Sur quelques liens entre théorie des topos et théorie de la démonstration.

Conférence faite à l'Université de la Méditerrannée le 29 mai 2007

Alain Prouté([1])

Dernière révision de ce texte : 28 mai 2016

Résumé

C'est en travaillant à la réalisation d'un logiciel de formalisation des mathématiques (délibérément) fondé sur la théorie des topos, que j'ai été amené à explorer cette structure d'un point de vue calculatoire. Les liens entre catégories cartésiennes fermées et normalisation forte du lambda-calcul sont maintenant bien connus, et c'est donc sur le contenu calculatoire du classifiant du foncteur des sous-objets que porte cet exposé. On aboutit ainsi à une conception de la correspondance preuve-programme beaucoup moins symétrique que celle proposée par les concepts Martin-Löf/Curry-Howard, mais certainement plus proche des mathématiques usuelles, et où on reparle d'axiome du choix et d'indiscernabilité des preuves.

Motivations.

Le but de mon travail est de créer un logiciel de formalisation des mathématiques (un "assistant de preuve") qui puisse être mis entre les mains des lycéens, donc a fortiori des étudiants et des chercheurs, sans qu'ils n'aient grand chose à apprendre pour s'en servir, et sans que son utilisation n'introduise une autre sorte de mathématiques que les mathématiques de "tout le monde".

La réalisation d'un tel logiciel est bien entendu un travail multidisciplinaire. La théorie qui doit lui servir de fondement n'est qu'un aspect du problème, et c'est seulement de cet aspect qu'il est question ici. En dehors de cet aspect "théorique" il faut résoudre des problèmes de syntaxe, d'ergonomie, d'aspect visuel (graphisme), de méthodologie, d'organisation (bases de données, bibliothèques), de possibilité d'import/export (en particulier vers TEX), etc... et même de psychologie.

On sait que le niveau de l'enseignement des mathématiques au lycée s'est fortement dégradé depuis plus d'une trentaine d'années. La première grosse alerte est venu le jour où les élèves ayant subi la "réforme Haby" sont entrés en première année d'université ou en "math–sup". On a pu constater une brusque division du niveau par un facteur de 3 ou 4. Une partie d'entre eux se sont retrouvés quelques années plus tard à la préparation du CAPES, à laquelle j'ai eu l'occasion de participer comme enseignant. J'ai pu alors constater qu'on entrait dans une boucle infernale en formant de futurs professeurs ayant un niveau très faible par rapport à leurs devanciers, et surtout une incompréhension de ce que sont véritablement les mathématiques. Ceci a provoqué un rétroaction sur l'enseignement secondaire qui n'a fait qu'aggraver la situation, et fait encore baisser le niveau. Aujourd'hui, on peut dire qu'on ne fait tout simplement plus de mathématiques au lycée. J'en veux pour preuve ma confrontation comme enseignant avec des

1. Université Denis Diderot–Paris 7. (alp@math.univ-paris-diderot.fr)

étudiants de première année à l'Université Paris 7, pendant les années 2000 à 2006. Non seulement les nouveaux étudiants arrivent à l'Université en ne sachant rien en mathématiques (et surtout pas ce que "démontrer" veut dire), mais pire, avec des idées fausses. Ceci n'a en fait rien d'étonnant quand on constate par exemple, que la définition de la continuité d'une fonction se résume pour eux au fait "de ne pas lever le crayon". J'étais moi–même au lycée il y a une quarantaine d'années, et je me souviens fort bien qu'on savait définir la continuité proprement en terminale, qu'on calculait avec des matrices 3×3 en première, et que d'une manière générale, on avait commencé à apprendre à démontrer en classe de quatrième, la chose la plus importante d'ailleurs étant qu'on considérait le fait de démontrer comme un jeu et qu'on s'amusait prodigieusement bien.

La situation étant ce qu'elle est, c'est-à-dire un cercle vicieux, une sorte de "trou noir" pédagogique dont il paraît impossible de sortir, il semble qu'aucune nouvelle réforme de l'enseignement ne pourra résoudre ce problème. Peut–être même n'a–t–on tout simplement pas la volonté, voire l'idée, de le résoudre. Les arguments des pédagogues de tous poils, estampillés par l'état ou non, allant en général dans le sens du laxisme sous prétexte d'éviter de "traumatiser" ces chers petits avec cette dangeureuse idéologie qu'on appelle du nom barbare de "mathématiques". On m'a rapporté récemment qu'au cours d'un congrès d'IREMs, des orateurs prônaient l'élimination totale des démonstrations au lycée. J'ai peur malheureusement, que ce soit inutile. C'est déjà fait.

Il est clair pour moi, et pour nombre de mathématiciens, fort heureusement, que si l'on retire des mathématiques la notion de démonstration, il ne reste tout simplement plus rien. Le fait d'apprendre par cœur des recettes de calcul auquelles on ne comprend rien n'a, à mon sens, pas le droit de s'appeler "mathématiques". Il résulte de tout cela que les jeunes ne commencent à faire vraiment des mathématiques qu'au mieux après 18 ans, et que c'est trop tard, car il en va des mathématiques comme de beaucoup d'autres disciplines, par exemple sportives ou musicales. C'est d'autant plus dommage, que les mathématiques ne sont en fait qu'un jeu. Faire une démonstration, c'est un peu comme faire une réussite, ou résoudre une énigme policière. C'est d'ailleurs d'autant plus intéressant que cela ne suppose pratiquement aucune connaissance. L'existence même de la notion de démonstration en mathématiques est une sorte de miracle, dont on prive nos jeunes. Quand on voit à quel point ils sont doués pour les jeux vidéo, et ce dès un âge précoce, on se dit qu'il est vraiment dommage qu'on ne leur fasse pas un logiciel pour jouer à ce jeu merveilleux, ce jeu des jeux, ce jeu par excellence, qui se nomme "mathématiques". C'est le but que je désire atteindre, en espérant qu'un tel logiciel pourra modifier une situation qu'aucun ministre de l'Éducation Nationale n'a le pouvoir de changer. C'est certainement possible, et ce ne serait pas la première fois qu'un logiciel bouleverserait les habitudes, et finirait par s'imposer dans l'enseignement. Cela a déjà été le cas pour le langage de programmation Pascal, pour le logiciel Mathematica, et pour TEX au niveau des chercheurs. Tout espoir n'est donc pas perdu.

Table des matières

1 Pourquoi la Théorie des Ensembles "ne suffit pas".

On prétend en général dans les milieux mathématiques, faire des mathématiques dans le cadre de la Théorie des Ensembles. En fait, ce n'est pas tout à fait vrai, car la Théorie des Ensembles omet de formaliser une bonne part de l'activité du mathématicien. Elle ne formalise que la partie la plus "consciente". Le mathématicien fait inconsciemment des opérations mentales qui échappent à cette théorie, et qui n'en sont pas moins essentielles. Je vais illustrer cela par un exemple. Bien entendu, on pourra dire que ces considérations sont inutiles, car personne n'aurait l'idée de considérer l'"absurdité" que je vais citer ci–dessous. Je répondrai qu'il est important pour moi de les considérer et de comprendre les mécanismes qui la font rejeter par les mathématiciens, car si je ne le fais pas, je ne parviendrai pas à réaliser le logiciel qui est le but de mon travail. Ce logiciel doit en effet "penser" comme un mathématicien, et en particulier être capable de mettre en œuvre tous les mécanismes utiles et en particulier ceux qui sont *inconscients* chez le mathématicien.

Par ailleurs, la Théorie des Ensembles ne formalise pas la logique (le maniement de $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$), mais bien entendu, on est sensé la connaître d'avance. Ce n'est pas de cela que je veux parler dans cette section, mais d'une notion de "type" qui est implicite dans tout le langage mathématique, et qui est distincte de celle d'"ensemble".

La Théorie des Ensembles n'est rien d'autre qu'une axiomatisation de la notion d'appartenance ($\in$). Elle présuppose par ailleurs, que tout objet mathématique est un ensemble. Dans les premiers développements de cette théorie, on construit, entre autres choses, la notion de "paire", en général en posant :

$$(a, b) = \{a, \{a, b\}\}.$$

On notera le caractère arbitraire de cette définition([2]), qui pourrait bien être remplacée par une autre tout à fait équivalente sur le plan opérationnel, comme par exemple : $(a, b) = \{b, \{a, b\}\}$.

Beaucoup plus tard, on définit (par exemple) la fonction exponentielle $\mathbb{R} \xrightarrow{\exp} \mathbb{R}$, ou l'anneau $\mathcal{M}_{2\times 2}(\mathbb{Z}/7\mathbb{Z})$ des matrices 2×2 à coefficients dans $\mathbb{Z}/7\mathbb{Z}$. Si l'on s'en tient à la lettre de la Théorie des Ensembles, ces deux "objets mathématiques" sont des ensembles. Par ailleurs, la Théorie des Ensembles nous dit que si X et Y sont deux ensembles quelconques, leur intersection $X \cap Y$ est

2. Les informaticiens la qualifieraient de "détail d'implémentation".

bien définie. Je me mets donc “dans la peau” de l’ordinateur sensé faire des mathématiques, et je ne trouve rien à redire quand mon utilisateur écrit :

$$\exp \cap \mathcal{M}_{2\times 2}(\mathbb{Z}/7\mathbb{Z})$$

Évidemment, si au lieu d’être un ordinateur, je suis un mathématicien, et en particulier un “prof de maths”, je vais immédiatement hurler.

On peut trouver cet exemple parfaitement absurde. Mais justement la question cruciale est de savoir pourquoi il l’est. Ce n’est certainement pas la Théorie des Ensemble qui peut apporter la réponse. D’une certaine façon, elle est trop “laxiste” pour pouvoir le faire.

En fait, si on s’en tient à nouveau à la lettre de la Théorie des Ensembles, cette intersection, toute absurde qu’elle puisse paraître, peut être calculée, et le plus étonnant est qu’elle n’est peut–être pas vide. En fait cela dépend de la façon dont on définit la paire (voir ci-dessus) et un certain nombre d’autres concepts de base de même farine. C’est le caractère arbitraire de ces définitions qui provoque ces variations de ”sens”. En fait, les mathématiciens ont un procédé essentiellement inconscient pour éliminer toute construction qui dépendrait de pareilles conventions arbitraires. Ce procédé est bien connu des logiciens. C’est l’utilisation d’une notion de “type”. Et bien entendu, cette notion est tout à fait distincte de celle d’ensemble (contrairement à une idée largement répandue en informatique).

Ce qu’il se passe est que chaque fois qu’on donne un nom à une nouvelle notion, par exemple, le nom de “matrice”, ou de “fonction”, ou de “nombre”, ou de “polynôme”, ou de “suite” ou de “série”, on crée de nouveaux objets mathématiques par “clonage” d’objets précédemment construits. Ces nouveaux objets n’ont donc rien en commun avec les objets précédents, en particulier, le simple fait de les comparer avec des objets précédents n’a pas de sens. Par exemple, la notion de fonction est un clonage de celle de graphe fonctionnel, et il n’est pas question de dire qu’une fonction est “égale” à son graphe. Il est clair pour le mathématicien, même s’il ne s’en rend pas toujours compte, qu’une fonction et le graphe d’une fonction sont deux objets de nature différente, c’est-à-dire de “type” différent. En mathématiques, c’est le fait de nommer qui crée les types.

Il n’y a dès lors rien d’étonnant à ce que les logiciens et informaticiens qui réalisent des assistants de preuve, fondent fortement leur travail sur une notion de type. Toute la question est de trouver la bonne, et en particulier, pour le projet qui est discuté ici, une notion de type qui ne nous éloigne pas des mathématiques “de tout le monde”. Je vais montrer dans la suite de cet exposé que les théories de types habituellement utilisées, ne sont pas satisfaisantes de ce point de vue, alors qu’au contraire, la Théorie des Topos nous apporte une vision du typage tout à fait compatible avec la préservation des mathématiques “de tout le monde”.

2 L’indiscernabilité des preuves.

Quiconque a fait un peu sérieusement des mathématiques sait ce qu’est une “preuve” (encore appelée “démonstration”), même si, la plupart du temps, il ne sait pas en donner une définition précise, comme il sait par contre le faire pour les groupes ou les espaces topologiques. Les logiciens, quant–à eux, ont donné diverses définitions (formelles) des preuves, et ces définitions définissent en général (toujours ?) les preuves comme des objets de nature purement syntaxique. En revanche, une question plus rarement posée est celle de la sémantique d’une preuve. Qu’est–ce qu’une preuve représente, si toutefois il est nécessaire qu’elle représente quelque chose ?

L'"indiscernabilité des preuves" est un principe qui peut s'énoncer ainsi :

Deux preuves quelconques d'un même énoncé sont toujours égales.

Bien sûr, par "égales", on entend qu'elles représentent la même chose, autrement–dit qu'elles ont la même sémantique, mais non pas qu'elles sont "identiques", c'est-à-dire écrites de la même manière. On peut prouver un énoncé de nombreuses façons différentes, et ce principe affirme seulement que ces différentes façons ont la même signification.

Les systèmes formels qui ont été proposés par les logiciens et les informaticiens pour rendre précise la notion de preuve ne vérifient pas tous ce principe. La première des thèses que je voudrais développer ici est précisément le fait que les preuves en mathématiques traditionnelles (les mathématiques des mathématiciens non logiciens, c'est-à-dire de "tout le monde") satisfont cette propriété.

2.1 Comment on nomme les théorèmes.

Quand un mathématicien découvre une preuve d'un énoncé qu'on ne savait pas démontrer auparavant, l'énoncé reçoit l'appellation de "théorème" et ce théorème reçoit le nom du mathématicien qui en a découvert la première preuve, ou éventuellement des mathématiciens qui ont collaboré à cette découverte. Si un jour, un autre mathématicien découvre une autre preuve plus élégante ou plus courte (donc probablement meilleure) du même énoncé, on ne change pas pour autant le nom du théorème. Cela reste le même théorème quelle que soit la façon de le démontrer.

Il apparaît donc que c'est le fait d'avoir démontré le théorème qui compte aux yeux des mathématiciens, et non pas la façon dont cela a été fait. Bien entendu, cela n'empêchera pas les mathématiciens de reprendre à leur compte la nouvelle preuve ne serait–ce, par exemple, que pour des raisons pédagogiques. Mais on aura beau redémontrer le théorème de Pythagore de cinquante façons différentes, ce sera toujours le théorème de Pythagore.

De plus, même si la façon de démontrer un théorème peut pour diverses raisons intéresser le mathématicien, ce dernier ne se soucie aucunement de la façon dont un théorème a été démontré pour l'appliquer. Et c'est heureux. Si pour appliquer un théorème, il fallait tenir compte de la façon dont il a été démontré, les mathématiques serait beaucoup plus complexes qu'elles ne le sont déjà.

