Incomplétude mathématique des formalismes

Marc Bagnol, Roland Casalis
Sous la direction de Giuseppe Longo

21 octobre 2008

Introduction

Les mathématiques du XIX$^{\text{è}}$ siècle ont été marquées par d'immenses progrès en analyse et en géométrie. Mais ces progrès vinrent avec l'apparitions d'objets étranges et... quelques démonstrations fausses. C'est en essayant de clarifier la situation qu'est née l'idée de formalisation.

L'objectif était d'arriver à voir les mathématiques comme un jeu de symboles avec des règles claires, permettant de vérifier les démonstrations, excluant l'à-peu-près. On empêchait du coup la démonstration d'absurdités. Par ailleurs, il fallait s'assurer que l'on "ne perdait rien", que le système permettait quand même de décider si une proposition était vraie ou fausse.

Ceci se résume en trois propriétés, que ce qu'on a appelé le "programme de Hilbert" se proposait de démontrer dans le cas de l'arithmétique formalisée :

- cohérence : impossibilité de démontrer une proposition et son contraire.
- effectivité : possibilité de vérifier qu'une démonstration est valide.
- complétude : si un énoncé n'est pas démontrable, alors son contraire l'est.

Le même Hilbert ayant montré que toute la géométrie pouvait se coder en arithmétique, les problèmes de fondation des mathématiques se trouvèrent ramenés à la ceux de l'axiomatique de Peano.

Gödel, avec ses désormais célèbres *théorèmes d'incomplétude*, apportera finalement une conclusion radicale (et négative) au programme de Hilbert en démontrant qu'aucun système formel dont l'expressivité suffit à "faire de l'arithmétique" ne peut vérifier simultanément ces trois propriétés.

Dans sa démonstration, Gödel construit, par un procédé de diagonalisation, une formule G non démontrable et dont le contraire n'est pas non plus démontrable. Seul problème, cette formule n'exprime pas une propriété mathématique intéressante (elle dit essentiellement *"G n'est pas démontrable"*). Il est alors naturel de se demander s'il existe des énoncés ayant un sens mathématique dans un système formel mais indécidables dans celui-ci.

Nous allons nous intéresser à un cas d'incomplétude concrète au travers du **système F** introduit par le mathématicien J.-Y. Girard. Ce système de λ-calcul très expressif joue un rôle important tant en logique (on peut y formaliser la logique et même l'arithmétique du second ordre) qu'en informatique, puisqu'il est à l'origine des langages de programmation fonctionnelle (CamL, Quest...).

Le système F vérifie une propriété algorithmique (le théorème de normalisation) qui assure la terminaison de ses calculs. Nous verrons que cette propriété tout à fait exprimable dans l'arithmétique du second ordre ne peut y être démontrée formellement.

Sommaire

1 Rappels sur les théorèmes de Gödel

1.1 Définitions et notations

D'une manière très générale, un système formel est un ensemble de symboles avec lesquels on écrit les formules, et un ensemble de règles permettant de manipuler ces formules.

Définition : Séquent
On appelle *séquent* une expression de la forme $\Gamma \vdash \Delta$ (lire « Γ thèse Δ ») où Γ et Δ sont des suites de formules.

Définition : Règles
Une règle de dérivation $\mathfrak{R}$ est la donnée de deux ensembles de séquents, $\mathscr{P}$ (pour "prémisses") et $\mathscr{C}$ (pour "conclusions"). On la note généralement sous la forme suivante :

$$\frac{\mathscr{P}}{\mathscr{C}}\ (\mathfrak{R})$$

On dira alors que le séquent $\Gamma \vdash \Delta$ est *dérivable* si on peut l'obtenir comme conclusion d'une suite de règles, chacune utilisant comme prémisse les conclusions des règles précédentes. Les règles de départ sont donc celles qui nécessitent aucune prémisse, c'est à dire les axiomes.

Plus intuitivement : on cherche à formaliser ce qu'est une démonstration mathématique. Pour cela, on considère que ce que fait le mathématicien qui écrit sa démonstration consiste à écrire des hypothèses (Γ), appliquer des règles de déduction ("Si j'ai $\mathscr{P}$, alors par $\mathfrak{R}$, j'ai $\mathscr{C}$") pour arriver à une conclusion (Δ).

Il faut donc lire « $\Gamma \vdash \Delta$ » comme « à partir des hypothèses Γ, on peut déduire les énoncés Δ ».

Donnons un exemple de démonstration formelle, les règles utilisées sont celles du système LJ que nous détaillerons dans la section suivante :

$$\dfrac{\dfrac{}{A \vdash A}\ (\text{axiome}) \qquad \dfrac{}{B \vdash B}\ (\text{axiome})}{A, A \Rightarrow B \vdash B}\ (\Rightarrow \vdash)$$

En formalisant ainsi, on a délimité avec précision ce qu'on avait le droit de faire, ce qui va permettre de raisonner *sur les démonstrations* elles-mêmes... On peut alors énoncer clairement les trois propriétés que l'on cherche à démontrer (cohérence, effectivité, complétude) ce qui constitue évidemment un premier pas vers leur démonstration éventuelle.

On rappelle enfin deux notions de calculabilité dont nous aurons besoin :

- Une fonction (qui n'est pas nécessairement totale) est dite *récursive* ou *calculable* s'il existe une machine de Turing qui la calcule.
- Un ensemble est dit *récursif* s'il existe une machine de Turing qui permet de décider pour tout objet, si cet objet est élément ou non de l'ensemble.

1.2 L'arithmétique de Péano et les théorèmes de Gödel

Le deuxième problème de la liste de Hilbert était la démonstration de la cohérence des axiomes de l'arithmétique. Cette démonstration devait bien sûr elle-même reposer sur des bases solides afin d'éviter tout cercle vicieux. Pour cela, Hilbert préconisait de se placer dans le seul cadre qui soit hors de tout soupçon : celui du fini. Dans le cadre qu'on a fixé, on se limitera donc à des séquents finis, des règles avec un nombre fini de prémisses et de conclusions, et à des dérivations finies (utilisant un nombre fini de règles). Enfin, l'ensemble des axiomes devra être récursif.

Dans ces conditions, il devient alors possible de faire vérifier la validité des preuves par une machine de Turing.

En plus des règles purement logiques, on se donne la liste des axiomes de l'arithmétique de Péano ($\mathscr{PA}$) qui définissent l'opération successeur, l'addition, le produit, et énoncent la validité du principe de récurrence.

Gödel remarque alors la chose suivante : à toute formule, on peut associer injectivement un entier via une fonction calculable[1], nous noterons $F \mapsto \underline{F}$ l'application qui à une formule F associe son code. Puisque les démonstrations ne sont que des suites finies de formules, il est également possible d'associer injectivement, à toute preuve d'une formule, un entier : le code de la démonstration.

