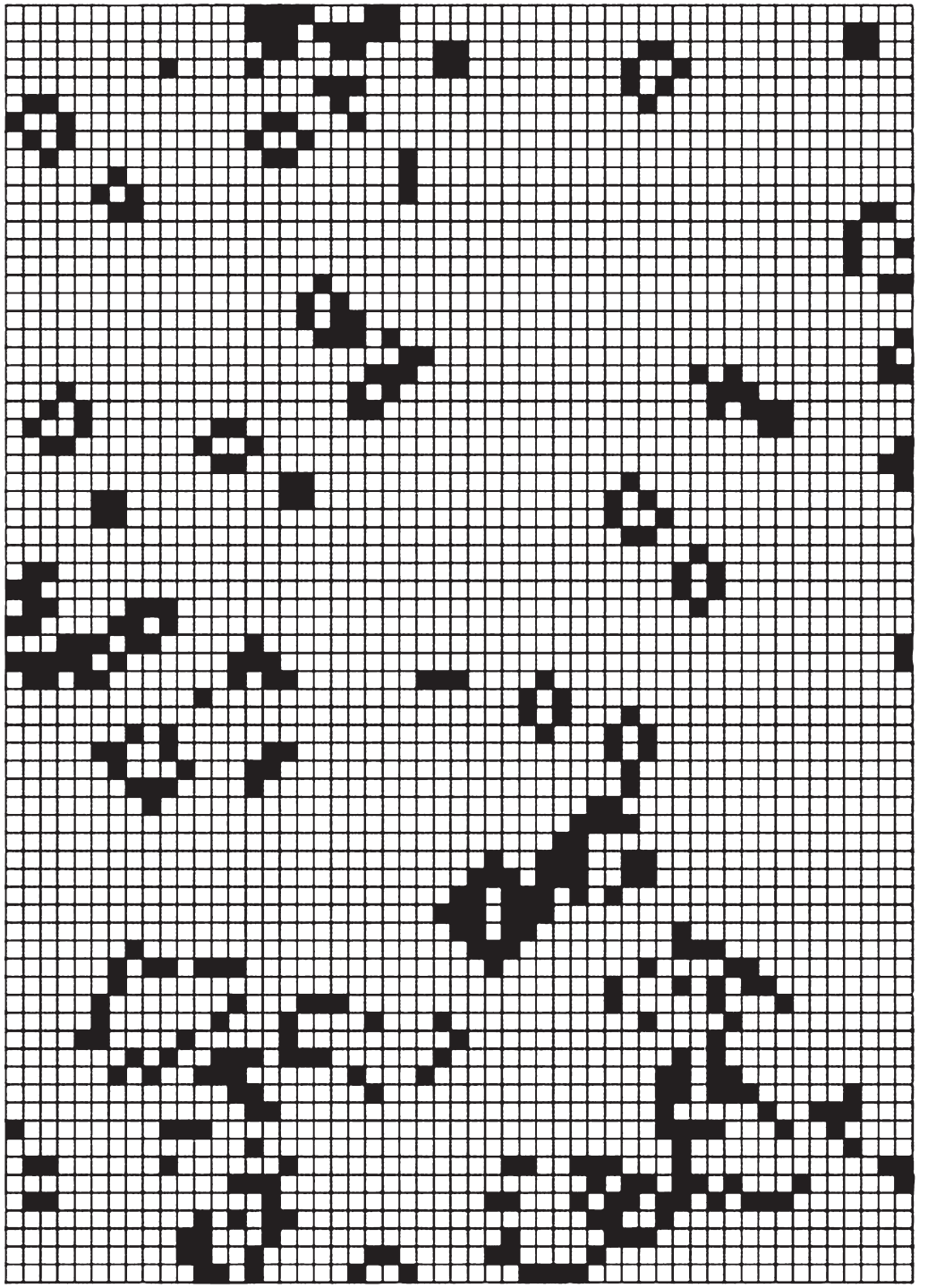
NOMBRES, CODES, PROGRAMMES



↔ *Game of Life* (jeu de la vie), John Horton Conway, 1970.

Conjectures des langages formels

Frédéric Migayrou

L'histoire du code est d'abord une histoire de la formalisation, celle de la fonction des signes de la définition de leur valeur sémantique et syntaxique, le code restant indissociable de l'écrit, code, caudex, codex, renvoyant à la fonction première de l'inscription. Le code est un système conventionnel organisé autour de symboles et articulé selon des règles combinatoires formant des ensembles, une capacité à produire des informations et à les transmettre, à former un système linguistique. Le code a aussi une valeur normative, il définit des règles d'usage, il impose une valeur prescriptive, il organise une logique interne, il accompagne l'extension d'un domaine juridique, il autorise et garantit la permanence des conventions, des contrats, des lois. Mais le code trouve dans sa signification la plus primitive une autre dimension conventionnelle, il accompagne la notion de chiffre, une forme d'écriture mêlant signes, lettres et nombres et liant de toute origine écriture et numération. Au-delà de l'histoire des nombres, de la notion de *cifra* issue de la numérotation arabe et qui introduira au Moyen Âge, contre la numération romaine, le concept mathématique du zéro, le chiffre associe au principe même de l'écrit, l'idée du calcul, de la numération. Le chiffre est l'élément d'écriture de tout système de numération, qu'il soit unaire, binaire ou décimal mais il porte en lui la structure symbolique du chiffrement, l'idée d'un cryptage où lettre et nombre peuvent entrer dans des jeux de substitution. Le code suppose une forme de dénombrement, la définition d'une chaîne numérique et finalement d'une combinatoire. Toute approche d'une histoire du domaine computationnel se heurte à la définition d'un langage formel le plus souvent assimilé à l'évolution d'une histoire du calcul, une certaine idée de la machine à calculer, et de son évolution technique accompagnant les étapes d'un développement de la mathématique et de la logique. Nombre d'histoires de l'informatique tissent une trame historique à partir de la généralisation des tables de calcul, des abaques, dans toute civilisation, de la création de la première machine arithmétique par Blaise Pascal p. 65, 74 à celle de Gottfried Wilhelm Leibniz p. 64, 74, 111, de l'apparition des cartes perforées pour les métiers à tisser de Joseph-Marie Jacquard p. 74 à leur utilisation pour programmer les données de la machine analytique de Charles Babbage p. 64, 74 puis de celle d'Herman Hollerith p. 65, 74, 112 utilisée pour le recensement des populations. L'idée que l'informatique ouvre le champ d'une formalisation liée à ce processus d'encodage