Il semble donc qu'il y ait quelque part un principe qui fait que deux preuves d'un même énoncé, même si elles constituent des textes notoirement différents, représentent en définitive la même chose. C'est cette chose "unique" que j'appelerai un "garant"([3]). Un garant "garantit" la vérité d'un énoncé, et pour chaque énoncé, il ne peut y avoir plus d'un seul garant. Toutes les preuves de cet énoncé représentent ce garant unique.

2.2 Anonymat des preuves.

En mathématiques on fait souvent des déclarations, comme par exemple : "*Soit* $\varepsilon > 0$.". On pose aussi des définitions locales, comme quand on écrit : "*Posons* $a = f(b)$.". Dans les deux cas,

3. Dans la version originale de ce texte, j'utilisais le mot "témoin" au lieu de 'garant". J'ai changé d'appellation le jour où je me suis aperçu que "témoin" était déjà employé en théorie de la démonstration, avec un sens distinct. De toute façon, "garant" est plus approprié pour le sens que je donne à ce concept.

on donne un nom à un objet mathématique supposé (dans le cas d'une déclaration), ou dont on dispose déjà (dans le cas d'une définition). Bien entendu, on se sert ensuite de ce nom pour représenter l'objet en question. On connait la puissance de ce procédé, qui fait toute la différence par exemple entre calcul numérique ($\frac{1}{3}+\frac{1}{2}=\frac{5}{6}$) et calcul symbolique ($(a+b)^2=a^2+2ab+b^2$). Les noms sont tout à fait nécessaires, car même en sachant très bien à quel ensembles appartiennent tels et tels objets, deux d'entre eux ne peuvent souvent être distingués que par leurs noms.

Les hypothèses, au contraire, qu'elles soient supposées (suppositions et axiomes) ou constatées (théorèmes et démonstrations précédentes), n'ont en général pas besoin de recevoir de nom. Le lecteur du texte mathématique a constaté que tel énoncé est vrai ou supposé, et n'a pas besoin qu'on donne un nom à cet énoncé (en réalité c'est au garant de cet énoncé qu'on donnerait un nom), pour être capable de l'utiliser au bon moment dans la suite de la preuve. La raison en est, bien entendu, que peu importe comment on a prouvé tel énoncé pourvu qu'on l'ait prouvé. Les preuves (en réalité les garants, car les preuves ne sont que des représentations des garants) n'ont donc en général pas besoin de nom. Un garant est comme un naufragé sur une île deserte. Tant que personne ne vient lui tenir compagnie, son nom (s'il en a un) ne lui sert à rien. Les preuves (en fait les garants) sont donc la plupart du temps anonymes.

2.3 Pourquoi les preuves supportent l'à peu près.

S'il y a un domaine où le style d'écriture peut largement varier, c'est bien celui des preuves mathématiques. Quel mathématicien ne s'est jamais écrié : Ce type écrit comme un cochon ! Il n'empèche qu'en faisant les efforts qu'il faut pour combler les manques, on arrive en définitive à lire et comprendre une preuve même mal écrite, c'est-à-dire omettant de nombreux détails. On n'écrit pas non plus de la même façon pour des mathématiciens chevronnés et pour des étudiants de première année.

Il y a donc une grande latitude dans le degré de précision (ou le niveau de détails) qu'on doit apporter à l'écriture d'une preuve, étant entendu d'ailleurs qu'on ne va en général jamais jusqu'à la précision extrême, celle des preuves complètement formelles, qui sont beaucoup trop difficiles à écrire pour des humains (mais le sont parfois par des programmes).

Les preuves supportent donc l'à peu près, contrairement aux calculs, qui doivent être beaucoup plus précis, voire très strictement écrits. Ceci tient encore à la même raison. La seule chose qui importe en matière de preuve (ou de sous–preuve, c'est-à-dire de partie de preuve) est d'en trouver une. Peu importe laquelle. Et il en est ainsi bien sûr, parce quelles représentent toutes la même chose.

2.4 Comparaisons avec les spécifications d'une bibliothèque de programmes.

la situation décrite ci–dessus contraste fortement avec celle de l'informatique en général. Afin de pouvoir faire cette comparaison, il est nécessaire d'établir une correspondance entre les concepts des mathématiques et ceux de l'informatique. Cette correspondance est bien connue, et fait jouer aux types de données le rôle des énoncés, et aux programmes le rôle des preuves. De même qu'une preuve prouve un certain énoncé dans un certain contexte, un programme représente (calcule) une donnée d'un certain type dans un certain contexte. Dans les deux cas, le contexte est constitué de tout ce qui a été préalablement déclaré (supposé dans le cas des garants) ou défini (démontré dans le cas des garants).

Une bibliothèque de programmes est alors l'analogue d'un livre de mathématiques. Les énoncés des théorèmes correspondent aux spécifications (types) des fonctions de la bibliothèque, et les preuves de ces théorèmes correspondent au code source de ces fonctions. Si les informaticiens avaient autant de chance que les mathématiciens, ils leur suffirait de lire les types des fonctions pour pouvoir les utiliser, puisqu'il suffit au mathématicien de lire l'énoncé d'un théorème pour pouvoir l'utiliser.

Il n'en va pas ainsi. C'est dû au fait que contrairement à un énoncé qui ne peut avoir plus d'un garant, un type de données peut avoir de nombreux éléments distincts. Le type d'une donnée (quelle soit une fonction ou autre chose) ne suffit donc pas à caractériser cette donnée, contrairement à un énoncé qui caractérise parfaitement son unique garant. L'informaticien devra donc aussi lire la documentation.

Toutefois, le fait que les preuves supportent l'à peu près, est une caractéristique très intéressante pour l'informatique. Imaginez un monde idéal dans lequel on pourrait écrire des programmes supportant l'à peu près, et qui seraient nécessairement juste quand même. Ce paradis des programmeurs n'est peut–être pas si loin de nous, et c'est la raison pour laquelle il est intéressant de formaliser la notion de preuve constructive, c'est-à-dire de preuve capable de produire des programmes. Au lieu de programmer, on prouverait, et les programmes sortiraient tout seuls des preuves. Bien entendu, une autre façon de voir ceci consiste à remarquer qu'une telle "technologie" reviendrait en fait à remplacer la documentation des bibliothèques de programmes par des énoncés de théorèmes caractérisant les objets proposés par cette bibliothèque. D'une certaine façon, on rendrait ainsi la documentation lisible par le compilateur, et même si on ne diminuerait que d'assez peu le travail du programmeur, on augmenterait par contre considérablement la sûreté.

2.5 Pourquoi les programmes ne sont pas tous des preuves.

Ainsi donc, la correspondance entre les preuves et les programmes a ses limites, et il est certainement abusif de parler d'*isomorphisme* ([4]) comme le font certains. Il y a bien une correspondance, car il y a bien effectivement une analogie structurelle entre preuves et programmes, mais il y a aussi cette différence fondamentale qui est l'unicité du garant. Bien entendu, on peut très bien concevoir un isomorphisme véritable entre preuves et programmes, mais dans ce cas il est à craindre que la sorte de mathématiques qui sera incarnée par un tel système ne soit fortement exotique([5]). On en verra une démonstration un peu plus loin.

Les preuves sont bien des programmes, plus précidément des cas particuliers de programmes, au vocabulaire près bien sûr, mais les programmes ne sont pas tous des preuves. En fait les preuves sont les programmes dont les types ne peuvent pas avoir deux éléments distincts.

Cette "nuance" est clairement perceptible quand on analyse le comportement des mathématiciens, comme on l'a fait ci–dessus, mais elle l'est aussi, comme on va le voir, quand on se base sur la structure de topos élémentaire pour formaliser les mathématiques. Par contre, si on approche le problème de manière purement opérationnelle, par exemple du point de vue de la normalisation forte, elle reste invisible.

4. Je veux parler bien sûr de cette "tarte à la crème" qu'on appelle "Isomorphisme de Curry–Howard".
5. Mon collègue Yves Lafont parle même d'"ésotérisme" (communication personnelle).

3 Les théories de types.

3.1 De Heyting à Curry et Feys, puis à Howard et Martin–Löf.

C'est en 1930 qu'Arend Heyting [8] énonce les règles maintenant connues sous le nom de "Sémantique de Heyting" ou "Interprétation de Brouwer–Heyting–Kolmogorov". Ces règles qui définissent le sens des énoncés en termes de démonstrabilité, constituent sans doute le véritable acte de naissance de la mathématique intuitionniste auparavant en gestation chez Brouwer. Heyting explique par exemple qu'une preuve d'une implication $E \Rightarrow F$ est une "méthode" produisant une preuve de F à partir d'une preuve de E. Il n'emploie pas le mot "algorithme", et encore moins celui de "lambda–expression", mais l'idée est fondamentalement la même. Par "méthode", il entend clairement "procédé mécanique", c'est-à-dire en fait algorithme.

Bien entendu, la mathématique intuitionniste a beaucoup souffert d'abord du fait qu'elle était une nouveauté incompréhensible pour beaucoup de mathématiciens([6]) (comment pouvait–on avoir l'audace de récuser le principe du tiers exclu, et d'envoyer ainsi les trois quarts des mathématiques aux orties ?), et aussi du fait de son appellation d'"intuitionniste", qui pouvait laisser supposer qu'il s'agissait d'une philosophie fumeuse basée sur l'"intuition", ce qui était pour une part vrai dans les premiers essais de Brouwer, mais démenti par les travaux de Heyting dès 1930. Le terme le plus adapté pour décrire cette "nouvelle" mathématique est probablement l'adjectif "constructive". Si les intuitionnistes, dans les années 30, avaient poussé leur reflexions dans le sens opératoire (calculatoire), ils auraient peut–être inventé l'informatique avant l'heure. Mais ce mode de pensée n'était pas dans l'air, et il faudra attendre Turing (lui–même peut–être moins inspiré par la logique que par les machines à écrire) pour que la notion de calcul prenne une place importante dans la logique.

En 1958, Curry et Feys [3] remarquent une analogie de structure entre le lambda–calcul et le raisonnement mathématique, du moins en ce qui concerne l'implication. Eux non plus ne poussent pas cette idée plus loin. Il est vrai qu'à cette époque, l'informatique et la logique sont encore assez disjointes.

En 1980, à l'occasion du 80^{e} anniversaire d'Haskell Curry, Howard [9] révèle son principe "formulae–as–types", qui circulait déjà sous forme de preprint depuis une dizaine d'années et qui est aujourd'hui si prisé des logiciens et informaticiens. C'est aussi l'époque où l'informatique passe des sphères privées au grand public, avec le premier "PC", rendant ainsi l'accès au calcul très facile pour tout le monde.

Vers la même époque, Martin–Löf présente son système de types intuitionniste [14], en se plaçant dans la lignée des travaux précédents. Nous allons parler de ce système plus loin.

Aucun de ces travaux ne fait référence à un quelconque principe d'indiscernabilité des preuves. Cela peut paraître un peu étrange, car l'analyse qui a été faite dans la première partie de cet exposé quant-aux habitudes des mathématiciens aurait bien pu être faite par tous ces chercheurs([7]). Une autre façon d'analyser ce manque est de constater qu'aucun de ces travaux ne fait allusion à la sémantique d'une preuve (pas de notion similaire à celle de "garant"). Tout se passe comme si il y avait constamment confusion entre signifiant (preuve) et signifié (garant). Par exemple, Heyting dit qu'une preuve de $E \wedge F$ est un couple (p, q) tel que p soit une preuve de E et q une preuve de F. Il ne faut sans doute pas entendre par là qu'une preuve est une donnée syn-

6. Ce qui est malheureusement encore le cas un siècle après !

7. Pour être honnête, je dois dire que je n'ai pas moi non plus eu cette idée d'emblée. Comme on va le voir plus loin, ma première confrontation avec l'indiscernabilité des preuves est venue de la Théorie des Topos.

taxique, car toutes les écritures de preuves de $E \wedge F$ ne sont pas nécessairement de la forme (p, q) quand elles ne sont pas en forme normale, et même si elles sont en forme normale, si le contexte contient suffisamment de déclarations. Martin–Löf prend au contraire la précaution de rappeler souvent que les preuves sont d'abord "calculées" pour être mises sous la forme requise, afin que les règles de calcul puissent s'appliquer. Toutefois, ce calcul n'est pas vraiment la même chose qu'une sémantique, et en particulier, Martin–Löf ne demande jamais un principe ressemblant à l'indiscernabilité des preuves. Au contraire, il insiste sur le fait que deux preuves d'un même énoncé peuvent être distinctes, et c'est d'ailleurs pour cette raison qu'il peut produire une preuve constructive de l'axiome du choix. Il annonce par ailleurs que ce fait est un des plus remarquables de son système. Je vais montrer ci–dessous qu'au contraire, c'est celui qui permet de détecter que son système est impropre à la formalisation des mathématiques "de tout le monde".

3.2 Incompatibilité entre Martin–Löf et Diaconescu.

Le théorème de Diaconescu (1975) nous dit que le tiers exclu est une conséquence de l'axiome du choix. Par ailleurs, ce théorème peut être prouvé en utilisant seulement des moyens strictement intuitionnistes. Il en résulte une chose assez surprenante, qui a été remarquée par moi–même en 1992 [16], puis par J.L. Bell [1], D. DeVidi [4], M.E. Maietti et S. Valentini [13], et certainement par nombre d'autres logiciens([8]), qui est que ce théorème ne peut pas être prouvé dans le système de Martin–Löf, si tant est qu'on considère que ce que Martin–Löf appelle "axiome du choix" est bien effectivement l'axiome du choix([9]). Il est intéressant d'en analyser les raisons.

Commençons par donner l'argument qui montre l'impossibilité de démontrer le théorème de Diaconescu dans le système de Martin–Löf. Le système de Martin–Löf est consistant([10]) et contient les entiers naturels. Il en résulte qu'il est incomplet au sens de Gödel, c'est-à-dire qu'il existe dans ce système un énoncé I indécidable. Par ailleurs, l'axiome du choix (tel qu'il est énoncé par Martin–Löf) est démontrable dans ce système. Si le théorème de Diaconescu était démontrable, on obtiendrait une preuve du tiers exclu. Il en résulterait, étant donné que $I \vee \neg I$ serait alors démontrable constructivement (puisque le système de Martin–Löf est constructif), que I serait démontrable ou que $\neg I$ serait démontrable, ce qui évidemment contredit l'argument gödelien ci–dessus.

Si donc on essaye de démontrer le théorème de Diaconescu dans le système de Martin–Löf, on doit nécessairement tomber sur une impossibilité. Que manque–t–il au système de Martin–Löf pour être capable de produire une telle preuve ?