Or les formules en question appartiennent au langage de l'arithmétique donc *parlent* des entiers. La diagonalisation n'est plus très loin.

On peut définir une fonction récursive $(m,n) \mapsto Sub(m,n)$ et une formule $Dem(k,n)$ décidable telles que :

$\mathscr{PA} \vdash Sub(m,\underline{F}) = \underline{F[m/x]}$ pour toute formule F.

$\mathscr{PA} \vdash Dem(k,\underline{F})$ si et seulement si k est le code d'une démonstration de F.

On rappelle que l'expression $F[m/x]$ est la formule F où l'on a remplacer toutes les occurrences libres de x par m. On en déduit une formule $Theor(n) := \exists k\, Dem(k,n)$ qui exprime la démontrabilité de la formule de code n dans $\mathscr{PA}$. Autrement dit, pour toute formule F on a :

$\mathscr{PA} \vdash Theor(\underline{F})$ si et seulement si $\mathscr{PA} \vdash F$.

Il ne reste plus qu'à poser $G(x) := \neg Theor(Sub(x,x))$ et à l'appliquer à $m := \underline{G(x)}$ pour faire "bugger" le système !

Informellement $G(m)$ dit « on ne peut pas démontrer $G(m)$ dans $\mathscr{PA}$ ». Ce qui n'est rien d'autre qu'une version syntaxique du paradoxe du menteur[2] dans lequel on aurait remplacé la "vérité" par la "démontrabilité" !

Le paradoxe se traduit ici par la possibilité d'avoir formellement $\mathscr{PA} \vdash G(m) \Leftrightarrow \neg Theor(G(m))$
On peut alors démontrer :

Théorème : Premier théorème d'incomplétude de Gödel

Si l'arithmétique de Péano $\mathscr{PA}$ est cohérente, alors il existe une formule $G_0 (= G(m))$ telle que $\mathscr{PA} \nvdash G_0$ et $\mathscr{PA} \nvdash \neg G_0$.

Preuve :

Par l'absurde :

• Si $\mathscr{PA} \vdash G(m)$ alors $\mathscr{PA} \vdash \neg Theor(G(m))$. Mais on a également $\mathscr{PA} \vdash Theor(G(m))$ puisque $\mathscr{PA} \vdash G(m)$. Donc $\mathscr{PA} \nvdash G(m)$.

• Mais si $\mathscr{PA} \vdash \neg G(m)$ alors $\mathscr{PA} \vdash \neg\neg Theor(G(m))$. Or, en logique classique $\neg\neg A \Leftrightarrow A$, ce qui donne : $\mathscr{PA} \vdash Theor(G(m))$ puis $\mathscr{PA} \vdash G(m)$. Donc $\mathscr{PA} \nvdash \neg G(m)$. □

On a fait tout cela sans parler de *vérité*, d'un point de vue purement formel. Si on prend une définition "naïve"de la vérité (on suppose connaître les valeurs de vérité des formules atomiques, puis on en déduit celle des formules plus complexes) alors $G(m)$ doit être vraie. En effet, si $\mathscr{PA}$ est cohérente, $G(m)$ est indémontrable dans $\mathscr{PA}$ et c'est justement ce qu'elle exprime.

[1] Il invente au passage le *codage*, idée qui depuis a fait son chemin en informatique...

[2] « cette phrase est fausse »

Ce raisonnement relativement imprécis (et en tout cas informel) peut en fait se formaliser dans $\mathscr{P}\mathscr{A}$, c'est *le second théorème d'incomplétude*.

La cohérence de $\mathscr{P}\mathscr{A}$ s'écrit de la façon suivante : $Coher_{\mathscr{P}\mathscr{A}} := \neg Theor(\bot)$, où $\bot$ est l' "absurde" par exemple "$0 = 1$".

Théorème : Second théorème d'incomplétude de Gödel

$$\mathscr{P}\mathscr{A} \vdash Coher_{\mathscr{P}\mathscr{A}} \Leftrightarrow G(m)$$

Avec un corollaire immédiat et problématique pour la "question des fondements" :

Corollaire :

Si $\mathscr{P}\mathscr{A}$ est cohérente, $Coher_{\mathscr{P}\mathscr{A}}$ n'est pas prouvable dans $\mathscr{P}\mathscr{A}$

Remarque :

Attention, il faut bien voir que ce n'est pas réellement $\mathscr{P}\mathscr{A} \nvdash Coher_{\mathscr{P}\mathscr{A}}$ qui pose problème (après tout si $\mathscr{P}\mathscr{A}$ est incohérente, $\mathscr{P}\mathscr{A} \vdash Coher_{\mathscr{P}\mathscr{A}}$) mais le fait que cela soit le cas pour tout sous-système, qui démontre moins de choses, car plus faible. En particulier cela interdit toute preuve de $Coher_{\mathscr{P}\mathscr{A}}$ à partir de principes plus simples dont la cohérence serait supposée ou démontrée : les mathématiques ne peuvent pas se justifier elles-mêmes.

En fait, on voit que ce raisonnement va s'appliquer à tous les systèmes suffisamment expressifs, capables – via un codage – de parler de leurs propres formules et de leurs démonstrations. Tout système contenant les axiomes de Péano pour l'addition et la multiplication est donc incomplet et ne prouve pas sa cohérence.

L'arithmétique n'est donc pas seulement incomplète, elle est *incomplétable*.

Évoquons enfin l'aspect algorithmique du théorème de Gödel : supposons un instant qu'il soit faux et partant que $\mathscr{P}\mathscr{A}$ soit cohérente, décidable ET complète. Dans ce cas, voici un algorithme qui permet de décider en temps fini si c'est A ou $\neg A$ qui est démontrable dans $\mathscr{P}\mathscr{A}$:

« $n := 0$; Vérifier si n est le code d'une démonstration de A ou de $\neg A$. Si oui s'arrêter, sinon recommencer avec $n := n + 1$ »

Vu sous cet angle, le mathématicien peut commencer à douter de la complétude, qui réduirait en quelque sorte son activité à celle d'une machine. Un des premiers à voir dans cette mécanisation une improbable trivialisation des mathématiques fut Weyl, un des élèves de Hilbert. On peut également citer Poincaré qui ne fut pas particulièrement avare de critiques à l'encontre du programme de Hilbert[3].

