La syntaxe et son importance

November 2023

1 Introduction

Ce document a été rédigé par un étudiant de L2 mathématiques au mois de Novembre, aussi n'a t-il aucune valeur académique ou pédagogique avérée. Il s'adresse à tous ceux qui sont intéressés par ce sujet, et en particulier aux non-mathématiciens étrangers aux concepts évoqués ici, mais les parties 2 et 3 sont plus faciles à comprendre pour un étudiant qui connait un peu les objets dont il est question, même si leur définition est rappelée. Dans mon (trop) court voyage au pays de l'informatique théorique, la leçon la plus importante que j'ai l'impression d'avoir compris est l'importance de la syntaxe. On peut facilement avoir l'impression, en mathématiques, qu'on manipule des idées, des intuitions, puis qu'on les formalise pour obtenir un résultat mathématique. Cependant, on oublie que tout ce qui se passe sur une copie (et idéalement, dans une boîte crânienne) n'est qu'une manipulation de symboles, en suivant des règles très précises. C'est une idée très déconcertante, quand on y pense. Toutes les fonctions qu'on a l'impression de visualiser, les groupes qu'on a l'impression de comprendre, tout ça n'est théoriquement qu'un empilement de symboles, qui interragissent entre eux de façon parfois étonnante. En premier lieu, nous expliquerons la différence entre syntaxe et sémantique (cette définition sera assez informelle, proche du sens usuel), puis nous donnerons trois exemples (par difficulté croissante de compréhension) qui illustrent la syntaxe qui se tapit derrière le sens des objets qu'on manipule. Le premier portera sur l'addition des entiers naturels, le second portera sur les groupes, et le troisième sera une introduction au λ-calcul, mets de choix pour les amateurs de beauté syntaxique.

2 La syntaxe et la sémantique

D'après CNTRL, la syntaxe c'est l'"Arrangement des mots et construction des propositions dans la phrase selon les règles de la grammaire." Il s'agit d'une définition qui convient pour le langage usuel, le français. Cependant, il existe une flagrante analogie entre ce qu'on dit en français et ce qu'on dit en mathématiques. D'ailleurs, la plupart des résultats mathématiques peuvent se dire en français, et en modélisant bien les problèmes décrits par des phrases en français on peut très souvent les décrire mathématiquement. Quand on était petit, on se demandait si l'on "avait le droit" de faire quelque chose. "Est-ce qu'on a le droit de passer à l'inverse dans une égalité ?" Oui, pour des membres non nuls. "Est-ce qu'on a le droit d'inverser dans une inégalité ?" Non. Cette idée d'avoir le "droit" retranscrit bien l'idée de syntaxe : depuis un résultat, on a besoin d'avoir une règle précise pour arriver à un autre résultats. Et c'est de cette façon qu'on construit toute la théorie : on part de ces règles "originelles" (qu'on appelle axiomes), puis on en construit de nouvelles (qu'on appelle lemme, théorème, résultat, identité, j'en oublie sûrement). C'est ce qui se passe aussi bien lorsqu'on écrit 1+1=2, que lorsqu'on démontre le théorème de pythagore. Toujours d'après CNTRL (cette fois ci j'ai choisi la définition la plus proche possible du sens usuel), la sémantique est l'"Étude d'une langue ou des langues considérées du point de vue de la signification; théorie tentant de rendre compte des structures et des phénomènes de la signification dans une langue ou dans le langage". C'est donner du sens aux mots, étudier leur signification. Par exemple, un isomorphisme entre deux structures algébriques est une application bijective qui préserve leur structure, mais nous l'interprétons comme une preuve de similitude : les deux structures sont en fait deux représentations d'un même objet, c'est le sens que nous donnons à l'isomorphisme. Souvent, on manipule des objets très basiques par leur sémantique, bien avant de les définir syntaxiquement. Par exemple, au lycée on manipule des entiers naturels depuis plus de 10 ans, mais très peu de lycéens peuvent donner une définition formelle, syntaxique, de ces derniers. C'est normal, bien sûr : nous sommes des humains et non des machines, nous manipulons bien mieux les "idées", le "sens", que des objets formels (à mon avis, ça s'explique assez facilement d'ailleurs. Une machine formelle aurait eu beaucoup de mal à traverser le processus évolutif par lequel l'espèce humaine passe depuis 300 000 ans).

