Une introduction au λ-calcul.

Ch. Berline

1995, légèrement raffraîchie 2002.

Table des matières

Remarque. On pourra compléter cette introduction informelle, qui date un peu, avec, par exemple, les premières pages de [7], et certains des articles “de base” qui y sont cités. On trouvera également sur ma page web[1] une introduction plus développée sur la Correspondance de Curry-Howard, qu'il pourra être intéressant de regarder par la suite. A l'inverse certains des aspects mentionnés ici (la logique combinatoire et les aspects fondationnels du lambda-calcul) ne seront pas abordés dans ce cours.

0.1 Qu'est-ce que le λ-calcul ?

C'est une *théorie mathématique des fonctions*, issue d'un système logique inventé par Church en 1930 [voir plus bas], dans un but de fondement des mathématiques et du raisonnement prédicatif. Le système originel complet de Church s'est avéré non consistant [Kleene et Rosser, 1935] (ce qui n'est pas un

[1] http ://www.pps.jussieu.fr/˜berline

cas unique à cette époque de recherches de fondements), mais la partie purement mathématique était, elle, très solide : c'est le λ-*calcul* dont il est question dans ce cours, et qui s'est avéré historiquement essentiel pour l'élaboration des notions mathématiques de calculabilite (travaux de Church et Kleene).

L'idée de prendre la notion de fonction et d'application comme primitive remonte à Frege (1890), le fondateur de la Logique Mathématique (voir [1]) ; elle est aussi a la base du système logique de Schönfinkel (1924) (voir [1]), dont la partie mathématique est la *Logique Combinatoire* (Schönfinkel et Curry), une théorie du premier ordre équivalente au λ-calcul, en un sens que nous esquisserons plus bas.

Le λ-calcul est un modèle à la fois très simple et très conceptuel de la calculabilité : deux règles de formation des termes plus, essentiellement, une règle de calcul (la β-réduction). Avec ça on peut représenter toutes les fonctions récursives partielles.

Le λ-calcul est revenu sur le devant de la scène avec le developpement de l'informatique et la création des langages de *programmation fonctionnelle* (LISP, crée par Mc Carthy dans les annees 60, ML, CAML etc.]dont il est le coeur [cette remarque semble due à Landin, 1966] [2]. Son importance théorique s'est encore accrue par l'élaboration actuelle de la Correspondance de Curry-Howard entre preuves et programmes (dont le λ-calcul est un intermédiaire crucial) et de la "*programmation par preuves*", qui produit des programmes dont la correction est assurée par des raisons mathématiques profondes.

On distingue le λ-calcul *pur* (ou "non typé"), qui a, de loin, le plus grand pouvoir d'expressivité au niveau des termes, et les λ-calculs typés, qui sont régulés au moyen de l'ajout d'une couche logique. La correspondance de Curry-Howard n'a de sens que pour les calculs typés.

0.1.1 Intuitions de base.

Intuitivement, tout terme du lambda-calcul représente une "fonction". La notion intuitive de fonction a ici à la fois un sens très général, puisqu'en particulier tout terme du lambda-calcul pur s'applique à tout terme (il n'y a ni domaine ni ensemble image), et un sens calculatoire.

0.1.2 Un calcul.

Au niveau le plus concret c'est, comme son nom l'indique, un calcul ; c'est *le* calcul général de l'évaluation des fonctions le plus naturel qu'on puisse imaginer. La syntaxe de base en est très simple : les termes sont formés à l'aide de variables, de l'opérateur d'abstraction et de l'application, tandis que le calcul consiste a substituer successivement (et correctement) les arguments aux variables correspondantes. Ce calcul (la β-conversion) est défini par un *ensemble*

[2]En théorie la programmation fonctionnelle s'oppose à la programmation impérative (ex : Pascal ou Fortran), à la programmation logique, à la programmation par objets ..). En pratique les langages fonctionnels incorporent aussi des traits impératifs ou autres, pour des raisons d'efficacité.

de règles de réécriture des termes du langage. Le choix d'une stratégie rend le calcul déterministe.