[3] « M. Hilbert pense que les mathématiques sont comme la machine à saucisses de Chicago : on y introduit des axiomes et des porcs, et en sortent des saucisses et des théorèmes »

2 Logique intuitionniste et arithmétique du second ordre

2.1 Le raisonnement déductif et le Hauptsatz

Le raisonnement le plus naturel, et avec lequel on fait des mathématiques repose autour de la règle que nos ancêtres appelaient le *modus ponens*. Avec des mots, elle dirait : *si A et $A \Rightarrow B$ sont vrais, alors il en est de même pour B*. Ce qui donne dans le calcul des séquents la règle d'élimination de l'implication, ou la coupure :

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash A \Rightarrow B}{\Gamma \vdash B} \text{ (modus ponens)}$$

Il est naturel de demander à un système logique une telle règle (ou bien une équivalente). Une autre règle tout à fait intuitive est la règle de déduction : avec des mots elle dirait : *si B est une conséquence de A, c'est que $A \Rightarrow B$ est vrai*. Cela donne pour le calcul des séquents la règle d'introduction de l'implication :

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \text{ (déduction)}$$

Ce sont ces deux règles qui traduisent dans le langage du calcul des séquents le sens du mot français *implique*. Rajoutons encore l'axiome *si on a A alors on a A* ce qui donne pour le calcul des séquents :

$$\frac{}{A \vdash A} \text{ (axiome)}$$

Sans ce dernier, nous n'avons pas de point de départ. Nous obtenons alors un système logique appelé système minimal intuitionniste. Dans cette logique il n'y a qu'un connecteur, l'implication, ce qui la rend beaucoup trop faible pour y faire des mathématiques. Modulo la correspondance de Curry-Howard, il s'agit du λ-calcul simplement typé, dont le théorème de normalisation forte permet d'affirmer que notre première règle, le modus ponens, est une règle redondante !

Cela peut sembler paradoxal, nous venons juste de dire que le raisonnement naturel tournait autour du modus ponens, or les formules que l'on peut démontrer en utilisant cette règle sont également prouvables sans l'utiliser. C'est une propriété, nommée Hauptsatz (théorème principal en allemand), que vérifient aussi les logiques classique et intuitionniste.

Théorème : HAUPTSATZ

La règle de coupure est redondante : tout séquent démontrable l'est également sans utiliser de coupures.

Nous démontrerons ce résultat pour la logique intuitionniste du second ordre qui correspond au système F modulo l'isomorphisme de Curry-Howard. Dans ce système, nous démontrerons le théorème de normalisation et celui-ci implique le théorème de l'élimination des coupures.

2.2 L'intuitionnisme

L'intuitionnisme est une position philosophique concernant les mathématiques s'opposant à l'approche classique que nous connaissons. Elle a été fondée par le logicien Brouwer qui cherchait à se passer de certaines règles du raisonnement classique, jugées non intuitives. En particulier, Brouwer trouvait douteux que l'on puisse démontrer une existence sans avoir réellement un moyen de construire l'élément dont on a prouvé l'existence. De plus, il refusait – pour des domaines infinis – le principe du tiers exclu affirmant que pour toute proposition A, on a $A \vee \neg A$, et qui permet de faire des raisonnements par l'absurde. Illustrons cette position avec l'exemple typique de raisonnement qui ne le satisfaisait pas :

$$\exists a, b \in \mathbb{R}\backslash\mathbb{Q}, a^b \in \mathbb{Q}$$

$\sqrt{2}$ étant irrationnel, on a :

– soit $\sqrt{2}^{\sqrt{2}}$ est rationnel, auquel cas c'est fini.
– soit $\sqrt{2}^{\sqrt{2}}$ est irrationnel, auquel cas on prend $a = \sqrt{2}^{\sqrt{2}}$, $b = \sqrt{2}$ et :
$a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\times\sqrt{2}} = 2$ et c'est bon.

On a donc démontré une existence, mais finalement on ne sait pas quels sont les a et b qui conviennent. Pire, on ne sait même pas si on peut les trouver. On a également montré une disjonction (c'est exactement $A \vee \neg A$) mais on ne peut savoir si l'une d'elle est démontrable...

La *logique intuitionniste* a été formalisée par les élèves de Brouwer. Nous allons l'expliciter à travers le calcul des séquents LJ. Les symboles utilisés sont l'implication $\Rightarrow$, l'absurde $\bot$, la conjonction $\wedge$, la disjonction $\vee$, ainsi que les quantificateurs universel $\forall$ et existentiel $\exists$. La négation y est définie par $\neg A := A \Rightarrow \bot$.

Un séquent intuitionniste est une expression de la forme $\Gamma \vdash A$ où Γ est un multi-ensemble fini d'énoncés (donc en particulier l'ordre n'a pas d'importance) et A un énoncé, $\Gamma \vdash A$ signifie intuitivement que partant des hypothèses Γ, on peut démontrer A. Les règles de dérivation sont les suivantes :

Règles structurelles

$$\frac{}{A \vdash A}\ (\text{axiome}) \qquad \frac{\Gamma \vdash A \qquad \Lambda, A \vdash B}{\Gamma, \Lambda \vdash B}\ (\text{coupure})$$

$$\frac{\Gamma \vdash A}{\Gamma, \Lambda \vdash A}\ (\text{affaiblissement}) \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ (\text{contraction})$$

Règles logiques

Règles gauches

$$\frac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C}\ (l\wedge\vdash) \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \wedge B \vdash C}\ (r\wedge\vdash)$$

$$\frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}\ (\vee\vdash)$$

$$\frac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \Rightarrow B \vdash C}\ (\Rightarrow\vdash) \qquad \frac{}{\bot \vdash A}\ (\bot\vdash)$$

Règles droites

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\vdash\wedge)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ (l\vdash\vee) \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ (r\vdash\vee)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ (\vdash\Rightarrow)$$

L'implication classique, qui était définie par $A \Rightarrow B := \neg A \vee B$, prend un autre sens : démontrer l'implication intuitionniste $A \Rightarrow B$ revient à donner un algorithme qui transforme une preuve de A en une preuve de B. Il manque peu de chose pour obtenir la logique classique : il suffirait de rajouter la règle $\frac{}{\neg\neg A \vdash A}$ ou une équivalente.

Comme les mathématiques se font avec des variables, il peut être intéressant de connaître les règles de dérivation des quantificateurs du premier ordre :

Quantificateurs du premier ordre

Règles gauches	Règles droites
$\dfrac{\Gamma, A[t/x] \vdash B}{\Gamma, \forall x A \vdash B}\ (\forall \vdash)$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash \forall x A}\ (\vdash \forall)$
$\dfrac{\Gamma, A \vdash B}{\Gamma, \exists x A \vdash B}\ (\exists \vdash)$	$\dfrac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x A}\ (\vdash \exists)$

Pour utiliser les règles $(\vdash \forall)$ et $(\exists \vdash)$, il faut que la variable x ne soit pas libre dans le contexte Γ. Quant aux règles $(\forall \vdash)$ et $(\vdash \exists)$, $A[t/x]$ signifie que l'on a remplacé les occurrences libres de la variable x dans A par le terme t, et que l'on peut utiliser cette règle seulement si les variables liées de A n'apparaissent pas dans t.

