tout* besoin ; nous discuterons à plusieurs reprises dans l'ouvrage en quel sens la dénomination de l'ordinateur comme *machine universelle* est pertinente). C'est ce fait essentiel et aisément observable de la programmation qui doit donc nous interroger.

Effectivité des ordinateurs

Cette question en appelle une autre. Que l'on parvienne à décrire un grand nombre d'activités humaines dans un langage générique des procédures est une chose, que ce langage soit effectif dans la réalité, sans l'intermédiaire d'un agent humain qui l'interprète en agissant en est une autre.

Répondre que c'est possible parce que justement le calcul exige une application « mécanique » des règles semble jouer sur les mots, et fournir une métaphore au lieu d'une explication. Que pourrait vouloir dire en effet pour une machine d'« appliquer des règles » ? Comment est-il concevable qu'un ensemble de formules symboliques, ou une suite de 0 et de 1 – ce à quoi se réduit ultimement le programme d'un ordinateur – puisse s'animer de manière à accomplir des fonctions signifiantes dans le monde, comme piloter un avion, ou même simplement déterminer l'affichage d'un écran ? Une réponse courante est d'évoquer la découverte par Claude Shannon, de la correspondance entre les opérateurs logiques élémentaires (conjonction, disjonction, négation, etc.) et certaines configurations des circuits électriques (interrupteurs en série, en parallèle, inversés)[4]. Cependant, il n'est pas du tout évident que le pilotage d'un avion puisse être réduit à un assemblage d'opérations logiques !

La possibilité d'une telle effectivité des significations – la possibilité de l'ordinateur tout court – n'est pas claire. Ainsi, par exemple, l'informaticien Tim Colburn, dans sa réflexion sur la nature de l'informatique, se résigne à faire appel au schéma leibnizien de l'harmonie préétablie entre l'âme et le corps pour expliquer l'accord merveilleux entre les représentations abstraites des programmeurs et le comportement matériel de l'ordinateur – le Dieu horloger étant ici remplacé par la longue chaîne anonyme des informaticiens ayant développé les compilateurs*[1], les systèmes d'exploitation* et les circuits intégrés* qui établissent cette liaison[5]. Cependant,

1. Le symbole (*) marque un terme technique explicité dans le glossaire en fin d'ouvrage. Cet astérisque n'apparaît qu'à la première occurrence.

on ne voit pas clairement comment une chaîne de traducteurs, à partir d'un texte initial, produirait autre chose que des traductions, c'est-à-dire d'autres textes. Que font donc ces programmeurs pour établir une telle harmonie, s'ils ne font eux-mêmes que manipuler des significations ?

Notre seconde question est donc tout simplement de comprendre comment des significations peuvent devenir *effectives* dans le monde, sans l'intermédiaire direct d'un interprétant humain. Cela exige de clarifier en quel sens nous comprenons ici *effectif*.

Selon les dictionnaires Robert et Larousse, ce terme a d'abord le sens très générique de « réel, tangible, positif ». Ce n'est pas en ce sens que nous souhaitons diriger notre questionnement, car cela nous obligerait à expliciter ce qu'il faut entendre par *monde*, *réalité*, etc. Il est peu probable que notre question, qui concerne un phénomène empirique et historique – les ordinateurs et leur rôle dans le monde humain –, doive recevoir des réponses différentes selon le cadre métaphysique dans lequel on se place.

Le Larousse marque également deux autres sens d'*effectif*. Il y a d'abord celui de « prendre effet » comme dans : « L'armistice sera effectif à 11 heures ». C'est le sens principal qui nous intéresse ici. Les ordinateurs aident *effectivement* les personnes à résoudre des problèmes auxquels elles sont confrontées, comme traiter des factures, contrôler l'accès à un bâtiment, piloter une sonde spatiale. Ils permettent à leurs demandes et à leurs spécifications de « prendre effet ».

En ce sens, l'effectivité se rapproche d'une compétence ou d'une capacité. Nous évitons le terme de *puissance* et des termes équivalents souvent employés pour décrire,

et parfois dénoncer, l'ampleur de la transition numérique : *L'Hyperpuissance de l'informatique, L'Empire numérique, La Siliconisation du monde* sont des titres d'ouvrages récents[6] qui évoquent un « règne » ou une domination de l'informatique. Notre ouvrage n'a pas l'ambition d'analyser et de critiquer les effets anthropologiques du numérique ; la discussion sur ces sujets est déjà dense. Nous visons plutôt à éclairer la possibilité même de tels effets, c'est-à-dire à montrer qu'ils relèvent, au-delà du progrès technique et mathématique, d'une option fondamentale (mais non nécessaire) de notre capacité à connaître et à agir.