3 Les entiers naturels

Intuitivement, les entiers naturels sont les nombres que l'on pourrait compter sur nos doigts si l'on avait une infinité de mains. Cependant, lorsqu'on déf¡init les entiers naturels par récurrence, on adopte la définition suivante : "Un entier naturel est soit 0, soit le successeur d'un entier naturel". On peut écrire cette définition de façon formelle via la "BNF" (Forme de Backus-Naur) suivante : $n ::= 0 | Sn'$. Cette écriture au premier abord un peu barbare a des fondations théoriques parfaitement claires, mais un peu trop longues pour ce document. Le lecteur intéressé pourra consulter la page wikipedia, voire un cours sur l'induction, mais nous allons décortiquer ici brièvement le sens qu'on donne à cette notation. Elle décrit l'ensemble des entiers naturels par n, la barre indique une disjonction : n est soit 0 soit S n'. On considère que n' est un entier naturel, et que S n' est son successeur. Voilà qui colle bien à notre définition par récurrence : un entier naturel n est soit 0, soit le successeur d'un autre entier naturel n'. Alors, le nombre 1 est simplement un raccourci pour S 0, 2 est un raccourci pour S (S 0), etc. Maintenant qu'on a des entiers naturels, on peut essayer de construire l'addition. L'addition est techniquement une fonction. Ce mot a différentes définitions (la troisième partie de ce document en parle un peu), mais ici on considère une fonction comme un processus qu'on peut suivre à partir d'un ou plusieurs arguments pour obtenir un résultat. Notons que cette définition ne garantit pas que le processus peut être suivi en un temps fini (ça, c'est une fonction calculable), ni n'inclut des considérations mathématiques usuelles (par exemple une fonction peut donner deux résultat différent sur un même élément. On peut dire que la fonction qui à une vache associe le résultat d'un dé qu'on aura lancé peut donner les résultats 1 et 3 sur la même vache...). Elle est donc assez légère. On définit la fonction addition suivante :

```
addition (a : entier naturel) (b : entier naturel) :
        si a = 0, b
        si a = S a', addition a' (S b)
```

Pour se convaincre que cette définition est correcte, le lecteur pourra essayer de faire des additions simples en se collant à cette définition, par exemple 0 + 3, 1+1, ou pour les plus téméraires 5 + 0. On a donc vu que la définition intuitive qu'on avait de l'addition colle parfaitement à cette définition syntaxique, pourtant au fond le sens des objets qu'on manipule reste le même. On a donc "mis sous le tapis" (et c'est une bonne chose au début) cet aspect formel, syntaxique, de l'addition qu'on utilise. Cependant, on n'a fait ici qu'effleurer la théorie derrière les additions. On sait que pour un entier naturel a, a + 0 = 0. Le lecteur pourra essayer de le prouver avec cette définition de l'addition, mais c'est un exercice assez difficile. Pour le traiter, et d'autres du même genre, on pourra s'essayer à Coq, un logiciel de preuves, notamment au chapitre 1 et 2 de 'Software Foundations' (https://softwarefoundations.cis.upenn.edu/lf-current/toc.html).

4 Les groupes, les mots

Un monoïde est la donnée de 2 objets : un ensemble M, et une loi (qu'on ne note pas, ici. On pourrait par exemple la note $\times$, mais on considère que ab est l'élément obtenu en appliquant la loi à a et b) qui a les propriétés suivantes :

- associativité : Pour tous éléments a, b, c de M, (ab)c = a(bc)
- la loi est interne : Pour tous éléments a et b de M, ab appartient à M
- existence d'un neutre : il existe un élément neutre, généralement noté e, tel que pour tout élément x de M, ex = xe = x