Enfin remarquons que les deux calculs suivants permettent de plonger la logique classique dans la logique intuitionniste, autrement dit, on n'a rien perdu en rejetant le principe du tiers exclu. On y a même gagné : les formules prennent un sens plus fin (voir la section sur le calcul sans coupure).

$$\dfrac{\dfrac{\dfrac{\vdots}{A \vdash B} \qquad \dfrac{}{\bot \vdash \bot}\ (\bot \vdash)}{A, \neg B \vdash \bot}\ (\Rightarrow \vdash)}{\neg B \vdash \neg A}\ (\vdash \Rightarrow)$$

Ce qui permet, à partir de $A \vdash B$ de déduire $\neg B \vdash \neg A$. On n'a pas de réciproque car elle est équivalente au principe du tiers exclus.

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\ (\text{axiome}) \qquad \dfrac{}{\bot \vdash \bot}\ (\bot \vdash)}{A, \neg A \vdash \bot}\ (\Rightarrow \vdash)}{A \vdash \neg\neg A}\ (\vdash \Rightarrow)$$

Ainsi, la première règle appliquée à $A \vdash \neg\neg A$ donne $\neg\neg\neg A \vdash \neg A$. C'était la règle manquante pour retrouver la logique classique, ainsi :

Théorème : TRADUCTION DE GÖDEL

A est démontrable classiquement ssi A^g est démontrable au sens intuitioniste, où A^g est obtenu à partir de A en insérant des doubles négations devant chaque sous formule atomique et chaque connecteur.

2.3 Calcul sans coupure

D'après le Hauptsatz, la règle de coupure est redondante. Observons les règles logiques autres que la coupure pour en obtenir certaines propriétés. Que peut-on dire sur les prémisses d'un séquent $\Gamma \vdash A$? Tant qu'on utilise la coupure, on ne peut rien dire car il y a des énoncés qui figurent dans les prémisses mais qui n'apparaissent pas dans la conclusion. En revanche, sans la coupure, les prémisses sont des sous-formules de la conclusion où la relation "être une sous-formule" est la clôture réflexive et transitive de la relation définie comme suit :

$$\begin{aligned} A &\preccurlyeq \neg A \\ A &\preccurlyeq A \wedge B, A \vee B, A \Rightarrow B, B \Rightarrow A \\ A[t/x] &\preccurlyeq \forall x A, \exists x A \end{aligned}$$

Ainsi comme $\bot$ n'a pas de sous-formule, il n'est pas possible de montrer l'absurde. On ne peut donc pas tout démontrer et la logique est cohérente ! **Le Hauptsatz implique la cohérence du système.**

Remarquons également que si l'on souhaite démontrer le séquent $\vdash A$, la dernière règle utilisée est nécessairement une introduction (ou règle droite). On voit encore que l'on ne peut pas démontrer l'absurde car il n'y a pas d'introduction de l'absurde. De plus si on souhaite démontrer un énoncé existentiel, la dernière règle utilisée est $(\vdash \exists)$. On voit alors que la prémisse de cette dernière règle permet de retrouver un élément dont on a prouvé l'existence. Ainsi, sans coupure, on démontre qu'il existe x tel que $A(x)$ ssi il existe x pour lequel on démontre $A(x)$. Mieux, grâce au théorème de normalisation qui affirme que tout algorithme pour éliminer les coupures termine, on sait qu'on peut obtenir cet élément. Malheureusement, la complexité de ces algorithmes est monstrueuse, et il est en pratique impossible de les appliquer. Tout ce qui a été dit pour un énoncé existentiel marche également pour les énoncés de la forme $A \vee B$, sans coupures, démontrer A ou B, c'est démontrer A ou démontrer B, et grâce au théorème de normalisation, il existe un moyen de retrouver l'énoncé que l'on a démontré.

Remarquons quand même qu'en présence d'axiomes, on perd le Hauptsatz : si A et si $A \Rightarrow B$ sont des axiomes, il est impossible de démontrer B sans la règle de coupure.

2.4 Le second ordre

Il est possible de rajouter au système les quantificateurs du second ordre. Ceux-ci permettent une quantification sur les relations. Par exemple, on pourra écrire l'axiome d'induction de l'arithmétique avec le seul axiome :

$$\forall_2 X, (X(0) \wedge \forall_1 n, X(n) \Rightarrow X(n+1)) \Rightarrow \forall_1 n, X(n)$$

Ici, on a ajouté les indices 1 et 2 pour indiquer l'ordre du quantificateur. Il y a un intérêt à introduire ces nouveaux quantificateurs : on augmente considérablement l'expressivité de la logique. Par exemple, il découle du théorème de complétude de Gödel qu'il n'est pas possible d'axiomatiser les ordres bien fondés, même avec un nombre infini d'axiomes. Cela provient du fait que la formule caractérisant les ensembles bien ordonnés

$$\forall X \neq \emptyset, \exists x \in X, \forall y \in X, x \leq y$$

n'a pas d'équivalent au premier ordre. De même, il est possible d'axiomatiser au second ordre les ensembles dénombrables. Ce qui est impossible au premier ordre d'après le théorème de Löwenheim-Skolem. Ainsi, on gagne en expressivité, mais on perd les théorèmes de complétude et de Löwenheim-Skolem. La syntaxe de la logique est quasiment identique, pour chaque $n \in \mathbb{N}$, on se donne en plus un ensemble infini $\{X_1^n, X_2^n, \ldots\}$ de variables de relations n-aires. A partir d'une relation n-aire X^n on obtient un prédicat en considérant la formule $P_{X^n}(x_1, \ldots, x_n) = X^n(x_1, \ldots, x_n)$. Inversement on a besoin d'un axiome, la compréhension :

Pour tout prédicat P à n variables libres : $\exists_2 X^n, \forall x_1, \ldots, x_n (X(x_1, \ldots, x_n) \Leftrightarrow P(x_1, \ldots, x_n))$

Autrement dit, si $n = 1$, on donne un sens à une expression de la forme $\{x, P(x)\}$. Donnons les règles de dérivation liées à ces nouveaux quantificateurs en terme de séquents LJ. Comme pour les quantificateurs du premier ordre, il y a une condition de liberté des variables pour utiliser ces règles, elles sont évidentes en pratique.

