Expressions Indéterminées, Constructivisme et Axiome du Choix

Alain Prouté*

Abstract: We introduce indeterminate expressions (in which substitution does not respect equality). This gives a convenient treatement of sub–types and quotient types in a construtive mathematical system. The Axiom of Choice is revisited at the light of this concept, and an example stresses the interest of the notion introduced.

1 Introduction

1.1 Algorithmes et Fonctions

On a souvent tendance à confondre algorithmes et fonctions, ou tout au moins à considérer qu'un algorithme est une fonction. En fait, il n'en est rien en général, car pour qu'un algorithme soit une fonction, il est nécessaire qu'il respecte l'égalité.

En effet, un algorithme opère non pas sur des objets mathématiques, mais sur des représentations d'objets mathématiques. Il n'y a donc aucune garantie en général (pour un algorithme quelconque), que deux expressions distinctes représentant le même objet mathématique (i.e: égales), soient transformées par l'algorithme en des expression égales.

En fait, dans la plupart des systèmes formels, la syntaxe des algorithmes est telle, que l'on peut en général montrer qu'il n'est possible d'écrire que des algorithme respectant l'égalité. Mais bien sûr, la remarque ci–dessus laisse entrevoir la possibilité d'imaginer des systèmes formels dans lesquels on utiliserait des algorithmes ne respectant pas l'égalité. Le problème est de savoir si cela présente un intérêt. Le but de cet exposé est de démontrer que oui.

Pour pouvoir developper cette idée, nous devons nous placer dans un contexte formel. Nous adoptons un système formel de type "Martin–Löf", auquel nous allons apporter des modifications. Il n'est pas indispensable que le lecteur soit familier avec la théorie de Martin–Löf, car nous allons rappeler (sans entrer dans trop de détails) les principles idées de cette "théorie des types".

Notations: Par "connecteur logique", nous entendons l'un des symboles suivants: $\top$ (vrai), $\bot$ (faux), $\wedge$ (et), $\vee$ (ou), $\Rightarrow$ (implique), $\exists$ (il existe), $\forall$ (pour tout) et $\neg$ (non). Ces connecteurs ne sont pas indépendants les uns des autres, et en particulier, on doit considérer que pour tout énoncé A, l'énoncé $\neg A$ n'est qu'une notation pour $A \Rightarrow \bot$. De même, $\top$ et $\bot$ peuvent être traités comme des cas particuliers de $\wedge$ et $\vee$, puisqu'ils ne sont que les éléments neutres de ces opérations.[1] Par ailleurs, nous utilisons des crochets, comme dans $E[x]$ pour signifier que E est

*Université Paris 7, ap@mathp7.jussieu.fr

[1] Par contre on n'a pas d'équivalence du genre $A \wedge B \simeq \neg(\neg A \vee \neg B)$ (loi de De Morgan), ou $\exists x \in E\ \varphi(x) \simeq \neg\forall x \in E\ \neg\varphi(x)$. En effet, nous n'admettons que les règles intuistionnistiquement valides (qui sont aussi classiquement

une expression contenant éventuellement des occurences libres de la variable x. Le résultat de la substitution de a à x dans $E[x]$ sera noté $E[a]$.

1.2 L'Interprétation BHK

L'interprétation de Brouwer-Heyting-Kolmogorov (interprétation BHK), est une liste d'*exigences* que l'on peut avoir quant–à la signification des connecteurs logiques. Ces exigences sont celles des intuitionnistes qui, comme on sait, réfutaient les preuves d'existence non constructives. En voici la liste.

- Une preuve de $A \wedge B$ est un couple formé d'une preuve de A et d'une preuve de B.
- Une preuve de $A \vee B$ est un couple formé d'un drapeau (valant 0 ou 1), et d'une preuve de A si le drapeau vaut 0, sinon d'une preuve de B si le drapeau vaut 1.
- Une preuve de $A \Rightarrow B$ est un algorithme construisant une preuve de B à partir d'une preuve quelconque de A.
- Une preuve de $\forall x \in E\ \varphi(x)$ est un algorithme construisant une preuve de $\varphi(a)$ à partir de toute expression a représentant un élément de E.
- Une preuve de $\exists x \in E\ \varphi(x)$ est un couple formé d'une expression a représentant un élément de E, et d'une preuve de $\varphi(a)$.