de la machine et au développement continu de sa performativité traduite en termes de vitesse, d'optimisation des processeurs, et de l'infinie augmentation des capacités de stockage mémoire, nourrit une doxa où la technique et plus largement les technologies, sont assimilées aux modèles définis par le machinisme. L'histoire de la miniaturisation, des transistors aux circuits imprimés, des puces gravées en nanomicrons aux nouvelles puces nanomorphiques, conforterait une certaine approche de la temporalité, de sa réfraction extrême selon la description d'un « accélérationnisme », d'une idéologie de la vitesse, d'un temps réel, ancré sur la définition d'un statut monolithique et médiant de la technique. Toute approche des langages formels serait, au-delà d'un questionnement sur la technique puis de la cybernétique, tenue à la tension critique posée entre l'affirmation des sources logiques du calcul numérique, de la calculabilité, de la mise en évidence de fonctions calculables mécaniquement par un algorithme ancrées sur le finitisme de David Hilbert, et le refus d'un tel positivisme assumé par Martin Heidegger récusant l'objectivisme de la technique pour réintroduire le principe fondamental d'une fracture ontologique. « L'être lui-même aurait-il été touché par une mise en demeure de faire apparaître l'étant dans la perspective de la calculabilité ? […] L'arraisonnement à plus d'être, s'il est encore permis de s'exprimer ainsi, que toutes les énergies atomiques et que toutes les machines du monde, plus d'être que la masse écrasante de l'organisation, de l'information et de l'automation »[1]. *Identité et différence*, pour reprendre le titre de la conférence de Martin Heidegger, l'ensemble du champ critique qui s'est constitué autour des technologies numériques pour aujourd'hui former le champ des *digital humanities* (humanités numériques) semble répondre à l'écho de cette ontologie extatique de l'étantité. Ainsi s'oppose la technique moderne et la *tekhnè* grecque, le temps différé et le temps réel des technologies de l'information, l'écriture, la trace, aux codes programmatiques.

Jean-François Lyotard [p. 63, 111] avait circonscrit ce domaine critique induit par les technologies de l'information et la généralisation du numérique comme la réversion du langage par rapport au réel à son référent matériel laissant place à une langue programmatique et formelle. En opposant au *technologos* une histoire du langage humain, la *poetike tekhne* des grecs chez Aristote, une synthétique

① Martin Heidegger, *Questions I*, Paris, Éditions Gallimard, 1968, p. 268-269.

du temps, d'un langage mémoire qu'il retrouve chez Emmanuel Kant ou Henri Bergson, Jean-François Lyotard [p. 62, 111] rejoint la thématique heideggerienne et accorde à l'écriture une fonction ontologique ou pré-ontologique au sens de Jacques Derrida [p. 115], « effet du miroir clair de l'être sur le miroir de l'étant »[2]. Évoquant pourtant une matérialité de la technique propre à la corporéité des organismes vivants, il envisage les possibles d'une pensée sans corps mais qui ne peut se résoudre à la structure des logiques binaires : « La déception procurée par ces organes de pensée sans corps provient de ce qu'ils opèrent en logique binaire, celle qui s'est imposée avec la logique mathématique de Russel et de Whitehead, la machine de Turing, le modèle neuronal de McCulloch et Pitts, la cybernétique de Wiener et von Neumann, l'algèbre de Boole, l'informatique de Shannon »[3]. D'une certaine manière, cette opposition entre logique et phénoménologique dirige encore aujourd'hui les questionnements autour du numérique, interpellant notre « être au monde computationnel ». L'extension du *logos*, au-delà de la notion de discours, à la logique, aux domaines de la rationalité, de la raison, reste l'enjeu premier de l'interprétation critique heideggerienne qui, reconfigurant une origine présocratique, héraclitéenne, mettra en exergue une dimension ontologique, une capacité du langage à laisser survenir le sens (*ereignis*), le rapport d'appartenance entre pensée et être. Mais au-delà d'une confusion entretenue avec la dimension technique de la logique aristotélicienne, peut-on simplement assimiler la formalisation des logiques computationnelles avec la langue, ou pour les distinguer plus avant des langages ou de l'écriture, qui restent des univers sémantiques flexibles et mouvants, soumis à des mutations et des évolutions morphologiques permanentes. Il est sans doute nécessaire de redéfinir une autre généalogie du computationnel qui trouve une source plus spécifique dans une histoire de la numération et de l'art combinatoire dont on retrouve des sources aussi bien en Grèce qu'en Chine. L'ouvrage majeur de Paolo Rossi, *Clavis Universalis*[4], reconstituait cette archéologie d'une intrication entre arts de la mémoire et logique combinatoire ayant, à partir de l'*Ars Magna* (1275) de Raymond Lulle [p. 74], organisé la structure du savoir selon des tables et des graphiques circulaires qui ne cesseront d'influer sur l'évolution d'une formalisation des théories de la connaissance et la quête d'une langue universelle, de Nicolas de Cues

② Jean-François Lyotard, « Logos, tekhnè, ou la télégraphie », *L'inhumain : Causerie sur le temps*, Paris, Éditions Galilée, 1988, p. 66.

③ Jean-François Lyotard, « Si l'on peut penser sans corps », *op. cit.*, p. 23.

④ Paolo Rossi, *Clavis Universalis*, Grenoble, Éditions Jérôme Millon, 1993.

à Petrus Ramus ou Giordano Bruno, de Francis Bacon à René Descartes et surtout à Gottfried Wilhelm Leibniz [p. 61, 74, 111] qui publiera sur la base de sa thèse son *De Arte Combinatoria* (1666). Ce n'est donc pas la logique formelle, le développement d'une idéologie de la technique, des âges du machinisme et d'une culture appareillée qui auraient seul présidé à l'apparition de l'ordinateur, mais plus évidemment la généalogie d'un art combinatoire qui trouve son premier accomplissement avec Gottfried Wilhelm Leibniz [p. 61, 74, 111] comme art de penser universellement mais aussi un *ars inveniendi*, un art d'inventer fondé sur des bases logiques (*logica inventionis semina*). « La combinatoire est certes ancrée dans l'arithmétique puisqu'elle consiste à trouver le nombre de toutes les combinaisons possibles pour des éléments et des conditions données. Mais son applicabilité n'est pas limitée à cette discipline puisque les éléments en question ne sont pas des nombres mais des caractères dont l'interprétation va de l'algèbre à la géométrie, en passant par la logique, la musique ou la cryptographie » ⑤.
La publication par Leibniz de l'*Explication de l'arithmétique binaire* (1703), qui fait suite à la conception de sa machine à calculer dotée d'une mémoire mécanique, reste un jalon dans l'histoire d'une utilisation des systèmes de numération binaires modernes, de Thomas Harriot à Francis Bacon et Leibniz, pour s'accomplir avec la logique algébrique de George Boole [p. 74, 111] (*Mathematical Analysis of Logic*, 1847) et trouve une première application dans le domaine du calcul numérique dans les recherches de Louis Couffignal [p. 113] (*Les machines à calculer : leurs principes, leur évolution*, 1933), dans la conception et dans la théorie des signaux de Claude Shannon [p. 67, 76, 104, 113, 143, 148, 230] (« A Mathematical Theory of Communication » [p. 67, 76, 113, 230], 1948). Le lien entretenu avec l'analyse combinatoire se démontre avec acuité dans la récurrence de l'emploi des damiers ou des échiquiers, ainsi pour la *Rhabdologie* (1617) de Jean Neper où le « calcul par l'échiquier » répondait à la numération binaire, le *tit-tat-to* imaginé en 1844 par Charles Babbage [p. 61, 74] ⑥ parallèlement à la création de sa machine analytique, ou l'automate joueur d'échecs (1914) de Leonardo Torres y Quevedo [p. 74], première machine à posséder une mémoire électromécanique. Même Alan Turing [p. 70, 95, 102, 154, 200] imaginant en 1936 sa célèbre machine s'attacha à la conception d'un programme d'échec (Turochamp, 1948), tout comme Christopher Strachey [p. 54, 70, 96, 102, 154], inventeur de la sémantique dénotationnelle en programmation qui, travaillant