0.1.3 Une théorie équationnelle.

La β-conversion engendre une relation d'équivalence sur les termes (la β-équivalence), qu'on peut présenter comme une théorie de l'égalité entre termes. Elle est définie par des axiomes (qui sont des équations entre termes) et des règles d'inférence de telles équations.

0.1.4 Une théorie du premier ordre.

Le λ-calcul pur n'est pas directement une théorie du premier ordre (telles qu'elles vous ont été définies dans le cours fondamental), à cause de "l"'opérateur d'abstraction. *La Logique Combinatoire* (LC) est une théorie du calcul des prédicats du premier ordre. Son langage contient seulement un symbole binaire pour l'application et deux symboles de constante K et S . Elle admet un système fini d'axiomes $\forall\exists$ très simple et est équivalente au λ-calcul en ce sens qu'il existe des traductions réciproques des termes de l'un dans les termes de l'autre qui échange les équations dérivables dans le λ-calcul avec les conséquences équationnelles de LC. Il en résulte que les deux théories ont le même "modèle des termes" canonique (quoique le mot "modèle" ait ici deux sens un peu différents). Il existe aussi un ensemble fini d'axiomes universels ($\forall$) du même langage (qui sont, à une exception près, des équations : les axiomes de Curry), qui a les mêmes conséquences équationnelles et inéquationnelles que LC. La correspondance précise entre LC et le λ-calcul est traitée dans [6] et [5].

0.1.5 Un modèle de la calculabilité.

La définition formelle basée sur le λ-calcul de la classe des fonctions (partielles) calculables est infiniment plus manipulable et plus transparente que la modèlisation par machines de Türing. Elle est plus riche de concepts que la (les) formalisations de fonctions récursives ; elle ne nécessite pas le choix d'un ensemble de base de fonctions et d'opérateurs (il y a tout de même un choix à faire : celui de la représentation des entiers par des termes, mais ce choix a un sens (logique).

Précisions et nuances dans la Section 0.4.

L'équivalence des trois classes de fonctions calculables a ete montrée avant 1937 et a conforté la Thèse de Church (voir la Section 0.4.1)

0.1.6 Une branche de la logique mathématique et de l'informatique théorique,

Très diversifiée et extrêmement féconde.

Liens forts et éclairants avec les autres (branches).

– **La récursivité**, historiquement et naturellement.

- **La théorie de la démonstration** : l'*isomorphisme de Curry-Howard* est en fait la mise en évidence d'une analogie profonde entre les démonstrations de théorèmes (dans certains systèmes logiques) et certains termes du lambda-calcul. Cette correspondance a lieu plus précisément entre un système de *"lambda-calcul typé"* et une logique associée. Dans un tel système un terme t est "typable" (ou typé) par une ou plusieurs formules A de la logique considérée (les "types" du système) ; l'analogie est :
  [type=théorème],
  [terme=démonstration] (du théorème dans la logique considérée).
  Plus précisément, le terme clos t est de type A ssi t code une démonstration de A.

Pour ceux d'entre vous qui suivent le cours de théorie de la démonstration je précise qu'il s'agit d'une vraie traduction entre systèmes logiques de déduction (déduction naturelle ou calcul des séquents) et systèmes de λ-calcul typé, et que son noyau est de la forme :

A est une premisse	$\longrightarrow$	variable de type A
preuve de A	$\longrightarrow$	terme de type A
preuve sans coupure de A	$\longrightarrow$	terme normal de type A
règle d'introduction de la flèche	$\longrightarrow$	λ-abstraction
règle d'élimination de la flèche	$\longrightarrow$	application
élimination d'une coupure	$\longrightarrow$	β_0-réduction
élimination des coupures	$\longrightarrow$	β-normalisation