Bien sûr, pour que cette interprétation soit possible, une preuve ne doit pas être confondue avec le texte (expression) qui la représente. En effet, ce n'est pas parce qu'une preuve de $A \wedge B$ est un couple, que toute expression représentant une telle preuve doive s'écrire sous la forme (P, Q).

On remarquera que toute preuve est alors soit un couple soit un algorithme, c'est–à–dire deux des concepts de base de l'informatique.

Remarquons aussi, et c'est là le cœur du sujet de cet exposé, qu'une preuve d'une implication ou d'un énoncé universellement quantifié sont des algorithmes, et non pas des fonctions. En effet, il n'entre pas dans nos exigences (ni dans celles des intuitionnistes) que ces algorithmes respectent l'égalité des preuves. Pour le mathématicien classique, cela va de soi, puisqu'il se moque éperduement de l'égalité entre preuves.

Remarquons enfin que la règle concernant $A \vee B$ montre que si A et B sont démontrables tous les deux, alors il y a au moins deux preuves *distinctes* de $A \vee B$, car une preuve de $A \vee B$ de la forme $(0, \textit{preuve de } A)$ est nécessairement distincte d'une preuve de la forme $(1, \textit{preuve de } B)$, sinon le drapeau ne sert plus à rien. Or ce drapeau est essentiel lorque l'on veut exploiter constructivement une hypothèse de la forme $A \vee B$ (raisonnement par disjonction des cas).

On peut ajouter dans la même veine, les deux exigences suivantes:

- il y a une et une seule preuve de $\top$,

valides). En fait, nous n'admettons que les règles qui découlent de l'interprétation BHK. Pour le lecteur qui n'est pas familier avec l'intuitionisme, signalons simplement que les intuitionnistes admettent moins d'axiomes que les classiques. Un classique peut donc admettre sans problème tous les principes intuitionnistes de démonstration, qui font tous partie des siens.

- il n'y a pas de preuve de $\bot$.

qui ne font qu'exprimer le fait que $\top$ est l'élément neutre du connecteur $\wedge$, et $\bot$ l'élément neutre du connecteur $\vee$.

1.3 Contructivisme contre Classisisme

Les exigence de l'interprétation BHK sont bien sûr des exigences "constructivistes", puisqu'elles impliquent en particulier que dès que l'on détient une preuve d'existence (exécutée), on détient une représentation de l'objet dont l'existence est affirmée par cette preuve. Bien sûr, ceci est subordonné au fait que les expressions représentant des éléments soient exécutables, et en particulier que l'exécution d'une expression représentant un couple donne effectivement un couple, etc... Nous ne pouvons entrer ici dans trop de détails à ce sujet, mais nous affirmons que notre système est fait de telle façon que toute expression est exécutable. Le lecteur qui sait comment est exécuté ou compilé un langage applicatif en sera convaincu sans peine.

On peut se demander qu'elle est la cause de la non constructivité des mathématiques classiques. Cette cause est le raisonnement par l'absurde.[2] Le principe du raisonnement par l'absurde peut s'exprimer par divers énoncés intuistionnistiquement équivalents. Nous n'en retiendrons qu'un, le principe du tiers exclus, que voici:[3]

$$\forall p \in \Omega \; p \vee (\neg p).$$

Ce principe dit donc que tout énoncé ne peut être que vrai ou faux (Aristote).

Par ailleurs, K. Gödel (1931)[5] a démontré qu'un système assez puissant pour permettre l'arithmétique contient nécessairement des énoncés indécidables.[4] On constate alors facilement que le constructivisme et le tiers exclus sont incompatibles. En effet, soit p un énoncé indécidable. D'après le tiers exclus, on a une preuve de $p \vee (\neg p)$, et d'après la deuxième des exigences de l'interprétation BHK, cette preuve contient soit une preuve de p, soit une preuve de $\neg p$, ce qui est impossible.

On doit donc choisir entre classisisme (acceptation du tiers exclus) et constructivisme (renonciation au tiers exclus et à tout ce qui l'entraine: raisonnement par l'absurde, double négation, formes trop fortes de l'axiome du choix).

1.4 Le Principe "Énoncé = Type"

Il y a une façon simple de satisfaire *de facto* aux exigences ci–dessus. Il suffit de décider que les énoncés sont des types[5]. On doit intuitivement comprendre qu'un énoncé, lorsqu'il est vu

[2] et dans une bien moindre mesure l'axiome du choix comme on va le voir dans la suite.