⑤ Hourya Benis Sinaceur, « Ars Inveniendi aujourd'hui », *Les Études Philosophiques*, N°2, 1989, Presses Universitaires de France, p. 202.

⑥ Devin Monnens, « I commenced an examination of a game called 'tit-tat-to': Charles Babbage and the 'First' Computer Game », *Proceedings of DiGRA 2013: DeFragging Game Studies*, vol. 7, août 2014, p. 1-14.

avec Turing, concevra un jeu de dames pour le Ferranti Mark I, selon un programme de mille instructions, le plus long jamais créé à l'époque (1952). Au-delà de la dimension du jeu, c'est bien celle du hasard qui hante toute l'histoire d'une formalisation d'abord mécanique puis numérique du calcul, celle des automates, des horloges discrétisant le temps, des machines simulant l'ordre du jeu, des automates musiciens de la Renaissance au XVIII^e siècle jusqu'au triomphe du programme Deep Blue sur le champion du monde d'échecs Garry Kasparov en 1997. L'automate est par essence un objet programmé mais l'une de ses premières applications, préfigurant concrètement l'âge numérique, ouvrira le vaste champ d'une modélisation des phénomènes aléatoires et celle de son application pour une machine à statistique à cartes perforées utilisée pour le recensement américain de 1850 par Herman Hollerith [p. 61, 74, 112]. Du calcul des probabilités initié par Blaise Pascal [p. 61, 74] et Pierre de Fermat à la théorie des nombres, les combinatoires algébriques formalisées par des algorithmes trouveront avec l'infinie puissance de calcul des ordinateurs une expansion qui de fait ont constitué le socle d'un domaine computationnel. C'est bien ici que l'interprétation du code comme une forme symbolique, et donc son assimilation à un langage formalisé fait question, ouvrant la voie à de multiples approches qui dans une filiation structuraliste, s'affirment comme des herméneutiques en capacité de décrypter le sens, d'organiser une sémantique des langages machines.

Peut-on simplement assimiler le computationnel à une formalisation entièrement définie selon une ontologie formelle et donc, au-delà des formalismes logiques fermés, retrouver comme une sauvegarde la dimension intuitionnelle du mathématique d'Henri Poincaré à Jean Cavaillès ouvrant à une ultime herméneutique. En soulignant l'évidente convergence numérique, soit la traductibilité de tous les domaines d'informations, que ce soit l'écrit, le son, l'image, la tentation est grande d'y voir l'accentuation d'un principe de déréalisation, d'opposer nos systèmes analogiques traditionnels, à cette accentuation des niveaux de discrétisation. La perspective d'une approche formaliste du langage, de la mise en évidence d'une possible grammaire universelle, une grammaire générative définie par une modélisation structuraliste de la langue telle que définie par Noam Chomsky [p. 66, 97, 114], constituera le plus généralement un cadre d'analyse récurrent

de l'encodage numérique. Toutefois, peut-on simplement assimiler les élaborations du code source, le code opérationnel des instructions machines, les phases d'assemblage et de compilation et la genèse des langages de programmation à une grammaire formelle pour transposer les normes ASCII ou Unicode en un alphabet, les symboles d'écritures en des règles de syntaxe, les instructions machine en un vocabulaire, pour idéalement constituer une sémantique générale du computationnel. La simple prise en compte d'une histoire de la programmation, de la méthode de calcul à partir de nombres de Bernoulli d'Ada Lovelace [p. 74, 88], au lambda-calcul d'Alonzo Church [p. 68, 95], aux instructions de Plankalkül de Konrad Zuse [p. 75, 96], au Short Code [p. 76] de John Mauchly [p. 75, 76], à l'Autocode [p. 76] d'Alick Glennie [p. 76], démontre la spécificité du langage machine définissant un champ ontologique propre. L'autonomisation de langages sources (et la liberté d'une implémentation sur de multiples machines) comme le Formula Translating System (Fortran [p. 70, 76, 77, 91, 103, 114, 168, 188, 208, 216]) de John Backus [p. 76] ou le FLOW-MATIC [p. 77], premier langage de traitement de données élaboré pour l'UNIVAC par Grace Hopper [p. 76] ont ouvert la voie à une pluralité de programmes du Cobol [p. 76, 77] à l'Algol [p. 77, 114, 122], de Lisp [p. 77, 114, 141] à C++ [p. 78] ou au BASIC [p. 77].
La production exponentielle de programmes, des langages orientés objets jusqu'au *generative programming* et aux pratiques de *scripting* généralisées par l'*open source* et l'universalisation d'Internet, ne peuvent voiler le fondement même d'un « langage » défini par le nombre, par la numération. Noam Chomsky [p. 65, 97, 114], alors qu'il affirme dans un texte séminal que « la grammaire doit être définie et approchée comme un domaine spécifique indépendamment des sémantiques », préserve dans son réductionnisme la structure transformationnelle des langues naturelles récusant « le modèle d'un langage à états finis du processus de Markov » ⑦. Cette référence à Andreï Markov est cruciale car elle marque une fracture décisive dans l'approche du langage envisagé non plus comme un médium linéaire mais comme une distribution aléatoire de signes, de voyelles et de consonnes, un système discret, binaire, qu'il soumet à une analyse statistique afin de dégager des modèles de régularité. Alors qu'il est invité à une conférence pour le bicentenaire de l'*Ars Conjectandi* (1690) de Jakob Bernoulli et sur le théorème des probabilités, Andreï Markov ne s'attachera pas aux nombres mais au roman en vers *Eugène Onéguine* (1831) d'Alexandre Pouchkine qui sera la base de sa théorie

