De la Construction au Calcul

La Théorie des Types de Martin-Löf

Matthias Puech*

14 janvier 2010

Table des matières

*Université Paris 7 – Università di Bologna

Introduction

Je vais vous parler d'un formalisme logique, ou plutôt d'une famille de formalismes – la Théorie des Types – qui a été introduite par le philosophe et logicien Per Martin-Löf dans les années 70.

Je m'y suis interessé parce que mon travail de thèse s'articule autour de l'assistant à la preuve Coq, qui est une des implémentations d'un formalisme descendant directement de la théorie des types, le calcul des constructions. C'est une théorie des types donc, intuitionniste, qui donne la part belle à la notion de calcul, à tel point que le langage de Coq peut être vu à la fois comme une logique intuitionniste d'ordre supérieur, et comme un véritable langage de programmation.

Je vais m'intéresser à deux questions dans cet exposé :

Quand Brouwer propose le programme intuitionniste pour la logique, il s'agit de rejeter toute forme de formalisme, les preuves faisant référence à des expérience de pensée : c'est ce qu'il appelle les constructions. Le mathématicien ne vérifie pas les démonstrations, il doit s'en convaincre, on se souvient de l'expression, "au plus profond de sa demeure intime". Cette vision profondément subjectiviste nous laisse avec la question – entière – de la nature de ces constructions. L'interprétation de cette vision par Heyting en 1930 fait émerger les notion de méthode, de problème comme des raffinement de la notion de construction, mais en vérité laisse le mystère presque entier.

Cinquante ans plus tard émergent des systèmes informatiques de vérification des démonstrations, totalement automatiques, et en lien direct avec l'intuitionnisme, en témoigne le nombre de ces outils qui le sont. Comment, du subjectivisme si profond de Brouwer ont pu émerger ces outils, qui témoignent d'une vision si externaliste de la logique ? Comment la construction, d'une expérience "intime", presque mystique, a pu voir sa nature changer si profondément qu'elle est devenu un procédé, un algorithme dont la vérification est complètement externe au mathématicien ?

Si cette évolution s'est amorcée bien avant Martin-Löf, notamment avec l'isomorphisme de Curry-Howard, sa Théorie des Types y est sans doute importante, et ce pour deux raisons. Elle est tout d'abord profondément ancré dans le programme intuitionniste et propose réellement une réalisation, une implémentation complète et fidèle de la notion de construction en répondant aux questions laissées en suspens par Curry-Howard. Mais de ce programme elle fait aussi émerger des méthodes et des idées qui vont au delà de celles de Brouwer. En poussant à l'extrême la notion de jugement, Martin-Löf en dégage ce qui sera la base de l'algorithmicité des preuves.

A travers donc la théorie des types, je vais tenter de donner quelques pistes de réflexion sur la nature de ces constructions, et le lien que la logique entretient depuis avec la programmation.

1 La méthode du jugement

1.1 Glossaire

Commençons par établir un petit glossaire de quelques notions logiques contemporaines telles qu'elles sont utilisée chez Martin-Löf. Pour cela, jouons un jeu, tentons de réduire la logique à *quia* ; c'est le jeu auquel aiment jouer les enfants quand ils demandent « pourquoi ? » de façon répétée, jusqu'à ce que leur interlocuteur ne puisse plus répondre que « parce que ». Cela ne nous apportera sans doutes pas de réponses intéressantes sur la nature de la logique, mais nous permettra de fixer notre vocabulaire d'une part, et aussi d'observer un phénomène qui sera nécessaire par la suite : celui de la précédence conceptuelle de certaines notions sur d'autre. Pour expliquer une notion, on doit avoir expliqué ou défini un certain nombre d'autres notions plus « primitive ». On pourrait appeler cet ordre l'*ordre des priorités conceptuelles*, c'est ce terme que propose Martin-Löf.

Qu'est-ce qu'une *règle logique* ? — Une règle logique, ou règle d'inférence, c'est une relation, indiscutable une fois qu'on se l'est fixé, entre un certain nombre d'objets – les prémisses – et un objet du même genre – la conclusion. — Alors quelle est la nature de ces objets ? Qu'est ce que c'est qu'une prémisse ou une conclusion ? — Sans doutes des *propositions*, ou *formules*. Une formule, c'est une forme particulière d'*expression*, qui prend sa construction dans un domaine particulier, celui dont on veut parler. Une proposition de l'arithmétique, c'est une expression que l'on construit seulement à partir des symboles de l'arithmétique et des connecteurs logiques standards. — Et une expression, qu'est-ce que c'est ? — Une expression, c'est juste un objet qu'on peut se représenter, qu'on peut reconnaître dans ses multiples occurrences, et qu'on peut reproduire en plusieurs copies. — D'accord, mais cela ne répond pas à la question : comment agissent les règles logiques sur ces propositions ? Quelle est la fonction d'une proposition ? — Une proposition, c'est quelque chose que l'on affirme ou que l'on nie. — Existe-t-il donc deux formes indiscernable de propositions, celles qu'on affirme et celles qu'on nie ? — Non, sûrement pas — Et ne veut-t-on pas pouvoir aussi exprimer d'autres caractères d'une proposition ? Son affirmation ou sa négation, mais aussi par exemple son caractère conditionnel, ou toute autre propriété ? — C'est vrai. Les prémisses et conclusions des règles logiques expriment bien implicitement le caractère « jugeable » associé aux propositions, et ce caractère devrait être explicite dans les prémisses et conclusions. La règle :

$$\frac{A}{A \vee B}$$

exprime implicitement le fait que si A est vraie, $A \vee B$ est vrai. Elle devrait être lue comme :

$$\frac{A \ \mathsf{vrai}}{A \vee B \ \mathsf{vrai}}$$

Revenons donc sur ce que l'on a dit, et appelons comme Kant un *jugement* l'ensemble de la proposition et de son caractère jugeable. C'est donc les jugements que l'on peut diviser par exemple en affirmatifs ou négatifs. On peut alors noter comme Gentzen $\vdash A$ l'affirmation