- Ceci est le principe de la *programmation par preuves,* où la correction complète des programmes sera assurée si la logique associée est suffisamment expressive et puissante.
  Une traduction analogue donne lieu à une correspondance de Curry-Howard entre des versions typées de la Logique Combinatoire et des systèmes logiques de style Hilbert (les constantes K et S étant alors les traductions respectives des deux axiomes de Hilbert, tandis que l'application code le modus ponens).
- **En programmation fonctionnelle** l'équation est : [terme=programme]. Dans les systèmes typés on ajoute : [type= spécification] (du programme). Plus la logique associée (et en particulier le langage des commentaires, ou spécifications) est expressive, plus le λ-calcul typé correspondant peut-être considéré comme un *langage évolué* de programmation ; les types permettent de plus une vérification automatique très partielle des programmes. Mais de toute manière l'exécution du programme ne fera intervenir que le terme du lambda-calcul pur sous-jacent.
- **Programmation "dure"**(de plus bas niveau). En fait le λ-calcul pur peut-etre considéré comme un *langage machine,* chaque symbole du langage pouvant etre vu (et implementé, ex. dans la "machine de Krivine") comme une instruction "machine", de telle manière que faire tourner la

"machine λ-calcul" sur un terme revient à le β-réduire en suivant une stratégie fixée à l'avance.

- Le lambda-calcul admet diverses sémantiques et diverses **théories des modèles**, d'inspiration variée. Beaucoup reste à faire au niveau des applications potentielles de ces sémantiques et de leur compréhension. Il est à noter ici que, bien que le lambda-calcul puisse être vu (partiellement) comme une théorie du premier ordre, à savoir *la Logique Combinatoire*, le point de vue sur les modèles est ici différent (d'une part parce qu'on ne s'intéresse souvent qu'à leur *théorie équationnelle*, mais surtout parce que les problèmatiques sont différentes). Le cadre abstrait le plus général pour parler de sémantique du λ-calcul est celui des *catégories cartésiennes closes,* que nous ne ferons qu'effleurer dans ce cours. ajouter qqs mots
- **Théorie des ensembles naïve.** Il existe aussi un "isomorphisme" "naïf" entre le λ-calcul et la théorie naïve des ensembles (+ calcul propositionnel). Ici il s'agit plutot d'un parallèle (une traduction sans démonstration d'équivalence à un niveau structurel profond ... pour autant que je le sache actuellement). Note : je ne donne dans les lignes suivantes que ce qui est strictement nécessaire pour traduire le paradoxe de Russel.

ensemble	$\longrightarrow$	fonction
appartenance	$\longrightarrow$	application
formation d'ensemble	$\longrightarrow$	λ-abstraction
égalité entre ensembles	$\longrightarrow$	β-équivalence
négation	$\longrightarrow$	une constante de terme "$\neg$"
équivalence de propositions	$\longrightarrow$	β-équivalence

La traduction du paradoxe de Russel en λ-calcul pur est ainsi :

$$(\lambda x.\neg(x)x)\lambda x.\neg(x)x =_\beta \neg\ (\lambda x.\neg(x)x)\lambda x.\neg(x)x$$

Mais ceci n'a rien de contradictoire (car en λ-calcul pur tout terme, y compris $\neg$, y admet des points fixes). Cette correspondance etait présente dans l'esprit de Church et l'a guidé dans l'élaboration de son système, lui permettant d'espérer qu'il n'etait pas contradictoire ; le contre-exemple de Kleene et Rosser est toutefois de la même famille. Dans son deuxieme article Church supprime quatre axiomes dont découlent cette première contradiction.

- **Map Theory** [Hors Cours !] est un système logique général dû à K. Grue [1992] et proposé par lui comme une alternative à ZFC (+calcul des prédicats). Il remplit les objectifs initiaux de Church en ce sens que c'est une théorie équationnelle qui étend le λ-calcul pur et où l'on peut interpréter les mathématiques (y compris la théorie des ensembles), et le raisonnement logique usuel. D'autre part il est directement capable de traiter de la calculabilité, au moyen du λ-calcul. Le système de Grue est consistant, très bien conçu, et beaucoup plus simple que celui de Church (qui ne comptait pas moins de 5 règles et 37 axiomes, axiomatisant en particulier le comportement de 6 symboles de constantes a visée logique). Le système de Grue ne comporte qu'un symbole à visée strictement logique : c'est essentiellement l'opérateur de Hilbert [3], dont un autre avatar est l'une des constantes logiques de Church. Cet opérateur, convenablement axiomatisé, suffit pour définir les quantificateurs et assurer la véracite de l'axiome du choix. Il existe une correspondance non triviale entre modèles de Map Theory et univers de la Théorie des ensembles. Il existe une interprétation syntaxique de ZFC (+ calcul des prédicats) dans Map Theory ; cette interpretation est différente toutefois de la traduction naïve évoquée plus haut .