Voici quelques exemples de monoïdes : $\mathbb{N}, +; \mathbb{N}, \times; \mathbb{R}^*, \times$... On s'intéresse à un monoïde particulier : un langage. On se donne un alphabet (qui est simplement un ensemble de symboles) et un symbole "neutre" noté e, et on considère le monoïde formé par les "mots" de cet alphabet : tout ce qu'on peut obtenir en concaténant des lettres de notre alphabet. Par exemple, avec un alphabet a,b,c,e (où e est le neutre), ab, abc, cccccccccccccccccc, ou e sont des mots de notre langage. Un groupe est un monoïde auquel on ajoute une règle : tous les éléments de notre ensemble (qu'on appelle désormais G) doivent avoir un inverse, c'est à dire que pour tout élément $g \in G$, il existe dans G un élément g^{-1} tel que $gg^{-1} = e$. Voici quelques exemples de groupe $\mathbb{Z}, +; \mathbb{Q}^*, \times$ les positions de la petite aiguille sur l'horloge (de 0 à 11), munie de l'addition... Pour un groupe dont l'ensemble sous jacent est $\{g_1, ..., g_n, g_1^{-1}, ..., g_n^{-1}\}$ (rappelons que l'un des g_i est le neutre) on peut dire que le groupe vaut $< g_1, ..., g_n >$. Cette notation peut être définie de plusieurs façons (notamment : $< g_i, ..., g_k >$ est le plus petit sous groupe contenant $g_1, ..., g_k$, ou encore l'intersection de tous les groupes contenant $g_1, ..., g_k$) , mais une définition appropriée ici est la suivante : $< g_1, ..., g_n >$ est l'ensemble des éléments qu'on peut générer avec $\{g_1, ..., g_n, g_1^{-1}, ..., g_n^{-1}\}$. D'un point de vue syntaxique, si on prend $\{g_1, ..., g_n, g_1^{-1}, ..., g_n^{-1}\}$ pour alphabet, on voit que le langage qu'on obtient est exactement $< g_1, ..., g_n >$. Cependant, ce langage est un ensemble, et donc il contient notamment e et $g_1g_1^{-1}$. Ce n'est pas incohérent : du point de vue du monoïde ces éléments sont bien différents. Mais si l'on adopte la sémantique du groupe, $g_1g_1^{-1}$ et e sont des éléments identiques. L'ajout de la règle de l'inversion nous fait donc passer du sens qu'on donne $g_1g_1^{-1}$ à une égalité syntaxique $g_1g_1^{-1} = e$.

5 Lambda calcul non typé

Cette section aborde un sujet avancé très lié à la syntaxe. Pour une introduction plus complète, je vous invite à consulter le début du cours de P Selinger (https://www.irif.fr/ mellies/mpri/mpri-ens/biblio/Selinger-Lambda-Calculus-Notes.pdf), qui est accessible pour tous ceux qui maitrisent les définitions et raisonnements par induction.

Pourquoi le lambda calcul ?

Le λ-calcul est une modélisation "informatique" du concept de fonction. En mathématiques, une fonction peut être définie par un graphe Γ, c'est à dire un ensemble de couples (x,y). On interprète l'appartenance de (x,y) à Γ comme l'égalité f(x)=y. On a alors défini f par toutes ses images. Cette définition a des différences conceptuelles avec celle de la partie 2, qui n'utilise pas d'ensemble. Elle présente des "trous" conceptuels. Par exemple, l'identité : c'est la fonction qui "ne fait rien", dont toutes les images valent leur antécédent. Pour définir l'identité, avec la définition mathématique classique, on a besoin de donner un ensemble de départ (les x décriront cet ensemble). Le souci est qu'intuitivement, l'identité n'utilise pas de donnée sur l'objet qu'elle prend en argument : il n'est pas possible (usuellement) de définir rigoureusement l'identité indépendamment de la nature des objets qu'elle prend en argument, alors que la façon dont l'identité "agit" (ou plutôt n'agit pas) sur ces objets ne dépend pas de leur nature. On peut définir l'identité sur les entiers naturels, l'identité sur $\{0, 1\}$, mais pas l'identité *en général*. On peut donc légitimement vouloir chercher des façons de décrire les fonctions qui permettent de définir l'identité *en général*. Ce n'est pas la raison (ni historique, ni "technique") d'être du λ-calcul, mais c'est un premier exemple de "trou syntaxique" dans les mathématiques "usuelles" : l'intuition d'une identité *en général* existe, mais la syntaxe ensembliste qu'on adopte en général n'est pas capable de la décrire. Le λ-calcul permet de recouvrir ce trou, mais ce n'est qu'une toute petite partie de son intérêt dans le paysage mathématique, informatique et logique.