Quantificateurs du second ordre

Règles gauches	Règles droites
$\dfrac{\Gamma, A[T/X] \vdash B}{\Gamma, \forall_2 X.A \vdash B}\ (\forall_2 \vdash)$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash \forall_2 X.A}\ (\vdash \forall_2)$
$\dfrac{\Gamma, A \vdash B}{\Gamma, \exists_2 X.A \vdash B}\ (\exists_2 \vdash)$	$\dfrac{\Gamma \vdash A[T/X]}{\Gamma \vdash \exists_2 X.A}\ (\vdash \exists_2)$

Enfin, remarquons que le passage au second ordre retire la propriété de la sous-formule. En effet, si on a une quantification universelle sur les relations, par exemple $A = \forall_2 X^1.B$, X^1 représente n'importe quelle partie du domaine, par exemple $\{x,A\}$. On tourne en rond, comment démontrer A pour tout X^1, même pour $X^1 = \{x,A\}$? On dit alors que le système est *imprédicatif*. Toute la difficulté de la preuve du théorème de normalisation vient de là : on n'a plus d'ordre bien fondé sur les formules.

2.5 Arithmétique

L'arithmétique est un système important pour plusieurs raisons. Originellement, Hilbert avait choisi ce système pour réaliser son programme, car il estimait que les axiomes de l'arithmétique ne pouvaient pas être mis en doute et car il avait réussi à y plonger la géométrie. De même, c'est le système du *codage* et donc de l'informatique. C'est grâce à ce codage que le pouvoir expressif de l'arithmétique est grand, et à cause de celui-ci que se manifestent les théorèmes d'incomplétude. Notons que si la théorie des ensembles est beaucoup plus expressive, cela vient de l'axiome de l'infini, ainsi il n'y a pas seulement une infinité d'éléments, il y a aussi un ensemble infini.

Voici les axiomes de l'arithmétique proposés par Péano (S désigne l'application successeur) :

$\forall x \neg(Sx = 0)$
$\forall x \exists y \neg(x = 0) \Rightarrow (Sy = x)$
$\forall x \forall y (Sx = Sy) \Rightarrow (x = y)$
$\forall x (x + 0 = x)$
$\forall x \forall y (x + Sy = S(x + y))$
$\forall x (x.0 = 0)$
$\forall x \forall y (x.Sy = (x.y) + x)$

On y ajoute le schéma d'induction : pour chaque formule P à $n+1$ variables libres :

$$\forall x_1 \ldots \forall x_n (P(0, x_1, \ldots, x_n) \Rightarrow (\forall x (P(x, x_1, \ldots, x_n) \Rightarrow P(Sx, x_1, \ldots, x_n))) \Rightarrow \forall x P(x, x_1, \ldots, x_n))$$

On obtient alors les systèmes $\mathscr{P}\mathscr{A}$ (ou $\mathscr{P}\mathscr{A}_1$ si l'on veut indiquer l'ordre) et $\mathscr{H}\mathscr{A}_1$ (pour Arithmétique de Heyting) selon que l'on se place en logique classique ou en logique intuitioniste du premier ordre. On définit également les systèmes $\mathscr{P}\mathscr{A}_2$ et $\mathscr{H}\mathscr{A}_2$ comme les systèmes formés des mêmes axiomes mais pour la logique classique ou intuitioniste du second ordre, auxquels on ajoute le schéma de compréhension.

3 Le système F

Le système F est un λ-calcul particulièrement expressif qui, comme tous les λ-calculs, présente des interactions fortes avec la logique (*cf.* 3.2). Pour cela, il a été introduit par le logicien J.-Y. Girard puis développé en informatique par J. C. Reynolds.

L'idée de départ du λ-calcul, inventé par Church en 1930, était double : formaliser la notion de fonction mathématique et proposer une nouvelle approche des fondements cherchant à esquiver la théorie des ensembles au profit du paradigme "tout est fonction". Cette tentative a échoué dans le sens où les problèmes de cohérence y sont les mêmes (on échappe pas si facilement aux théorèmes de Gödel) et le pouvoir expressif s'est révélé moins important. En revanche le λ-calcul, en plus d'avoir ouvert des horizons en logique, est en un sens le premier langage de programmation, et est donc une des idées fondatrices de l'informatique.

3.1 Syntaxe et conventions

Le système fonctionne sur deux niveaux : on a les **termes**, qui sont les objets que le système manipule. Comme on l'a dit, en λ-calcul "tout est fonction", les termes jouent donc un double rôle d'argument et de fonction un peu déroutant de prime abord. Les **types**, se situent au niveau au dessus et correspondent intuitivement à des ensembles de termes.

Définition : Types

Avant de définir les termes, définissons par induction les *types* du système F :

- les variables de type (notées X, X_i, Y, Y_i,...) sont des types.
- si U et V sont des types, $U \to V$ est un type.
- si T est un type et X une variable de type, $\forall X.T$ est un type.

Définition : Termes

Les termes du système F sont définis par induction de la façon suivante :

- pour tout type T, les variables (notées x^T, x_i^T, y^T, y_i^T,...) sont des termes de type T.
- si t est un terme de type U et x^V une variable de type V, alors $\lambda x^V.t$ est un terme de type $U \to V$. On appelle cette opération l' "abstraction".
  Intuitivement : à partir de la formule "$g(x)$" on forme la fonction "$x \mapsto g(x)$".
- si t et u sont des termes de types respectifs $U \to V$ et U , alors tu est un terme de type V. On appelle cette opération "application".
  À partir des fonctions "f" et "g", on forme la fonction "$f(g)$".
- si t est un terme de type T, alors $\Lambda X.t$ est un terme de type $\forall X.T$. On appelle cette opération l'abstraction universelle.
  Si "f_X" est une fonction dont le type dépend de "X", on forme "$X \to f_X$".
- si t est un terme de type $\forall X.T$, et si U un type, alors tU de type $T^{[U/X]}$ (où $T^{[U/X]}$ est le type où l'on a remplacer toutes les occurences libres de X par U. On appelle cette opération l'application universelle.
  Dans le cas précédent, cela revient à appliquer "$X \to f_X$" à "U".

Il existe d'autres formes de λ-calcul. Par exemple le λ-calcul non typé. Comme son nom l'indique les termes n'ont pas de types, ils sont construit avec les trois premières règles (variables, application et abstraction). De même, on peut citer le λ-calcul simplement typé. Là encore la construction est analogue à celle du système F. Les types sont définis par induction avec les deux premières règles (pas de quantification) et les termes avec les trois premières.

Selon qu'elles sont abstraites par un λ ou non, les variables peuvent être libres ou liées. On ne définira pas en détail cette notion, qui correspond exactement à la notion de variables libres et liées par "$\forall x$: ... " en logique. Donnons tout de même un exemple :

Dans $\Lambda Y.\lambda x^X.y^{X\to Y}x^X$, Y *et* x^X *sont liées, tandis que* $y^{X\to Y}$ *et* X *sont libres*

Dès lors qu'une variable est liée, on pourra omettre ses indications de typage pour alléger les notations.

Les variables liées par un λ peuvent être renommées sans changer le comportement du terme, à condition de ne pas modifier les indications de typage sur les variables de termes. On définit ainsi (modulo des difficultés techniques) une relation : la α-équivalence ou α-renommage. On raisonnera toujours par la suite en notant "$t = v$" pour t et v α-équivalents.