d'une proposition A et $\nvdash A$ sa négation, ou plus simplement A vrai et A faux, mais on a maintenant toute la liberté d'introduire de nouvelles *formes de jugement*, comme par exemple A est une formule de l'arithmétique, ou A est jaune. — D'accord, donc ce que l'on peut lire aux prémisses et conclusions règles d'inférence, ce sont des jugements. Mais qu'est ce que c'est qu'un jugement ? Quelle est sa fonction ? — Le jugement, c'est ce que l'on juge. Attention, le terme est ambigü en français : c'est à la fois la chose que l'on juge, l'objet syntaxique, et l'acte de juger. Et cette ambiguïté doit être distinguée au niveau sémantique : appelons donc le jugement comme objet, celui que l'on peut juger, un *énoncé*, ou *pré-jugement*, dans le sens où c'est l'objet avant son acte. Maintenant, appelons simplement *jugement* l'acte lui-même de juger, et enfin, appelons *jugement manifeste* (« evident judgment ») un énoncé jugé. Comme on le verra, l'acte de juger, celui qui transforme un énoncé en jugement manifeste est ce qui constitue la *preuve* de cet énoncé.

$$\text{énoncé} \xrightarrow{\text{jugement/acte de juger}} \text{jugement manifeste}$$

De ce dialogue se détache bien une certaine notion de précédence conceptuelle, et aussi un bouleversement de cet ordre quand on en arrive à la notion de jugement. En effet, à la différence de la proposition, qui dépend directement de concepts syntaxiques (l'expression), la notion de jugement, ou plus exactement de forme de jugement, en est indépendante : le pré-jugement, ou énoncé, est seulement défini par rapport à son acte associée, celui de le juger. Une expression, dans ce cas, est simplement attachée au jugement. On passe donc de l'ordre :

règle logique > formule > expression,

(on doit définir la notion de formule avant celle de règle logique), à l'ordre plus souple :

règle logique > formes de jugement > acte de juger :

pour définir une forme de jugement, on doit d'abord définir ce qui rend un jugement de cette forme manifeste. Autrement dit, on doit définir quel *processus* autorise à le juger. Ce n'est qu'au dessus de cette construction que pourront se définir les règles de la logique.

Le jugement prend donc une nouvelle place, centrale, au détriment de la proposition. Cette liberté nouvelle, l'indépendence du jugement par rapport à la proposition, est illustrée dans la transformation suivante, qui sera une des clés de la compréhension de la Théorie des Types.

1.2 De la proposition au jugement

La présentation habituelle d'un système formel, par exemple la logique du premier ordre, passe par deux étapes bien distincte :

- la définition inductive des formules, au moyen par exemple d'une grammaire, qui identifie l'ensemble des expressions bien formées,
- la définition des axiomes et règles d'inférence, qui définissent un sous-ensemble de ces formules : les théorèmes.

En un sens, on procède donc par deux raffinement successifs pour identifier l'ensemble des expressions intéressantes. La faiblesse de cette approche la suivante. Considérons une règle logique comme l'introduction de la disjonction :

$$\frac{A}{A \vee B}$$

Cette règle contient en prémisse et en conclusion les variables A et B. Implicitement, ces variables représentent des formules, c'est à dire des instances d'un objet figé et immuable au moment ou l'on énonce les règles d'inférence du système. Autrement dit, dans l'ordre des priorités conceptuelles, la formule vient traditionellement avant la règle logique : les règles logiques sont dépendantes du langage des formules. En particulier, la définition de l'ensemble des termes – la signature, qui dans le cas de la logique du premier ordre impose une théorie particulière (l'arithmétique de Peano, ZF...), précède celle d'une règle – $\vee_{\text{intro}}$ – qui ne parle pourtant en aucun cas de cette théorie. Donc si je définis l'arithmétique de Peano, et la théorie des ensemble de ZF, les deux disjonctions que j'emploie ne sont pas les mêmes objets, ce sont deux versions inutilement spécialisées du même objet conceptuel.

Cette remarque suggère le traitement plus libre, plus lâche, de la notion de formule. Remplaçons la même par la notion d'expression qui elle n'est pas définie inductivement, mais constitue un concept ouvert. L'hypothèse implicite d'avant (A est une formule) peut alors être explicitée en séparant le jugement unique (et implicite) précédent en deux jugements distincts :

– A est une proposition (A prop)
– A est vraie (A vrai)

La règle précédente devient alors :

$$\frac{A\ \mathsf{prop} \qquad B\ \mathsf{prop} \qquad A\ \mathsf{vrai}}{A \vee B\ \mathsf{vrai}}$$

Pour ainsi dire, la charge de travail laissé précédemment à la notion de formule (discriminer les propositions bien et mal formées) est reportée sur les règles d'inférence, et une dérivation dans le système consiste non plus seulement en l'établissent de la véracité d'une formule mais aussi de sa bonne formation. Pour chaque symbole, on doit bien sûr se donner les règles adéquate :

$$\frac{A\ \mathsf{prop} \qquad B\ \mathsf{prop}}{A \vee B\ \mathsf{prop}} \qquad\qquad \frac{A\ \mathsf{prop}}{\neg A\ \mathsf{prop}} \qquad\qquad \ldots$$

La syntaxe des formules fonctionne donc en « vase communiquant » avec l'établissement des règles logiques. Cette remarque, si elle paraît anodie, revêt une importance nouvelle quand on cherche, comme dans notre cas, à définir une théorie ouverte, extensible, où les constantes et les règles logiques peuvent être ajoutées « dynamiquement » à condition qu'elles soient justifiées.

2 Le jugement comme connaissance

2.1 Juger, savoir, prouver

Mais comment les justifier ? Pour répondre à cette question, revenons en arrière et demandons-nous : Qu'est-ce qu'un jugement ? Qu'est-ce que l'acte de juger ? Autrement posé, qu'est qui constitue le jugement d'un énoncé ?