[3] Ω est l'ensemble intuitionniste des "valeurs de vérité", c'est–à–dire le quotient de l'ensemble de tous les énoncés par la relation d'équivalence engendrée par la déductibilité. Autrement–dit, deux énoncés représentent le même élément de Ω si et seulement si leur équivalence logique est démontrable. Ce type ne figure pas dans la théorie de Martin–Löf. Il est inspiré de la théorie des topos élémentaires. Il faut donc se souvenir qu'un énoncé a un double rôle. D'une part, il représente un élément de Ω, d'autre part il représente un type. Les règles d'introduction pour Ω sont celles de la formation des énoncés. Les règles d'élimination sont précisément les règles de formation des sous–types et des types quotients. Quant–à la règle de conversion, elle dit simplement qu'un énoncé est vrai (égal à $\top$) dès qu'on en a une preuve.

[4] c'est–à–dire indémontrables, de même que leur négation. Ceci montre que Ω a au moins trois éléments. En fait, on peut montrer que Ω a une infinité d'éléments.

[5] Cette idée merveilleuse semble avoir été exprimée pour la première fois dans Curry et Feys (1958)[3]. Elle a été popularisée par Howard (1980)[6] à l'aide du lambda–calcul, mais c'est Martin–Löf (1975)[13] et (1982)[14] qui

comme un type, est simplement l'*ensemble des preuves* de cet énoncé. On peut alors traduire l'interprétation BHK en langage mathématique usuel, ce qui donne la *définition* suivante des connecteurs logiques:[6]

- $A \wedge B = A \times B$,
- $A \vee B = A \amalg B$,
- $A \Rightarrow B = \overline{B^A}$
- $(\forall x \in E\ \varphi(x)) = \overline{\prod}_{x\in E}\varphi(x)$,
- $(\exists x \in E\ \varphi(x)) = \coprod_{x\in E} \varphi(x)$,
- $\top = \{*\}$ (ensemble ayant un seul élément),
- $\bot = \emptyset$.

où $\overline{B^A}$ représente le type des algorithmes (ne respectant pas nécessairement l'égalité) de A vers B (et non pas celui des fonctions de A vers B, qui est noté B^A), et $\overline{\prod}_{x\in E}A(x)$ le type des algorithmes (ne respectant pas nécessairement l'égalité) décrivant des familles d'éléments des types $A(x)$ quand x varie dans E (et non pas le produit au sens habituel).

Commentons un peu cette "traduction". Il faut se souvenir que l'on a confondu tout énoncé A avec l'ensemble de ses preuves. On peut donc lire la première définition ci–dessus comme suit: "l'ensemble des preuves de $A \wedge B$ est le produit cartésien de l'ensemble des preuves de A et de l'ensemble des preuves de B". Ce n'est qu'une façon d'énoncer la première règle de l'interprétation BHK.

Les autres définitions s'interprètent de même. Nous nous attarderons juste un peu sur celle du connecteur $\exists$. Il faut imaginer qu'un élément de l'union disjointe $\coprod_{x\in E} F(x)$ de la famille d'ensembles $(F(x))_{x\in E}$ est la même chose qu'un couple (i, x), où i est un élément de E (que l'on pourrait appeler "l'indice" de (i, x)), et x un élément dont le type dépend de la valeur de i. En fait, il y a la même différence entre ce concept et celui de produit cartésien, qu'entre la notion d'espace fibré et celle de produit cartésien (fibré trivial) d'espaces topologiques. Les informaticiens ont baptisé cette construction "produit dépendant", pour signifier un ensemble de couples dont le type du deuxième élément dépend de la valeur du premier élément. Le lecteur pourra essayer d'interpréter le produit $\prod_{x\in E} F(x)$ comme un ensemble de "fonctions dépendantes" (en fait l'analogue d'une section d'un espace fibré). C'est pourquoi nous adoptons la notation des fonctions $x \mapsto f(x)$ pour représenter un élément d'un tel produit. L'ensemble $\overline{\prod}_{x\in E}F(x)$ est interprété comme un ensemble d'algorithmes dépendants. Bien sûr, le produit dépendant $\coprod_{x\in E} F(x)$ a deux projections, la première sur E (projection sur la base d'un fibré),

en fait l'exploitation la plus systématique. Il n'en reste pas moins vrai qu'elle n'est que le prolongement naturel de l'interprétation BHK, et donc essentiellement dûe à Brouwer (1923)[1], Heyting (1930)[7], (1931)[8], (1934)[9] et Kolmogorov (1932)[11].