⑦ Noam Chomsky, *Syntactic structures*, Mouton publishers, 1957, p. 106.

tel que le démontre David Link. « Cette méthode s'applique à tous les autres calculs de Markov. Elle ne concerne pas le style d'un auteur en particulier et une façon d'arranger les mots mais le langage en général. La conclusion est donc qu'à partir de cette dispersion de valeurs, le processus de formation qui génère l'ensemble de toutes les chaînes de lettres autorisées est un processus aléatoire » ⑧.
Et David Link d'étendre son analyse à un texte contemporain qui paradoxalement est la source de toute compréhension structuraliste du langage, le *Cours de linguistique générale* de Ferdinand de Saussure qui dresse une analogie comparable entre la formation de la langue et le système combinatoire du jeu d'échecs. « Mais de toutes les comparaisons qu'on pourrait imaginer, la plus démonstrative est celle qu'on établirait entre le jeu de la langue et une partie d'échecs… un état du jeu correspond bien à un état de la langue. La valeur respective des pièces dépend de leur position sur l'échiquier, de même que dans la langue chaque terme à sa valeur par son opposition à tous les autres » ⑨. Claude Shannon [p. 64, 76, 104, 113, 143, 148, 230] exploitera dans « A Mathematical Theory of Communication » [p. 64, 76, 113, 230] (1948) les chaînes de Markov pour établir des modèles statistiques suggérant que toute source transmettant des informations est une chaîne de Markov vers une transcription digitale des signaux analogiques afin qu'ils puissent être encodés, une discrétisation permettant d'établir un processus ergodique préservant la structure des messages face à la déperdition des bruits et de l'entropie. Les chaînes de Markov, le processus de Markov à temps discret et à espace d'états discrets, reste par ailleurs une source fondamentale des modèles de simulations informatiques appliqués en physique, croisant la théorie des ensembles, théorie basique de l'intégration et la théorie des probabilités, telle qu'énoncée avec la méthode de Monte-Carlo de Nicholas Metropolis [p. 76] et Stanislaw Ulam [p. 76, 115] ⑩.

La méthode de Monte-Carlo qui tient son nom des jeux de casino s'organise selon des méthodes algorithmiques permettant d'affecter, à partir d'un échantillon aléatoire (*random sampling*) choisi au sein d'un large ensemble (une population), une distribution de probabilités, une simulation dont l'algorithme permet de façon récursive de définir une valeur numérique approchée, et donc la probabilité d'occurrence la plus efficiente. Pleinement développée par Stanislaw Ulam [p. 76, 115] et John von Neumann [p. 68, 75, 95, 115, 116] pour évaluer le souffle

⑧ David Link, « Traces of the mouth, Andrei Andreyevitch Markov's Mathematisation of writing », in *Archeology of algorithmic artefacts, univocal*, 2016, p. 42.

⑨ Ferdinand de Saussure, *Cours de linguistique générale*, Paris, Payot, 1931, p. 125-126.

⑩ Nicholas Metropolis et Stanislaw Ulam, « The Monte-Carlo Method », *Journal of the American Statistical Association*, vol. 44, N°247, septembre 1949.

de la bombe atomique, la méthode de Monte-Carlo répondait au développement de la théorie des ensembles de von Neumann-Bernays-Gödel et à la limite posée par les théorèmes d'incomplétude de Kurt Gödel[p. 75, 116]. Von Neumann multipliera ainsi les champs d'application notamment dans les domaines de la dynamique des fluides, de la prévision météorologique, de la prévision économique publiant avec Oskar Morgenstern la *Théorie des jeux et comportements économiques* (1944), dans le domaine informatique redéfinissant l'architecture des ordinateurs modernes, et surtout dans la modélisation selon des automates cellulaires (*Theory of Self-Reproducting Automata*, 1966) ouvrant la voie aux modèles d'auto-réplication et du constructeur universel (*universal constructor*) et aux modélisations orientées agent (*agent-based modeling*). « Von Neumann s'est demandé s'il était possible pour un seul automate dans un environnement donné d'être un constructeur universel, c'est-à-dire capable de construire tous les autres automates si l'on en donnait une description appropriée. Il s'est ensuite interrogé afin de déterminer si un automate de ce type pouvait construire un autre automate identique à lui-même »⑪. De tels modèles de simulations génériques établissent un modèle ontologique propre qui dépassent la simple opposition entre réalisé et l'abstraction d'un domaine déréalisé renvoyant vers un univers virtuel où le calcul n'a pas de structure signifiante en tant que telle, un système formel comme le lambda-calcul d'Alonzo Church[p. 66, 95] ou une fonction récursive primitive qui n'est pas assimilée à un système de messages. Il convient donc *a contrario* de s'attacher, hors toute approche techno-logique du computationnel (en une quête du sens, d'un *logos* mis à mal par la structure médiate de la *tekhnè*), à une ontologie du nombre et au statut de la récursivité pour des structures de données. Les réseaux neuronaux récurrents (RNN) qui modélisent les langages de nos réseaux numériques répondent à des modèles ontologiques ancrés dans une histoire des combinatoires et d'un redéploiement de la notion de nombre, des ordinaux conçus comme des ensembles ordonnés selon John von Neumann[p. 67, 75, 95, 115, 116] aux nombres surréels incluant les nombres réels et les nombres ordinaux transfinis de Georg Cantor tels qu'initiés par John Horton Conway[p. 77, 162], l'inventeur du *Game of Life*[p. 77, 162] (jeu de la vie, 1970) et popularisés par Donald Knuth[p. 97, 103, 105], l'auteur de *The Art of Computer Programming* (1974). Stephen Wolfram[p. 78, 157, 176], concepteur du langage de programmation Mathematica[p. 78, 116, 176]

⑪ William Stevens, « A self-replicating programmable constructor in kinematic simulation environment », *Robotica*, vol. 29, 2011, p. 154.

basé sur les automates cellulaires, souligne cette évolution imposée par les développements d'une théorie combinatoire des nombres. « Depuis l'apparition des ordinateurs électroniques, de grands efforts ont été faits pour définir le continuum, les nombres réels aussi précisément que possible. Cela fonctionnait pour étudier des systèmes au comportement assez simple. Mais pour des systèmes plus complexes, il apparaissait évident que cette approximation échoue et il n'y avait d'autres choix que de redéfinir de façon explicite notre représentation des nombres »⑫. Cerner le champ épistémologique qui borne l'histoire du code, c'était donc revenir sur l'essence même du numérique, sur la mutation du statut des nombres et sur leurs interactions. En s'attachant aux disciplines de création qui, sur près d'une cinquantaine d'années, se sont impliquées dans le domaine digital, il est frappant de voir la congruence des recherches qui de l'art à la littérature, de la musique à l'architecture ou à la danse, ont saisi, au-delà de la simple poursuite des évolutions technologiques, d'un accès de plus en plus effectifs aux ordinateurs et aux logiciels, l'essence même d'une formalisation par le nombre, le système binaire, de base 2, ayant très vite été transcrit visuellement sous forme de grilles, de pixels ou de voxels et ce dès les années 1960. Peut-être faut-il revenir, au-delà des langages, à la notion primaire du programme, un programme source qui peut être compilé vers une forme binaire et qui décrit un registre d'instructions exécutables par un processeur.