Afin de le déterminer, commençons par rappeler une preuve de ce théorème. On considère d'une part le type $\mathbf{2}$ (ou **Boole**) des booléens, c'est-à-dire la somme de types $\mathbf{1} + \mathbf{1}$, où $\mathbf{1}$ est le singleton canonique, c'est-à-dire le produit de zéro facteurs. Puis on considère son ensemble ou type des parties $\mathcal{P}(\mathbf{2})$. Bien entendu, ce type ne doit pas être confondu avec $\mathbf{2^2}$, car nous

8. Probablement par tous ceux qui connaissent à la fois le système de Martin–Löf et le théorème de Diaconescu, car l'argument est très simple. Toutefois, il est assez remarquable que W.W. Tait, alors qu'il publie en 1994 un article intitulé "The law of excluded middle and the axiom of choice" [19], n'ait pas eu à cette époque connaissance du théorème de Diaconescu, déjà vieux de 19 ans (confirmation par Tait lui–même, rapportée par DeVidi [4]).

9. On peut en douter pour différentes raisons. D'abord parce que la démonstration de Martin–Löf ne procède à aucun moment à ce qu'on pourrait appeler un "choix". Tout y est déterminé d'avance. Ensuite parce que les personnes qui continuent le développement du logiciel Coq, ont introduit une variante de l'axiome du choix, en particulier afin de pouvoir rendre le théorème de Diaconescu démontrable en Coq. L'axiome du choix originel dans ce système, basé sur une théorie inspirée par celle de Martin–Löf, etait donc peut–être mal nommé.

10. S'il ne l'est pas, il sera de toute façon sans intérêt. Il semble toutefois que sa consistance absolue ait été prouvée par les méthodes de la normalisation forte (Tait, Girard, Coquant,...).

ne somme pas, à ce point de la démonstration, dans un cadre de mathématiques classiques. Par contre, on peut l'identifier à $\Omega^{\mathbf{2}}$.

On considère alors la partie suivante de $\mathcal{P}(\mathbf{2})$:

$$X = \{S \in \mathcal{P}(\mathbf{2}) \mid 0 \in S \vee 1 \in S\}$$

où bien entendu, 0 et 1 sont les deux éléments de $\mathbf{2}$. On peut facilement démontrer l'énoncé suivant :

$$\forall_{S \in X}\ \exists_{x \in \mathbf{2}}\ x \in S.$$

En effet, après avoir déclaré $S \in X$, on dispose de l'hypothèse $0 \in S \vee 1 \in S$. On peut alors faire un raisonnement par disjonction des cas. Sous l'hypothèse $0 \in S$, on exhibe facilement un élément de $\mathbf{2}$ qui est dans S, de même dans l'autre cas. C'est là qu'on utilise l'axiome du choix, dont l'instance qui nous intéresse s'écrit :

$$(\forall_{S \in X}\ \exists_{x \in \mathbf{2}}\ x \in S) \Rightarrow (\exists_{f \in \mathbf{2}^X}\ \forall_{S \in X}\ f(S) \in S).$$

On a donc prouvé que $\exists_{f \in \mathbf{2}^X}\ \forall_{S \in X}\ f(S) \in S$. On peut donc déclarer un tel f. Soit maintenant E un énoncé quelconque. Notre but est de démontrer $E \vee \neg E$. Considérons les deux sous–ensembles (ou sous–types) suivants de $\mathbf{2}$:

$$A = \{x \in \mathbf{2} \mid x = 0 \vee E\} \qquad \text{et} \qquad B = \{x \in \mathbf{2} \mid x = 1 \vee E\}$$

Il est facile de montrer que A et B sont des éléments de X. En effet, on a clairement $0 \in A$ et $1 \in B$. Ceci permet d'appliquer la fonction f à A et à B. On peut donc poser $a = f(A)$ et $b = f(B)$, et on a $a \in A$ et $b \in B$, d'après la conclusion de l'axiome du choix obtenue plus haut. Ceci montre que $(a = 0 \vee E) \wedge (b = 1 \vee E)$, ce qui implique $(a = 0 \wedge b = 1) \vee E$([11]).

De $a = 0 \wedge b = 1$, on déduit $a \neq b$([12]). On a donc montré $a \neq b \vee E$.

Comme par ailleurs f est une fonction, de $A = B$, on déduit $a = b$. On a donc $a \neq b \Rightarrow A \neq B$, et de $A \neq B$ on peut déduire $\neg E$. En effet, si on suppose E, on a clairement $A = B$. On a donc prouvé $E \vee \neg E$. □

Où se trouve la faille dans cette démonstration ? Car il y en a nécessairement une, quand la démonstration est faite dans le système de Martin–Löf. Bell, qui fait dans [1] une analyse analogue, n'apporte pas de conclusion très tranchée. Il se demande d'abord si le problème ne vient pas d'un défaut de la condition d'"éliminabilité" :

$$a \in \{x \in A \mid E\} \qquad \text{entraine} \qquad E[a/x].$$

Pour pouvoir répondre à cette question, il faut d'abord se demander comment la syntaxe de compréhension $\{x \in A \mid E\}$ est définie dans le système de Martin–Löf. Ce n'est pas un concept primitif. Il est défini comme suit :

$$\{x \in A \mid E\} = \coprod_{x \in A} E$$

11. Cette dernière implication est valable en logique intuitionniste. En effet, les hypothèses $a = 0 \vee E$ et $b = 1 \vee E$ vont conduire à distinguer quatre cas par disjonction des cas. Dans le premier cas, on a $a = 0$ et $b = 1$ comme hypothèses, d'où la conclusion. Dans les trois autres cas, on a E comme hypothèse.

12. C'est évidemment une conséquence du fait que $0 \neq 1$. Par ailleurs, cette dernière assertion résulte de la possibilité, étant donné deux éléments quelconques u et v dans un type, de définir une fonction envoyant 0 sur u et 1 sur v. En particulier, on peut les envoyer sur les éléments "true" et "false" de Ω, ce qui permet de conclure "false" sous l'hypothèse $0 = 1$.

ce qui signifie qu'un élément du "type" $\{x \in A \mid E\}$ est en fait une paire (dépendante) (a, p), où a est dans A et où p est une preuve de $E[a/x]$. Si telle est la définition des "sous–ensembles", l'éliminabilité est simplement la seconde projection (dépendante) $(a, p) \mapsto p[a/x]$, et ne pose pas de problème particulier, du moins dans le système de Martin–Löf.

Il se demande ensuite si le problème ne viendrait pas du fait qu'il serait impossible de déduire $a = b$ de $A = B$. Son argumentation consiste à dire que A et B étant des éléments du "sous–ensemble" X, chacun d'entre eux est en fait une paire. Par exemple, $A = (A_1, q)$ où A_1 est un élément, non pas de X, mais de $\mathcal{P}(\mathbf{2})$. Il en conclut que l'application f est plus ou moins mal définie, et donc qu'on ne peut peut-être pas déduire $f(A) = f(B)$ de $A = B$.

En fait, le problème ne semble pas venir de là, car le fait que $A = B$ entraine $f(A) = f(B)$ résulte explicitement des règles d'égalité dans le système de Martin–Löf. L'égalité se transmet à toutes les constructions, ou si l'on préfère, la substitution respecte l'égalité. Autrement–dit, si a et b sont deux expressions égales et substituables à x dans une expression E, alors $E[a/x]$ et $E[b/x]$ sont des expressions égales.([13])

Le problème vient plutôt du fait que dans un système comme celui de Martin–Löf, il est impossible de démontrer de manière générale que l'injection canonique d'un sous–ensemble dans un ensemble est injective. Par exemple, considérons le type ou ensemble $\mathbf{1}$, et considérons le sous–ensemble suivant de $\mathbf{1}$:

$$X = \{x \in \mathbf{1} \mid x = 0 \vee x = 0\}$$

où 0 est l'unique élément de $\mathbf{1}$. Dans le système de Martin–Löf, on a

$$X = \coprod_{x \in \mathbf{1}} x = 0 \vee x = 0$$

On peut exhiber les éléments suivants de X :

$$a = (0, i_1(*)) \qquad \text{et} \qquad b = (0, i_2(*))$$

où i_1 et i_2 sont les deux injections (également appelées "canoniques") de $0 = 0$ dans $0 = 0 \vee 0 = 0$, et où $*$ est une preuve de $0 = 0$. L'injection canonique de X dans $\mathbf{1}$ est donnée par la projection sur le premier facteur, c'est-à-dire par :

$$a \mapsto 0 \qquad \text{et} \qquad b \mapsto 0$$

Bien entendu, a et b sont distincts, car leurs deuxièmes projections le sont. Dans ce système, une démonstration d'une disjonction doit spécifier lequel des deux énoncés de la disjonction est démontré, et une preuve de la disjonction via le premier énoncé ne peut pas être égale à une preuve de la disjonction via le second énoncé (la disjonction est une somme disjointe).

Ici, on voit nettement les limites de la notion de "formulae–as–type" qui voudrait donner exactement le même statut aux types et aux énoncés. On voit bien que le fait qui empêche de démontrer que l'inclusion de X dans $\mathbf{1}$ est injective est le fait qu'une disjonction peut avoir des démonstrations non égales.

On peut bien entendu critiquer les arguments ci–dessus en disant qu'après tout il y a peut–être dans le système de Martin–Löf une autre manière de définir la syntaxe de compréhension

13. Il semble difficile de renoncer à cette règle en mathématiques usuelles. Toutefois, dans [16], je suggère un système dans lequel cette règle ne serait pas vérifiée. Depuis cette époque, j'ai renoncé à cette idée, elle aussi fortement "exotique".

que :

$$\{x \in A \mid E\} = \coprod_{x \in A} E.$$

Toutefois, quelle que soit cette façon, il n'en reste pas moins vrai que le théorème de Diaconescu demeurera indémontrable (si bien sûr on conserve comme axiome du choix l'énoncé que Martin–Löf appelle "axiome du choix"). Or quand on examine les arguments utilisés, on constate que tous sont strictement intuitionnistes sauf le fait que l'égalité de A et B comme éléments de X entraine leur égalité comme éléments de $\mathcal{P}(\mathbf{2})$. Il est donc impossible de prouver dans le système de Martin–Löf, que cette inclusion est injective, et ceci quelle que soit la notion de sous–ensemble qu'on adopte. Tant que l'axiome du choix sera démontrable constructivement, on en restera là.

Du coup, on peut avoir envie également d'examiner cette démonstration constructive de l'axiome du choix pour déterminer les principes de démonstration qui seraient "abusifs". Sous l'hypothèse que deux preuves d'un même énoncé sont toujours égales, il doit nécessairement y en avoir un. La preuve de Martin–Löf est très simple. L'énoncé de l'axiome du choix est le suivant :

$$(\forall_{x \in A}\ \exists_{y \in B}\ \varphi(x,y)) \Rightarrow (\exists_{f \in B^A}\ \forall_{x \in A}\ \varphi(x, f(x))).$$

où A et B sont des ensembles (des types dans ce cas), et φ une relation binaire entre A et B. Comme les énoncés sont des types, l'énoncé ci-dessus n'est rien d'autre que le type des fonctions de $\prod_{x \in A} \coprod_{y \in B} \varphi(x,y)$ dans $\coprod_{f \in B^A} \prod_{x \in A} \varphi(x, f(x))$, ce qui fait qu'on peut en écrire la preuve dans le langage mathématique usuel comme suit :

$$p \mapsto (x \mapsto \pi_1(p(x)), x \mapsto \pi_2(p(x)))$$

où π_1 et π_2 sont les deux projection du coproduit (qui n'est rien d'autre qu'un produit dépendant de deux facteurs). Ce qui est ici en contradiction avec le principe de l'indiscernabilité des preuves est le fait que π_1 ne peut pas respecter l'égalité. En effet, $p(x)$ appartient à $\coprod_{y \in B} \varphi(x,y)$, donc sa première projection appartient à B. Or, les éléments de $\coprod_{y \in B} \varphi(x,y)$ sont des garants, alors que ceux de B sont des données ordinaires. Si on identifie tous les garants d'un même énoncé, l'application π_1 est mal définie (elle ne passe pas au quotient !).

Le fond du problème est donc me semble–t–il le fait que le système de Martin–Löf est en contradiction avec le principe de l'unicité du garant, ou de l'indiscernabilité des preuves. Il en va de même de tous les systèmes de type "Curry–Howard", surtout si on insiste sur le mot "isomorphisme". Dans de tels systèmes, il y a des injections canoniques de sous–ensembles dont il est impossible de prouver qu'elles sont injectives. Ceci disqualifie clairement ces systèmes pour ce qui est de la formalisation des mathématiques "de tout le monde".

4 Les topos.

La raison de considérer la structure de topos pour la création d'un langage de formalisation des mathématiques, est que c'est apparemment le seul modèle théorique connu qui soit à la fois constructif et dans lequel l'interprétation des mathématiques soit garantie non "exotique", ce dernier point tenant au fait que la catégorie des ensembles elle–même est un topos.

Il est probable que l'une des raisons qui a fait que la théorie des topos élémentaire n'a pas encore (à ma connaissance) été utilisée dans ce sens, est que son caractère "algébrique" n'est

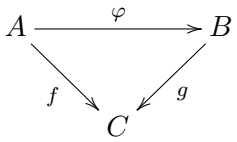
pas évident. On ne voit pas très bien a priori comment implémenter le concept de classifiant du foncteur des sous–objets, qui repose fortement sur la notion de monomorphisme, a priori non formalisable sous forme d'équations. Toutefois, A. Burroni [2] a démontré que la structure de topos est bien algébrique sur celle de graphe, et bien entendu, ce résultat hante tout mon travail, même s'il n'y est pas utilisé directement dans cet exposé.

4.1 Quelques rappels.

Un topos est une catégorie $\mathcal{T}$ qui a un objet final (noté 1), un produit fibré $\mathcal{T}^D \longrightarrow \mathcal{T}$ (i.e. un adjoint à droite du foncteur diagonal $\mathcal{T} \xrightarrow{\Delta} \mathcal{T}^D$, où D est un diagramme formé de deux flèches de même cible, et qui bien sûr couvre le cas particulier du produit ordinaire $A \times B$ de deux objets A et B) que nous utiliserons par la prise de pullbacks, des exponentielles $B \mapsto B^A$ (i.e. pour chaque objet A, un adjoint à droite du foncteur $X \mapsto X \times A$), et un monomorphisme universel $1 \xrightarrow{\top} \Omega$ (i.e. dont tout monomorphisme est équivalent([14]) à un unique pullback). Un "morphisme logique" entre topos est un foncteur qui respecte (strictement) toute cette structure.