- Il existe **d'autres systèmes fondationnels,** en général basés sur des lambda-calculs typés. Le plus connu est la Théorie des Types de Martin-Löf.

Comme vous avez pu remarquer, le λ-calcul est un sujet a multiples facettes.

0.2 Ce que nous ferons dans le cours.

(Se reporter à la brochure du DEA de l'année !)

- Nous présenterons la syntaxe du lambda-calcul pur, celle du lambda calcul simplement typé (qui correspond à la logique propositionnelle intuitioniste du premier ordre construite sur $\rightarrow$ uniquement), d'un système typé (le système E) qui ne correspond pas à une logique "raisonable" mais qui offre un intérêt pour l'étude du λ-calcul pur et qui a un lien profond avec les sémantiques que nous étudierons.
- Nous présenterons les sémantiques fonctionnelles basiques du λ-calcul, à savoir les sémantiques continues et stables. Nous donnerons des méthodes pratiques de construction de modèles du λ-calcul non typé, et nous discuterons de leurs théories équationnelles.
- Nous étudierons les systèmes F et $AF2$, qui correspondent au calcul propositionnel (resp. au calcul des prédicats) intuitioniste du second ordre sur $\rightarrow$ et $\forall$, et qui sont suffisamment puissants pour qu'on puisse se rendre compte pleinement, et mathématiquement, de l'intérêt de la correspondance de Curry-Howard [voir les intros des chapitres correspondants du cours].

Références

[1] J. van Heijenoort, From Frege to Godel, A source book in Mathematical Logic, 1879-1931, ed. by van Heijenoort, Harvard University Press, 3rd printing 1977.

[2] P. Odifreddi, Classical Recursion Theory, Studies in Logic vol.125, North Holland, 1989.

[3](utilisé aussi par Bourbaki pour présenter la théorie des ensembles)

[3] A. Church, A set of postulates for the foundations of logic I, Ann. Math 33, p.346-366 (1932).

[4] A. Church, A set of postulates for the foundations of logic II, Ann. Math 34, p.839-864 (1933).

[5] H. Barendregt, The λ-calculus, Studies in Logic, North Holland 1984.

[6] J.L. Krivine, λ-calcul, types et modèles, Masson 1990 ; version anglaise étendue chez Ellis et Horwood (1993).

[7] C. Berline, From computation to foundations via functions and application : the λ-calculus and its webbed models, TCS 249, p.81-161, 2000.

[COMPLETER] avec la bibliographie donnée dans l'introduction de [7], à laquelle on pourra ajouter par exemple le gros article de synthèse de Barendregt sur les systèmes typés (1992 ?), dans le Handbook of Logic for Computer Science, tome II ?..

0.3 Commentaires sur la bibliographie.

Les articles de Church sont uniquement donnés a titre de curiosité historique. Ils sont intéressants en ce qui concerne la genèse des grandes idées mathématiques et logiques, ainsi d'ailleurs que le livre de van Heijenoort [1], excellent pour la culture logique mais qui ne contient que des articles antérieurs au λ-calcul ; on y trouvera toutefois l'article de Schönfinkel (1924) sur la naissance des combinateurs . Les préfaces aux divers articles de [1] sont remarquablement bien faites. Les premières pages de l'article de Church expliquent très clairement la distinction des niveaux intuition/formalisme.