Nous avons vu qu'avec l'irruption de ce concept Kantien, et le remplacement par lui de la notion de proposition comme on l'entendait depuis Aristote, il devient le concept central de la logique puisqu'il est celui qui doit être expliqué avant tous les autres. A tel point que Martin-Löf écrit :

> « Il y a donc une relation intime entre la réponse à la question "qu'est-ce qu'un jugement" et la question même "qu'est-ce qu'est la logique". » [ML96]

Il y donne une réponse directe, qui était selon lui déjà derrière toute l'analyse de Kant de la notion de jugement. C'est cette réponse qui servira de fil conducteur à tout le développement de la théorie des types :

> « Un jugement n'est rien d'autre qu'un acte de savoir. » [ML96]

Autrement dit, en explicitant l'ambiguité du terme jugement, l'acte juger, c'est l'acte de savoir (comprendre), et ce qui est jugé, un bout, un objet de savoir (« a piece of knowledge »). Le jugement logique est donc défini en terme de connaissance, et on peut dresser la correspondance entre toutes les notions déjà établie et ce nouveau fondement pour le jugement :

juger	acquérir un savoir comprendre
énoncé pré-jugement	objet de savoir
jugement manifeste	savoir acquis connaissance

Qu'est qu'une preuve dans ce contexte ? Wikipédia – il faut savoir vivre avec son temps – nous apprend qu'une preuve, c'est « ce qui établit la véracité d'une proposition, d'un fait ». Cette définition doxique se traduit de façon directe dans notre nouveau cadre : « la véracité d'une proposition » tout ensemble devient « un énoncé », « la véracité » étant une forme de jugement, et « une proposition » une expression quelconque, et « établir » devient juger. La preuve, c'est donc l'acte qui transforme un pré-jugement, un énoncé, en un jugement manifeste (« A proof is what makes a judgment evident »). En somme, *prouver, c'est juger* !

Comment maintenant définir, spécifier cet acte, cette transformation qu'est la preuve d'une forme de jugement donnée ? C'est maintenant clair : Pour définir une forme de jugement, il faut que je définisse ce qu'il faut savoir pour avoir le droit d'en émettre un jugement.

C'est ce qui est résumé ici :

Je sais que $\boxed{A}$ est vraie

« A »	=	expression
« est vraie »	=	forme de jugement
« A est vraie »	=	énoncé
« je sais que »	=	jugement, preuve

Martin-Löf va même plus loin dans cette idée :

> « Si "théorie de la preuve" est compris [...] comme l'étude des preuves dans le sens originel du mot, alors "théorie de la preuve" est la même chose que "théorie de la connaissance", qui est à son tour la même chose que la logique dans son sens originel, comme l'étude du raisonnement. » [ML96]

Forts de ce constat, il ne nous reste plus qu'à étudier, d'un point de vue opérationnel, les mécanismes qui régissent (une simplification de) l'acquisition de la connaissance, et nous aurons une base pour justifier la logique, en dehors, ou en dessous d'elle même.

2.2 Une théorie du jugement

Un énoncé ou pré-jugement, s'assimile à donc un objet de savoir, et un jugement manifeste, ou un jugement accompagné de sa preuve, à une connaissance acquise. Ce dernier devient un objet statique, fixé dans notre mémoire. On l'appellera un *jugement catégorique*. Utiliser cette connaissance, c'est l'intégrer dans un nouvel un acte de prouver qui transformera à son tour un jugement énoncé en un jugement manifeste. L'acte d'acquérir des connaissances est donc l'acte de construire une série d'objet de connaissance, des jugements catégoriques, qui viennent à leur tour enrichir notre connaissance globale, et qui pourront être réutilisés dans le futur.

L'établissement de connaissances est donc un acte qui se déroule dans le temps. C'est cette idée de mouvement, de transformation que l'on va essayer de capturer en introduisant formellement l'idée de *jugement hypothétique*, qui modélisera un « instantané » de ces situations temporelles.

Dès que j'ai un énoncé, quel qu'il soit, c'est que je sais ce qu'il faut savoir pour avoir le droit de le juger. Intuitivement, je peux tout à fait le supposer, émettre l'hypothèse de son existence, c'est-à-dire supposer que je le connais, que je l'ai prouvé. La règle pour faire une supposition est donc : dès que j'ai un énoncé, dès que je sais ce qui en constitue une preuve, je peux le supposer. Cela donne naissance à la notion de jugement hypothétique, ou jugement sous hypothèse :

$$(U)\ V$$

Ici U est un *énoncé hypothétique* et V un énoncé, le tout forme un énoncé hypothétique. Intuitivement, on dira que V *dépend* de U, dans le sens où pour juger (comprendre) V, on peut utiliser la connaissance (hypothétique) de U. En mathématiques, on dirait que V est

une *famille* de jugements indexés par U et on écrirait $\{V(x)\}_{x\in U}$. Qu'est-ce qui constitue une preuve d'un tel énoncé hypothétique ? C'est une preuve – partielle – dont les parties manquantes sont des jugements de U. Autrement dit, un jugement hypothétique c'est une preuve qui, complétée par une preuve de son hypothèse U, devient une preuve catégorique de sa conclusion V.

Et cette idée peut être itérée autant de fois que l'on veut, ainsi dans :

$$(\dots((U_1)U_2)\dots)U_n$$

que l'on écrira simplement :

$$(U_1)(U_2)\dots U_n$$

$U_1 \dots U_n$ et V sont des énoncés, dont chacun dépend de tous les précédents : U_2 dépend de U_1, …, U_n dépend de $\{U_i\}_{i<n}$. Grâce à ce mécanisme, on capture un phénomène très commun du raisonnement mathématique. considérons par exemple l'énoncé :

> Soit R une relation sur un ensemble A et R^* sa clôture réflexive, alors pour tout $x \in A$, on a $R^*(x, x)$.