Pour qui connait un peu la théorie des catégories, les topos([15]) ont donc une définition très simple, ce qui rend d'autant plus remarquable le fait que chacun d'entre eux (pourvu qu'il ne soit pas dégénéré([16])) est un univers mathématique complet.

On aura besoin de la notion de catégorie relative. Si $\mathcal{C}$ est une catégorie, et C un objet de $\mathcal{C}$, on peut considérer la catégorie "relative" $\mathcal{C}/C$, dont les objets sont les flèches de $\mathcal{C}$ de cible C, et dont les flèches de $A \xrightarrow{f} C$ vers $B \xrightarrow{g} C$ sont les triangles commutatifs :



Les topos ont une propriété remarquable qui peut s'énoncer ainsi :

Théorème 1 *(P. Freyd [5]) Toute catégorie relative $\mathcal{T}/A$ d'un topos $\mathcal{T}$ est un topos, et si $A \xrightarrow{f} B$ est une flèche de $\mathcal{T}$, le foncteur de pullback le long de f : $\mathcal{T}/B \xrightarrow{\mathcal{T}/f} \mathcal{T}/A$ est un morphisme logique qui a un adjoint à gauche et un adjoint à droite.*

Ce théorème joue un rôle important dans l'interprétation du langage qui est présenté ci–dessous. En effet, la façon simple et naturelle d'interpréter les expressions de ce langage relatives à un contexte Γ donné, est de les interpréter dans le topos relatif $\mathcal{T}/\overline{\Gamma}$, où $\overline{\Gamma}$ est l'interprétation du contexte Γ comme objet de $\mathcal{T}$. Cette interprétation peut se traduire en une interprétation directement dans $\mathcal{T}$, si on se donne la peine d'expliciter les constructions dont l'existence est affirmée par le théorème de Freyd. Or cette explicitation n'est pas simple, ce qui est une façon de dire que le théorème de Freyd n'est pas très facile à démontrer.

14. Deux monomorphismes de même cible sont équivalents si l'un est le composé de l'autre par un isomorphisme de leurs sources.

15. La définition ci–dessus est celle des topos dits "élémentaires" de Lawvere et Tierney, et non pas de ceux de Grothendieck.

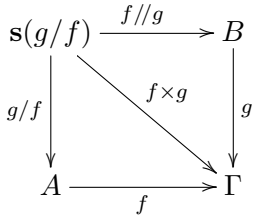
16. Un topos est dégénéré s'il est équivalent, comme catégorie, à la catégorie à un seul objet et une seule flèche. Tout objet du topos est alors final, et a un et un seul élément global.

Nous n'allons pas démontrer ce théorème, aussi appelé "théorème fondamental des topos". Le lecteur en trouvera la démonstration dans tous les ouvrages généraux sur les topos, par exemple, pages 190 et suivantes dans [12], ou dans l'article fondateur de Freyd [5]. Nous allons nous contenter de fixer quelques notations.

La source d'une flèche f sera notée $\mathbf{s}(f)$. La flèche identité de l'objet A sera notée 1 ou 1_A.

Si A et B sont deux objets de $\mathcal{T}$ on notera $A \times B$ leur produit, π_1 et π_2 les deux projections. Si $f : X \longrightarrow A$ et $g : X \longrightarrow B$ sont deux flèches, on notera $<f, g> : X \longrightarrow A \times B$ l'unique flèche telle que $\pi_1 \circ <f, g> = f$ et $\pi_2 \circ <f, g> = g$. On notera de même $f \times g$ le produit de deux flèches quelconques, puisque $\times$ est un foncteur. On a $f \times g = <f \circ \pi_1, g \circ \pi_2>$.

Si $f : A \longrightarrow \Gamma$ et $g : B \longrightarrow \Gamma$ sont deux flèches de $\mathcal{T}$ de même cible Γ, on peut considérer le produit des deux objets f et g dans la catégorie $\mathcal{T}/\Gamma$. En fait, ce produit n'est rien d'autre (c'est une tautologie) que le composé $f \circ (g/f)$ où g/f est le pullback de g le long de f. En d'autres termes, on a le carré cartésien suivant dans $\mathcal{T}$:



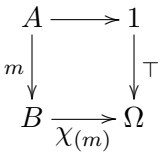
La raison de noter la flèche du haut $f//g$ et non pas f/g est que le produit dans la catégorie $\mathcal{T}/\Gamma$ n'est pas commutatif. Il l'est seulement à isomorphisme près. La notation avec une seule barre rappelle qu'il s'agit de la première projection, celle avec deux barres de la deuxième. Ces deux projections pourront aussi être notées, comme c'est l'usage, π_1 et π_2.

Par ailleurs, si $x \xrightarrow{\varphi} f$ et $x \xrightarrow{\psi} g$ sont deux flèches de source x et de cible respectives f et g dans $\mathcal{T}/\Gamma$, on a la flèche $<\varphi, \psi>$, qu'on notera plutôt $<\varphi, \psi>_\Gamma$, afin de ne pas la confondre avec une flèche de $\mathcal{T}$ (qui en est clairement distincte en général).

L'exponentielle de deux objets A et B de $\mathcal{T}$ sera notée B^A. L'évaluateur sera noté $B^A \times A \xrightarrow{\text{ev}} B$. Pour toute flèche $X \times A \xrightarrow{f} B$, on notera $\Lambda_A(f)$ l'unique flèche telle que $\text{ev} \circ (\Lambda_A(f) \times 1_A) = f$. Bien entendu, $(A, B) \mapsto B^A$ n'étant pas un foncteur, mais un bifoncteur, la notation g^f n'a pas de sens dans $\mathcal{T}$, mais en a un dans $\mathcal{T}/\Gamma$, puisque d'après le théorème de Freyd, $\mathcal{T}/\Gamma$ a des exponentielles.

Bien sûr, nous aurons aussi à utiliser la notation $\Lambda_f(g)$. Quant–à l'évaluateur dans le topos $\mathcal{T}/\Gamma$, on le notera éventuellement ev_Γ, pour éviter toute confusion.

Dans un topos, l'objet Ω, cible du monomorphisme universel $\top$, doit être compris comme le type ou l'ensemble des "valeurs de vérité", et le monomorphisme universel $\top$ lui–même comme la valeur de vérité "vrai". Pour chaque monomorphisme $A \xrightarrow{m} B$, on a une flèche $B \xrightarrow{\chi(m)} \Omega$ telle que le carré suivant soit cartésien :



$\chi(m)$ est appelée la "flèche caractéristique de m".

Par ailleurs, pour chaque flèche $A \xrightarrow{f} \Omega$ dans $\mathcal{T}$, on a un carré cartésien :

$$\begin{array}{ccc} \mathbf{s}(\top/f) & \longrightarrow & 1 \\ {\scriptstyle \top/f}\downarrow & & \downarrow{\scriptstyle \top} \\ A & \xrightarrow[f]{} & \Omega \end{array}$$

Bien entendu, $\top/f$ est un monomorphisme comme pullback d'un monomorphisme. Cette flèche joue un rôle important dans l'interprétation du langage décrit plus loin, puisque, comme on le verra, elle permet d'interpréter ce qui jouera le rôle de "type des garants".

Dans le topos $\mathcal{T}/\Gamma$, le rôle de l'objet Ω est joué par la projection $\Gamma \times \Omega \xrightarrow{\pi_1} \Gamma$, vue, bien sûr, comme un objet de $\mathcal{T}/\Gamma$, et le rôle de $\top$ est joué par la section $<1_\Gamma, \top>$ de cette flèche, que nous noterons $\top_\Gamma$.

On sait que dans un topos, toute flèche f se décompose en une composition $m \circ e$ d'un épimorphisme e avec un monomorphisme m (analogue à la décomposition "canonique" en surjection suivie d'une injection d'une application entre ensembles), qu'on appelle une "épi–mono" décomposition. Le sous–objet de la cible de f représenté par m doit être compris comme l'"image" de f. Cette décomposition est unique à isomorphisme près. Il en résulte que toute flèche d'un topos qui est à la fois un épimorphisme et un monomorphime est un isomorphisme. En effet, si f est un telle flèche alors $1 \circ f$ et $f \circ 1$ sont des épi–mono décompositions de f. Par unicité de cette décomposition, on voit que f est isomorphe à 1, donc est un isomorphisme.

Le théorème de Freyd nous apprend par ailleurs que pour toute flèche $C' \xrightarrow{f} C$, le foncteur de pullback $\mathcal{T}/C \xrightarrow{\mathcal{T}/f} \mathcal{T}/C'$ a un adjoint de chaque coté. L'adjoint à gauche est trivial, puisqu'il consiste en la composition (à gauche) avec la flèche f. Par contre, l'adjoint à droite est plus difficile à construire. Nous le noterons Π_f, comme cela se fait traditionnellement en théorie des topos. Il transforme bien entendu les objets de $\mathcal{T}/C'$, mais aussi les flèches.

4.2 Un langage formel.

La définition des topos telle qu'elle a été donnée ci–dessus mène tout naturellement à la conception d'un langage pour la représentation des objets et des flèches d'un topos quelconque. La définition de ce langage peut être présentée sous la forme d'un ensemble de "règles". Ces règles définissent à la fois la syntaxe et les conditions de validité des expressions du langage. De ce fait ce langage, inspiré par la définition de la structure de topos, est "adhoc" pour la représentation des objets et des flèches d'un topos (même s'il ne prétend pas les représenter tous). Le premier langage de cette sorte a été proposé indépendamment par Mitchell, Bénabou et Joyal. Celui qui est proposé ici met l'accent sur le notion de contexte, et formalise également les preuves.

Bien entendu, il s'agit de définir un langage formel aussi proche que possible du vernaculaire mathématique. En particulier, ce langage est symbolique (on peut y faire des déclarations et des définitions), contrairement aux langages dits "combinatoires", comme celui de Curry [3], où celui qui est utilisé dans [17]. Il y donc des symboles (variables) et une notion de contexte. Un contexte, qui est une liste (ordonnée) de déclarations([17]), s'écrit soit ϕ s'il est vide (aucune

17. Il est inutile pour ces considérations théoriques d'y mettre des définitions, qui n'apportent rien d'autre qu'un

déclaration), soit $\Gamma; x \in A$ (respectivement $\Gamma; p \vdash E$) s'il résulte de l'ajoût de la déclaration $x \in A$ (respectivement de l'hypothèse $p \vdash E$) au contexte déjà construit Γ.

Toutefois, le langage proposé ici n'est en aucun cas le langage qui sera celui de l'utilisateur du logiciel qui est le but de ce travail. Le langage ci–dessous est à usage interne du compilateur seulement. Le langage utilisateur est une surcouche qui cache un certain nombre de concepts, dont en particulier celui de type. Dans le langage de l'utilisateur, la notion centrale est celle d'ensemble. Celle de type est présente, mais implicite, comme en mathématiques. Aucune expression du langage utilisateur ne permet de définir directement un type ou de s'y référer directement. Par contre, des explications sur la notion de type figurent nécessairement dans le manuel de l'utilisateur, ne serait–ce que parce que le compilateur doit parler de type dans certains messages d'erreur, par exemple dans ceux qui résultent de l'écriture d'une égalité entre deux données de types différents. De plus, les preuves supportant l'à peu près, et les preuves formelles étant exclues au niveau de l'utilisateur, le compilateur doit comporter un algorithme de recherche de preuve, capable de trouver en une fraction de seconde une preuve de tout énoncé sensé être "évident". La notion d'évidence dépend bien sûr de la façon dont cet algorithme est construit, mais joue de toute manière un rôle de premier plan dans un tel système. En fait, l'un des rôles du compilateur est de produire des preuves formelles à partir des preuves "informelles" écrites par l'utilisateur. C'est de toute façon indispensable pour vérifier que les preuves, même incomplètes, sont correctes.

La définition du langage se fait en énonçant un certain nombre de règles qui définissent (comme le ferait un programme écrit en Prolog) le sens de certains "jugements". Les jugements sont les méta–énoncés qui nous permettent de manipuler correctement les expressions du langage, c'est-à-dire de porter un "jugement" sur le fait qu'elles sont ou non bien construites. Dans le cas qui nous occupe, il y a quatre jugements, qui peuvent s'écrire (où Γ, T, a, E et p sont des méta–symboles, c'est-à-dire représentent des expressions quelconques) :

$/\Gamma$	lire : "Γ est un contexte correct"
T/Γ	lire : "Γ est un contexte correct, et T est un type correct dans le contexte Γ"
$a \in T/\Gamma$	lire : "Γ est un contexte correct, T est un type correct dans le contexte Γ, et a est un terme correct de type T dans le contexte Γ"
$p \vdash E \in \Omega/\Gamma$	lire : "Γ est un contexte correct, E est un terme correct de type Ω, et p est une preuve correcte de E"

Bien entendu, Ω est une constante et non pas un méta–symbole. Le jugement $p \vdash E \in \Omega/\Gamma$ sera abrégé en $p \vdash E/\Gamma$, la présence du signe $\vdash$ le rendant non ambigu. Les termes de type Ω seront appelés des "énoncés". Le jugement $p \vdash E/\Gamma$ se lit :"p prouve E (dans le contexte Γ)".

Ici, nous avons délibérément insisté lourdement sur l'adjectif "correct", pour bien faire comprendre que ces règles ne sont pas seulement des règles de syntaxe.

Par ailleurs, nous utiliserons la notation $a[b/x]$ pour représenter le résultat de la substitution de b à toutes les occurrences libres de la variable x dans l'expression a. Nous supposons que le lecteur connait les règles standard de la substitution (en particulier l'obligation de renommer

confort d'écriture. Mais bien entendu, la situation est différente pour un "vrai" langage, avec compilateur et surtout utilisateurs.

parfois une variable pour éviter sa capture). Bien entendu, les opérateurs qui lient une variable sont tous ceux qui sont suivis (parfois précédés) d'une déclaration de cette variable.

On voit que les énoncés sont vus comme des sortes de types, puisque le signe $\vdash$ joue pour les énoncés et les preuves un rôle analogue à celui que le signe $\in$ joue pour les types et les termes. Ce système n'est donc pas complètement étranger au principe "formulae–as–type", même s'il s'interdit de considérer un énoncé directement comme un type.

Les règles qui définissent le langage sont en fait des "clauses de Horn"([18]). La conclusion se trouve dans la colonne "On a", les conditions (premisses) dans la colonne "pourvu que". La colonne "Commentaires" est soit une traduction de la règle en langage usuel, soit une indication pour en faciliter la compréhension. Les méta–symboles Γ, T, U, E, a, b, f, p, n sont implicitement universellement quantifiés, autrement–dit, ces règles sont valides pour toutes valeurs de ces méta–symboles. Le méta–symbole x représente un symbole quelconque du langage lui–même. Ces règles sont réparties en quatre groupes : celles qui définissent les contextes, celles qui définissent les types, celles qui définissent les termes, et celles qui définissent les preuves.