Pour remplir cet objectif, la notion d'effectivité nous semble adéquate, étant plus « modeste » que celle de puissance ou de règne. Contrairement à ces termes, elle est *locale* et circonscrite au problème dont elle s'occupe. Elle désigne la propriété d'un objet servant de manière essentielle à effectuer *ce qui est demandé*. Cette formulation ne présuppose pas que ces problèmes soient les bons, ni qu'ils augmentent le pouvoir des êtres humains sur le monde, le bien-être individuel ou l'utilité générale. On peut utiliser les ordinateurs pour résoudre de mauvais problèmes, ou des problèmes insignifiants. Pis encore, il se peut que les ordinateurs induisent les personnes à se poser des problèmes qu'elles ne se seraient jamais posés sans leur présence, ou à les poser d'une manière qui rendent ces machines indispensables.

Si donc la programmation est une activité de résolution de problème à l'aide d'une machine, en quel sens faut-il entendre ici *problème* ? Nous ne limitons pas le champ de cette notion aux problèmes intellectuels auxquels s'est d'abord intéressée l'informatique : problèmes logiques et

mathématiques, jeux, énigmes. *Problème* doit être pris dans un sens bien plus général, celui de la représentation discursive d'une situation d'insatisfaction[7]. Néanmoins, nous ne prenons pas ici le terme dans son acception la plus grande, et ne considérons pas les larges classes de problèmes qu'on appelle *philosophiques, éthiques, esthétiques*, etc. – où les notions même de *résolution* et d'*effectivité*, au sens où nous les entendons ici, sont généralement considérées comme non pertinentes.

L'effectivité comme « prise d'effet » est donc relative à une demande, à une prescription, à un problème spécifique. La réalité du comptable peut être différente de celle de l'astronaute et de celle du mathématicien, et chacun d'eux exprime ses problèmes dans un langage spécialisé. La question de l'effectivité des significations se pose à ce niveau local, elle concerne la manière par laquelle un état de choses désiré, *tel qu'il est décrit dans le langage pertinent*, peut être réalisé de manière à rendre valide cette description.

Cette interprétation est confirmée par le troisième sens d'*effectif* indiqué par le Larousse, qui atteste son usage technique en informatique : « Se dit d'une méthode qui, appliquée à la résolution d'un problème, en donne la solution au bout d'un nombre fini d'étapes (Une opération est dite effective s'il existe un algorithme permettant de la calculer)[8] ». Nous aurons amplement l'occasion de revenir sur cette dernière définition dans le cours de ce travail ; son articulation avec le sens plus général d'*effectif* (« prendre effet ») apparaîtra progressivement, ainsi qu'avec le mot *algorithme* qui apparaît dans la définition citée. En informatique théorique, en effet, *algorithme* est synonyme de *procédure effective*. Nous

emploierons de préférence cette seconde expression, du fait de l'attention que nous portons à chacun des deux termes qui la composent.

L'explication technique

On pourrait objecter que l'on se pose de faux problèmes. L'effectivité des ordinateurs est une réalité technique et doit donc être expliquée techniquement. Pour expliquer la diversité des applications de l'informatique, le mieux n'est-il pas tout simplement d'étudier les programmes sous-jacents à chacune, ligne à ligne et pas à pas, ou encore les circuits intégrés qui les implémentent ?

Chaque phénomène informatique individuel, qu'il s'agisse du contrôle d'un bras robotique, de celui d'un serveur vocal interactif, ou d'une simulation scientifique, est en effet parfaitement explicable, du moment que l'on peut accéder à l'organisation logique des programmes qui le pilotent et au code individuel de ceux-ci. Il n'y a même rien de plus explicable que cela, *en principe*, si l'on y consacre le temps nécessaire. De simples raisonnements logiques permettent de rendre compte de ce qui se passe, pourvu que les termes employés par le code des programmes soient définis, et que les hypothèses concernant le comportement des composants matériels de l'ordinateur soient spécifiées. Certes, la vitesse et la quantité des opérations effectuées rendent de telles explications impossibles pratiquement ; mais la possibilité de les réaliser sur n'importe quelle partie du système

proportionnée à nos capacités doit nous convaincre qu'il n'y a nul mystère derrière tout cela.

En d'autres termes, nos questions ne se dissipent-elles pas sous le regard avisé du programmeur, qui est capable d'analyser et de comprendre les programmes informatiques, de la même manière qu'un ingénieur est capable d'expliquer pourquoi un pont tient debout ? L'étonnement du grand public serait ainsi celui de néophytes qui, ne pouvant expliquer des phénomènes du fait d'un bagage technique insuffisant, les rapportent de ce fait à la magie.