Définitions

Sans se soucier de considérations ensemblistes, on note la fonction qui à x associe f(x) $\lambda x.f(x)$. Par exemple, la fonction carré se note $\lambda x.x^2$.
On définit le λ-calcul par $M, N ::= x|\lambda x.M|MN$. Cette forme de Backus-Naur se lit comme suit : un élément du λ-calcul (appelé λ-terme) est soit une variable x, soit une application qui à x associe M (appelée λ-abstraction), soit une composition de deux λ-termes M et N (notée MN). Dans ce document on essaie de faire l'impasse sur toutes les considérations de nommage de variable, on considère que toutes les variables utilisées sont différentes, c'est à dire que dans un λ-terme toutes les variables (x,y,z,m,....) qui sont "liées" (c'est à dire après un λ et avant un point) sont deux à deux différentes.

La Béta réduction

Nos objets sont bien définis comme des fonctions, mais on n'a rien pour "faire marcher" ces fonctions. Par exemple, $(\lambda x.x^2)y$ est clairement la fonction carrée appliquée à y, mais aucune règle ne nous permet d'affirmer que c'est *égal* à y^2. En réalité, ce ne sont pas deux expressions *égales* mais plutôt β-équivalentes. Comme vu dans la partie 1, en mathématiques l'égalité est "conceptuelle" : on peut la définir sur des entiers naturels (comme on l'a fait pour l'addition), ou sur des rationnels ($\frac{p}{q} = \frac{p'}{q'}$ ssi pq' = qp'), par exemple. Ces deux définitions sont clairement différentes, pourtant on utilise le même signe pour les désigner (ce qui est plutôt naturel puisque dans notre tête l'égalité est la même) : l'idée d'égalité n'est pas *bien* retranscrite syntaxiquement. C'est pour ça que dans un monde syntaxiquement parfait, on n'utilise pas l'égalité mathématique usuelle, on définit notre "propre" égalité, en tout cas notre propre relation capable de décrire la similitude de deux objets. Cette notion est la β-équivalence. Mais pour ça, définissons d'abord la β-réduction par les règles suivantes : La β-réduction peut être définie par les règles suivantes :

$$(\beta)\ \frac{}{(\lambda x.M)N \rightarrow_\beta M[N/x]}$$

$$(cong1)\ \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M'N}\ (cong2)\ \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta MN'}$$

$$(\xi)\ \frac{M \rightarrow_\beta M'}{(\lambda x.M) \rightarrow_\beta (\lambda x.M')}$$

Une règle de la forme

$$\frac{h_1, ..., h_n}{c}$$

Se lit "Sous les hypothèses $h_1, ..., h_n$ on peut déduire c". La flèche $\rightarrow_\beta$ se lit "se β-réduit en". Voici la retranscription et l'explication du contenu de ces règles :

- (β) : $(\lambda x.M)N$ se β-réduit en M[N/x], qui signifie "M dans lequel toutes les occurrences de x sont remplacées par N". C'est la règle "la plus importante", celle dans laquelle l'essentiel de la mécanique d'une β-réduction réside. Par exemple, c'est celle ci qui nous permet de dire que $(\lambda x.x^2)y \rightarrow_\beta y^2$
- (cong1) et (cong2) disent que si un terme M se réduit à un terme M', alors en composant par un terme N à droite où à gauche notre réduction est "préservée" : on a bien $NM \rightarrow_\beta NM'$ et $MNrightarrow_\beta M'N$.
- (ξ) dit que si un terme M se réduit à un terme M', en "liant" ce terme à une nouvelle variable, la réduction est préservée.