Il se traduit en le jugement hypothétique :

(A est un ensemble)
(R une relation sur A)
(R^* la clôture réflexive de R)
($x \in A$) $R^*(x, x)$

Chaque hypothèse dépend potentiellement des hypothèses précedentes. Ici, l'ordre des hypothèse est donc important, à la différence par exemple de l'ordre des hypothèse dans le séquent de la déduction naturelle : changer leur ordre, c'est potentiellement casser leur dépendence.

L'opération dynamique, que l'on appelera *substitution*, qui consiste à compléter une connaissance partielle (une preuve d'un jugement hypothétique) grâce à une connaissance de ses parties manquante (une preuve de ses hypothèses) est triviale : remplissez juste les trous et vous obtiendrez une preuve catégorique !

Cette mini-théorie reflète, de façon partielle bien sûr, le mécanisme d'acquisition de connaissance et leur réutilisation dans le temps. En même temps, on y voit émerger, par sa dynamique, une certaine *opérationalité*, un algorithme de transformation, qui est le premier indice de son lien avec le calcul : la machine, grâce à sa mémoire et sa capacité de manipuler des informations statiques (des preuves) est un support tout à fait valable pour l'*implémentation* de cette théorie.

2.3 La ligne directrice du constructivisme

Un des tournants méthodologiques majeurs qu'initie Brouwer avec l'intuitionnisme se situe dans l'abandon de l'interprétation sémantique des propositions. Il suggère que la dénotation

habituelle des propositions, comme des valeurs de vérité, est tout simplement inadéquate, et ce pour quatre raisons :

- C'est tout d'abord philosophiquement une platitude (*cf.* Tarski : $A \wedge B$ vrai si A vrai et B vrai. Le "et" *méta* ne nous aide en rien à comprendre le $\wedge$... *« It's turtles all the way down »*).
- Justifier la quantification est indécidable, puisqu'on doit la justifier sur tout le « domaine du discours ».
- Le théorême de Fermat dénote ici la même chose que $2+2=4$. Est-ce bien raisonnable de laisser si peu de place à la preuve ?
- Enfin, cette interprétation ne passe tout simplement pas à l'échelle de l'ordre supérieur.

Brouwer, au contraire, adopte un approche beaucoup plus subjectiviste de la notion de preuve. Pour lui, elle se rapproche même de la notion juridique de preuve (*evidence*) : une preuve de A, c'est l'ensemble des procédés qui emportent la conviction que A est vraie (je dois cette analogie à Alexandre Miquel). C'est une pièce à conviction, un objet dont l'observateur – le juge ou le mathématicien – peut se convaincre, soit directement (c'est le *flagrant délit*, la preuve immédiate), soit par une suite d'étapes, un mécanisme (médiate, par assimilation à la juridiction médiate) qui pourraient être « rejoué » autant de fois qu'on a besoin de s'en convaincre. Le slogan intuitionniste – « there are no unexperienced truth » – prend alors tout son sens. L'expérience, ce que Brouwer appelle la construction, est la seule preuve (*evidence*) qui constitue la vérité. Par exemple, le calcul $4 * 3 = 12$ est une preuve que 12 n'est pas un nombre premier.

C'est Heyting vers 1930, et plus tard Kolmogorov qui vont formuler plus précisément cette idée : la proposition (l'énoncé, le pré-jugement dans notre cas) doit être compris comme un problème, une attente ou une intention. Mieux, une tâche. A la lumière de l'assimilation du jugement et de la connaissance, connaître un énoncé (ou un problème, une tâche), c'est savoir ce qui est considéré comme une solution, une vérification de ce pré-jugement. Heyting et Kolmogorov proposent dans ce leur fameuse interprétation :

$$\begin{aligned}
[\![A \Rightarrow B]\!] &= f \in [\![A]\!] \to [\![B]\!] \\
[\![\forall x \in A,\ B]\!] &= f \in \{[\![B_x]\!]\}_{x \in [\![A]\!]} \\
[\![\exists x \in A,\ B]\!] &= \langle a, b\rangle \in [\![A]\!] \times [\![B[x/a]]\!] \\
[\![A \wedge B]\!] &= \langle a, b\rangle \in [\![A]\!] \times [\![B]\!]
\end{aligned}$$

Une précision supplémentaire sur cette assimilation est apportée dans les années 70 sous la forme de la correspondance de Curry-Howard : ce qui était considéré jusqu'alors comme un programme – un terme du λ-calcul – devient (isomorphe à) une preuve de ce qui était alors appelé le *type* de ce programme (sa spécification), et qui devient lui même un pré-jugement. Juger, c'est exécuter (on retrouve l'opérationnalité du jugement).

En se permettant un certain anachronisme, on peut aujourd'hui relire et compléter cette correspondance, grâce au jour nouveau que fait émerger, d'une part la théorie de la programmation, et d'autre part la « méthode du jugement » décrite précédemment (*judgemental method*, Frank Pfenning).

preuve	programme
énoncé, pré-jugement	type, spécification
juger, comprendre	exécuter
jugement manifeste	valeur, résultat de l'exécution

En composant ce tableau et la remarque précédente, on obtient :

> Pour définir une forme de jugement, il suffit de définir ce que l'on considère comme une vérification, un jugement manifeste de celle-ci.

Autrement dit, de l'autre côté de la correspondance :

> Pour définir un nouveau type, il suffit de définir (la forme de) ses valeurs.
>
> Un type n'est défini que par l'ensemble de ses valeurs.

Il apparaît donc que cette correspondance s'adapte parfaitement aux concepts que l'on a introduit jusque là : ils s'y traduisent presque mot-à-mot. En fait, comme on le verra, Martin-Löf prend l'interprétation de Heyting-Kolmogorov comme un fil conducteur pour la justification de sa théorie des types.