[6]Pour définir un système formel, on doit énoncer les familles de règles suivantes: règles de formation des types et de leurs éléments, qui permettent d'introduire les notations pour les types et leurs éléments, règles d'élimination, qui permettent d'exploiter les expressions construites, règle de calcul (ou de conversion) qui définissent les égalités entre expressions et donc la sémantique du système formel. Comme nous ne définissons pas complètement notre système formel dans cet exposé, nous laissons le soin au lecteur de deviner une bonne partie de ces règles, qui sont de toute façon très naturelles. Vers la fin de cet exposé, nous utilisons sans le dire des règles que nous n'avons pas précisées.

la deuxième "dépendante" (projection sur la fibre au dessus du point qui est la projection sur la base).

Le système de Martin–Löf contient d'autres constructeurs de types que ceux cités plus haut. En gros, il y a un type des entiers **N**, un schéma de construction de types récursifs (dont **N** n'est qu'un cas particulier), et des univers U_1, U_2,... permettant de parler de la classe des ensembles, etc ...

Par contre, il n'y a pas de constructeur de sous–types, ni de constructeur de types quotients. Ce sont ces concepts que nous allons introduire dans la section suivante, et nous allons voir comment ils nous mènent à renoncer au respect de l'égalité par la substitution. Ceci ne veut pas dire que nous renonçons à l'interprétation extensionnelle de l'égalité entre fonctions. Nous allons revenir sur cette question dans la section suivante.

Nous terminons cette section par un exemple (bien classique) de démonstration écrite dans l'esprit de l'interprétation BHK. Soient A et B deux types, et $R(x, y)$ une relation binaire entre A et B. Alors, il est bien connu que l'on a:

$$\exists x \in A\ \forall y \in B\ R(x, y) \Rightarrow \forall y \in B\ \exists x \in A\ R(x, y).$$

En voici la preuve. Il s'agit de trouver un algorithme $p \mapsto E(p)$, où p désigne une preuve quelconque de $\exists x \in A\ \forall y \in B\ R(x, y)$, et $E(p)$ une preuve de $\forall y \in B\ \exists x \in A\ R(x, y)$. Notez que p appartient à $\coprod_{x\in A} \overline{\prod}_{y\in B} R(x, y)$. p a donc deux projections, que l'on va noter (en style informatique) $p.0$ et $p.1$. $p.0$ est un élément de A, et $p.1$ est un élément de $\overline{\prod}_{y\in B} R(p.0, y)$.

Nous devons maintenant produire un élément de $\overline{\prod}_{y\in B} \coprod_{x\in A} R(x, y)$. En voici un: $y \mapsto (p.0, p.1(y))$. Ce qui fait que la démonstration cherchée s'écrit:

$$p \mapsto (y \mapsto (p.0, p.1(y))).$$

Nous donnerons d'autres exemples plus intéressants de telles démonstrations dans la suite de cet exposé.

2 Sous–Types et Types Quotient

2.1 Expressions Indéterminées

Une expression $E[x]$ contenant éventuellement des occurences libres de la variable x est dite *déterminée en* x, si pour deux expression égales a et b, les expressions $E[a]$ et $E[b]$ sont égales. Si l'on ne sait pas prouver que $E[x]$ est déterminée en x, on dira que $E[x]$ est *indéterminée* en x.

Notez que le fait que l'expression $E[x]$ soit indéterminée en x, ne signifie pas qu'il y ait des doutes sur l'existence de $E[x]$. Il n'y a de doute que sur sa "classe d'égalité". Du point de vue informatique, cela signifie que $x \mapsto E[x]$ est un algorithme qui termine effectivement et produit un élément du bon type, mais qui par contre ne respecte pas l'égalité.

Cette notion impose une vigilence nouvelle lorsque l'on fait des calculs. En effet, nos habitudes veulent que de $a = b$ on déduise en général $E[a] = E[b]$.

