Hyper-calcul et vérification

Alan Turing est généralement connu en informatique et en logique pour avoir conçu le modèle de calcul aujourd'hui appelé « machine de Turing » [Turing, 1936]. En informatique car la machine de Turing est le modèle théorique des ordinateurs modernes, en logique puisqu'elle est la formalisation de la notion de fonction calculable par procédure effective.

Turing est pourtant aussi à l'origine de l'« O-machine » qui est une machine de Turing équipée d'un « oracle », à savoir une boîte noire dont le fonctionnement interne n'est pas spécifié, capable de fournir pour certains arguments les résultats de fonctions non calculables par procédure effective [Turing, 1939]. L'O-machine, de part son architecture et sa puissance calculatoire n'est pas un modèle de calcul au sens de la théorie de la calculabilité mais est un modèle d'« hyper-calcul », terme renvoyant à la possibilité de calculer des fonctions non calculables par machine de Turing (non Turing-calculables) [Copeland, 2002a].

Bien que l'hyper-calcul soit possible en logique, cette notion ne fait pas l'unanimité en physique et en philosophie. En effet, la position selon laquelle un modèle d'hyper-calcul pourra être un jour construit physiquement est sévèrement critiquée. En particulier, un des problèmes récurrents lié à cette position est le problème de la vérification [Copeland, 2002b] (p. 491) [Shagrir and Pitowski, 2003] (p. 90) [Davis, 2006] (p. 13).

Le problème de la vérification peut être formulé de la façon suivante : supposons que nous diposons d'un modèle d'hyper-calcul construit physiquement, pouvons-nous vérifier que ce modèle calcule au moins une fonction non Turing-calculable ? Non, puisque pouvoir vérifier chaque résultat calculé par la machine reviendrait à disposer d'une procédure effective nous permettant de suivre le calcul de la donnée initiale au résultat, ce qui est impossible d'après l'hypothèse selon laquelle la fonction n'est pas Turing-calculable. Par conséquent il nous est impossible de déterminer si la machine est capable de calculer une fonction non Turing-calculable.

Le but de ma présentation est de montrer pourquoi la principale réponse proposée par les défenseurs de l'hyper-calcul n'est pas satisfaisante. Cette réponse consiste à dire que le problème de la vérification ne concerne pas

uniquement l'hyper-calcul mais qu'il est aussi valable pour le calcul effectif [Cleland, 2004] (p. 223) [Shagrir and Pitowski, 2003] (p. 99). La raison en est qu'il est impossible de vérifier qu'un ordinateur calcule une fonction totale [1] telle que l'addition car les ressources limitées de l'ordinateur permettent uniquement de vérifier que ce dernier calcule une fonction partielle, fonction qui sera consistante avec une infinité de fonctions différentes de l'addition.

Cette réponse n'est pas satisfaisante car le problème de la vérification appliqué à l'hyper-calcul est plus complexe que celui appliqué au calcul effectif. Plus précisément, la réponse des défenseurs de l'hyper-calcul n'est pertinente que pour un type particulier de vérification, à savoir une vérification en pratique, mais est insatisfaisante pour une vérification en principe.

Je distingue en effet deux types de vérification :

1. Une vérification en principe qui fait abstraction des ressources computationnelles.
2. Une vérification en pratique qui prend en compte les ressources computationnelles.

La réponse des défenseurs de l'hyper-calcul est pertinente en pratique car il est impossible en pratique de vérifier qu'un ordinateur calcule une fonction totale pour chacun de ses arguments. Par exemple, nous n'avons aucun moyen pratique de vérifier qu'un ordinateur calcule correctement la n-ième décimale de π car il est actuellement impossible, à cause d'un manque de ressources computationnelles, de vérifier que la 10^{12}-ème décimale de π est bien égale à 5. Néanmoins, je vais montrer que cette réponse n'est plus pertinente en principe.

Je définis dans ce but deux conditions nécessaires afin de résoudre le problème de la vérification :

1. Pouvoir vérifier que la machine fournie en sortie un résultat correct à partir d'une donnée en entrée.
2. Pouvoir vérifier que la machine calcule une fonction donnée, c'est-à-dire pouvoir vérifier, pour toute donnée en entrée, que la machine fournie en sortie un résultat correct.

En principe, la première condition est satisfaite dans le cas du calcul effectif mais n'est pas satisfaite dans le cas de l'hyper-calcul. Un ordinateur moderne peut en effet être étudié à partir de son modèle théorique qui est la machine de Turing. Etant une formalisation de la notion de procédure effective, la machine de Turing satisfait la contrainte suivante : un être humain doit pouvoir suivre l'algorithme étape par étape, de la donnée initiale

1. Une fonction totale, contrairement à une fonction partielle, produit une valeur pour chaque nombre appartenant à son ensemble de départ.

au résultat indépendamment des contraintes de temps et d'espaces mémoire [Copeland, 2002a] (p. 1). Le nombre d'étapes de calcul d'un ordinateur étant fini, il est donc possible en principe de vérifier un résultat fourni par ce dernier. En revanche, nous ne pouvons pas procéder de la même manière avec un « hyper-ordinateur » car le modèle théorique de ce dernier, un modèle d'hyper-calcul, n'est pas une formalisation de la notion de procédure effective et ne satisfait pas la contrainte énoncée ci-dessus.

De plus, bien que la seconde condition ne soit pas satisfaite en principe à la fois pour le calcul effectif et l'hyper-calcul [Gold, 1965], il est tout de même possible dans le cas du calcul effectif de falsifier certaines hypothèses contradictoires avec nos observations afin de se rapprocher de l'identification de la fonction calculée par l'ordinateur. Par exemple, les résultats $f(1) = 2, f(2) = 4, f(3) = 9$ falsifient notre hypothèse que la fonction calculée est la multiplication par 2. Mais dans le cas où la machine est un hyper-ordinateur, nous sommes incapables de falsifier l'hypothèse selon laquelle la fonction calculée est non Turing-calculable car si nous faisons une telle hypothèse, aucun nombre fini d'observations ne nous permettra de la falsifier et nos observations pourront toujours s'accorder avec une fonction Turing-calculable.

En résumé, même si le calcul effectif et l'hyper-calcul sont tous les deux soumis au problème de la vérification, ce dernier est davantage problèmatique dans le cas de l'hyper-calcul.

Références

C.E. Cleland. The Concept of Computability. *Theoretical Computer Science*, 317 pp. 209-225, 2004.

B.J. Copeland. The Church-Turing Thesis. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2002a.

B.J. Copeland. Hypercomputation. *Minds and Machines*, 12 pp. 461-502, 2002b.

M. Davis. The Myth of Hypercomputation. *in Christof Teuscher (ed), Alan Turing : the life and legacy of a great thinker, Springer*, 2006.

M. Gold. Limiting Recursion. *The Journal of Symbolic Logic*, 30, pp. 28-48, 1965.

O. Shagrir and I. Pitowski. Physical Hypercomputation and the Church-Turing Thesis. *Minds and Machines*, 13 pp. 87-101, 2003.

A.M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1936.

A.M. Turing. Systems of Logic Based on the Ordinals. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1939.