

## Histoire de $\lambda$ -calcul

Qu'est-ce que c'est un  $\lambda$ -calcul ? Les codeurs sont familiers avec une notion de  $\lambda$ -fonction - une fonction anonyme qui est utilisée dans les morceaux du code qui ne méritent pas d'avoir une méthode nommée : clé de trie, les petits transformations dans les requêtes à la sql etc. Cependant, la notion de  $\lambda$ -fonction a été prise d'un système de calcul aussi puissant que la machine de Turing (est inventé dans les mêmes années 30s). Dans cet article on va discuter l'histoire de son invention pour mieux comprendre le concept.

### Plan

1. Crise des fondements
2. Axiomes de Peano
3. 1ere version de Lambda calculs
4. Theoreme de Goedel
5. Machine de Turing et calculabilité
6. Thèse de Church-Turing
7. Applications

### Crise des fondements

À la fin de XX siècle les mathématiciens ont construit une théorie des ensembles : tout le monde est familier avec ça version naïve depuis l'école. Ce désir de formaliser ses fondements a donné lieu des travaux comme les livres de Nicolas Bourbaki, spécialement connu pour sa définition de zéro. */spoiler/*. Cependant, on est vite tombé sur les nombreux paradoxes. Considérons par exemple un paradoxe de Russel qui réclame que l'ensemble des ensembles n'appartenant pas à eux-mêmes est impossible. Effectivement, considérons  $X := \{t | t \notin t\}$ . Soit  $X \in X$ , soit  $X \notin X$ , mais l'un est l'autre sont contradictoire avec une définition de  $X$ . Evidemment, l'existence de ce paradoxe relève des doutes sur les autres domaines de mathématique : et si on trouve les mêmes paradoxes en analyse, algèbre etc.? Cette période est connue comme une "Crise des fondements".

La solution a été de "revisiter" les bases de mathématique est d'étudier l'existence éventuelle des autres paradoxes. Cette crise a été reflétée sous le numéro 2 dans une liste des 23 fameux problèmes de Hilbert qui ont déterminé le développement du mathématique en XXème siècle. Plus précisément,

la deuxième problème a été de déterminer la consistance de l'arithmétique. Même si l'énoncé est simple, la solution nécessite de passer par deux grands étapes : - formuler les axiomes de l'arithmétique - i.e., trouver les proposition "minimales" telles que on peut en déduire tout ce que l'on connaît jusqu'au présent. - prouver que en partant de ces axiomes, il n'existe pas d'une proposition  $X$ , tel que les axiome implique  $X$  ainsi que "non  $X$ ". Pour justifier l'importance de ce problème, voici quelques noms des mathématiciens qui ont contribué dans la solution: Péano, Dedekind, Gödel, Church, Russell, Kleene, Rosser etc.

### **Axiomes de Peano**

Le résultat ? 1) Aux années 1889, Peano a proposé les axiomes de l'arithmétique les plus connues : ...

### **1ère version de Lambda calcul**

2) Vers les années 1932 Church a proposé une autre construction qui a été appelé lambda-calcul. Malheureusement, son étudiant, Kleene a prouvé que cette construction n'a pas été consistante.

$\lambda$ -calcul a formalisé une application d'une fonction. L'écriture envisage la compréhension d'une fonction comme une "règle". Et l'écriture classique  $f(x)$  pointe plutôt sur le résultat de ce règle.

### **Théorème de Gödel**

3) Le résultat le plus connu, a été prouvé par Kurt Gödel : il a prouvé que la consistance d'un système d'axiomes ne peut pas être prouvée en n'utilisent que ces axiomes. Donc pour prouver la consistance d'arithmétique il faut ajouter les axiomes supplémentaires (qui a été vite fait, en 1936). Le seul problème est que maintenant il faut prouver une autre système...

Pour résumer le sujet de l'arithmétique, disons que lambda-calcul a été l'un des modèles qui pourrait formaliser les axiomes de l'arithmétique. Son version actuel a été prouvée consistante et publiée en 1936. Cette construction devait rester un sujet purement théorique qui a intéressé les rares génies de mathématique qui a étudié ses fondaments (on rappelle que beaucoup des personnages de cet article se sont finis mal...).

### **Machine de Turing et calculabilité**

Cependant, comme il est souvent en science, il faudrait étudier le même domaine de point de vue un peu différent. Cela a été fait sur l'autre conti-

nent par un jeune étudiant Alan Turing. Il a cherché une solution pour un problème de la décision posé en 1928 par Hilbert et Ackermann : "trouver un algorithme qui détermine dans un temps fini, s'il un énoncé est vrai ou faux". La formalisation d'un terme algorithme a conduit au concept de machine de Turing connu par tout le monde. Entre autre, le théorème de Gödel a été reformulé en termes d'une machine de Turing. Le résultat a été aussi négatif, connu comme un théorème de Turing-Church: "il existe des énoncés pour lesquels on ne peut pas déterminer" (vérifier l'énoncé et le nom d'un théorème).

### **Thèse de Church-Turing**

Le résultat positif. S'il existe des fonctions, qu'il peuvent pas être décidées, on se pose la question, qu'est-ce que ce sont les fonctions simples, i.e. les fonctions que l'on peut effectivement calculer. Intuitivement, c'est dont la valeur peut être calculée avec un crayon si on a suffisamment de papier et du temps. Mais vous comprenez déjà que les mathématiciens n'acceptent pas les solutions intuitives... Le problème de décision est lié avec un problème de calculabilité. Qu'est-ce que signifie qu'une fonction peut être calculée ? Souvent on se réfère sur "des méthodes d'un crayon et de papier". Indépendamment, chaque des deux a proposé que toute fonction calculable en termes de crayon et papier peut être calculée par son méthode (lambda-calcul ou la machine de Turing). Les deux propositions - ne sont pas des théorèmes, peuvent pas être prouvées car on ne peut pas formaliser autrement calculabilité. (On doit remarquer ici qu'il y avait le troisième mécanisme de déterminer la calculabilité - les fonctions récursives primitives). Relativement vite il a été prouvé que tous les 3 mécanismes sont équivalents. Donc, n'importe lequel peut être utilisé comme une définition de fonction effectivement calculable.

### **Impacts de $\lambda$ -calculs**

1. Formalisation d'une notion de calculabilité.
2. Preuves de calculabilité.
3. Preuves formelles.
4. Programmation fonctionnelle.