2.2 Égalité Extensionnelle et Égalité Intentionnelle

On distingue deux sortes de système formels, suivant que la sémantique de l'égalité y est *extensionnelle* ou *intentionnelle*. Pour expliquer cette distinction, et le sens de ces adjectifs, il est nécessaire d'évoquer le théorème d'incomplétude de Gödel, qui affirme que dans tout système formel assez puissant pour que l'on puisse y faire de l'arithmétique, il existe des énoncés indécidables. C'est le cas bien sûr du système qui nous intéresse. Il en résulte en particulier que l'égalité extensionnelle (c'est–à–dire au sens habituel des mathématiques: $f = g \Leftrightarrow \forall x\ f(x) = g(x)$) entre fonctions de $\mathbf{N}$ vers $\mathbf{N}$ n'est pas testable par algorithme, ou encore que l'égalité vue comme une fonction de $\mathbf{N}^{\mathbf{N}} \times \mathbf{N}^{\mathbf{N}}$ vers $\{\top, \bot\}$ n'est pas programmable.

Il y a donc un décalage entre ce que l'on aimerait que l'égalité soit, et ce que nous somme capables de calculer. La notion d'égalité que l'on obtient en s'en tenant à des algorithmes testant l'égalité des fonctions (et aussi d'autres types d'objets) se nomme *égalité intentionnelle*. Elle est plus faible que l'égalité extensionnelle, en ce sens que si deux objets sont intentionnellement égaux, alors il sont extensionnellement égaux, mais pas réciproquement. Bien sûr, une définition précise de l'égalité intentionnelle dépend fortement d'une définition précise du système formel. Nous ne nous y livrerons pas ici.[7]

Toutefois, il y a des types pour lesquels les deux notions d'égalité se confondent. Quand c'est le cas, nous dirons que le type est "normal". Le type $\mathbf{N}$ des entiers est normal.[8]

Du point de vue informatique, un type normal est un type pour lequel il existe un algorithme de normalisation confluent et Noethérien, c'est–à–dire un algorithme transformant toute expression de ce type en une expression normale (forme normale) telle que deux expressions soient (extensionnellement) égales si et seulement si leurs formes normales sont syntaxiquement identiques.

Supposons maintenant que nous effectuons les calculs en "appel par valeur", ce qui signifie que l'argument a d'un terme applicatif (de la forme $f(a)$) est calculé avant que ne lui soit appliqué la fonction ou l'algorithme f. Alors, si A est normal, f va nécessairement respecter l'égalité. En effet, le calcul de l'argument a pour effet de remplacer l'expression représentant cet argument par sa forme normale, et il en résulte que deux expressions égales de type A seront toujours remplacées par des expressions *syntaxiquement identiques*, avant même que l'algorithme représentant f ne leur soit appliqué. Comme bien sûr un algorithme donne des résultats identiques sur des expressions identiques, les résultats obtenus seront égaux parce qu'identiques. Un des axiomes cruciaux de notre système est donc le suivant (Axiome de détermination):

Si A est un type normal, x une variable de type A, et $E[x]$ une expression quelconque, alors

[7] Martin–Löf semble hésiter fortement entre un système extensionnel et un système intentionnel, comme le montrent les revirements dont il fait preuve au long de ses écrits. Pour notre part, nous estimons que seule l'égalité extensionnelle est mathématiquement signifiante, et nous tenons le problèmes de "l'approximation la meilleure possible" de l'égalité extensionnelle par une égalité intentionnelle (donc calculable) comme le problème d'intelligence artificielle crucial pour l'automatisation de la recherche de preuves. Bien sûr, il résulte du théorème de Gödel qu'une telle approximation sera toujours perfectible, et ne sera donc jamais la meilleure possible.

[8] Ce qui est très loin d'être trivial. La démonstration dépend d'ailleurs de l'ensemble des constructeurs du système formel. Elle repose sur un théorème de "normalisation", dont les diverses versions présentes dans la littérature trouvent leur origine dans les travaux de G. Gentzen sur l'élimination des coupures. Le cœur de la preuve est une induction assez complexe, équivalente à un raisonnement par récurrence transfinie sur l'ordinal ε_0, ordinal limite de la suite ω, ω^ω, ω^{ω^ω},... C'est grâce à ce théorème que Gentzen a pu montrer la consistance de l'arithmétique, ce qui, compte tenu du théorème d'incomplétude de Gödel, montre qu'une telle induction n'est pas formalisable dans l'arithmétique. Voir par exemple Lambek et Scott (1986)[12], pour une démonstration complète, dans le cadre d'un système formel à base de topos.

$E[x]$ est déterminée en x.

2.3 Deux Constructeurs de Types

En mathématiques on utilise très souvent des ensembles quotients, ou des sous–ensembles. On imagine d'ailleurs difficilement faire sérieusement des mathématiques sans ces notions. Aussi allons nous ajouter deux constructeurs de types au système présenté précédemment.