Comme vous le savez, toutes ces idées sont déjà réalisées dans le λ-calcul simplement typé, qui est le premier support, la première l'incarnation de l'interprétation de Heyting-Kolmogorov. Mais celui-ci ne correspond qu'à une logique très pauvre, et c'est l'intention d'étendre la correspondance de Curry-Howard qui donne naissance à la Théorie des Types Intuitionniste de Martin-Löf [MLS84]. C'est une extension à un cadre non seulement plus expressif (la logique intuitionniste d'ordre supérieure) mais aussi extensible : un objet logique peut être défini et ajouté à condition qu'il soit justifié, c'est à dire que l'on définisse ses valeurs.

3 La théorie des types

On peut maintenant commencer à exposer la théorie des types. Elle est composée d'un coeur minimal, un noyau, qui n'est rien d'autre qu'une instance de la théorie des jugements que l'on a défini, noyau au dessus duquel on déclarera les objets logiques ou mathématiques.

Il existe de nombreuses présentations et de nombreuses variantes de cette théorie, j'ai choisi d'exposer la première version que Martin-Löf ait exposée, que l'on appelle « Bibliopolis » (du nom de l'éditeur chez qui ont été publiée les notes d'un cours de 1980), et ce pour la raison suivante : la théorie des types est, encore aujourd'hui, un sujet de recherche très actif et n'est, trente ans après, toujours pas un objet stable incarné par une version définitive. Il n'y a donc aucune légitimité à présenter une version plutôt qu'une autre. J'ai choisi la première car elle est le plus – selon moi – en adéquation avec les idées philosophiques qu'il a développé pour la justifier.

Elle repose sur deux notions primitive. La première est appelée indifféremment *type* ou *ensemble*, et correspond intuitivement à l'idée – intuitionniste – qu'un type n'est que l'ensemble de ses valeurs. Attention, *ensemble* n'est pas ici à comprendre dans le sens global de la

théorie des ensemble, mais juste comme « ce que l'on définit par induction ». Contrairement à la plus récente notation " :", on note $\in$ l'appartenance à un tel ensemble.

La deuxième notion est celle de jugement d'égalité (*judgemental equality*). C'est une forme de jugement (en fait deux) qui définissent l'ensemble des opérations de remplacement purement calculatoires que l'on peut faire respectivement sur les ensembles et les habitants de ces ensembles. Je ne m'étend pas pour le moment sur cette question du calcul, pensez simplement que l'ensemble des règles sur les jugements sont définies *modulo* ces jugements d'égalité.

3.1 Formes de jugements

On introduit les formes de jugements suivantes :

1. A est un ensemble (A set)
2. A et B sont des ensembles égaux ($A = B$)
3. a est un élément de A ($a \in A$)
4. a et b sont deux éléments égaux de A ($a = b \in A$)

Pour justifier – expliquer – ces formes de jugements, on doit comme je l'ai expliqué répondre pour chacun à la question : « Que doit-on savoir pour avoir le droit d'émettre un tel jugement ? ». Comme on va le voir, la réponse à cette question va faire émerger un nouvel ordre des précédence conceptuelles, qui nous prescrira l'ordre de définition des jugements.

3.1.1 Explication

Formation d'ensemble Un ensemble n'est donc défini que par la « collection » de ses éléments. Cette réponse peut sembler imprécise quand on la compare à la réponse apportée par la théorie des ensembles. Elle nous dicte pourtant déjà une réponse assez précise à la question : pour juger qu'un objet est un ensemble, il faut savoir comment former ses éléments.

C'est ce qu'on fait quand on définit les entiers, non pas à la manière de Cantor, mais de celle plus « intuitive » de Peano :

$$\frac{}{\mathsf{0} \in \mathbb{N}} \qquad\qquad \frac{n \in \mathbb{N}}{\mathsf{S}\ n \in \mathbb{N}}$$

Un objet comme $2 + 2$ (ou $\mathsf{plus}(\mathsf{S}\ \mathsf{0}, \mathsf{S}\ \mathsf{0})$) n'a pourtant pas cette forme : on appelle ces objets *non-canoniques*, par opposition aux objet *canoniques* tels que 0 ou 42 (on les a appelé *valeurs* précédemment). Alors que le jugement de $42 \in \mathbb{N}$ est immédiat, celui de $2 + 2 \in \mathbb{N}$ requiert une étape intermédiaire, de raisonnement ou de calcul, $2 + 2 = 4 \in \mathbb{N}$. On doit donc aussi indiquer comment juger de l'égalité de deux éléments de l'ensemble :

$$\frac{}{\mathsf{0} = \mathsf{0} \in \mathbb{N}} \qquad\qquad \frac{n = m \in \mathbb{N}}{\mathsf{S}\ n = \mathsf{S}\ m \in \mathbb{N}}$$

Un ensemble A est donc défini en indiquant :

– comment est formé un élément canonique de A,
– comment former deux éléments égaux de A

Égalité d'ensembles Un ensemble est uniquement déterminé par ses éléments. Si donc $A = B$, on a $a \in A$ ssi $a \in B$. C'est la signification de ce jugment.

Appartenance à un ensemble Suivant la ligne directrice de l'interprétation de Heyting, $a \in A$ signifie que a est une *méthode* qui, une fois éxecutée, produit un élément canonique de A. Pour cela, on doit définir ce qu'est cette exécution, qui correspond au jugement suivant.

Égalité d'éléments d'un ensemble On l'a déjà presque dit, deux éléments arbitraires d'un ensemble A sont égaux ($a = b \in A$) si après raisonnement (ou calcul), ils produisent des éléments canoniques égaux.

3.1.2 Interprétation

Vous ne serez pas étonné d'apprendre que chacune de ces formes de jugement accepte différentes lectures, puisque leurs définitions sont basées sur l'interprétation de Heyting-Kolmogorov.