Par ailleurs, il faudrait ajouter à ce système au minimum un type des entiers naturels et les constructions qui vont avec (zéro, successeur, itération). En fait, dans un logiciel réellement utilisable pour faire des mathématiques (et éventuellement de la programmation) il faut prévoir un système plus souple de "types récursifs", incluant les arbres, les listes etc... Nous les avons omis sciemment, car ce n'est pas le sujet de cet exposé.

Contextes :

On a	pourvu que	Commentaire
$/\phi$		Le context vide est correct.
$/\Gamma; x \in T$	T/Γ	On peut ajouter la déclaration $x \in T$ au contexte Γ, si T est un type dans le contexte Γ.
$/\Gamma; p \vdash E$	$E \in \Omega/\Gamma$	On peut ajouter l'hypothèse $p \vdash E$ au contexte Γ, si E est un énoncé dans le contexte Γ.

Types :

On a	pourvu que	Commentaire
$1/\Gamma$	$/\Gamma$	1 est un type dans tout contexte.
$T \times U/\Gamma$	$T/\Gamma,\ U/\Gamma$	On peut faire le produit de deux types.
U^T/Γ	$T/\Gamma,\ U/\Gamma$	On a des types fonctionels.
Ω/Γ	$/\Gamma$	On a un type Ω (des valeurs de vérité).

18. Qu'on peut aussi voir comme des "séquents" de la déduction naturelle.

Termes :

On a	pourvu que	Commentaire
$() \in 1/\Gamma$	$/\Gamma$	Le type 1 contient un élément noté $()$.
$(a,b) \in T \times U/\Gamma$	$a \in T/\Gamma\ b \in U/\Gamma$	Construction des paires.
$\pi_1(a) \in T/\Gamma$	$a \in T \times U/\Gamma$	Première projection canonique.
$\pi_2(a) \in U/\Gamma$	$a \in T \times U/\Gamma$	Seconde projection canonique.
$x^T \mapsto a \in U^T/\Gamma$	$a \in U/\Gamma; x \in T$	Construction des fonctions.
$f(a) \in U/\Gamma$	$f \in U^T/\Gamma\ a \in T/\Gamma$	Application d'une fonction à un argument.
$\exists_{x\in T}\ E \in \Omega/\Gamma$	$T/\Gamma,\ E \in \Omega/\Gamma; x \in T$	Quantification existentielle.
$\forall_{x\in T}\ E \in \Omega/\Gamma$	$T/\Gamma,\ E \in \Omega/\Gamma; x \in T$	Quantification universelle.
$(\xi \vdash E) \wedge F \in \Omega/\Gamma$	$E \in \Omega/\Gamma,\ F \in \Omega/\Gamma; \xi \vdash E$	Conjonction (dépendante).
$(\xi \vdash E) \Rightarrow F \in \Omega/\Gamma$	$E \in \Omega/\Gamma,\ F \in \Omega/\Gamma; \xi \vdash E$	Implication (dépendante).

Preuves :

On a	pourvu que	Commentaire
$<a,p> \vdash \exists_{x\in T}\ E/\Gamma$	$a \in T/\Gamma,\ p \vdash E[a/x]/\Gamma$	"Paire de Heyting".
$\delta(p) \in T/\Gamma$	$p \vdash \exists!_{x\in T}\ E/\Gamma$	Description.
$\epsilon(p) \vdash E[\delta(p)/x]/\Gamma$	$p \vdash \exists!_{x\in T}\ E/\Gamma$	
$\upsilon(p, x \in T, \xi \vdash E, q) \vdash C/\Gamma$	$p \vdash \exists_{x\in T}\ E/\Gamma,\ q \vdash C/\Gamma; x \in T; \xi \vdash E$	
$\lambda_{x\in T}\ p \vdash \forall_{x\in T}\ E/\Gamma$	$p \vdash E/\Gamma; x \in T$	Preuve déclarative.
$p.a \vdash E[a/x]/\Gamma$	$p \vdash \forall_{x\in T}\ E/\Gamma,\ a \in T/\Gamma$	Particularisation.
$<p,q> \vdash ((\xi \vdash E) \wedge F)/\Gamma$	$p \vdash E/\Gamma,\ q \vdash F/\Gamma; \xi \vdash E$	"Paire de Heyting".
$\delta(p) \vdash E/\Gamma$	$p \vdash ((\xi \vdash E) \wedge F)/\Gamma$	
$\epsilon(p) \vdash E[\delta(p)/\xi]/\Gamma$	$p \vdash ((\xi \vdash E) \wedge F)/\Gamma$	
$(\lambda_{\xi\vdash E}\ p) \vdash ((\xi \vdash E) \Rightarrow F)/\Gamma$	$p \vdash F/\Gamma; \xi \vdash E$	
$p.q \vdash F[q/\xi]/\Gamma$	$p \vdash ((\xi \vdash E) \Rightarrow F)/\Gamma,\ q \vdash E/\Gamma$	Modus ponens.

Les deux sous–ensembles de règles ci–dessus pour les preuves sont évidemment presque les mêmes. Dans le premier, toutes les quantifications ont lieu sur des types, alors que dans le deuxième toutes les quantifications ont lieu sur des énoncés. Or précisément, ces deux sous–ensembles diffèrent par la façon de détruire les paires (dites "paires de Heyting", car l'un au moins des deux composant de la paire est une preuve).

On a vu, en discutant le système de Martin–Löf, que le principe qui est incompatible avec l'indiscernabilité des preuves est le fait de pouvoir extraire librement le premier composant d'une paire de Heyting dans le cas où ce composant n'est pas une preuve. C'est la raison de la présence de la règle :

$$\upsilon(p, x \in T, \xi \vdash E, q) \vdash C/\Gamma \qquad \text{pourvu que} \qquad p \vdash \exists_{x\in T}\ E/\Gamma,\ q \vdash C/\Gamma; x \in T; \xi \vdash E$$

qui réalise une extraction "contrôlée" des composants de la paire de Heyting représentée par p. En effet, les deux déclarations $x \in T, \xi \vdash E$, qui déclarent ces deux composants, ont une portée limitée à la preuve q. Or, q étant une preuve, elle représente non pas une donnée, mais un garant, qui est toujours le même, quelle que soit la donnée (de type T) représentée par x. De cette façon, on ne viole pas le respect de l'égalité. En bref, l'extraction de ces composants n'est autorisée que si on ne les utilise que pour produire une preuve, pas pour produire une donnée. Bien entendu,

dans le deuxième groupe de règles, cette précaution est inutile, puisque le premier composant de la paire de Heyting représente un garant.

Cette même règle est présente dans le système de Martin–Löf. Toutefois, elle ne contrôle pas l'extraction, car elle ne limite pas le type de l'expression $\upsilon(p, x \in T, \xi \vdash E, q)$ à un énoncé. Il en résulte qu'elle est alors équivalente, comme le démontre Martin–Löf, aux règles qui définissent les deux projections δ et ϵ.

Par contre, si p est une preuve d'existence et d'unicité, on peut se permettre d'extraire séparément chaque composant de la paire de Heyting, sans limiter la portée de cette extraction, puisque dans ce cas, grâce à l'unicité, x ne peut pas représenter deux données différentes. Evidemment, l'extraction du deuxième composant ne pose pas le même problème, puisqu'il s'agit encore d'une preuve. Toutefois, ce deuxième composant étant dépendant du premier, la possibilité de l'extraire est nécessairement liée à la possibilité d'extraire le premier.

Hyland et Pitts [10] dans leur présentation de la "théorie des constructions", fortement inspirée par celle de Martin–Löf, imposent la même restriction que ci–dessus quand à l'extraction des composants d'une paire représentant un garant dans le cas où le premier composant ne représente pas un garant. Toutefois, leurs motivations sont, semble–t–il, seulement liées au modèle catégorique qu'ils proposent. Il ne semble pas qu'ils aient eu l'intention de considérer un principe de l'unicité du garant. De plus, le vocabulaire qu'ils utilisent est assez étrange, puisqu'ils nomment "type" ce qui est ici nommé "énoncé" et nomment "ordre" ce qui est ici nommé "type". Ceci évidemment ne favorise pas l'introduction du principe de l'unicité du garant. Il semble par contre que Pavlović [15] se soit approché de plus près du point de vue développé ici.

Ce système contient donc un faible nombre de règles. Toutefois, il contient toutes les opérations logiques habituelles en mathématiques. Voici comment on définit "vrai", "faux", la disjonction, la négation, l'égalité :

$\bot$	$\forall_{q \in \Omega}\ q$	"faux".
$\neg E$	$E \Rightarrow \bot$	Négation.
$\top$	$\neg\bot$	"vrai".
$E \vee F$	$\forall_{q \in \Omega}\ ((E \Rightarrow q) \wedge (F \Rightarrow q)) \Rightarrow q$	"De Morgan" version constructive.
$a = b$	$\forall_{p \in \Omega^T}\ p(a) \Rightarrow p(b)$	Règle de Leibnitz.

Ces définitions sont bien connues. Elles sont valables aussi bien en mathématique intuitionniste qu'en mathématique classique, puisqu'en supposant qu'on axiomatise de la manière habituelle ces connecteurs au lieu de les définir, on parvient sans peine à démontrer les équivalences ci–dessus.

Il n'était pas absolument indispensable d'axiomatiser la quantification existentielle. En effet, on a équivalence entre :

$$\exists_{x \in T}\ E \qquad \text{et} \qquad \forall_{q \in \Omega}((\forall_{x \in T}(E \Rightarrow q)) \Rightarrow q)$$

(variante constructive de $\exists_{x \in T}\ E = \neg(\forall_{x \in T}\ \neg E)$). Toutefois, le fait de l'axiomatiser simplifie la présentation de l'interprétation de la description.

À ce point on peut faire quelques remarques. La plupart des gens (y compris s'ils sont logiciens) ont tendance à considérer que la conjonction de deux énoncés est symétrique :

$$E \wedge F \qquad \text{équivalent à} \qquad F \wedge E.$$

Pourtant, les mathématiques usuelles, même élémentaires, de même que la langue naturelle sont là pour nous inciter à penser le contraire. En effet, considérons une partie A de l'intervalle $[0, 1]$

de l'ensemble des nombres réels. On pourra écrire l'énoncé :

$$(A \neq \phi) \wedge (\sup(A) \leq 1).$$

Toutefois, tous les potaches savent qu'on ne peut pas écrire :

$$(\sup(A) \leq 1) \wedge (A \neq \phi),$$

car $\sup(A)$ risque de ne pas être bien défini, tant que rien ne nous dit que A n'est pas vide.

De même, dans la langue naturelle, on peut dire : "Il a un ami, et il lui a parlé de cette question". Par contre, la phrase : "Il lui a parlé de cette question et il a un ami" est pour le moins obscure, et en tout cas ne peut avoir le même sens que la précédente. Ceci tient au fait que le pronom "lui" dans la proposition "il lui a parlé de cette question" n'a de sens que si l'autre proposition "Il a un ami" a été énoncée précédemment, car "lui" désigne l'ami.

Dans le cas de notre exemple mathématique, l'expression $\sup(A) \leq 1$ n'a de sens que si on dispose d'un garant du fait que A n'est pas vide. Autrement–dit la proposition $(A \neq \phi) \wedge (\sup(A) \leq 1)$ s'écrit en réalité :

$$\exists_{\xi \vdash A \neq \phi}\ (\sup(A) \leq 1)$$

et l'expression $\sup(A) \leq 1$ contient une occurrence libre implicite du symbole de preuve ξ (de même qu'une occurrence libre implicite d'un symbole de preuve du fait que A est majoré). C'est uniquement encore une fois parce que les preuves sont généralement anonymes que ce symbole ne figure pas explicitement dans l'expression $\sup(A) \leq 1$. Ceci est encore un argument en faveur du principe d'indiscernabilité des preuves.

Le lecteur pourra d'ailleurs tenter de traduire l'énoncé $\sup(A) \leq 1$ dans le langage défini ici. Il constatera qu'il a besoin de l'opérateur de description pour définir le symbole "sup", et que cet opérateur prend en argument précisément la preuve p qui est nécessaire pour donner un sens à "sup".

Pour les mêmes raisons, l'implication est dépendante. On pourra examiner l'énoncé $(A \neq \phi) \Rightarrow (\sup(A) \leq 1)$ à la lumière des mêmes arguments. Par contre, la disjonction n'a aucune raison d'être dépendante.

La conjonction et l'implication dépendante sont aussi présentes dans le système de Martin–Löf, où elles sont définies par :

$$E \wedge F = \coprod_{\xi \in E} F \qquad \text{et} \qquad E \Rightarrow F = \prod_{\xi \in E} F$$

Signalons enfin que malgré son aspect asymétrique, la règle de Leibnitz est parfaitement symétrique. Le lecteur pourra en effet s'assurer qu'il a compris le maniement des règles ci–dessus en vérifiant que :

$$\lambda_{a \in T} \lambda_{b \in T} \lambda_{p \vdash a=b} \lambda_{q \in \Omega^T} \lambda_{r \vdash q(b)}\ p.(x^T \mapsto x = a).(\lambda_{q \in \Omega^T} \lambda_{s \vdash q(a)}\ s)$$

est bien une preuve de $\forall_{a \in T}\ \forall_{b \in T}\ a = b \Rightarrow b = a$ autrement–dit une preuve de la symétrie de l'égalité, du moins, bien sûr, une preuve "interne" au langage qui vient d'être décrit.

Une chose particulièrement importante est la distinction entre "types" et "ensembles". Dans le système présenté ci–dessus, nous avons introduit une notion de type, et non pas d'ensemble. En fait, les ensemble peuvent être définis dans ce système (de même qu'en théorie des constructions

de Hyland–Pitts [10]). Ceci est dû à la présence du type Ω. En effet, en théorie des topos, on sait bien que l'interprétation intuitive "standard" de l'objet fonctionnel Ω^X est "l'objet des parties de X". Dès lors, on peut définir les ensembles comme les parties des types. Les ensembles sont alors des données et non pas des types, ce qui rend leur manipulation beaucoup moins rigide que celle des types.

Les types fonctionels U^T ne sont d'ailleurs pas indispensables. Les théoriciens des topos savent bien (voir par exemple [12] ou [11]) qu'on peut se contenter des types de la forme Ω^X, que pour le coup on va renommer $\mathcal{P}(X)$. La discussion ci–dessus concernant les ensembles est à rapprocher de l'interprétation que font Lambek et Scott [11] de ce qu'ils appellent "théorie de types"([19]) dans un topos. Dans cette interprétation, les objets du topos ne sont pas les types, mais bien les données des types de la forme $\mathcal{P}(X)$.