Exemple : $\Lambda Y.\lambda x^X.y^{X\to Y}x$ *et* $\Lambda Z.\lambda z^X.y^{X\to Z}z$ *sont* α*-équivalents.*

On a également une notion de substitution de terme ou de type à une variable (libre) qui correspond à la notion de substitution en logique. La principale difficulté étant donc d'éviter la "capture de variable", ce que l'ont fait grâce à l'α-renommage. On notera : "$t\,[u/x]$" le terme dans lequel u a été substitué à x dans t et $T^{[U/X]}$ la substitution de types de U par X dans T.

Exemple : si $t = \lambda x^X.y^{X\to X}x$ *et* $u = \lambda z^X.z$*, alors* $t\left[u/y\right] = \lambda x^X.(\lambda z^X.z)x$.

On peut alors définir la notion de β-réduction, qui va être la "règle de calcul" du système. On part pour cela des relations :

$$(\lambda x.t)u \to t\,[u/x] \text{ et } (\Lambda X.t)U \to t^{[U/X]}$$

qu'on étend en posant $t \to_1 t'$ si une des relations "$\to$" lie des sous-termes de t et t'. On dit alors que t' est un réduit immédiat de t (on obtient t' en effectuant une et une seule β-réduction de t). On notera parfois "$=_\beta$" la clôture réflexive-symétrique-transitive de cette relation, et "$\rightsquigarrow$" ou "$\to_\beta$"sa clôture réflexive-transitive.

Définition : Normalisation

- On dit qu'un terme est *normal* si on ne peut pas lui appliquer de β-réduction. Autrement dit c'est un élément minimal pour $\rightsquigarrow$.
- On dit qu'un terme t est *faiblement normalisant* s'il existe un terme normal t' tel que $t \rightsquigarrow t'$. Autrement dit, il existe une suite finie $t \to_1 t_1 \to_1 \ldots \to_1 t'$. On dit alors que t' est une *forme réduite normale* de t.
- On dit qu'un terme t est *fortement normalisant* si toute suite $t \to_1 t_1 \to_1 \ldots$ est finie. En particulier t est faiblement normalisant. On notera $\mathscr{SN}_T$ l'ensemble des termes fortement normalisant de type T.

Illustrons ces définitions avec trois exemples venant du λ-calcul non typé :

- $\Omega = (\lambda x.xx)(\lambda x.xx)$ n'est pas faiblement normalisant car tout réduit de ce terme est égal Ω.
- $(\lambda x.y)\Omega$ est faiblement normalisant si y est une variable, mais pas fortement normalisant.
- $(\lambda x.xx)y$ est fortement normalisant si y est une variable.

Une fois posées ces définitions, on peut démontrer (mais nous ne le ferons pas ici bien que les démonstrations ne soient pas dures) les propriétés suivantes, qui assurent que le système ne se soucie pas de l'ordre dans lequel on fait les calculs et que la β-réduction préserve le typage :

Théorème : Confluence forte (souvent appelée "propriété de Church-Rosser")

Si t et t' sont des termes tels que $t =_\beta t'$, alors il existe u tel que $t \rightsquigarrow u$ et $t' \rightsquigarrow u$. En particulier, si t et t' sont normaux alors $t = t'$; ainsi, si un terme est faiblement normalisant il y a unicité de la forme réduite normale.

Théorème : Autoréduction

Si t est de type T et $t =_\beta t'$, alors t' est de type T.

Dans la présentation que l'on a choisie, le type se construit en même temps que le terme. Il témoigne en quelque sorte de sa "structure générale", de la forme des ses arguments et des résultats du calcul.

On peut voir le typage des termes sous un angle un peu différent (mais équivalent) en traduisant les constructions des termes dans des règles de déduction. On fait alors comme si on dérivait « le terme t est de type T dans le contexte Γ » à l'aide de ces règles.

Définition : Terme typable dans un contexte

Un contexte est un ensemble fini de déclarations de typage de la forme " $x : U$ ", où x est une variable et U est un type, dans lequel chaque variable du langage apparait au plus une fois.
On dit qu'un terme t est typable de T dans le contexte Γ, et on note " $\Gamma \vdash t : T$ " si l'on peut le dériver à l'aide des règles suivantes :

$$\frac{}{\Gamma, x^T : T \vdash x^T : T}\ \text{(Axiome)}$$

$$\frac{\Gamma, x^U : U \vdash t : V}{\Gamma \vdash \lambda x^U.t : U \to V}\ (\to i) \qquad \frac{\Gamma \vdash t : U \to V \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : V}\ (\to e)$$

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash \Lambda X.t : \forall X.T}\ (\forall i) \qquad \frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash tU : T^{[U/X]}}\ (\forall e)$$

3.2 La correspondance de Curry-Howard

Le typage a été initialement introduit pour assurer la terminaison de la β-réduction, en rendant impossible la construction de fonctions dont le calcul pourrait ne pas terminer comme $(\lambda x.xx)(\lambda x.xx)$. Cependant, regardé sous un éclairage différent, il fait apparaître une parenté forte entre λ-calcul et logique (intuitionniste, en particulier).

Voyons sur un exemple ce qui se passe. Le terme $\Lambda X.\lambda x^X.x$ admet pour dérivation de typage :

$$\cfrac{\cfrac{\cfrac{}{x : X \vdash x : X}\ \text{(Axiome)}}{\vdash \lambda x^X.x : X \to X}\ (\to i)}{\vdash \Lambda X.\lambda x^X.x : \forall X.X \to X}\ (\forall i)$$

Si on ne regarde que les types, la dérivation précédente ressemble à s'y méprendre à la démonstration de " $\forall X\ X \Rightarrow X$ " !

C'est exactement ce qui fonde la correspondance de Curry-Howard. On va raisonner selon le paradigme suivant :

- Les types sont des propositions.
- Les termes sont des preuves.

Vu les symboles choisis pour les types, ceux-ci correspondent évidemment à une proposition en logique du second ordre. Mais on voit alors que les règles de typage des termes correspondent à des règles de déduction de la logique, et que la β-réduction peut être vue comme l'élimination des coupures.