Techniquement la β-réduction est donc la plus petite relation qui respecte ces règles, et qui est transitive (i.e pour des λ-termes M, N et P si on a $M \rightarrow_\beta N$ et $N \rightarrow_\beta P$ alors on a $M \rightarrow_\beta P$). La β-équivalence, notée $=_\beta$ est sa "clotûre réflexive symétrique", c'est à dire que c'est une relation réflexive (i.e pour tout terme M, M $=_\beta$ M), et symétrique (i.e si M $=_\beta$ N, alors N $=_\beta$ M). Nous avons défini une façon pour les λ-termes de se comporter entre eux (β-réduction) et d'être comparés (β-équivalence). Voyons un court exemple de pourquoi le lambda calcul permet d'encoder certaines données et d'obtenir tous les résultats qu'on connaît dessus sans mettre de syntaxe sous le tapis.

Encodage des booléens

On appelle les termes $\lambda x.\lambda y.x$ et $\lambda x.\lambda y.y$ "VRAI" et "FAUX". Ici, aucun changement dans notre syntaxe n'a été fait : on donne simplement un nouveau nom à deux termes. À priori, ce nommage est étonnant, lu littéralement il signifie que "VRAI" serait une fonction qui prend en argument une fonction, ce qui est assez éloigné de la définition usuelle du mot (ou de son utilisation

mathématique). C'est ce que cette partie essaie de montrer : en se plaçant dans une syntaxe, et en empruntant des concepts à une sémantique, on peut "voir leur équivalent" dans la syntaxe qu'on adopte. Ici par exemple, on va voir qu'en se servant de la syntaxe on va pouvoir retrouver les éléments de la logique booléenne depuis ces deux termes. Commençons par retrouver la fonction "OU" comme λ-terme. Il s'agit d'une fonction, donc le λ-terme correspondant commence par $\lambda a.....$ Imaginons que cette fonction soit composée avec "VRAI", alors elle doit se réduire en VRAI (on dit que VRAI est absorbant pour OU). Sinon, elle doit se ramener à l'élément suivant avec lequel elle sera composée : OU prend intuitivement deux arguments, mais nos lambdas termes ne prennent qu'un seul argument au maximum. En fait, il est possible de simuler le fait de prendre plusieurs arguments par des fonctions à un argument : c'est la Curryfication. Par exemple, la somme prend deux entiers et s'évalue en un troisième. On peut aussi la voir comme la fonction qui prend un entier n, et s'évalue en une fonction qui prend un entier m et qui s'évalue en n+m. Alors, (le terme n'est pas correct, il retranscrit l'idée d'une fonction à deux arguemnts) $\lambda(n, m).m + n$ est $\lambda n.\lambda m.n + m$. Notre fonction commence donc probablement par $\lambda a.\lambda b....$, et se réduit en VRAI si le premier argument est VRAI. En revenant à la définition de VRAI, on s'apperçoit que VRAI se comporte comme une fonction qui choisit le premier argument qui lui est donné, et "mange" le second sans rien en faire. On peut d'ailleurs pour deux termes M et N réduire VRAI M N : $VRAIMN = (\lambda x.\lambda y.x)MN \to_\beta (\lambda y.M)N \to_\beta M$ (on applique deux fois la règle (β)). Pour en revenir à notre OU, il doit choisir VRAI si on lui donne VRAI comme premier argument, et le second argument sinon. On peut donc essayer de voir si $\lambda a.\lambda b.aVRAIb$ fonctionne, puisqu'il répond à ces conditions nécessaires. Et surprise : en réduisant ce terme, ça fonctionne à chaque fois. En voici une preuve :
$OU\ VRAI\ M = (\lambda a.\lambda b.a\ VRAI\ b)\ VRAI\ M = (\lambda a.\lambda b.a\ VRAI\ b)(\lambda x.\lambda y.x)\ M \to_\beta (\lambda b.(\lambda x.\lambda y.x)\ VRAI\ b)\ M \to_\beta (\lambda b.(\lambda y.VRAI)b)\ M \to_\beta (\lambda y.VRAI)M \to_\beta VRAI.$
On a d'abord prouvé que VRAI est absorbant à gauche, c'est à dire que pour tout terme M, OU VRAI M est β-équivalent à VRAI (en particulier, il se β-réduit en VRAI). Montrons maintenant que VRAI est un élément absorbant à droite :
$OU\ M\ VRAI = (\lambda a.\lambda b.a\ VRAI\ b)\ M\ VRAI = (\lambda a.\lambda b.a\ VRAI\ b)\ M\ VRAI \to_\beta (\lambda b.M\ VRAI\ b)VRAI \to_\beta MVRAIVRAI$ À ce stade, quel que soit M (VRAI ou FAUX), l'expression se β-réduira en VRAI. C'est l'occasion de rappeler que ces termes n'ont pas d'autre sens que celui que l'on leur donne : ici OU est appliqué à des termes dont on suppose qu'ils sont VRAI ou FAUX, sinon ça n'a *pas de sens*. D'ailleurs, en λ-calcul on peut aussi bien encoder des entiers, et si l'on ne se restreint pas aux entiers tels qu'on les encode en tant qu'argument, on peut très bien trouver un point fixe de la fonction qui à un entier associe son successeur.
Donc OU M VRAI et OU VRAI M "sont tous les deux vrais" (en fait, ils sont β-équivalents au terme VRAI), quel que soit M. Il ne reste plus qu'à prouver