Pour faire des mathématiques dans un tel système, on doit plus ou moins oublier les types, et les "remplacer" par les ensembles. Dès lors, on doit définir, successivement la notion d'ensemble des parties d'un ensemble (on ne connait jusqu'ici que le type des parties d'un type), la notion de produit cartésien d'ensembles, et la notion d'ensemble fonctionnel. En ce qui concerne cette dernière notion, on le fait bien sûr comme à l'école, en définissant d'abord une relation entre deux ensembles comme une partie de leur produit cartésien, puis en délimitant une partie de cet ensemble de relations pour définir l'ensemble des fonctions entre ces deux ensembles.

Dès lors, on aura aucun problème pour montrer en toute généralité que l'inclusion d'un sous–ensemble dans un ensemble est injective. Ceci tient au fait que le graphe de cette inclusion sera définie comme une partie de la diagonale du plus gros des deux ensembles. La propriété à démontrer se réduira à :

$$\forall_{x\in A}\ \forall_{y\in A}\ x = y \Rightarrow x = y$$

puisqu'alors, les éléments de l'ensemble et du sous–ensemble A ont le même type. On réalise ainsi une idée déjà ancienne de De Bruijn sur les "prototypes" ce qui laisse penser que le problème évoqué ici avait déjà été détecté par lui.

Enfin, il faudra sans doute ajouter un axiome permettant de prouver que deux énoncés équivalents sont égaux :

$$\forall_{p\in\Omega}\ \forall_{q\in\Omega}\ (p \Leftrightarrow q) \Rightarrow (p = q).$$

puisque ceci ne semble pas résulter des méthode de preuves du système. Toutefois, ceci n'a d'intérêt en mathématiques "de tous les jours" que si on parle de l'ensemble des valeurs de vérité, c'est-à-dire de Ω (qui est éventuellement isomorphe à $\mathbf{2}$). En effet, les mathématiciens, quand ils ne font pas de logique, envisagent rarement la notion d'égalité entre énoncés. Bien entendu, cet axiome est vrai dans l'interprétation que nous décrivons ci–dessous.

4.3 Interprétation dans un topos.

Le langage défini ci–dessus s'interprète de façon naturelle dans tout topos $\mathcal{T}$. Cette interprétation n'est pas forcément utile pour réaliser un compilateur. Son rôle est surtout de nous rassurer sur les choix faits dans la définition du langage. On veut vérifier que ce langage s'interprète comme on le désire. En particulier, le fait de tester cette interprétation sur le topos des ensembles lui–même joue un rôle important dans notre démarche.

19. Et qui est très différente des théories de types à la Martin–Löf.

Pour définir précisément cette interprétation, il faut donner une règle d'interprétation pour chacune des règles définissant le langage. Comme cette interprétation est bien entendu récursive, il faut aussi une notation pour l'interprétation elle même. On notera $\overline{\Gamma}$ l'interprétation du contexte Γ, $[T]_\Gamma$ l'interprétation de T quand on a T/Γ, et $[a]_\Gamma$ l'interprétation de a quand on a $a \in T/\Gamma$.

Nous allons interpréter les contextes comme des objets de $\mathcal{T}$. Si T est un type dans le contexte Γ, l'interprétation de T sera une flèche dont la cible est l'interprétation de Γ. Notez que ceci revient à dire que l'interprétation de T est un objet dans le topos relatif à l'interprétation de Γ. Enfin, si a est un terme type T dans le contexte Γ, on va interpréter a comme une section de la flèche représentant T. Notez que ceci revient à dire que l'interprétation de a est une flèche de l'objet final vers l'interprétation de T dans le topos relatif à l'interprétation de Γ. On aura donc, pour tout terme a de type T dans le contexte Γ, les flèches suivantes :

$$\overline{\Gamma; x \in T} \underset{[a]_\Gamma}{\overset{[T]_\Gamma}{\rightleftarrows}} \overline{\Gamma} \circlearrowright 1$$

avec $[a]_\Gamma$ section de $[T]_\Gamma$, c'est-à-dire $[T]_\Gamma \circ [a]_\Gamma = 1_{\overline{\Gamma}}$.

Pour soutenir notre intuition, le mieux est de supposer que $\mathcal{T}$ est le topos des ensemble. Dans ce cas, on a des éléments dans les objets. On peut d'ailleurs identifier ces éléments avec des flèches de 1 vers les objets. De telle flèches sont ce qu'on appelle des "éléments globaux" en théorie des topos.

Un élément de $\overline{\Gamma}$ est alors une "instance du contexte". Se donner un tel élément revient à donner des valeurs à tous les symboles déclarés dans le contexte Γ. Cet élément peut aussi être vu comme la valeur (le contenu) de la "pile" de notre machine à l'exécution.

La flèche $\overline{\Gamma; x \in T} \xrightarrow{[T]_\Gamma} \overline{\Gamma}$ doit être comprise comme une sorte de "fibré". Précisément, l'image réciproque d'un point x de $\overline{\Gamma}$ par cette application (autrement–dit la "fibre" au dessus de x) est la "valeur" du type T pour cette instance du contexte. Rappelons que les types sont éventuellement dépendants (en particulier à cause de la présence des types de garants ; voir plus loin).

La flèche $[a]_\Gamma$ qui est une section de $[T]_\Gamma$ nous donne donc, pour chaque instance x du contexte Γ, un élément de la fibre au dessus de x. Elle doit donc être vue comme un élément de T, dans le contexte Γ.

Les énoncés ont trois interprétations (essentiellement équivalentes). Comme Ω s'interprète comme l'objet $\overline{\Gamma} \times \Omega \xrightarrow{\pi_1} \overline{\Gamma}$ dans $\mathcal{T}/\overline{\Gamma}$ (qui est bien l'objet Ω du topos relatif), un énoncé, en tant que terme, s'interprète donc comme une section de cette flèche. Mais évidemment, une telle section est déterminée par sa deuxième composante. On peut donc aussi interpréter cet énoncé comme une flèche de $\overline{\Gamma}$ vers Ω :

$$\overline{\Gamma} \xrightarrow{[E]_\Gamma} \Omega$$

Enfin, on peut aussi considérer le pullback de $\top$ le long de cette dernière flèche, qui est encore une donnée équivalente. L'intérêt de cette dernière interprétation est qu'elle fait apparaître l'interprétation d'un énoncé comme étant de même nature que celle d'un type, puisqu'il s'agit alors d'une flèche (dans ce cas un monomorphisme) de cible $\overline{\Gamma}$.

Afin d'éviter toute confusion, nous noterons $[E]_\Gamma$ l'interprétation de l'énoncé E dans le contexte Γ comme flèche de $\overline{\Gamma}$ vers Ω, et $[W(E)]_\Gamma$ l'interprétation de ce même énoncé comme

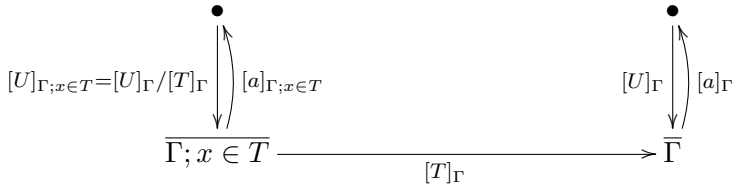
pullback de $\top$ le long de $[E]_\Gamma$. Cette notation rappelle que E est alors interprété comme une sorte de type, précisément, le "type des garants de E", qui est donc noté $W(E)$([20]).

En ce qui concerne l'interprétation des contextes, on a :

$\overline{\phi}$	$=$	1	où 1 est l'objet final de $\mathcal{T}$
$\overline{\Gamma; x \in T}$	$=$	$\mathbf{s}([T]_\Gamma)$	
$\overline{\Gamma; \xi \vdash E}$	$=$	$\mathbf{s}([W(E)]_\Gamma) = \mathbf{s}(\top/[E]_\Gamma)$	

Autrement–dit, on démarre sur l'objet final 1, et à chaque déclaration $x \in T$, on a une flèche $\overline{\Gamma; x \in T} \xrightarrow{[T]_\Gamma} \overline{\Gamma}$, qui définit l'interprétation du nouveau contexte. Évidemment, dans le cas d'un énoncé, c'est l'interprétation $[W(E)]_\Gamma$ qui est utilisée, et non pas $[E]_\Gamma$, puisqu'il faut alors voir E comme un type.

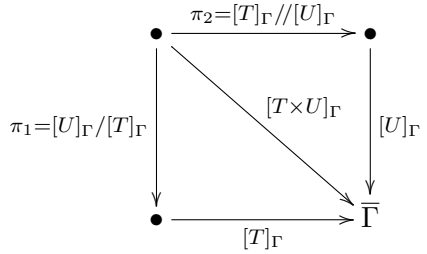
Le foncteur de pullback le long de $[T]_\Gamma$ est, comme l'indique le théorème de Freyd, un foncteur logique. Ceci signifie qu'il respecte toute la structure de topos. Dans ce cas, toutes les constructions que nous allons faire dans $\mathcal{T}/\overline{\Gamma}$ ont pour pullbacks ces mêmes constructions dans $\mathcal{T}/\overline{\Gamma; x \in T}$. En particulier, si a est un terme de type U dans le contexte Γ, alors a est toujours un terme de type U dans le contexte étendu $\Gamma; x \in T$. L'interprétation du type U dans le contexte $\Gamma; x \in T$ est le pullback de celle de U dans le contexte Γ, et sa section $[a]_\Gamma$ a pour pullback la section $[a]_\Gamma; x \in T$. A priori, tout cela n'est vrai qu'à isomorphisme naturel près, mais il est possible de faire en sorte que ce soit vrai exactement.



Je ne vais pas décrire l'interprétation des types, des termes et des preuves par des formules explicites, bien que cela soit possible. En effet, pour obtenir de telles formules, il faut avoir des formules explicites pour les constructions résultant du théorème de Freyd, et de toute façon les formules obtenues ne sont guère parlantes. On pourra facilement l'imaginer d'après la description précise mais non complètement formalisée que je vais donner ci–dessous de ces interprétations. Cette description est l'occasion de dessiner quelques diagrammes certainement plus parlants que des formules.

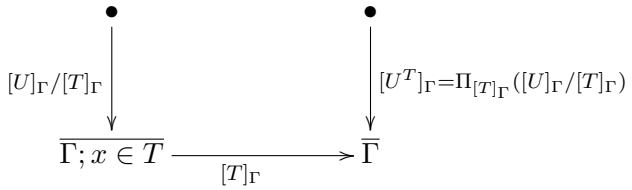
Le type 1 est interprété comme la flèche identique de $\overline{\Gamma}$, c'est-à-dire comme l'objet final du topos relatif $\mathcal{T}/\overline{\Gamma}$.

Le type $T \times U$ est interprété comme le produit dans $\mathcal{T}/\overline{\Gamma}$ des interprétations des types T et U, c'est-à-dire qu'on a le carré cartésien suivant dans $\mathcal{T}$:



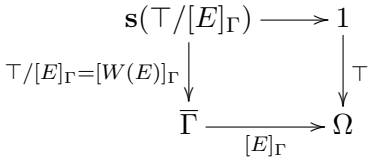
20. W comme "witness", signifiant "garant" en anglais.

Le type U^T est interprété comme l'exponentielle dans le topos relatif $\mathcal{T}/\overline{\Gamma}$ des interprétations des types T et U. On sait que cette exponentielle peut être reconstruite à partir du foncteur $\mathcal{T}/\overline{\Gamma;x\in T} \xrightarrow{\Pi_{[T]_\Gamma}} \mathcal{T}/\overline{\Gamma}$, sous la forme $\Pi_{[T]_\Gamma}([U]_\Gamma/[T]_\Gamma)$, cette dernière formule n'étant qu'une version "catégorique" de l'égalité $U^T = \prod_{x\in T} U$, dans le cas où U n'a pas d'occurrence libre de x.



Le type Ω est interprété comme le classifiant du foncteur des sous–objets dans le topos relatif $\mathcal{T}/\overline{\Gamma}$, c'est-à-dire comme la projection $\overline{\Gamma}\times\Omega \xrightarrow{\pi_1} \overline{\Gamma}$.

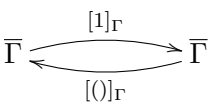
Le "type des garants" $W(E)$ est interprété comme le pullback de $\top$ le long de l'interprétation de E, qui est une flèche de $\overline{\Gamma}$ vers Ω :



Ceci se justifie de la façon suivante. À quelle condition l'énoncé E est–il vrai ? La condition est que la flèche $[E]_\Gamma$ se factorise à travers la flèche $1 \xrightarrow{\top} \Omega$, puisque $\top$ représente "vrai". Or, le carré étant cartésien, cette condition est équivalente à l'existence d'une section de $\top/[E]_\Gamma$. Une telle section peut donc être considérée comme un "garant" de la vérité de E, ce qui fait que la flèche $\top/[E]_\Gamma$ elle–même ne peut être que l'interprétation du type $W(E)$ des garants de E. En remarquant qu'un monomorphisme ne peut pas avoir plus d'une seule section, on tombe tout naturellement sur le principe de l'indiscernabilité des preuves([21]).

On remarquera que la correspondance entre les monomorphismes et leurs flèches caractéristiques est une façon de reformuler le principe "formulae–as–types". En effet, une flèche de $\overline{\Gamma}$ vers Ω ("formula") peut se voir comme un monomorphisme de cible $\overline{\Gamma}$ ("type"). En définitive, le principe qui fonde la définition même des topos peut être vu comme une façon de revisiter l'"Isomorphisme de Curry–Howard".

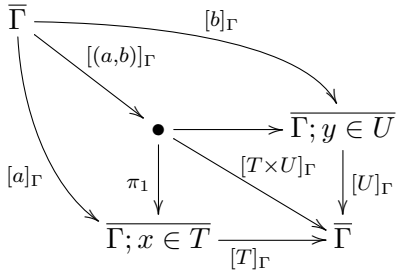
Le terme () de type 1 dans le contexte Γ, est interprété comme l'identité de $\overline{\Gamma}$, c'est-à-dire en fait l'unique flèche de l'objet final vers lui–même dans le topos relatif $\mathcal{T}/\overline{\Gamma}$. On a le diagramme suivant, dans lequel les deux flèches sont l'identité de $\overline{\Gamma}$:



Remarquer que $[()]_\Gamma$ est une section de $[1]_\Gamma$ comme il se doit.