Cette correspondance peut sembler incomplète à priori, puisque tous les symboles de la logique ne se retrouvent pas dans le système F ($\vee$, $\wedge$, $\exists$...). En fait, grâce à la quantification du second ordre, il est possible de définir l'absurde, la conjonction, la disjonction et le quantificateur existentiel du second ordre à partir des seuls symboles $\Rightarrow$ et $\forall_2$. Par exemple, on peut définir l'absurde par $\forall X.X$, l'équivalence entre ces formules est immédiate :

$$\frac{}{\bot \vdash \forall_2 X.X}\,(\bot\vdash) \qquad \frac{\dfrac{}{\bot\vdash\bot}\,(\bot\vdash)}{\forall_2 X.X \vdash \bot}\,(\forall_2\vdash) \text{ puisque } \bot = X[\bot/X]$$

On retrouve alors la règle d'élimination de $\bot := \forall X.X$:

$$\text{Pour tout type } A\text{, on a : } \quad \frac{\Gamma \vdash u : \bot}{\Gamma \vdash uA : A}\,(\forall e)$$

De même, on peut définir

$$\begin{array}{rcl} A \wedge B & := & \forall X.(A \to B \to X) \to X \\ A \vee B & := & \forall X.(A \to X) \to (B \to X) \to X \\ \exists X.A & := & \forall Y.(\forall X.A \to Y) \to Y \end{array}$$

et vérifier que l'on a des notions équivalentes.

On a vu que le processus de β-réduction correspond à l'élimination des coupures. En particulier, si le système F *normalise fortement*, c'est à dire, si tout terme est fortement normalisant, la logique du second ordre admet l'élimination des coupures[4] et l'on dispose d'une "méthode" pour produire des preuves sans coupure (n'importe quelle stratégie termine et conduit à l'unique forme normale).

De plus on sait que l'élimination des coupures entraîne la cohérence de la logique du second ordre. Modulo la correspondance de Curry-Howard, il en va donc de même pour la normalisation forte du système F. Ceci indique que l'incomplétude pourrait bien se manifester par la suite, la logique du second ordre étant le cadre logique qui fait essentiellement la différence entre $\mathscr{P}\mathscr{A}_1$ et $\mathscr{P}\mathscr{A}_2$...

[4] C'est même pour ça que le système F a été introduit à l'origine.

4 Le théorème de normalisation

4.1 Génèse de la preuve

Alors que le théorème de Curry-Howard montre une correspondance entre le système F et la logique, nous allons voir qu'on peut également y "faire de l'arithmétique", au moyen d'une représentation des entiers dans le système :

Définition : Entiers de Church
On appelle $n^{\text{ème}}$ *entier de Church*, et on note $\widehat{n}$, le terme $\Lambda X.\lambda f^{X\to X}.\lambda x^X.\underbrace{f\ldots f}_{n \text{ fois}}\ x$ de type $int := \forall X.(X \to X) \to (X \to X)$.

Remarque :
Le terme $\widehat{n}$ peut être interprété comme l'application qui, à une fonction de X dans X, associe sa $n^{\text{ème}}$ itérée.

Définition : Représentation des fonctions
Une fois fixée cette représentation des entiers, on peut définir la représentation de fonctions : On dit que le terme t *représente* la fonction f (de $\mathbb{N}$ dans $\mathbb{N}$) si

$$\forall n \in \mathbb{N}, t\widehat{n} \rightsquigarrow \widehat{f(n)}$$

Donnons quelques exemples :
- La fonction successeur est représentée par $\widehat{S} = \lambda n^{int}.\Lambda X.\lambda y^{X\to X}.\lambda x^X.y(((nX)y)x)$.
- L'addition est représentée par : $\lambda n^{int}.\lambda m^{int}.\Lambda X.\lambda y^{X\to X}.\lambda x^X.((nX)y)((mX)y)x$.
- La multiplication est représentée par : $\lambda n^{int}.\lambda m^{int}.\Lambda X.\lambda y^{X\to X}.\lambda x^X.((nX)((mX)y))x$.
- L'exponentiation $(n, m) \mapsto n^m$ est représentée par : $\lambda n^{int}.\lambda m^{int}.\Lambda X.(m(X \to X))(nX)$.

On a en fait le résultat suivant qui s'obtient en codant les preuves de $\mathscr{P}\mathscr{A}_2$ par des termes :

Théorème :
Une fonction récursive f de $\mathbb{N}$ dans $\mathbb{N}$ dont on peut prouver la terminaison dans $\mathscr{P}\mathscr{A}_2$ (on dit "prouvablement totale"), est représentable en système F.

Ce théorème va nous donner un indice précieux sur la preuve de la normalisation. En effet, tout cela pourrait avoir un rapport avec la cohérence de $\mathscr{P}\mathscr{A}_2$. Gödel et son second théorème commencent à rôder dangereusement dans les parages. Plus précisément :

On se donne un codage bijectif des termes, noté $\overline{t}$, pour l'entier codant t et $\underline{n}$ pour le terme codé par n (attention : ce codange n'a rien à voir avec la *représentation* $\widehat{n}$!) Si on peut prouver dans $\mathscr{P}\mathscr{A}_2$ que F normalise, la fonction $\mathscr{N} : n \to$ (le code de la forme normale de $\underline{n}$) y est prouvablement totale.

On a de plus, les fonctions totales :
- $A : (m, n) \to \overline{tu}$ (où $\underline{m} = t$ et $\underline{n} = u$), qui code l'application de termes.
- $\# : n \to \overline{(\widehat{n})}$, *i.e.* : le code de la représentation de n.
- $\flat : m \to n$ si $\overline{(\widehat{n})} = m$, 0 sinon ($\flat$ est l'inverse partiel de $\#$).

Ainsi, si m est le code d'un terme t, $A(m,\#n)$ est le code de $t\widehat{n}$, et si t représente f, $\mathcal{N}(A(m,\#n))$ est le code de $\widehat{f(n)}$, ie $\mathcal{N}(A(m,\#n)) = \#f(n)$.

On pose alors $D := \flat(\mathcal{N}(A(n,\#n))) + 1$. Cette fonction est toujours prouvablement totale et il lui correspond donc un terme t, de code d.

Vu ce qui précède $\#D(d) = \mathcal{N}(A(d,\#d))$ puis $\flat(\mathcal{N}(A(d,\#d)) = D(d)$, c'est à dire $D(d) = D(d)+1$! On a donc obtenu une contradiction, l'hypothèse faite étant que $\mathscr{P}\mathscr{A}_2$ démontrait la normalisation.

Voilà donc notre indice : inutile de chercher une preuve qui resterait dans $\mathscr{P}\mathscr{A}_2$[5] ! La caractéristique essentielle de $\mathscr{P}\mathscr{A}_2$ étant l'axiome de compréhension :

$$\text{Pour toute formule } A(y),\ \exists X\ \forall y\, A(y) \Leftrightarrow y \in X$$

On peut espérer trouver une solution en utilisant une forme plus générale de cet axiome, en considérant comme des "ensembles" des familles de termes qui n'en sont pas au sens de $\mathscr{P}\mathscr{A}_2$.