que OU FAUX FAUX est β-équivalent à FAUX :
$OU FAUX FAUX = (\lambda a.\lambda b.a\ VRAI\ b)\ FAUX\ FAUX \rightarrow_\beta (\lambda b.FAUX\ VRAI\ b)\ FAUX \rightarrow_\beta FAUX\ VRAI\ FAUX = (\lambda x.\lambda y.y) VRAI\ FAUX \rightarrow_\beta (\lambda y.y) FAUX \rightarrow_\beta FAUX$
On a donc bien : OU VRAI FAUX, OU VRAI VRAI, OU FAUX VRAI sont tous β-équivalents à VRAI, alors que OU FAUX FAUX est β-équivalent à FAUX, ce qui est exactement la table de vérité de OU. Donc notre syntaxe nous a permis d'intuiter des éléments de notre sémantique. Si cet exemple peut paraître simpliste, il faut savoir que toute la logique booléenne peut se réinterpréter à partir de ce dernier : la négation, la conjonction, en particulier, s'encodent également comme des λ-termes, qu'on peut intuiter de la même façon. Et certains résultats sur le lambda calcul peuvent donc s'interpréter comme des résultats de logique booléenne (un exemple simple est que toutes les expressions de la forme $e := VRAI|FAUX|OU\ e1\ e2|ET\ e1\ e2|NOT\ e1$ peuvent se voir comme des formules de logique, mais aussi comme des termes, dont la β-réduction donnera l'évaluation de ces expressions).

Le formalisme

Le formalisme (en philosophie des maths) est l'idée selon laquelle les résultats mathématiques sont le fruit de manipulation d'éléments purement syntaxiques, via des règles non moins syntaxiques. Comme si les mathématiques n'étaient qu'un jeu d'échec, et que les résultats ne correspondaient qu'aux positions qu'on peut atteindre au cours d'une partie. Dans ce cas, les résultats ne correspondent pas à des vérités absolues, mais plutôt à des affirmations vraies sous une certaine façon de voir les choses (qu'on peut nommer *syntaxe*). Vous l'aurez compris, ce document constitue plutôt un texte formaliste, il est même un plaidoyer formaliste en celà qu'il met de côté toute la vérité apparente de l'affirmation "1+1=2" pour se concentrer sur l'aspect purement syntaxique de cette affirmation. Il n'empêche que cette syntaxe, cette façon de voir les choses, vient bien de quelque part. Même en regardant les mathématiques comme un gigantesque édifice purement syntaxique, il faut bien que cette syntaxe trouve ses origines dans l'esprit de quelqu'un. C'est un exemple de glissement des mathématiques vers la philosophie : si les mathématiques étudient des résultats obtenus à partir d'axiomes (dont on considère qu'ils le sont par des règles et des preuves précises, d'un point de vue formaliste), la philosophie (ou en tout cas l'ontologie) fait le processus inverse : on s'intéresse à une réalité, ou ce qui nous semble en être une, et on tente d'inférer des axiomes. C'est ça, ce processus qui nous fait voir *d'où vient la syntaxe*.