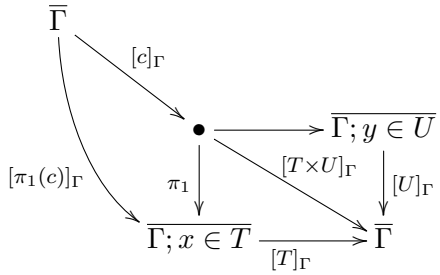
21. C'est d'ailleurs de cette façon que je m'y suis intéressé pour la première fois, et non pas en examinant les habitudes des mathématiciens.

Le terme (a, b), où a et b sont de types respectifs T et U dans le contexte Γ, est interprété dans le topos relatif $\mathcal{T}/\overline{\Gamma}$ comme l'unique flèche de l'objet final vers le produit $T \times U$, telle que les composés avec les projections canoniques soient les interprétations de a et b. Ceci se traduit dans le topos $\mathcal{T}$ par le diagramme :



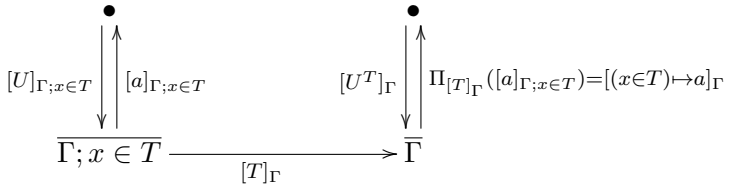
dans lequel le carré en bas à droite est cartésien. Bien entendu, on a $[T]_\Gamma \circ [a]_\Gamma = 1_{\overline{\Gamma}}$ et $[U]_\Gamma \circ [b]_\Gamma = 1_{\overline{\Gamma}}$, ce qui donne la flèche $[(a, b)]_\Gamma$, qui est alors une section de $[T \times U]_\Gamma$.

Le terme $\pi_1(c)$, où c est de type $T \times U$ dans le contexte Γ, est interprété comme la composition de π_1 (du diagramme précédent, qui est d'ailleurs égale à $[U]_\Gamma / [T]_\Gamma$) avec $[c]_\Gamma$:



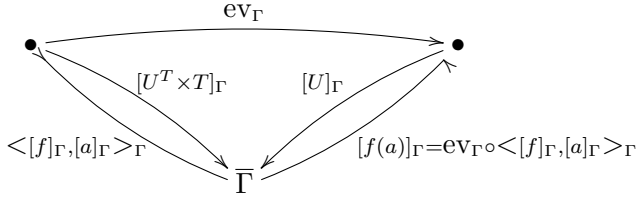
On interprète bien sûr $\pi_2(c)$ de manière analogue.

Le terme $(x \in T) \mapsto a$ de type U^T dans le contexte Γ, est interprété comme l'image par le foncteur $\Pi_{[T]_\Gamma}$ de l'interprétation de a. On a donc le diagramme suivant :

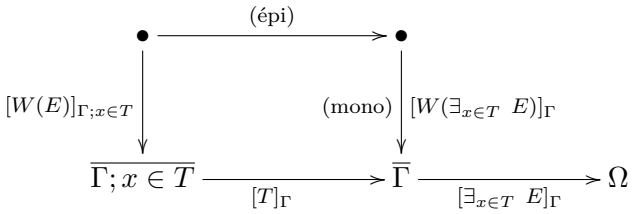


et $[(x \in T) \mapsto a]_\Gamma = \Pi_{[T]_\Gamma}([a]_{\Gamma;x\in T})$. Bien entendu, $[a]_{\Gamma;x\in T}$ comme argument du foncteur $\Pi_{[T]_\Gamma}$ est vu comme une flèche de $\mathcal{T}/\overline{\Gamma}$ et non pas comme une flèche de $\mathcal{T}$.

Le terme $f(a)$, où f est de type U^T et a de type T dans le contexte Γ, est interprété comme $\mathrm{ev}_\Gamma \circ <[f]_\Gamma, [a]_\Gamma>_\Gamma$, où $[f]_\Gamma$ et $[a]_\Gamma$ sont vus comme des flèches de $\mathcal{T}/\overline{\Gamma}$ et non pas comme des flèches de $\mathcal{T}$.

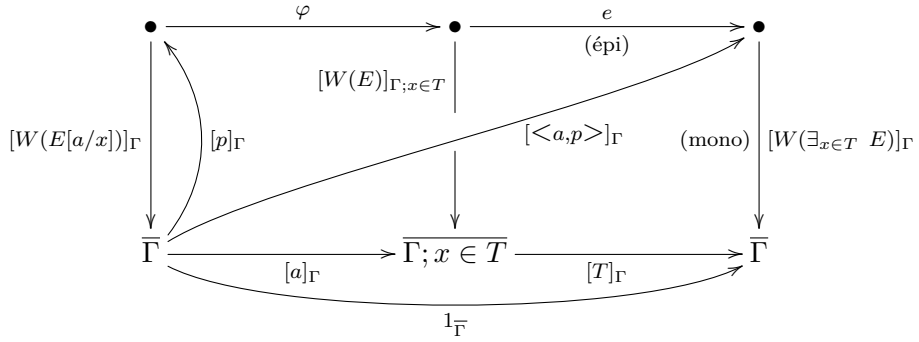


Le type $W(\exists_{x\in T}\ E)$ dans le contexte Γ, est interprété comme la partie monomorphisme de la décomposition de la flèche $[T]_\Gamma \circ [W(E)]_{\Gamma;x\in T}$ en épimorphisme et monomorphisme, et bien entendu, le terme $\exists_{x\in T}\ E$ est interprété comme la flèche caractéristique de ce monomorphisme :



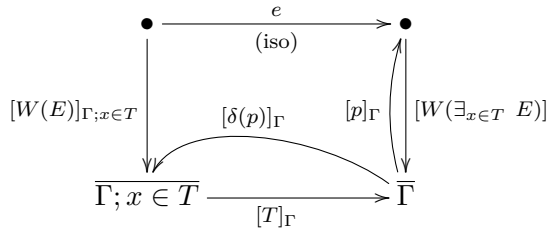
On remarquera que si l'énoncé $\exists_{x\in T}\ E$ est prouvable dans le contexte Γ, alors les flèches $[T]_\Gamma$ et $[T]_\Gamma \circ [W(E)]_{\Gamma;x\in T}$ sont des épimorphismes. En effet, une preuve p de $\exists_{x\in T}\ E$ dans le contexte Γ, fournit une section du monomorphisme $[W(\exists_{x\in T}\ E)]_\Gamma$, qui est alors un isomorphisme. Il en résulte que $[T]_\Gamma \circ [W(E)]_{\Gamma;x\in T}$ est un épimorphisme, et qu'il en est donc de même de $[T]_\Gamma$.

Supposons maintenant qu'on ait un terme a de type T, donc une flèche $\overline{\Gamma} \xrightarrow{[a]_\Gamma} \overline{\Gamma; x \in T}$ et une preuve p de $E[a/x]$ dans le contexte Γ, donc une flèche $[p]_\Gamma$, section de $[W(E[a/x])]_\Gamma$. On peut alors interpréter $<a, p>$ comme le composé $e \circ \varphi \circ [p]_\Gamma$ dans le diagramme suivant :



Il s'agit bien d'une section de $[W(\exists_{x\in T}\ E)]_\Gamma$.

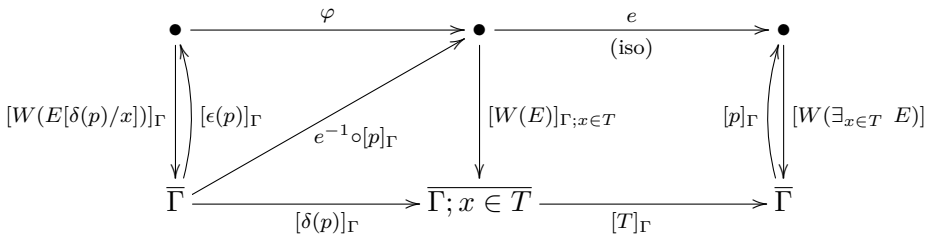
Dans le cas où p est une preuve de $\exists!_{x\in T}\ E$ dans le contexte Γ, l'épimorphisme e du diagramme précédent est un isomorphisme, car l'unicité implique que $[T]_\Gamma \circ [W(E)]_{\Gamma;x\in T}$ est un monomorphisme([22]). On obtient alors l'interprétation de $\delta(p)$ comme la composition $[W(E)]_{\Gamma;x\in T} \circ e^{-1} \circ [p]_\Gamma$, comme dans le diagramme suivant :



Il est immédiat que $[\delta(p)]_\Gamma$ est une section de $[T]_\Gamma$. De plus, la flèche $e^{-1} \circ [p]_\Gamma$ est un relèvement de $[\delta(p)]_\Gamma$ le long de $[W(E)]_{\Gamma;x\in T}$. Il existe donc une section de $[W(E[\delta(p)/x])]_\Gamma$ qu'on va noter

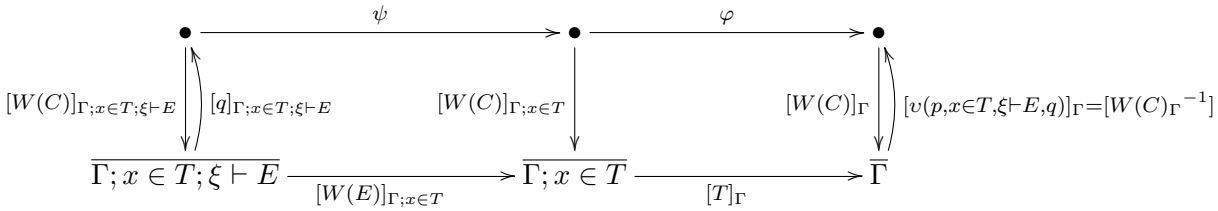
22. C'est un exercice pas complètement évident.

$[\epsilon(p)]_\Gamma$, comme dans le diagramme ci–dessous :



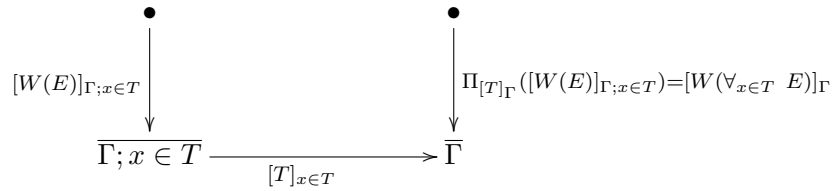
et qui sera donc l'interprétation de la preuve $\epsilon(p)$ dans le contexte Γ.

Pour interpréter $\upsilon(p, x \in T, \xi \vdash E, q)$ dans le contexte Γ, nous nous trouvons avec la configuration suivante :



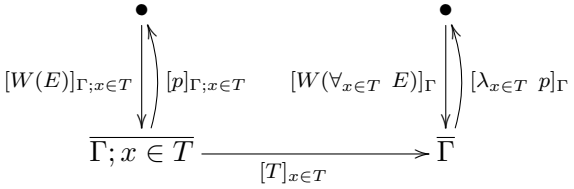
Comme on a une preuve p de $\exists_{x\in T}\ E$, la flèche $[T]_\Gamma \circ [W(E)]_{\Gamma;x\in T}$ est un épimorphisme. La présence de la flèche $[q]_{\Gamma;x\in T;\xi\vdash E}$ montre que $[W(C)]_{\Gamma;x\in T;\xi\vdash E}$ est un isomorphisme. Il en résulte que $[W(C)]_\Gamma$ est un épimorphisme, donc un isomorphisme. L'interprétation de $\upsilon(p, x \in T, \xi \vdash E, q)$ dans le contexte Γ est alors l'inverse de cet isomorphisme.

L'interprétation de $W(\forall_{x\in T}\ E)$ est l'image de l'objet $[W(E)]_{\Gamma;x\in T}$ de $\mathcal{T}/\Gamma; x \in T$ par le foncteur $\Pi_{[T]_\Gamma}$:

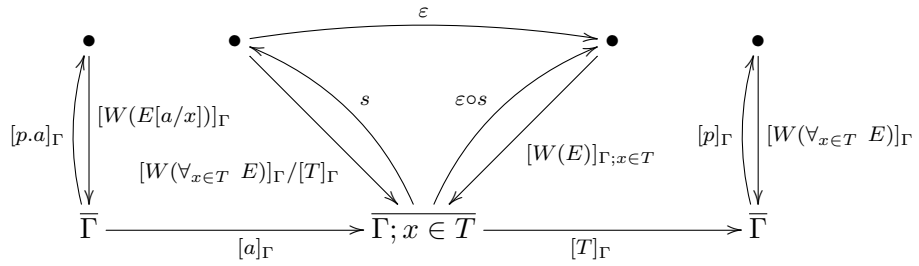


Notez que comme les adjoints à droite préservent les monomorphismes, et comme la flèche $[W(E)]_{\Gamma;x\in T}$, est un monomorphisme non seulement dans $\mathcal{T}$, mais aussi dans la catégorie relative $\mathcal{T}/\overline{\Gamma; x \in T}$ (il s'agit d'une flèche dont la cible est l'objet final), la flèche $[W(\forall_{x\in T}\ E)]_\Gamma$ est un monomorphisme dans $\mathcal{T}/\overline{\Gamma}$. Or, toujours comme adjoint à droite, $\Pi_{[T]_\Gamma}$ envoie l'objet final sur un objet final (il respecte les produits), c'est-à-dire sur un isomorphisme de $\mathcal{T}$ de cible $\overline{\Gamma}$. Il en résulte que $[W(\forall_{x\in T}\ E)]_\Gamma$ est un monomorphisme dans $\mathcal{T}$, ce qui fait qu'on ne viole pas par cette définition le principe de l'unicité du garant.

Ce foncteur transforme aussi une section $[p]_{\Gamma;x\in T}$ de la flèche $[W(E)]_{\Gamma;x\in T}$ (c'est-à-dire un élément global de l'objet $[W(E)]_{\Gamma;x\in T}$), provenant d'une preuve p de E dans le contexte $\Gamma; x \in T$, en une section de $[W(\forall_{x\in T}\ E)]_\Gamma$, qui sera notée $[\lambda_{x\in T}\ p]_\Gamma$.