A set	$a \in A$	
A est un ensemble	a est un élément de A	A est non-vide
A est une proposition	a est une preuve/un jugement de A	A est vrai
A est un problème	a est une méthode pour résoudre A	A est soluble
A est une spécification	a est un programme réalisant A	A est habité

Autrement dit, A set peut être vu (interprété) comme le jugement que l'on a introduit au début, A prop. De même, en « dégradant » le jugement $a \in A$ en oubliant le a, il se lit comme A vrai. Ces remarques n'ont qu'une valeur indicative : elles vont nous permettre, une fois la « vraie logique » réintroduite, de retomber sur nos pieds, vérifier que les constantes logiques sont bien celles dont on a l'habitude, en déduction naturelle par exemple.

3.2 Règles de typage

Commençons à construire un système de règles, justifiée par la construction informelle sous-jacente.

Égalité comme équivalence On se donne tout d'abord les règles usuelles sur l'égalité : réflexivité, symétrie, transitivité. Celles-ci doivent être répétée pour les deux formes de jugement d'égalité. Elles sont trivialement vraie, par définition, pour les éléments canoniques :

$$\textsc{Refl/EqS}\ \frac{A\ \mathsf{set}}{A = A} \qquad \textsc{Refl/EqE}\ \frac{a \in A}{a = a \in A}$$

$$\textsc{Sym/EqS}\ \frac{A = B}{B = A} \qquad \textsc{Sym/EqE}\ \frac{a = b \in A}{b = a \in A}$$

$$\textsc{Trans/EqS}\ \frac{A = B \qquad B = C}{A = C} \qquad \textsc{Trans/EqE}\ \frac{a = b \in A \qquad b = c \in A}{a = c \in A}$$

Voici par exemple une justification détaillée de la règle Trans/EqE : a, b et c sont trois éléments de A, potentiellement non-canoniques, qui « renvoient » (qui sont égaux par calcul) respectivement à trois éléments canoniques, a', b', c' tels que $a' = b'$ et $b' = c'$. Il s'agit alors d'une égalité d'expression, syntaxique, qui est par définition transitive. On en conclue $a' = c'$, donc $a = c \in A$.

Remarquez aussi les prémisses des règles de réflexivité. Elles expriment directement et forcent la dépendance entre les forme de jugement. Ainsi, on sait qu'un jugement $A = B$ quel qu'il soit suppose forcément que A set et B set. De même pour l'autre forme.

Conversion d'ensemble Ces règles décrivent l'utilisation des jugement d'égalité de type dans les jugements qui en dépendent. Ils sont justifiés trivialement par la signification des jugements de type $A = B$.

$$\textsc{Conv/E}\ \frac{a \in A \qquad A = B}{a \in B} \qquad \textsc{Conv/EqE}\ \frac{a = b \in A \qquad A = B}{a = b \in B}$$

Substitution Ces règles sont juste des instances de l'opération de substitution, qui consistait à compléter un jugement hypothétique par ses hypothèses. Il y en a donc en théorie une par paire de jugement respectant l'ordre de dépendance. Celles qui manquent sont dérivables.

Je ne donne ici que les règles pour une hypothèse, mais elles se généralisent facilement au cas avec plusieurs.

$$\textsc{Subst/E/S}\ \frac{a \in A \qquad (x \in A)\ B\ \mathsf{set}}{B[x/a]\ \mathsf{set}} \qquad \textsc{Subst/E/EqS}\ \frac{a \in A \qquad (x \in A)\ B = D}{B[x/a] = D[x/a]} \qquad \textsc{Subst/E/E}\ \frac{a \in A \qquad (x \in A)\ b \in B}{b[x/a] \in B[x/a]}$$

$$\textsc{Subst/E/EqE}\ \frac{a \in A \qquad (x \in A)\ b = d \in B}{b[x/a] = d[x/a] \in B[x/a]} \qquad \textsc{Subst/EqE/S}\ \frac{a = c \in A \qquad (x \in A)\ B\ \mathsf{set}}{B[x/a] = B[x/c]}$$

C'est tout ce que contient le noyau de la Théorie des Types ! En soi, elle ne constitue pas une logique, mais ce qui a été appelé par la suite un *Logical Framework*, c'est-à-dire un formalisme supportant potentiellement une logique écrite comme un langage objet. Déclarons maintenant les constantes logiques qui nous intéresse.

3.3 Déclaration des constantes logiques

Leur déclaration va suivre un motif commun. Pour chaque symbole, on a quatre règles :
- la *règle de formation* nous dit comment former un certain ensemble à partir d'autres ensembles,
- la *règle d'introduction* spécifie quelle forme ont les éléments canoniques de l'ensemble,
- la *règle d'élimination* montre comment définir des fonctions sur l'ensemble,
- *les règles d'égalité* relient les introductions et éliminations en montrant comment réduire (au sens de Prawitz) une fonction définie par élimination.

3.3.1 Produit cartésien

Le *produit cartésien*, ou *produit dépendant*, est un un opérateur de bas niveau, qui nous permettra de définir deux constantes usuelles. On peut voir un produit dépendant comme une fonction (calculable) dont le codomaine dépend du domaine, une famille.

Formation

$$\Pi\textsc{-Form}\ \frac{A\ \mathsf{set} \qquad (x \in A)\ B\ \mathsf{set}}{\Pi x \in A.\ B\ \mathsf{set}}$$

D'un jugement catégorique de A est un ensemble, et d'un jugement hypothétique de B est un ensemble paramétré par une preuve d'appartenance à A, je peux construire l'ensemble produit dépendant (de A et B). Dans la prémisse de droite, B peut dépendre de l'élément choisi de A (x). On nomme ce x dans la conclusion par commodité, de manière à voir Π comme un lieur dans le B de la conclusion.