- Si A est un type, et $p[x]$ un énoncé contenant d'eventuelles occurences de la variable x (de type A), alors
$$\{x \in A \mid p[x]\}$$
est un type (appelé un "sous–type" de A),
- Si A est un type, et R une relation binaire (non nécessairement d'équivalence) sur A, alors A/R est un type (appelé un "type quotient" de A).

Bien sûr, il faut comprendre $\{x \in A \mid p[x]\}$ comme l'ensemble des éléments x de A, pour lesquels $p[x]$ est vrai, et A/R comme le quotient de A par la relation d'équivalence *engendrée* par R. Dans cet exposé nous n'étudirons que le constructeur de sous–types.

Comment fait–on pour construire un élément de $\{x \in A \mid p[x]\}$? On commence par construire un élément a de A, puis on prouve $p[a]$. On voit donc, que si l'on désire conserver une trace de cette preuve dans la représentation de l'élément obtenu[9], il faut représenter un élément d'un sous–type de A comme un couple, formé d'un élément de A et d'une preuve que cet élément satisfait l'énoncé caractérisant ce sous–type.

Toutefois, l'inclusion de $\{x \in A \mid p[x]\}$ dans A doit être injective. Il en résulte que si on démontre $p[a]$ à l'aide de deux preuves différentes (disons P et Q), les deux couples
$$(a, P) \qquad \text{et} \qquad (a, Q)$$
représentent le même élément de $\{x \in A \mid p[x]\}$, même si la preuve P n'est pas égale à la preuve Q.

Comme un élément b de $\{x \in A \mid p[x]\}$ est un couple, interprétons les deux projections. $b.0$ est le premier élément du couple, autrement–dit, $b \mapsto b.0$ est l'inclusion canonique de $\{x \in A \mid p[x]\}$ dans A. $b.1$ est une preuve de $p[b.0]$. La discussion précédente montre clairement que l'expression $b.1$ est indéterminée en b. Pour mettre ce phénomène bien en évidence dans les notation, nous abandonnons la notation $b.1$ dans cette situation et la remplaçons par la notation $\$b$. La caratère $ sera dans toute la suite réservé pour les notations introduisant de "l'indétermination" dans les expressions.

3 L'Axiome du Choix

L'axiome du choix est l'énoncé suivant, dans lequel A et B sont des types quelconques, et $R(x, y)$ une relation entre A et B.
$$\forall x \in A\ \exists y \in B\ R(x, y) \Rightarrow \exists f \in B^A\ \forall x \in A\ R(x, f(x)).$$

[9] si l'on veut réellement réaliser un système constructif, il est primordial de ne rien jetter à la poubelle.

Autrement–dit, si on sait que pour tout x de A il existe un y de B, tel que $R(x,y)$, alors il existe une fonction f de A vers B permettant pour tout x de *choisir* un tel y.

3.1 Preuves de l'Axiome du Choix

Martin–Löf *prouve* l'axiome du choix dans son système.[10] Voici sa preuve:

$$p \mapsto (x \mapsto p(x).0, x \mapsto p(x).1).$$

Nous laissons au lecteur le temps d'admirer la simplicité, l'élégance et la parfaite symétrie de cette preuve avant de lui fournir quelques explications.

p est une preuve de la premisse, à savoir $\forall x \in A\ \exists y \in B\ R(x,y)$. Il en résulte donc que $p(x)$ (pour x donné dans A) est un couple formé d'un élément $p(x).0$ de B et d'une preuve $p(x).1$ de $R(x, p(x).0)$. Ceci nous donne la fonction f cherchée: $x \mapsto p(x).0$, et une preuve de $\forall x \in A\ R(x, f(x))$, à savoir $x \mapsto p(x).1$.

Cette preuve n'est par contre pas correcte dans notre système. En effet, la preuve p est un élément de $\overline{\prod}_{x\in A} \ldots$, et par conséquent, l'expression $p(x)$ est indéterminée en x. On ne peut donc pas considérer l'expression $x \mapsto p(x).0$ comme une fonction, mais seulement comme un algorithme. Il est dès lors clair, que dans notre système, les deux énoncés plus faibles suivants sont démontrés:

"Axiome du Choix Indéterminé" (Notez le remplacement de B^A par $\overline{B^A}$):

$$\forall x \in A\ \exists y \in B\ R(x,y) \Rightarrow \exists f \in \overline{B^A}\ \forall x \in A\ R(x, f(x)).$$

"Axiome du Choix Normal" (On suppose que A est normal):

$$\forall x \in A\ \exists y \in B\ R(x,y) \Rightarrow \exists f \in B^A\ \forall x \in A\ R(x, f(x)).$$

En particulier, l'axiome des choix dénombrables ($A = \mathbf{N}$) est démontrable dans notre système. On voit donc que l'axiome du choix, à condition de ne pas en utiliser des versions trop générales n'est pas un obstacle à la constructivité.