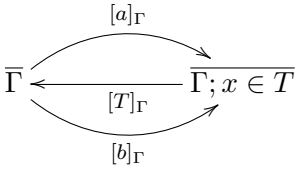
Si maintenant p est une preuve de $\forall_{x\in T}\ E$ dans le contexte Γ, donc représentée par la section $[p]_\Gamma$ de $[W(\forall_{x\in T}\ E)]_\Gamma$, alors le foncteur pullback le long de $[T]_\Gamma$ transforme $[p]_\Gamma$ en une section s de $[W(\forall_{x\in T}\ E)]_{\Gamma;x\in T}$. Par ailleurs, la coünité de l'adjonction entre ce foncteur et le foncteur $\Pi_{[T]_\Gamma}$ donne une flèche ε telle que $[W(E)]_{\Gamma;x\in T}\circ\varepsilon = [W(\forall_{x\in T}\ E)]_\Gamma/[T]_\Gamma$:



Donc, si a est un terme de type T, il suffit d'utiliser le foncteur pullback le long de $[a]_\Gamma$ sur la flèche $\varepsilon\circ s$ de $\mathcal{T}/\overline{\Gamma;x\in T}$, pour obtenir la flèche $[p.a]_\Gamma$, qui sera donc l'interprétation de la preuve $p.a$ de $E[a/x]$ dans le contexte Γ.

5 Le calcul.

Nous avons donc un langage et une sémantique dans un topos quelconque. Bien entendu, des termes de même type dans le même contexte, disons $a \in T/\Gamma$ et $b \in T/\Gamma$, sont interprétés comme des flèches parallèles, précisément comme des sections de la flèche représentant leur type commun :



et de tels termes a et b distincts peuvent éventuellement avoir la même sémantique : $[a]_\Gamma = [b]_\Gamma$. La sémantique induit donc une notion d'égalité entre les termes, qui est l'égalité au sens des mathématiques "de tout le monde".

La question qui nous intéresse est le calcul de cette sémantique. On sait bien qu'un tel calcul est impossible, d'abord parce que les signifiés eux–même sont trop abstraits pour nous être accessibles, et ont eux–mêmes besoin d'être représentés, par exemple par les notation introduites plus haut pour décrire les topos et leurs catégories relatives, et d'autre part parce que quand bien même on aurait accès aux signifiés, le signifié ne peut pas être calculable (au sens de Turing). S'il était calculable, toute démonstration serait inutile car les mathématiques seraient triviales.

On en est donc réduit à "approximer" le calcul idéal par un calcul engendrant une égalité plus faible. Ce calcul peut se définir par un ensemble de "règles de réécriture". Bien entendu, ce calcul n'est compatible avec notre sémantique que si les deux membres d'une règles de réécriture représentent toujours des signifiés égaux. On peut appeler cette propriété la "robustesse du calcul". Elle signifie tout simplement qu'appliquer les règles de calcul donne des calculs "justes". Les réécritures successives d'un terme, qui sont bien sûr des termes différents en général doivent représenter toutes la même flèche de notre topos. Il faut bien sûr s'assurer de cette robustesse, ce que nous ne ferons pas dans cet exposé. Toutefois, c'est un exercice assez facile.

Par ailleurs, il est indispensable que le calcul soit noethérien et préférable qu'il soit confluent. La noetherianité résulte très certainement d'une variante du théorème de Tait/Girard. Il est d'ailleurs probable que la notion de "candidat à la réductibilité" de Girard [6], soit indispensable, comme elle l'est pour le système F ou pour la théorie des constructions, car le système décrit ici autorise la quantification sur les termes de type Ω, qui sont par ailleurs considérés aussi comme des types (types de garants). Au jour d'aujourd'hui, je n'ai que partiellement adapté la démonstration de Tait/Girard. Cette démonstration fera sans doute l'objet d'un document séparé.

5.1 Règles de calcul.

Nous donnons maintenant les règles de calcul (ou de "réécriture") des termes de notre langage. Elles sont bien connues car héritées du lambda–calcul. Le système décrit ici est à deux niveaux (deux lambda–calculs parallèles), comme tous les systèmes qui formalisent à la fois les données usuelles (nombres, etc...) et la logique (énoncés, preuves). Ces deux niveaux "fonctionnent" essentiellement de la même façon ("formulae–as–type"). Comme le lambda-calcul ne prévoit de réduire que les termes et non pas les types, il n'est pas très étonnant qu'il n'y ait pas de règle de calcul spécifique associée au type Ω, puisque les énoncés jouent un rôle parallèle à celui des types. On pourrait introduire de telles règles. Toutefois, l'efficacité de ces règles resterait probablement très limité, et de toute façon, les techniques connues de recherche de preuve couvrent facilement ce genre d'optimisation. Nous nous en tiendrons donc à zéro règle spécifique concernant Ω. En d'autres termes : "un énoncé ne se calcule pas" (mais éventuellement, il se prouve).([23])

Nous avons séparé les règles de calcul en deux groupes : d'une part celles qui concernent les termes ordinaires (c'est-à-dire autres que les preuves), d'autre part celles qui concernent les preuves.

$\pi_1((a,b))$	$\rightsquigarrow$	a	
$\pi_2((a,b))$	$\rightsquigarrow$	b	
$(\pi_1(c),\pi_2(c))$	$\rightsquigarrow$	c	
$(x^T \mapsto b)(a)$	$\rightsquigarrow$	$b[a/x]$	
$x^T \mapsto f(x)$	$\rightsquigarrow$	f	si x n'a pas d'occurence libre dans f
$\delta(<a,p>)$	$\rightsquigarrow$	a	ces trois règles valables
$\epsilon(<a,p>)$	$\rightsquigarrow$	p	pour les deux sortes de
$<\delta(p),\epsilon(p)>$	$\rightsquigarrow$	p	δ et ϵ
$\upsilon(<a,p>, x \in T, \xi \vdash E, q)$	$\rightsquigarrow$	$q[a/x, p/\xi]$	remplacement en parallèle de x et ξ
$\upsilon(p, x \in T, \xi \vdash E, <x,\xi>)$	$\rightsquigarrow$	p	
$(\lambda_{x\in T}\ p).E$	$\rightsquigarrow$	$p[E/x]$	
$\lambda_{x\in T}\ p.x$	$\rightsquigarrow$	p	si x n'a pas d'occurrence libre dans p
$(\lambda_{\xi\vdash E}\ p).F$	$\rightsquigarrow$	$p[F/x]$	
$\lambda_{\xi\vdash E}\ p.x$	$\rightsquigarrow$	p	si x n'a pas d'occurrence libre dans p

23. L'une des conséquences de ce non–calcul des énoncés est que les données de type Ω sont représentées sur zéro bit à l'exécution. Mais elles sont représentées quand même. On peut très bien par exemple exécuter des fonctions à valeurs dans Ω. Mais bien sûr cela a un intérêt calculatoire faible, puisque le résultat aura zéro bits. Ceci est à rapprocher du fait que les types ne sont pas représentés à l'exécution.

5.2 Les mathématiques "de tout le monde".

Bien entendu, le système présenté ci–dessus est intuitionniste. Or, les mathématiques "de tout le monde" ne le sont pas, puisqu'elles sont "classiques". Ceci n'est pas un problème. En effet, il suffit d'ajouter au système le "schéma d'axiome du choix" autrement–dit un mécanisme qui permet de faire appel à n'importe quelle instance de ce schéma. Par ailleurs, il faut mettre la preuve du théorème de Diaconescu (qui peut être faite dans ce système) dans la bibliothèque de base, de même que les quelques théorèmes qui en découlent (double négation, par exemple) pour rendre possible le raisonnement par l'absurde. L'utilisation de l'axiome du choix se traduira alors (en interne dans le compilateur, mais d'une manière invisible pour l'utilisateur) par la présence d'un opérateur nouveau γ.

L'opérateur γ est en fait de type :

$$W((\forall_{x\in A}\ \exists_{y\in B}\ \varphi(x,y)) \Rightarrow (\exists_{f\in B^A}\ \forall_{x\in A}\ \varphi(x,f(x))))$$

Une question intéressante est de savoir si le compilateur peut déterminer si l'axiome du choix est indispensable dans une preuve. Il y a des circonstance dans lesquelles le choix peut être effectué. En effet, supposons qu'on ait une preuve p de l'énoncé $\forall_{x\in A}\ \exists_{y\in B}\ \varphi(x,y)$ de la forme (après normalisation éventuelle) :

$$\lambda_{x\in A}\ <a,q>$$

Compte tenu du type de γ, le compilateur pourra se trouver en présence de la preuve suivante :

$$\gamma.(\lambda_{x\in A}\ <a,q>)$$

qui sera donc une preuve de :

$$\exists_{f\in B^A}\ \forall_{x\in A}\ \varphi(x,f(x)).$$

Or, dans cette situation, il est facile de construire une autre preuve de ce même énoncé, à savoir :

$$<(x\in A)\mapsto a, \lambda_{x\in A}\ q[a/y]>$$

On aura bien sûr remarqué l'analogie frappante avec la preuve que Martin–Löf donne de l'"axiome du choix". Notre compilateur peut donc "optimiser" la preuve $\gamma.(\lambda_{x\in A}\ <a,q>)$ en la remplaçant par $<(x\in A)\mapsto a, \lambda_{x\in A}\ q[a/y]>$. Autrement–dit, on peut introduire une nouvelle règle de calcul :

$$\gamma.(\lambda_{x\in A}\ <a,q>) \rightsquigarrow <(x\in A)\mapsto a, \lambda_{x\in A}\ q[a/y]>$$

Cette règle "effectue" le choix, en éliminant l'opérateur γ. Le compilateur peut donc éventuellement par optimisation (l'optimisation utilise la réécriture) rendre constructive une preuve qui ne l'est pas a priori. Dans tous les cas, il peut renseigner l'utilisateur sur le caractère constructif de ses preuves. Par ailleurs, il peut aussi transformer les preuves constructives en programmes.

Notez qu'une règle comme celle–ci, qui réécrit des preuves et non pas des termes, respecte nécessairement l'égalité d'après le principe de l'unicité du garant. Il est donc loisible de l'utiliser aussi bien à l'optimisation qu'à l'exécution. On remarquera que le terme a qui est "prisonnier" dans la paire de Heyting $<a,q>$ reste prisonnier dans la paire de Heyting du membre de droite. La prison est simplement un peu plus grande. Seul l'opérateur de description permet de sortir de cette prison sans violer les propriété de l'égalité.

5.3 Choisir n'est pas calculer.

On pourrait être tenté d'introduire une règle plus forte que la précédente, en fait une règle analogue à celle qui concerne l'opérateur de description, qui permet de sortir de la "prison" mentionnée ci–dessus : $\delta(<a,p>) \rightsquigarrow a$ c'est-à-dire quelque chose comme : $\tau(<a,p>) \rightsquigarrow a$ où τ serait un "opérateur de choix de Hilbert", c'est-à-dire de type :

$$T^{W(\exists_{y\in A}\ E)}$$

où le type T est l'hôte de l'ensemble A, c'est-à-dire que A est de type Ω^T.

Sur le plan opératoire, cela ne pose pas de problème, et une telle règle de réécriture sera applicable dans les mêmes circonstances que celle concernant l'opérateur de description. En particulier l'arrêt du calcul résulte des mêmes théorèmes.

Toutefois, cette règle viole les principes de l'égalité. En effet, deux preuves p et q de l'énoncé $\exists_{y\in A}\ E$, qui sont nécessairement égales, pourraient par cette réduction produire des termes non égaux, violant ainsi la transitivité de l'égalité, et provoquant l'effondrement logique du système.

En d'autres termes, une telle règle n'est pas une "règle de calcul", puisqu'elle ne remplace pas une expression par une expression égale. On peut résumer cet argument par le slogan "choisir n'est pas calculer".

Il semble donc exclu d'utiliser cette règle à l'exécution des programmes. Par contre, on peut envisager de l'utiliser à la compilation, et ainsi d'effectuer le choix qui n'a pas été effectué par l'utilisateur, alors qu'il aurait pu éventuellement le faire. La réponse à cette question dépend clairement du droit qu'on a de "choisir". Elle dépasse le cadre de cet exposé, mais pourrait en tous cas être traitée sous forme d'interaction avec l'utilisateur.

Références

[1] **J.L. Bell** *The Development of Categorical Logic.* in D. Gabbay and F. Guenther, eds. *Handbook of Philosophical Logic.* Dordrecht : Kluwer.

[2] **A. Burroni** *Algèbres Graphiques.* Cahiers de Topologie et Géométrie Différentielle, vol. XXII, 3, 1981, pages 249–265.

[3] **H.B. Curry, R. Feys** *Combinatory Logic.* vol. 1, North–Holland, Amsterdam, 1958.

[4] **D. DeVidi** *Choice Principles and Constructive Logics.* Philosophia Mathematica (3) vol. 12, pages 222–243.

[5] **P. Freyd** : *Aspects of Topoi.* Bull. Austral. Math. Soc. **7** (1972), 1–76.

[6] **J-Y. Girard** : *Cours de D.E.A. 1986–87, Lambda–Calcul Typé.* Polycopié Université Paris 7.

[7] **R. Goldblatt** *Topoi.* 551 pages, Studies in Logic, North–Holland, 1984.

[8] **A. Heyting** *Die formalen Regeln der intuitionistischen Logik.* (1930) Sitzungsber. Preuss. Akad. Wiss., Phys. – Math Kl., 1930, pages 42–56.

[9] **W.A. Howard** *The formulae–as–type notion of construction.* To H.B. Curry. Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, London, 1980, pages 479–490.

[10] **J.M.E. Hyland, A.M. Pitts** *The theory of constructions : Categorical semantics and topos–theoretic models.* Contemporary Mathematics, vol. 92, 1989, pages 137–199.

[11] **J. Lambek, P.J. Scott** : *Introduction to higher order categorical logic.* Cambridge University Press, Cambridge 1986.

[12] **S. Mac Lane, I. Moerdijk** : *Sheaves in Geometry and Logic.* Universitext, 629 pages, Springer–Verlag, 1992.

[13] **M.E. Maietti, S. Valentini** *Can you add power–sets to Martin–Löf intuitionnistic set theory ?* Math. Logic Quarterly 45, pages 521–532.

[14] **P. Martin–Löf** *Intuitionistic Type Theory.* Notes by G. Sambin of a series of lectures given in Padua, June 1980. Bibliopolis, Napoli 1984.

[15] **D. Pavlović** *Predicates and Fibrations.* Thèse. Rijksuniversiteit Utrecht, 1990.

[16] **A. Prouté** : *Expressions Indéterminées, Constructivisme et Axiome du Choix.* Cahiers de Topologie et Géométrie Différentielle Catégoriques XXXIII, 1992, pages 279–287.

[17] **A. Prouté** : *On the Role of Description.* J. of Pure and Applied Algebra 158 (2001), 295–307.

[18] **W. W. Tait** : *Intentional Interpretation of functionals of finite type I.* J. Symbolic Logic 32, pages 198–212.

[19] **W.W. Tait** : *The law of excluded middle and the axiom of choice.* in Alexander Georges, ed. Mathematics and Mind. Oxford University Press, pages 45–70.