Le livre de Barendregt [5] a été longtemps le seul livre de mathématiques sur le lambda-calcul [nuancer : livres de Stoy et Hindley]. C'est toujours un livre de référence basique, et il contient de plus de nombreux très bons exercices ; en revanche il ne traite quasiment que du λ-calcul non typé, et en particulier pas de la Correspondance de Curry-Howard. Le livre de Krivine [6] est beaucoup plus ciblé et donne les bases mathématiques de la correspondance de Curry-Howard ; c'est le livre auquel je me réfèrerai pour la partie syntaxique du cours et l'étude des systèmes F et $AF2$, et aussi, en partie seulement, pour les méthodes de construction de modèles qui y sont présentées.

Les 200 premieres pages du livre de Odifreddi [2] sont très agréables à lire, et a mon avis irremplaçables pour qui veut comprendre ce qui se cache derrière les diverses formalisations mathématiques de la notion de calculabilité. On en trouvera 7 ou 8 d'esprit différent, y compris le λ-calcul, livrées avec les intuitions du sujet, la description des liens, la démonstration des équivalences (avec rigueur mais sans le côté fastidieux des techniques), des liens avec la programmation impérative, diverses variations autour de la Thèse de Church, les commentaires historiques, etc.

0.4 Remarques diverses.

0.4.1 Sur la notion de fonction.

La notion de fonction est une notion ambigüe.

En mathématique les fonctions sont la pluspart du temps considérées du point de vue “extensionnel”, ce qui veut dire qu'on ne s'interesse qu'à savoir si la fonction est définie sur son argument et, si oui, à la valeur qu'elle y prend, en résumé on ne s'intéresse qu'au résultat et pas à la façon dont on y est parvenu [si ce n'est pour vérifier sa correction mathématique]. Ceci revient à identifier une fonction et son graphe, en ajoutant éventuellement une ”valeur” indéfinie $\uparrow$ pour pouvoir parler des fonctions partielles.

En théorie des ensembles (par ex. ZF), ceci est rendu de façon particulièrement nette puisque les fonctions y sont définies comme des ensembles, à savoir des graphes.

Dans la vie courante, qu'elle soit concrète ou mathématique, et évidemment dès que l'on calcule, il en va forcément différemment. Pour des raisons d'efficacité d'une part, et, plus fondamentalement, parce que nous ne pouvons manipuler concrètement que du fini, et ne pouvons donc manipuler l'infini que potentiellement, c'est a dire sous des descriptions finies ou plus généralement localement finies (schémas de règles). *Un exemple :*

L'apprentissage des fonctions les plus usuelles que sont l'addition et de la multiplication se fait d'abord par leurs tables, qui sont des morceaux de graphes ; puis par l'apprentissage d'algorithmes (les techniques opératoires usuelles) et des propriétés des opérations (commutativité, etc). On peut voir ça aussi comme l'apprentissage d'un certain nombre de propositions dont on admet qu'elles sont ”vraies” et de règles (redondantes) pour en déduire d'autres propositions utiles et vraies sur le même sujet. Et tout ceci repose évidemment sur une notion d'entier prise comme primitive : “le” sens est uniquement au niveau intuitif, non explicite, en général non commenté, et succeptible d'évolution. Les réflexions sur cette notion primitive d'entier et sur le choix et la validité des règles générales d'utilisation relèvent du domaine de la philosophie, de la logique, et des “fondements” (des mathématiques et de la logique).

Dans les théories qui s'occupent de calculabilité, le mot ”fonction” a, suivant les cas, une signification extensionnelle, comme ci-dessus, ou une signification intentionnelle précise. Il s'agit ici de fonctions de $N^n \rightarrow N$, $n \geq 1$.