Nous allons voir un peu plus loin que la forme générale de l'Axiome du Choix donnée au début de cette section ne peut pas être démontrée dans notre système.

3.2 Le Théorème de Diaconescu

Radu Diaconescu (1975)[4] à démontré (d'une manière non formelle) que l'axiome du choix entraine le tiers exclus. Nous allons donner sa démonstration dans notre langage formel (sans l'écrire complètement).

Remarquons tout de suite que le théorème de Diaconescu ne peut pas être démontré dans le système de Martin–Löf. En effet, s'il pouvait l'être, le tiers exclus pourrait être démontré dans un système constructif contenant l'arithmétique, ce qui est impossible.

Commençons par introduire deux fonctions α et β de Ω vers Ω, par les formules suivantes:

$$\alpha(x) = x \vee (\neg x \wedge p), \qquad \beta(x) = \neg x \vee (x \wedge p).$$

[10] Ce qui peut surprendre, quand on pense que le système de Martin–Löf est constructif. Comme nous allons le voir, ceci démontre en fait une insuffisance de ce système.

Il est alors immédiat de prouver :

$$\alpha(\top) = \top, \qquad \beta(\bot) = \top, \qquad p \Rightarrow (\alpha = \beta).$$

Posons $\Gamma = \{g \in \Omega^\Omega \mid g = \alpha \vee g = \beta\}$. Alors $\overline{\alpha} = (\alpha, *)$ et $\overline{\beta} = (\beta, *)$ sont des éléments de Γ, où $*$ désigne des preuves d'énoncés "évidents", comme d'ailleurs dans la suite de cette démonstration. Les règles définissant la sémantique des sous–types donnent une preuve de $\alpha = \beta \Rightarrow \overline{\alpha} = \overline{\beta}$, donc de $p \Rightarrow \overline{\alpha} = \overline{\beta}$.

On peut alors prouver $\forall X \in \Gamma\ \exists x \in \Omega\ (X.0)(x)$, comme ceci:

$$X \mapsto [q \mapsto (\top, *), q \mapsto (\bot, *)](\$X).$$

où $[g, h]$ désigne l'unique fonction ou algorithme de $E \amalg F$ vers G dont les composantes sont g et h (règle d'élimination du constructeur $\amalg$).

Soit maintenant Δ une preuve de notre prémisse (c'est–à–dire de l'axiome du choix), alors Δ appliqué à la preuve ci–dessus donne une preuve de

$$\exists f \in \Omega^\Gamma\ \forall X \in \Gamma\ (X.0)(f(X))$$

On pose donc

$$f = (\Delta(X \mapsto [q \mapsto (\top, *), q \mapsto (\bot, *)](\$X)).0$$
$$K = (\Delta(X \mapsto [q \mapsto (\top, *), q \mapsto (\bot, *)](\$X)).1$$

(K est une preuve de $\forall X \in \Gamma\ (X.0)(f(X)))$, et on peut conclure que $\alpha(f(\overline{\alpha}))$ (par $K(\overline{\alpha})$) et $\beta(f(\overline{\beta}))$ (par $K(\overline{\beta})$).

Par définition de α on a:

$$(f(\overline{\alpha}) \vee (\neg f(\overline{\alpha}) \wedge p))$$

ce qui entraine $f(\overline{\alpha}) \vee p$. De même, on déduit $\neg f(\overline{\beta}) \vee p$, et de ces deux énoncés on déduit $\Big(f(\overline{\alpha}) \wedge \neg f(\overline{\beta})\Big) \vee p$.