La notion d'un art programmé dépassera très vite la simple notion du *Computer Art*, celle d'une poursuite des recherches de l'art cinétique touchant directement les notions d'image, de visualisation, et de cognitionv engageant le spectateur. La série d'expositions « Nove Tendencije »[p. 88] organisée à Zagreb (1961-1973) questionnait cette ambiguïté maintenue entre représentation et programmation tout comme l'exposition « The Responsive Eye »[p. 83, 89] organisée au MoMA en 1965. Frieder Nake[p. 71, 82, 85, 89, 92, 194, 214], artiste qui dès 1963 écrit ses propres programmes, s'étonnera lors de « Tendencije 4 » que le concours qui y est organisé concerne plus les objets et non les programmes. Pour l'exposition « Arte programmata »[p. 84, 88], conçue en 1962 par Bruno Munari[p. 52, 84], Umberto Eco[p. 52, 84, 88] définira les enjeux ouverts par la création numérique. « Il ne sera donc pas impossible de délimiter avec la pureté linéaire d'un programme mathématique, des champs des cas

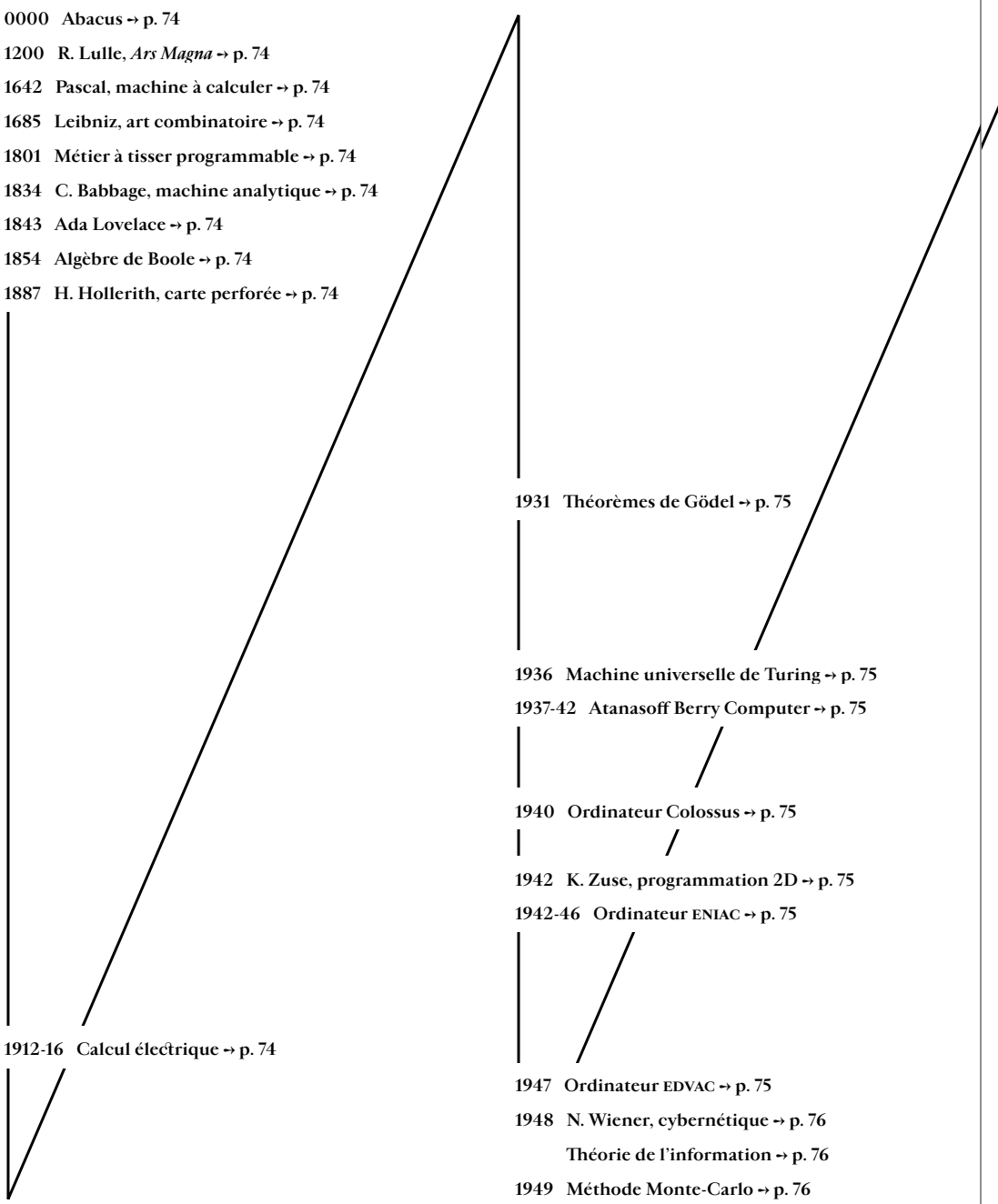
⑫ Stephen Wolfram, History of numbers, *A new kind of science*, Wolfram Media, 2002, p. 901.