Introduction On décrit maintenant les élément canonique du produit dépendant. Ce sont des expressions commençant par le symbole λ. Toute la partie droite de la prémisse peut dépendre de x et A. Le λ doit donc être vue comme un lieur dans b et B.

$$\frac{(x \in A)\ b \in B}{\lambda x.\ b \in \Pi x \in A.\ B}\ \Pi\text{-Intro}$$

Elimination On introduit maintenant l'opérateur d'élimination du produit : @.

$$\frac{c \in \Pi x \in A.\ B \qquad a \in A}{@(c, a) \in B[x/a]}\ \Pi\text{-Elim}$$

Expliquons ce qu'il signifie. $@(c, a)$ est une méthode pour obtenir un élément canonique de type $B[x/a]$ (une certaine instance de B). On sait que $c \in \Pi x \in A.\ B$, c'est-à-dire que c est une méthode qui renvoie un élément canonique $\lambda x.\ b$ (par la règle d'introduction). Prenons $a \in A$ et substituons-le dans b à la place de x (règles de substitution). Alors $b[x/a] \in B[x/a]$. On appelle ce dernier élément $@(c, a)$ pour dénoter l'opération que nous venons de faire. Notez que cette opération n'est pas directement à effectuer (quand on rencontre une application). C'est simplement une expérience de pensée qui justifie la règle.

Autrement dit, nous venons de justifier la règle d'élimination du produit par sa règle d'introduction et la dynamique des preuves (jugements). C'est l'idée, originellement de Prawitz, que les règles d'introduction et d'élimination dépendent l'une de l'autre, en vertue seulement d'un principe de réduction sous-jacent.

Égalité On justifie alors très facilement la règle d'égalité :

$$\frac{a \in A \qquad (x \in A)\ b \in B}{@(\lambda x.b, a) = b[x/a] \in B[x/a]}$$

Elle, effectue le calcul de substitution proprement dit, et elle se justifie de la même façon par les règles d'introduction, d'élimination et de substitution.

Remarquez que cet opérateur, le produit, est un peu « triché » : Il copie au niveau logique le fonctionnement qui existait déjà au niveau sub-logique (des jugements). Il est nécessaire que je l'introduise en premier mais je vais vous montrer juste après que ce n'est pas toujours le cas.

Applications du produit Regardons les deux règles (intro/elim) que l'on vient de déclarer, et « dégradons » les jugements comme on l'a expliqué précédemment. Comme le symbole n'a plus le même sens, je remplace Π par un autre, au hasard celui-ci : $\forall$. J'obtiens :

$$\frac{(x \in A)\ B\ \mathsf{vrai}}{\forall x \in A,\ B\ \mathsf{vrai}} \qquad \frac{a \in A \qquad \forall x \in A,\ B\ \mathsf{vrai}}{B[x/a]\ \mathsf{vrai}}$$

Vous connaissez ces règles ? C'est le quantificateur universel de la déduction naturelle ! Une autre : Je définis $A \supset B \equiv \Pi x \in A.\ B$, où x n'apparaît pas dans B. En refaisant la même opération j'obtiens :

$$\frac{(A\ \mathsf{vrai})\ B\ \mathsf{vrai}}{A \supset B} \qquad\qquad \frac{A \supset B\ \mathsf{vrai} \qquad A\ \mathsf{vrai}}{B\ \mathsf{vrai}}$$

C'est l'implication logique ! Si je résume, le produit dépendant se spécialise donc en deux opérateur suivant le sens que l'on donne à sa dépendance : la quantification universelle et l'implication. C'est une version très concrète d'une idée déjà présente dans l'interprétation de Heyting : l'implication $A \Rightarrow B$ était une fonction $A \to B$ et l'universel une famille $\{B_x\}_{x \in A}$. Or une famille n'est qu'une fonction, dont le codomaine dépend du domaine, donc une fonction est une famille particulière ; l'implication est un universel particulier (si j'ose dire), un universel « non-dépendant ».

3.3.2 Union disjointe

Voyons un autre exemple de connecteur de bas niveau : l'union disjointe.

Formation Même règle de formation que pour le produit :

$$\frac{A\ \mathsf{set} \qquad (x \in A)\ B\ \mathsf{set}}{\Sigma x \in A\ .B\ \mathsf{set}}$$

Introduction On introduit une nouvelle construction, la paire, comme unique élément canonique de l'union.

$$\frac{a \in A \qquad b \in B[x/a]}{(a, b) \in \Sigma x \in A.\ B}$$

Elimination $\mathsf{E}(c, (x, y)d)$ est un nouvel opérateur, d'élimination du produit. Son premier argument est un programme qui produit un élément canonique de la somme, et son deuxième argument un programme dépendant de programmes x et y dans le produit (attention, B peut toujours dépendre de A).

$$\frac{c \in \Sigma x \in A.\ B \qquad (x \in A)\ (y \in B)\ d \in C}{\mathsf{E}(c, (x, y)d) \in C[x/c]}$$

Cette règle se justifie, comme précédemment, par l'introduction et par substitution. Sa justification décrit une opération sur les preuves, que la règle d'égalité exprime :

Égalité

$$\frac{a \in A \qquad b \in B \qquad (x \in A)\ (y \in B)\ d \in C}{\mathsf{E}((a, b), (x, y)d) = d[x/a][y/b] \in C[x/a][y/b]}$$

Application de la somme Regardons les deux règles (intro/elim) que l'on vient de déclarer, et « dégradons » les jugements comme avant. Comme le symbole n'a plus le même sens, je remplace Σ par un autre, au hasard $\exists$. J'obtiens :

$$\frac{a \in A \qquad B[x/a] \text{ vrai}}{\exists x \in A.\ B \text{ vrai}} \qquad\qquad \frac{\exists x \in A.\ B \text{ vrai} \qquad (x \in A)\ (B \text{ vrai})\ C \text{ vrai}}{C \text{ vrai}}$$

On obtient (une variante) des règles d'introduction et d'élimination du quantificateur existentiel ! On continue : Soit la définition $A \wedge B \equiv \Sigma x \in A.\ B$ où B ne dépend pas de x. Les règles se « dégradent » en :

$$\frac{A \text{ vrai} \qquad B \text{ vrai}}{A \wedge B \text{ vrai}} \qquad\qquad \frac{A \wedge B \text{ vrai} \qquad (A \text{ vrai})\ (B \text{ vrai})\ C \text{ vrai}}{C \text{ vrai}}$$

Encore une fois, ce sont (une variante) des règle de la conjonction ! En somme, la conjonction n'est qu'un existentiel particulier, un existentiel non-dépendant.