Il y a deux problèmes à cette fin de non-recevoir. D'abord, posséder l'explication d'un programme ne suffit pas toujours à dissiper l'étonnement concernant ses résultats. De manière régulière, les programmeurs ont fait part de leur *propre* étonnement ou fascination vis-à-vis de leur activité, comme si celle-ci était dotée d'une opacité propre qui résistait aux simples explications de la raison[9].

Ensuite, il n'est pas évident de circonscrire le « bagage technique » de la programmation. S'agit-il de comprendre comment fonctionnent les ordinateurs, d'avoir des rudiments d'électronique ou faut-il connaître au moins un langage de programmation ? Ces savoirs, s'ils sont utiles, ne sont pas essentiels. Comme nous l'avons dit plus haut, la simple logique est souvent suffisante pour comprendre le fonctionnement d'un programme, du moment que l'on se laisse expliquer le sens des termes qu'il utilise. On a ainsi avancé que la programmation n'était pas une discipline technique mais mathématique ou logique.

La logique n'est cependant pas suffisante, car programmer ne requiert pas seulement de coder, mais également de décrire un problème concret en termes informatiques afin

de rendre le codage possible. Or traduire exige de comprendre, et l'une des difficultés majeures que rencontre tout programmeur généraliste est de pénétrer la nature très diverse des problèmes qui lui sont proposés, d'en saisir la logique interne en quelque sorte, de manière à pouvoir la transcrire informatiquement.

Ainsi, il n'est pas aisé de déterminer ce qui manque à l'homme de la rue pour que se dissipe son étonnement face à l'effectivité générale des ordinateurs. Il ne semble pas s'agir essentiellement d'un bagage technique, ni d'un savoir mathématique, ni encore de méthodes professionnelles, même si tous ces ingrédients sont utiles à la formation d'un programmeur. La preuve en est que l'on peut commencer à programmer très jeune, dès l'âge de sept ans environ[10]. Aussi l'informaticien Gérard Berry, lors de sa leçon inaugurale au Collège de France, ne rapporte-t-il pas cet étonnement à des lacunes concernant des savoirs positifs concrets, mais plus généralement à des « schémas mentaux inadaptés » :

> Si la locution *monde numérique* s'entend partout, ses fondements restent largement ignorés du public, qui semble en permanence surpris par les innovations techniques et les transformations sociales associées. Or, au moins sur le plan technique, l'évolution est largement prévisible, et il n'y a pas de raison d'être surpris par du prévisible. La surprise permanente est plutôt le signe d'un schéma mental mal adapté, ce qui n'est pas étonnant car l'information synthétique est encore pauvre dans ce domaine qui ne repose pas sur des bases enseignées classiquement. L'ambition de cette leçon inaugurale est d'aider à construire un schéma mental approprié, autrement dit un *bon sens informatique*, en expliquant pourquoi le monde devient numérique, comment les transformations

correspondantes se passent, et quels concepts et outils elles utilisent[11].

L'expression de « bon sens informatique » est d'une certaine manière un oxymore : car elle indique que le bon sens simple, celui que possède justement l'homme de la rue, n'est pas suffisant pour appréhender le phénomène informatique. Berry semble avancer par là, peut-être malgré lui, que l'informatique n'est pas seulement une question technique, mais qu'elle requiert une *réforme de l'entendement* – ce qui peut légitimement susciter une surprise encore plus grande que celle dont nous sommes partis.

Que propose donc Gérard Berry pour adapter nos « schémas mentaux » à ce « bon sens informatique » ? Quel est, en particulier, le langage générique capable de décrire toute action, que nous cherchions tout à l'heure ? Sa réponse est claire : ce langage est celui de l'information et du calcul. Le pilier sur lequel repose l'informatique est « l'idée de représenter et de manipuler de façon homogène toute information quelle que soit sa nature », et partant de résoudre tous les problèmes par le codage de fonctions numériques.

Cependant, si Berry montre bien comment coder en nombres le texte, les images, les sons, les températures, les mouvements et les forces, il n'explique pas *ce que l'on peut faire* avec ces codes numériques. Par exemple, pendant très longtemps la numérisation des images s'est limitée à des applications extrêmement sommaires : transmission, compression, découpages et recollages, distorsions, analyses de fréquences, etc. Mais ce que nous faisons avec des images – reconnaître une personne, une situation, un paysage – n'est pas du tout aisé à programmer. Une telle

opération n'est devenue accessible aux ordinateurs que très récemment, les premiers systèmes commerciaux de reconnaissance faciale datant de la fin des années 1990. La raison en est que le codage numérique des images perd toute l'information « structurelle » que nous savons y reconnaître. Ce n'est qu'avec l'apprentissage profond* – donc après 2010 – que de telles opérations sont devenues fiables à grande échelle.