Il suffit donc pour terminer de montrer que $f(\overline{\alpha}) \wedge \neg f(\overline{\beta})$ entraine $\neg p$ (i.e: $p \Rightarrow \bot$). Mais ceci résulte immédiatement de ce que p entraine $\overline{\alpha} = \overline{\beta}$, et de ce que f *est une fonction* et non point seulement un algorithme. □

Il est important de remarquer dans la preuve ci–dessus que l'on utilise la forme forte de l'axiome du choix. En effet, le type Γ, bien que ne semblant ne contenir que deux éléments est très loin d'être un type normal.[11]

Par ailleurs, nous avons déjà remarqué que cette démonstration ne peut pas être faite dans le système de Martin–Löf (même si on y ajoute le type Ω). Ceci montre que notre système apporte quelquechose de plus. En fait il apporte tout simplement une notion convenable de sous–type. Un système comme Nuprl (Constable et al. (1986)[2]) qui est essentiellement une réalisation informatique du système de Martin–Löf prétend avoir des sous–types. En fait, il définit l'ensemble des x de A satisfaisant $p[x]$ comme $\coprod_{x \in A} p[x]$, c'est–à–dire qu'il considère comme nous qu'un élément du sous–type doit être représenté par un couple, mais sa notion d'égalité sur le sous–type n'est pas la bonne, puisque des couples (a, P) et (a, Q) avec P et Q distinct représentent alors des éléments distincts du sous–type. En particulier, l'injection canonique du sous–type dans A n'est pas injective, ce qui est ennuyeux.

[11] En fait cette démonstration montre qu'il n'est pas normal, car s'il était (prouvablement) normal, on pourrait démontrer le tiers exclus dans notre système constructif, ce qui n'est bien sûr pas possible. De toute façon, on voit bien que Γ dépend de p (l'expression qui le définit n'est pas fermée), ce qui montre que sans hypothèse sur p, on ne peut même pas savoir si Γ a deux éléments ou un seul.

3.3 Conclusion

Nous espérons avoir démontré que l'introduction des expressions indéterminées permet d'avoir des sous–types ayant une sémantique en accord avec celle des mathématiques usuelles. Par ailleurs, nous avons aussi vu que l'absence de cette notion rend impossible la démonstration de certains théorème pourtant élémentaires, dont le théorème de Diaconescu n'est qu'un exemple.

References

[1] **L.J.E. Brouwer** *(1923) Intuïtionistische splitsing van mathematische grondbegrippen (Hollandais).* Nederl. Akad. Wetensch. Verslagen. 32, 877-880. (Traduction Allemande dans: Jahresber. Dtsch. Math.-Ver. 33, 251-256.

[2] **R.L. Constable et al.** *(1986) Implementing Mathematics with the Nuprl Proof Development System.* Prentice-Hall, Englewood Cliffs, New-Jersey.

[3] **H.B. Curry et R. Feys** *(1958) Combinatory Logic I.* North-Holland, Amsterdam.

[4] **R. Diaconescu** *(1975) Axiom of Choice and Complementation.* Proc. Amer. Math. Soc. 51, 176-178.

[5] **K. Gödel** *(1931) Über formal unentscheidbare Sätze der Principia Mathematica und verwandter System I.* Monatsh. Math. Phys. 38, 173-198.

[6] **W.A. Howard** *The formulae-as-types notion of construction.* dans: Seldin et Hindley: *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism.* Academic Press, New-York 480-490.

[7] **A. Heyting** *(1930) Sur la logique intuitionniste.* Acad. Roy. Belg. Bull. Cl. Sci. (5) 16, 957-963.

[8] **A. Heyting** *(1931) Die Intuitionistische Grundlegung der Mathematik.* Erkenntnis. 2, 106-115.

[9] **A. Heyting** *(1934) Mathematische Grundlagenforschung. Intuitionismus. Beweisstheorie.* Springer Berlin. Réimprimé en 1974. Traduction Française augmentée: voir Heyting 1955.

[10] **A. Heyting** *(1955) Les fondements des mathématiques. Intuitionnisme. Théorie de la démonstration.* Gauthier-Villars, Paris.

[11] **A.N. Kolmogorov** *(1932) Zur Deutung der intuitionistischen Logik.* Math. Z. 35, 58-65.

[12] **J. Lambek et P.J. Scott** *(1986) Introduction to Higher Order Categorical Logic.* Cambridge Studies in Advanced Math. 7.

[13] **P. Martin-Löf** *(1975) An Intuitionistic theory of types: predicative part. dans: Rose and Shepherdson: Logic Colloquium '73* North-Holland, Amsterdam, 73-118.

[14] **P. Martin-Löf** *(1982) Constructive Mathematics and Computer Programming.* dans: Cohen et al.: *Logic, Methodology and Philosophy of Science VI* North-Holland, Amsterdam, 153-175.