Si le temps le permettait, on pourrait continuer ces déclarations, et construire des preuves sur ces objets à l'infini. Ainsi, l'article introductif de Martin-Löf continue et donne des règles pour :

- Le coproduit, qui s'interprète en la disjonction,
- la contradiction comme l'ensemble privé d'élements,
- L'égalité propositionnelle, c'est-à-dire l'égalité sur laquelle on peut raisonner dans la logique,
- Les entiers, les booléens
- ...

Mais arrêtons-nous là et observons ce que l'on a construit. Nous avons une théorie sublogique (protologique ?) basée sur une notion dynamique, exécutable de connaissance, c'est ce que j'ai appelé le coeur de la Théorie des Types. Nous avons un jugement d'égalité qui lui aussi prend le sens de l'exécution, et qui est justifié par la première.

3.4 Typage et calcul

Examinons ces notions d'exécution plus en détail, et voyons en quoi elles peuvent nous aider à voir la Théorie des Types comme un langage de programmation.

L'analogie entre programmation et logique telle qu'elle est traditionnellement exprimée dans le λ-calcul simplement typé est la suivante : un programme, ou λ-terme, peut-être réduit, et il possède une forme normale s'il est bien typé. Son type est exprimé dans la logique du premier ordre, et ce type constitue alors un théorême. Exécuter ou réduire un λ-terme typé, c'est modifier la preuve de son type de façon à la rendre "normale". Cette opération est indifférente au mathématicien qui écrirait ses preuves comme des λ-termes : ce qui compte, c'est l'existence de la preuve, pas sa forme normale.

De l'autre côté, quand on utilise le λ-calcul comme un langage de programmation (par exemple quand on programme en Lisp ou en ML), les types ne sont utiles que comme des

« garde fous », un moyen de spécifier de façon très sommaire ce qu'un programme doit faire, et par faire, j'entends son execution, même uniquement son résultat. Ainsi, un programme de type $int \to bool$ est un programme qui attend en entrée un entier, et dont le résultat de l'execution sera un booléen, c'est-à-dire vrai ou faux. Le typage n'est donc qu'une indication sommaire, qui a presque la même valeur qu'un commentaire accompagnant le programme « donnez à ce programme un entier et il vous renverra un booléen » (à part bien sûr que cette vérification est automatique).

Pour résumer, en λ-calcul simplement typé, la vérification de type est le concept central à gauche de l'isomorphisme de Curry-Howard, tandis que la normalisation est le concept central à droite.

La théorie des types présente une situation totalement différente, et offre un nouveau regard sur le processus de calcul. Réduction et vérification de typage y sont en effet totalement entrelacée, car le calcul se situe en effet à deux niveau,

- Au niveau des preuves (des éléments d'un ensemble) par le biais du jugement $a = b \in A$, cette réduction est celle à laquelle on est habitué,
- Au niveau des types (des ensembles) par le biais du jugement $A = B$. Celle-ci est moins habituelle, puisqu'elle agit sur l'objet qui est censé rester stable par réduction : le type. La règle chargée d'indentifier deux ensembles égaux, Conv/E, nous assure même que l'on peut utiliser deux ensemble indifféremment s'ils sont égaux modulo calcul :

$$\frac{a \in A \qquad A = B}{a \in B}\ \textsc{Conv/E}$$

Ainsi, pour vérifier le bon typage d'une preuve, il est possible et même fréquent de faire appel au calcul, à la réduction d'un type. Prenons un exemple simple :

Considérons l'énoncé $\forall x \in \mathbb{N},\ x < 1 + x$. Il se prouve par induction sur x, et le cas de base est $0 < 1+0$. Nous savons que $0 < 1$, mais qu'en est-il de $0 < 1+0$? Laissons l'énoncé se calculer par l'intermédiaire du jugement d'égalité $1 + 0 = 1 \in \mathbb{N}$. Il se *réduit* donc vers $0 < 1$ et on peut le prouver directement par notre lemme. On a donc calculé un type, $0 < 1 + 0$, uniquement pour vérifier le bon typage de la preuve d'un énoncé.

Typage, ou vérification de preuve, et calcul sont donc maintenant indifférenciables, l'un étant défini en fonction de l'autre : il est nécessaire d'executer certains termes de preuves pour vérifier le bon typage.

De cette constatation à la notion de programmation, il reste un très court chemin à parcourir. Les deux mondes, celui des mathématiques et de la vérification de démonstrations dans lequel l'exécution joue un rôle anecdotique, et celui de la programmation dans lequel on écrit des programmes fait pour être exécutés, et qui sont bien typés seulement s'ils réalisent une certaine spécification, ces deux mondes sont dans la théorie des types indiscernables : les démonstrations calculent autant que les programmes, ils sont le même objet.

Terminons par un extrait de l'introduction d'un article paru en 1984 sous le nom « Constructive Mathematics and Computer Programming » :

« Si l'on entend par programmation non pas l'acte d'instruire une machine particulière pour lui faire exécuter une tâche, mais de façon plus générale comme l'invention de méthodes de calcul qui sont du ressort de l'ordinateur d'executer, alors il n'est plus possible de distinguer l'acte de programmer de la pratique des mathématiques constructives. » [ML84]

Références

[ML84] P. Martin-Löf. Constructive Mathematics and Computer Programming. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, pages 501–518, 1984.

[ML96] P. Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1) :11–60, 1996.

[MLS84] P. Martin-Löf and G. Sambin. *Intuitionistic type theory.* Bibliopolis Naples, 1984.