Tout ce que l'on peut donc dire, c'est que la programmation parvient, tant bien que mal, à répliquer certains de nos processus de connaissance, puisqu'elle s'occupe d'agencer des processus mêmes de manipulation et de création d'informations. L'hypothèse à laquelle nous sommes amenés est donc la suivante : c'est parce que nos activités doivent accéder à des savoirs positifs sur le monde ou, plus généralement, sur le problème à résoudre, que l'informatique parvient si bien à les assister. La programmation transposerait sur des machines nos processus de connaissance – calcul, mémoire, perception, comparaison, etc. – et c'est pour cela qu'elle aurait la polyvalence – voire l'universalité – de la connaissance elle-même.

L'hypothèse computationnelle de l'esprit

L'idée que la programmation pourrait répliquer la complexité de tous les comportements, parce qu'elle porte sur la connaissance elle-même, suscita l'enthousiasme d'une nouvelle génération de psychologues, au début des

années 1950, et donna naissance à la science cognitive, dans sa première version, dite *computationnelle*. La psychologie américaine, à cette époque, était dominée par le béhaviorisme, qui avait forgé le concept de *comportement* comme objet d'étude propre de la psychologie, le distinguant aussi bien de la physiologie du système nerveux que de l'étude des états mentaux. Le béhaviorisme visait à expliquer le comportement par des processus de fixation de réactions sensori-motrices (le fameux couple « stimulus/réponse ») face à des situations récurrentes et par leur composition subséquente en réactions complexes. Les béhavioristes en étaient cependant venus à admettre l'idée que l'intelligence, même animale, ne saurait s'expliquer par la simple plasticité de ces mécanismes internes, et qu'une certaine capacité d'auto-reconfiguration devait leur être conférée. Ainsi, Edward Tolman, un psychologue béhavioriste, propose dans les années 1940 la notion de carte cognitive (*cognitive map*) dont la métaphore, qui prépare celle de l'ordinateur, est celle du standard de commutation téléphonique. Un standard téléphonique n'est pas seulement le lieu central où passent tous les chemins possibles, il est surtout le lieu où *tel* chemin est *sélectionné* plutôt qu'un autre[12].

L'idée de programme fut à ce titre une libération conceptuelle pour les jeunes psychologues comme George Miller et Jerome Bruner. Plutôt que se contenter de l'image étriquée d'une composition mécanique de circuits comportementaux entrée-sortie, la cybernétique et l'informatique montraient la puissance de circuits à rétroaction pour modéliser à peu près tout ce que l'on voulait. Comportements simples et complexes, compositions de comportements, convergence de comportements aléatoires vers un comportement stable,

la mémoire, le langage, etc. – l'informatique montrait la possibilité de tout cela avec une clarté confondante, sans rien sacrifier au parti pris mécaniste imposé par le béhaviorisme à la psychologie, garant de sa scientificité. L'idée de *programme* permettait même de réhabiliter les notions d'*intention*, puisqu'un programme qui en appelle un autre peut apparaître comme la finalité de ce dernier, et de *réflexivité*, puisqu'un programme peut en analyser et en modifier un autre.

Il s'agissait donc d'inverser la perspective : plutôt que de partir de l'étude de rats et de pigeons pour comprendre comment des comportements élémentaires pouvaient progressivement se composer en comportements complexes, en prenant pour paradigme l'image du circuit élémentaire entrée-sortie, il fallait partir tout de suite du comportement complexe de l'être humain, capable de s'adapter et de se reconfigurer lui-même. La psychologie devenait ainsi *essentiellement* cognitive, au sens où, par ce terme, il ne s'agissait plus seulement de désigner cette branche de la psychologie s'occupant des phénomènes de cognition, mais d'expliquer tout type de comportement à partir de la cognition.

Ces idées furent exprimées dans toute leur clarté – et avec une certaine naïveté – par George Miller, Eugene Galanter et Karl Pribram, dans un ouvrage fondateur, au titre explicite : *Les plans et la structure du comportement* (*Plans and the Structure of Behavior*). Ils y affirment que le fonctionnement de l'esprit repose sur l'exécution de *Plans* qui sont, « pour un organisme, essentiellement la même chose qu'un programme pour un ordinateur[13] ». C'est même toute l'architecture de l'ordinateur moderne qui est