- Toute formalisation précise de la notion de fonction calculable, que ce soient les diverses classes de fonctions récursives générales (au sens de Gödel), les classes correspondant aux diverses familles de machines de Türing ou autres, ou la classe de fonctions représentables dans le λ-calcul (une fois fixée une représentation des entiers), introduit les fonctions en tant qu'algorithmes, c'est à dire comme procédures précises de calcul. Une même fonction “extensionnelle”, autrement dit un graphe de fonction calculable, aura une infinité de représentants “intentionnels” dans chaque classe.
- L'affirmation de l'équivalence des diverses formalisations de la notion in-

tuitive de fonction calculable, que ce soit à l'intérieur de la théorie de la récursivité ou dans les autres théories de la calculabilité, se place au niveau extensionnel : elle signifie précisément qu'au niveau des graphes les classes sont indistinguables. La Thèse de Church exprime qu'il en sera ainsi de toute formalisation un jour imaginable ; on peut lui donner un contenu plus intentionnel, on tombe alors dans les "super thèses de Church" (on trouvera dans [2] des variations de toute nature sur ce thème).

0.4.2 Sur la notion d'entier et sa représentation en λ-calcul.

Dans N et dans l'arithmétique les entiers (intuitifs) sont "chez eux".

L'entier n a de facon naturelle une interprétation de niveau 0 : qu'on se le représente comme n bâtons ou sous la forme plus sophistiquée du n-ième successeur de zéro, c'est dans tous les cas un objet qu'on peut aisement manipuler mais dont l'action sur les autres requiert une intervention, une technologie, extérieure.

En Théorie des ensembles les entiers ont des représentations très concrètes (par ex. celle de Von Neumann), et se voient ornés d'une signification plus abstraite en tant que cardinaux finis ; ils n'ont toujours aucun caractère opératoire.

En Théorie de la récursivité on passe son temps à donner des possibilités opératoires aux entiers puisque tout entier est le code d'une fonction récursive... une fois fixés un ensemble de fonctions et d'opérateurs primitifs ainsi qu'un codage arbitraire non transparent des éléments de cet ensemble ; en d'autre termes un entier a des possibilités opératoires (conférées arbitrairement par l'extérieur) mais pas de sens opératoire.

En λ-calcul le choix de départ, et en fait le seul, est sur la représentation des entiers, c'est à dire sur le sens opératoire qu'on leur donnera : du fait même de la nature de la théorie et des motivations qui sont derrière, on est en mesure d'interpréter du sens, et du sens opérationnel ; la représentation que nous utiliserons dans ce cours est celle adoptée par Church (1933) ; un choix différent est fait dans [5].

On peut alors définir directement *la* classe des fonctions de N dans N qui sont représentables dans le λ-calcul, et qui se trouve coïncider avec toutes les autres classes de fonctions calculables. Nul codage, nul arbitraire.

L'entier de Church $\underline{n}$ est le terme $\lambda f.\lambda x.(f)(f)...(f)x$ qui contient n exemplaires de f et dont le sens est : "*faire n fois*". Appliqué a un terme t, il rend essentiellement le composé de t n fois avec lui même.

0.4.3 Universalité des termes du λ -calcul pur.

Un terme du λ-calcul *pur* "est" une fonction, sans notion de domaine ni d'ensemble image ; tout s'applique à tout, en particulier à soi-même. Les notions de domaine et d'image peuvent être réintroduites par le typage qui peut traduire partiellement l'intention : t est de type $A \to B$ si t est une fonction de A dans B. Les sémantiques fonctionnelles du λ-calcul, que nous étudierons, permettent d'interpréter les termes par de vraies fonctions et les types par les domaines correspondants.

0.4.4 Sur le rôle opérationnel des termes.

Chaque terme a un rôle opérationnel non déterministe unique ; les règles de calcul sont des règles d'évaluation fixées une fois pour toute ; en les raffinant on peut spécifier une stratégie de calcul (ex : reduction gauche), chaque terme aura alors un rôle opérationnel déterministe (unique).

0.4.5 Pluri-intentionalité potentielle des λ-termes.

L'exemple le plus simple en est le terme $\lambda x.\lambda y.y$ qui peut représenter suivant la nécessité : la deuxieme projection π_2^2, l'entier 0, le booleen F ($Faux$). Son comportement opérationnel (unique) est en accord avec chacune de ces trois intentionnalités.