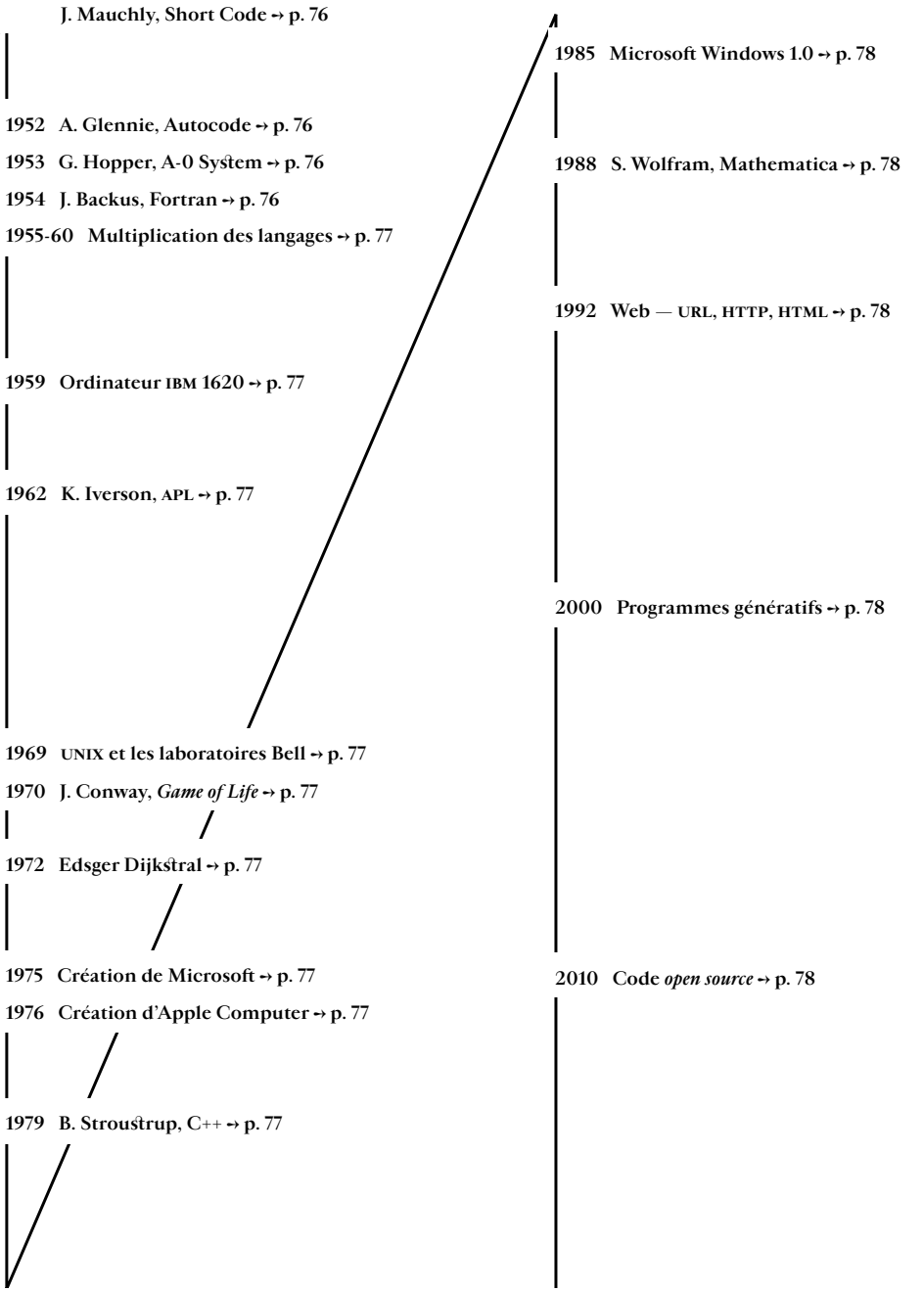
possibles dans lesquels peuvent se vérifier des processus casuels. C'est ainsi que nous aurons une singulière dialectique entre cas et programmes, entre mathématique et hasard, entre conception planifiée et libre acceptation de ce qui arrive… Nous pouvons alors parler d'art programmé »[13]. C'est un nouveau champ esthétique et critique qui s'ouvrait, définissant au-delà de tout formalisme, des critiques réitérées contre ce nouveau positivisme technologique, ouvrant le champ d'un accès aux domaines de l'aléatoire et de la complexité, où sont apparues de nouvelles méthodologies directement extraites de l'approche computationnelle, de la théorie des groupes, des formes de sérialisation, des nombres aléatoires, des automates cellulaires, rassemblant tous les domaines de création, intuition qui animait déjà Jasia Reichardt [p. 54, 82, 90], la commissaire de l'éponyme exposition « Cybernetic Serendipity » [p. 54, 82, 90, 1] (1968). Saisir en cohérence ce qui unit les domaines de créations dans les pratiques digitales les plus contemporaines, c'était donc recomposer les phases d'une archéologie du code, et saisir au-delà des œuvres la congruence d'une intelligence partagée de méthodes de programmation. Christopher Strachey [p. 54, 64, 96, 102, 154], qui a étroitement travaillé avec Alan Turing [p. 64, 95, 102, 154, 200], et l'auteur de *Systems Analysis and Programming* (1966), est tout aussi bien une source pour l'histoire des littératures digitales, auteur du premier poème écrit sur le Manchester Mark I (*Love letter*, 1952) que celui qui fait jouer au même ordinateur *God Save the Queen*, la première pièce d'informatique musicale. On pourrait de même comprendre les correspondances entre des créations basées sur l'utilisation de l'IBM 7070 et celles du langage Fortran [p. 66, 76, 77, 91, 103, 114, 168, 188, 208, 216] comme le poème *Tape Mark I* [p. 96, 102] (1961) de Nanni Balestrini [p. 96, 102], les chorégraphies de Jeanne Beaman [p. 130, 138] et du programmateur Paul Le Vasseur [p. 130, 138] ou les compositions graphiques établies avec Mini-Explor par Ken Knowlton [p. 81, 85, 89, 91, 92, 194]. Les chaînes de Markov restent autour de l'introduction de l'aléatoire dans les créations une référence récurrente, aussi bien pour Iannis Xenakis [p. 54, 149, 150, 154, 155, 156, 157, 238], initiateur de la musique stochastique (*Metastasis*, 1954), que pour la composition de la *Suite Illiac* [p. 148, 154, 186] de Lejaren Hiller [p. 148, 154, 155, 186], mais aussi pour l'écriture de l'*Autopoem* (1959) de Gerhard Sticker, de celle de *Castle* (1959) de Theo Lutz [p. 96, 102] recombinant des éléments du *Château* [p. 96, 102] de Franz Kafka [p. 96, 102], ou des premiers agencements

[13] Umberto Eco, *Arte programmata*, catalogue de l'exposition « Arte programmata », n.p, 1962.

graphiques de Frieder Nake [p. 69, 82, 85, 89, 92, 194, 214].
Les mêmes mises en parallèle pourraient être établies chronologiquement pour l'usage des automates cellulaires, des systèmes multi-agents, dans la création contemporaine. Au-delà des multiples archéologies qui se multiplient quant à l'expansion du digital dans les disciplines culturelles, il est temps de recomposer une archéologie croisée offrant une lisibilité plus efficiente, celle d'une esthétique générale du code, provision pour une compréhension des évolutions les plus contemporaines. « Une couche supplémentaire est venue s'ajouter aux langues et aux signes qui les rendent visibles, au langage non artificiel écrit des nombres, celle de l'encodage binaire et des langages artificiels propres aux machines… Soudain, chiffres et lettres se cachent sous des nombres, mais ceux-ci ne sont pas des idéalités, ce sont des nombres statuels, des signaux selon l'état de la matière »[14].

