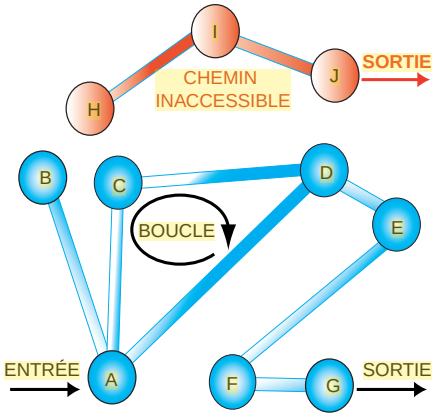
nombre n'est pas défini. Concrètement, le calcul se poursuit à l'infini. C'est ce que Kleene appelle le mouvement perpétuel. Ce mécanisme de recherche à l'infini offert par le schéma *mu* se retrouve dans les boucles *While* des programmes informatiques ; le programme «boucle» tant que $|x^2 - 10|$ est différent de 0.

Traditionnellement, un algorithme était une recette de calcul qui fournissait toujours un résultat après un nombre fini d'opérations ; avec le schéma *mu*, l'univers des algorithmes s'élargit et, dans certains cas, le calcul se prolonge à l'infini et ne donne jamais de résultat. On dénomme parfois de tels algorithmes, des semi-algorithmes.



LA RECHERCHE DE LA SORTIE dans un labyrinthe peut être infinie. Il faut donc borner la recherche en excluant, par exemple, les boucles telle que ACDA.

Maîtriser l'infini

Le schéma *mu* permet de définir des semi-algorithmes qui recherchent un certain objet dans un ensemble infini, sans même savoir si un tel objet existe. Un programme informatique qui recherche la sortie dans un labyrinthe est une illustration du schéma *mu*. Par exemple, on peut chercher un chemin qui va de A à G dans le labyrinthe de l'illustration ci-contre, et l'on trouvera peut-être le chemin A, D, E, F, G. Si l'on cherche un chemin allant de A à J, on n'en trouvera pas.

Comment procède-t-on pour trouver le chemin qui va de A à G?

Le nombre de chemins dans un labyrinthe est infini : on peut très bien se perdre en parcourant une boucle, en passant de A en D, de D en C et de C en A, sans jamais prendre le couloir qui va de D vers E. Pour trouver la sortie d'un labyrinthe, il ne suffit pas de faire la liste des chemins en allant toujours droit devant soi ; il faut énumérer les chemins de manière systématique afin de n'en oublier aucun, par exemple en prenant tous les chemins de longueur 1 (qui sont en nombre fini), puis tous ceux de longueur 2 (qui sont également en nombre fini), puis ceux de longueur 3… L'énumération de tous les chemins de tailles finies du labyrinthe permet de trouver la sortie, si celle-ci existe et correspond à un chemin de taille finie, mais énumérera les chemins éternellement s'il n'en existe pas. Ce risque de recherche à l'infini, dans le cas où le problème n'a pas de solution, est inévitable dans un ensemble infini.

Dans le cas du labyrinthe, il est possible d'éviter cette recherche dans un ensemble infini. Pour traverser un labyrinthe, il n'est jamais nécessaire de passer deux fois par le même carrefour ; s'il y a un chemin qui comporte une boucle, on la supprime et l'on obtient un autre chemin qui mène directement au même endroit.

On se restreint ainsi à chercher une solution parmi les chemins sans boucle. Ces chemins sont alors en nombre fini, et s'il n'y a pas de solution (comme quand on essaie d'aller de A à J), après avoir énuméré tous les chemins et constaté qu'aucun d'eux ne convient, on saura que les deux points ne sont pas reliés. Qu'il y ait une solution ou non, la recherche arrivera à son terme.

Restreindre ainsi l'espace de recherche de manière à le rendre fini, sans pour autant perdre la solution, est le problème principal quand on cherche à concevoir certains algorithmes, par exemple des algorithmes de sortie de labyrinthe : cela s'appelle borner le schéma *mu*. Parfois – comme dans ce cas – cela est possible, mais dans d'autres cas, en particulier quand le problème qu'on cherche à résoudre est indécidable, cela ne l'est pas. Lorsque le problème est indécidable, on ne peut pas toujours, après avoir effectué un nombre fini d'étapes, prédire si la solution existe ou non. Le problème demande une recherche dans un ensemble infini, et donc une recherche à l'infini quand il n'y a pas de solution.

Dans certaines situations, on ne peut éviter la recherche à l'infini, et le schéma *mu* est le bon outil. Mais dans d'autres, par exemple quand on cherche à définir la fonction d'Ackermann, dont les étapes sont certes imprévisibles, mais sont toujours finies, le schéma *mu* semble un outil mal adapté, car trop puissant, puisqu'il convoque l'infini dans sa recherche. En introduisant le schéma *mu*, Kleene a fait entrer le loup dans la bergerie : l'infini dans la théorie des algorithmes qui était *a priori* le cadre du fini.

Cela est-il dû à une maladresse de Kleene ou cette intrusion de l'infini dans l'univers des algorithmes était-elle nécessaire?

Chasser l'infini?

Church et Kleene ont montré qu'il ne s'agissait pas d'une maladresse et que cette intrusion était inévitable. En effet, tous les procédés de définition d'algorithmes qui se limitent à définir des fonctions qui donnent toujours un résultat après un nombre fini d'opérations (c'est-à-dire qui excluent ce mécanisme de recherche à l'infini) sont incomplets : on pourra toujours construire une fonction, calculable par un algorithme en un nombre fini d'étapes, mais qui n'est pas définissable par ce procédé. Cette fonction se construit en utilisant la méthode de la diagonale introduite par Georg Cantor pour étudier les cardinaux infinis (voir l'article de Jean-Paul Delahaye, *L'infini est-il paradoxal en mathématiques?*, page 30).

Il n'y a donc pas de définition qui englobe toutes les fonctions donnant un résultat après un nombre fini d'étapes et rien qu'elles. Certains procédés, comme le schéma *mu*, définissent toutes ces fonctions, mais aussi certaines fonctions qui ne donnent pas de résultat après un nombre fini d'étapes. D'autres procédés comme ceux – plus complexes – proposés par Kurt Gödel en 1941 ou Jean-Yves Girard en 1970 ne permettent de définir que des fonctions qui donnent un résultat en un nombre fini d'étapes. Cependant, ils ne les définissent pas toutes.

Chassez l'infini, il reviendra au galop...

Gilles DOWEK est chargé de recherches à l'Institut national de recherche en informatique et en automatique.

Jean-Luc CHABERT et al., *Histoire des algorithmes*, Belin, 1995.

Richard LASSAIGNE et Michel DE ROUGEMONT, *Logique et fondement de l'informatique*, Hermès, 1993.

René CORI, *Logique mathématique*, Masson, 1993.