[14] Clarisse Herrenschmidt, *Les trois écritures, langue, nombre, code*, Paris, Gallimard, 2007, p. 454-455.


0000 Abacus ↦ p. 74
1200 R. Lulle, *Ars Magna* ↦ p. 74
1642 Pascal, machine à calculer ↦ p. 74
1685 Leibniz, art combinatoire ↦ p. 74
1801 Métier à tisser programmable ↦ p. 74
1834 C. Babbage, machine analytique ↦ p. 74
1843 Ada Lovelace ↦ p. 74
1854 Algèbre de Boole ↦ p. 74
1887 H. Hollerith, carte perforée ↦ p. 74
1912-16 Calcul électrique ↦ p. 74
1931 Théorèmes de Gödel ↦ p. 75
1936 Machine universelle de Turing ↦ p. 75
1937-42 Atanasoff Berry Computer ↦ p. 75
1940 Ordinateur Colossus ↦ p. 75
1942 K. Zuse, programmation 2D ↦ p. 75
1942-46 Ordinateur ENIAC ↦ p. 75
1947 Ordinateur EDVAC ↦ p. 75
1948 N. Wiener, cybernétique ↦ p. 76
Théorie de l'information ↦ p. 76
1949 Méthode Monte-Carlo ↦ p. 76


J. Mauchly, Short Code ↦ p. 76
1952 A. Glennie, Autocode ↦ p. 76
1953 G. Hopper, A-0 System ↦ p. 76
1954 J. Backus, Fortran ↦ p. 76
1955-60 Multiplication des langages ↦ p. 77
1959 Ordinateur IBM 1620 ↦ p. 77
1962 K. Iverson, APL ↦ p. 77
1969 UNIX et les laboratoires Bell ↦ p. 77
1970 J. Conway, *Game of Life* ↦ p. 77
1972 Edsger Dijkstral ↦ p. 77
1975 Création de Microsoft ↦ p. 77
1976 Création d'Apple Computer ↦ p. 77
1979 B. Stroustrup, C++ ↦ p. 77
1985 Microsoft Windows 1.0 ↦ p. 78
1988 S. Wolfram, Mathematica ↦ p. 78
1992 Web — URL, HTTP, HTML ↦ p. 78
2000 Programmes génératifs ↦ p. 78
2010 Code *open source* ↦ p. 78

0000 L'Abacus est un instrument facilitant le calcul et la numération que l'on retrouve dans toutes les civilisations (abaques, bouliers, bâtons, tablettes, etc.).

1200 Philosophe et théologien, Raymond Lulle [p. 63] est le concepteur de l'*Ars Magna*, une machine logique dont les roues organisent la réflexion théorique selon des figures géométriques.

1642 Blaise Pascal [p. 61, 65] conçoit la première machine à calculer opérationnelle réalisée en huit exemplaires. Elle effectue différentes opérations algébriques.

1685 Gottfried Wilhelm Leibniz [p. 61, 64, 111] construit en 1694 une machine à calculer dotée d'un cylindre cannelé qui introduit une mémoire permettant la réutilisation d'un nombre.

1801 Joseph-Marie Jacquard [p. 61] conçoit le premier métier à tisser mécanique programmable, sélectionnant les fils de chaîne à l'aide d'un programme inscrit sur des cartes perforées.

1834 Alors qu'il développe une machine à calculer destinée à l'impression de tables mathématiques, Charles Babbage [p. 61, 64] propose d'incorporer au système les cartes perforées du métier de Jacquard. Jamais réalisée, cette « machine analytique », préfigure l'ordinateur dans le sens où la lecture séquentielle introduit la possibilité d'instructions et de données.

1843 À la suite d'un texte écrit pour Charles Babbage [p. 61, 64] au sujet de la Machine analytique, Ada Lovelace [p. 66, 88] développe une séquence de nombres rationnels qui anticipe la logique des programmes informatiques.

1854 George Boole [p. 64, 111], mathématicien et philosophe, est l'initiateur d'une logique formelle, à la fois symbolique et mathématique permettant de traduire les concepts en équations ouvrant à des applications dans le domaine informatique.

1887 Herman Hollerith [p. 61, 65, 112] conçoit une machine de statistique à cartes perforées permettant de traiter un grand nombre d'informations. Il fonde en 1896 la Tabulating Machine Company.

1912-1916 Ingénieur et mathématicien, Leonardo Torres y Quevedo [p. 64] substitue aux éléments mécaniques des machines de calcul des éléments électriques. À partir de sa théorie des automates cellulaires, il crée un jeu d'échecs automatisé.

1931 Publication des théorèmes d'incomplétude de Kurt Gödel [p. 68, 116] mettent en évidence des énoncés mathématiques qui bien que vrais resteront à jamais indémontrables.

1936 La machine de Turing est avant tout un modèle universel de calcul qui répond à la question : peut-on découvrir un algorithme qui nous permet de décider si une suite logique de termes est valide ou non ? Elle est souvent considérée comme le premier modèle conceptuel de l'ordinateur, avant même son apparition physique.

1937-1942 Conçu par John Vincent Atanasoff et Clifford Berry, l'Atanasoff Berry Computer est aujourd'hui reconnu comme le premier ordinateur utilisant le mode binaire sur la base de calculs électroniques pour représenter les nombres et les données.

1940 L'ordinateur Colossus est le second calculateur électronique fondé sur le système binaire, soit un système reposant sur deux valeurs, notées par convention 0 et 1. Construit par une équipe dirigée par Thomas Flowers à Bletchley Park, il réalise 5 000 opérations par seconde et est utilisé pendant la Seconde Guerre mondiale pour la cryptanalyse du code Lorenz.

1942 L'ingénieur allemand Konrad Zuse [p. 66, 96] est un pionner du calcul programmable. En 1941, il réalise la première machine de calcul électromécanique programmable automatisée. Pour cela, il emploie le Z3, un programme binaire à virgule flottante (méthode d'écriture fréquemment utilisée dans les ordinateurs).

1942-1946 Conçu par John Mauchly [p. 66, 76] en 1942, l'ENIAC (Electronic Numerical Integrator And Computer) est construit entre 1943 et 1945. Il s'agit du premier grand ordinateur entièrement électronique utilisé pour des calculs balistiques.

1947 Comme l'ENIAC, l'EDVAC (Electronic Discrete Variable Automatic Computer) est conçu pour des calculs balistiques. Contrairement au premier qui fonctionne en mode décimal (utilisation d'une base de dix chiffres), il se fonde sur un système binaire. John von Neumann [p. 67, 68, 95, 115, 116] définit la structure de l'ordinateur comme étant composée d'une unité

arithmétique et logique, d'une unité de contrôle, d'une mémoire et d'un dispositif entrée-sortie. Avec Stanislaw Ulam [p. 67, 115], il entreprend un travail de modélisation avec les automates cellulaires qui ouvre la voie à la simulation.

1948 En fondant la cybernétique, Norbert Wiener [p. 82, 113] introduit la notion de *feedback* (rétroaction) qui aura des implications dans les domaines de l'ingénierie, des contrôles de système, l'informatique, la biologie, la psychologie, la philosophie et l'organisation de la société.

Publié par Claude Shannon [p. 64, 67, 104, 113, 143, 148, 230] en 1948, l'article « A Mathematical Theory of Communication » [p. 64, 67, 113, 230] pose les bases de la théorie de l'information et de la communication. Il introduit le mot *bit*, soit la quantité minimale d'information pour la transmission d'un message, qui devient l'unité de mesure élémentaire en informatique.

1949 La méthode Monte-Carlo, méthode algorithmique, est utilisée pour calculer une valeur numérique approchée grâce à des procédés aléatoires et probabilistes. Elle est inventée en 1947 par Nicholas Metropolis [p. 67] et publiée en 1949 avec Stanislaw Ulam [p. 67, 115].

Le Short Code [p. 66] conçu par John Mauchly [p. 66, 75] est l'un des premiers langages élaborés pour l'ordinateur, constitué de formules mathématiques et non de simples instructions machines, il est implémenté sur l'ordinateur BINAC puis sur l'UNIVAC (Universal Automatic Computer).

1952 Le terme d'« autocode » renvoie à une famille de systèmes de codage simplifié. Le premier d'entre eux est conçu par Alick Glennie [p. 66] pour l'ordinateur Mark I de l'université de Manchester. L'Autocode [p. 66] est un langage de programmation de haut niveau, soit un programme utilisant des mots usuels de la langue anglaise.

1953 Grace Hopper [p. 66] conçoit pour le fabricant Remington Rand le compilateur nommé A-0 System pour le premier ordinateur réalisé et commercialisé aux États-Unis UNIVAC I. Elle initie un programme proche de l'anglais plutôt que calqué sur le langage machine. De cette idée naîtra le langage Cobol [p. 66, 77] en 1959.

1954 Fortran [p. 66, 70, 77, 91, 103, 114, 168, 188, 208, 216] (Formula Translator) conçu par John Backus [p. 66] est le plus ancien langage de programmation de haut niveau, développé en premier lieu pour l'ordinateur IBM 704. Il est encore utilisé aujourd'hui, en particulier pour le calcul scientifique.

1955-1960 À la suite de Fortran [p. 66, 70, 76, 91, 103, 114, 168, 188, 208, 216], de nouveaux langages de programmation sont développés comme FLOW-MATIC [p. 66] (1955), Lisp [p. 66, 114, 141] (1958), Cobol [p. 66, 76] (1959), Algol [p. 66, 114, 122] (1959) avec leurs propres syntaxes et lexiques.

1959 L'ordinateur IBM 1620 est largement diffusé, équipant les entreprises, mais aussi les universités. Il offre un premier accès public aux langages informatiques et à la programmation.

1962 Conçu par Kenneth Iverson, l'APL (initialement A Programming Language), est un langage de programmation simplifié et flexible grâce à l'utilisation de symboles graphiques.

1969 Au sein des laboratoires Bell que Ken Thomson et Dennis Ritchie développent le système UNIX, l'un des premiers systèmes d'exploitation multitâches et multi-utilisateurs.

1970 Le *Game of Life* [p. 68, 162], « jeu de la vie » inventé par John Horton Conway [p. 68, 162] est un automate cellulaire, un modèle binaire en grille ou chaque état conduit à l'état suivant à partir de règles préétablies. Malgré sa simplicité, ce jeu permet de calculer tout algorithme et configure un système de simulation.

1972 Edsger Dijkstral reçoit le prix Turing après avoir participé au développement du langage Algol [p. 66, 114, 122]. Il se spécialise dans la compréhension des langages de programmation, de leur représentation et de leur implémentation.

1975 Premier programme utilisant le langage BASIC [p. 66] (1964). À partir de celui-ci, Bill Gates et Paul Allen développent en 1975 l'Altair BASIC pour l'ordinateur Altair 8800 et fondent la société Microsoft.

1976 Création par Steve Jobs et Steve Wozniak de la société Apple Computer qui produit les premiers ordinateurs utilisant un interpréteur BASIC [p. 66] et un clavier QWERTY connecté à un téléviseur.

1979 Le langage C a été créé au début des années 1970 dans les laboratoires Bell pour réécrire UNIX. Alors que le langage

se stabilise, Bjarne Stroustrup, informaticien, développe le C++ p.66 en 1979, l'un des langages informatiques les plus utilisés dans le monde (applications et jeux vidéo). Aujourd'hui normalisé par l'ISO.

1985 Windows 1.0 est la première interface graphique de Microsoft basée sur MS-DOS. Elle ouvre à de multiples applications, traitement d'images, calculs, graphes, gestion de la documentation, etc.

1988 Travaillant dès 1979 sur les automates cellulaires, Stephen Wolfram p. 68, 157, 176 développe une première version de Mathematica p. 68, 116, 176, un logiciel pour le calcul algébrique et la création de programmes.

1992 L'HyperText Markup Language (HTML) est un langage de balisage représentant les pages Web et permettant de naviguer entre elles. Développé par Tim Berners-Lee, Robert Cailliau, il est l'une des trois principales technologies du Web avec les adresses Web (URL), l'Hypertext Transfer Protocol (HTTP).

2000 Le *generative programming* est un processus de génération d'un code source automatisé ouvrant à de nouvelles plateformes de « métaprogrammations » fondées sur l'implémentation de nouvelles entités algébriques.

2010 La généralisation de l'*open source* soit le « code source ouvert » offre un accès direct au code source des logiciels et des programmes qui peuvent être lus ou améliorés, offrant de nouvelles interfaces de programmation et ouvrant à des systèmes partagés.

ALGORISTES