Une méthode pour prouver la normalisation forte pour le λ-calcul simplement typé (qui est un système plus faible que F) utilise une "interprétation" des types (notée $[\![T]\!]$ pour tout type T) de la même manière que l'on définirait la valeur de vérité d'une formule dans un modèle, sauf que "les valeurs de vérité" sont ici des ensembles. A priori $[\![T]\!]$ est inclus dans $\mathscr{SN}_T$, l'ensemble des termes fortement normalisants de type T. On montre ensuite[6] que tout terme de type T est dans $[\![T]\!]$, ce qui prouve la normalisation.

Il peut être intéressant de remarquer que $[\![T]\!]$ "ressemble beaucoup" à l'ensemble des termes de type T (on définit $[\![U \to V]\!]$ comme $\{t / \forall u \in [\![U]\!] : tu \in [\![V]\!]\}$). Si ces ensembles sont égaux au bout du compte, leurs définitions diffèrent. Celle de $[\![T]\!]$ est beaucoup plus restrictive et apporte plus d'hypothèses d'induction ; c'est ce qui permet de faire la preuve. Seulement ces propriétés ne sont pas exprimable dans le système nécessaire pour énoncer la propriété de départ : « les termes du λ-calcul simplement typé normalisent fortement ». Nous allons essayer d'adapter cette méthode au système F dans la partie suivante.

4.2 La preuve

Nous allons maintenant donner la preuve du théorème, en commençant par une série de définitions. Une fois ces dernières données, la preuve n'est vraiment pas compliquée, il ne s'agit que de manipulations des définitions. La difficulté de la preuve vient surtout du fait que l'on ne se représente pas facilement les objets introduits. Il s'agit typiquement d'une démonstration où l'ordinateur se débrouille mieux qu'un humain.

4.2.1 Candidats de réductibilité

Définition :

Si t est un terme fortement normalisant, on définit $v(t) \in \mathbb{N}$ comme la longueur de la plus grande suite de β-réduction partant de t.

Cette définition semble tout à fait correcte mathématiquement, mais il y a derrière le lemme de Koenig affirmant que tout arbre infini à branchement fini possède une branche infinie. Si un terme t est fortement normalisant on peut construire son arbre de β-réduction. Toute branche est finie par hypothèse et les branchements sont évidemment finis, par le lemme, on en déduit que l'arbre est fini, donc possède une hauteur et $v(t)$ est bien défini. Le problème, c'est qu'il n'y a aucun moyen de trouver la branche infinie... Et la démonstration du théorème s'appuie fortement sur ce lemme non constructif.

[5] Ce qui signifie sortir de l'arithmétique courante : on quitte rarement $\mathscr{P}\mathscr{A}_2$ quand on manipule des objets finis.

[6] En réalité, comme souvent dans une preuve par induction, on montre beaucoup plus, la difficulté étant d'identifier la *bonne* hypothèse d'induction.

Définition : Simplicité

On dit qu'un terme est simple s'il ne commence pas par une abstraction.

Définition : Candidats de réductibilité

Soit A un type, on appelle *candidat de réductibilité* de type A un ensemble $\mathscr{C}$ de termes de types A satisfaisant les trois conditions suivantes :
CR1 : Tout terme de $\mathscr{C}$ est fortement normalisant.
CR2 : Si $t \in \mathscr{C}$ et $t \rightsquigarrow t'$ alors $t' \in \mathscr{C}$.
CR3 : Si t est simple et si tous ses réduits immédiats sont dans $\mathscr{C}$, alors $t \in \mathscr{C}$.

Pour tout type A, il existe des candidats de réductibilité : en effet, l'ensemble des termes fortement normalisant de type A est un candidat de réductibilité de type A.

(CR3) implique que si t est un terme simple et en forme normale alors $t \in \mathscr{C}$ pour tout $\mathscr{C}$. Ainsi les candidats de réductibilité de type C contiennent les variables de type C.

Soient $\mathscr{C}, \mathscr{D}$ des ensembles de termes de type C et D respectivement. Alors on note $\mathscr{C} \to \mathscr{D}$ l'ensemble de termes de type $C \to D$ tel que :

$$t \in \mathscr{C} \to \mathscr{D} \qquad \Leftrightarrow \qquad \forall u (u \in \mathscr{C} \Rightarrow tu \in \mathscr{D})$$

Le lemme suivant est important pour la démonstration du théorème :

Lemme :

Si $\mathscr{C}$ et $\mathscr{D}$ sont des candidats de réductibilité alors $\mathscr{C} \to \mathscr{D}$ l'est aussi.

Preuve :

CR1 : Soit $t \in \mathscr{C} \to \mathscr{D}$. Par définition, quel que soit $u \in \mathscr{C}$, $tu \in \mathscr{D}$. Soit x une variable de type C, par la remarque ci-dessus, $x \in \mathscr{C}$ ainsi, $tx \in \mathscr{D}$ et par (CR1), tx est fortement normalisant. Or si on a une suite de β-réduction :

$$t \to_1 t_1 \to_1 t_2 \to_1 \cdots \to_1 t_n \to_1 \cdots$$

on obtient une suite

$$tx \to_1 t_1 x \to_1 t_2 x \to_1 \cdots \to_1 t_n x \to_1 \cdots$$

finie car tx est fortement normalisant. Il en est donc de même pour la première et t est bien fortement normalisant.

CR2 : Soit $t \in \mathscr{C} \to \mathscr{D}$ et supposons $t \rightsquigarrow t'$. Soit $u \in \mathscr{C}$, alors $tu \in \mathscr{D}$ et $tu \rightsquigarrow t'u$. Par (CR2), $t'u \in \mathscr{D}$. Ceci étant vrai pour tout $u \in \mathscr{C}$, $t' \in \mathscr{C} \to \mathscr{D}$.

CR3 : Soit t un terme simple de type $C \to D$ dont les réduits immédiats sont dans $\mathscr{C} \to \mathscr{D}$, nous allons montrer que $t \in \mathscr{C} \to \mathscr{D}$, ie $\forall u \in \mathscr{C}, tu \in \mathscr{D}$ par récurrence sur $\nu(u)$ (qui est bien défini d'après (CR1). Comme t est simple, un réduit immédiat de tu est nécessairement de la forme $t'u$ avec $t \to_1 t'$ ou de la forme tu' avec $u \to_1 u'$. Dans le premier cas, $t' \in \mathscr{C} \to \mathscr{D}$ et donc $t'u \in \mathscr{D}$. Dans le second, u' est réductible par (CR2) et comme $\nu(u') < \nu(u)$, $tu' \in \mathscr{D}$. Ainsi tous les réduits immédiats de tu sont dans $\mathscr{D}$, on conclut en remarquant que tu ne commence pas par une abstraction et en appliquant (CR3).

□

4.2.2 La réductibilité paramétrée

Définition : Réductibilité

On définit la *réductibilité* d'un terme par induction sur son type (on note $[\![A]\!]$ l'ensemble des termes réductibles de type A) :
Variable : Si $A = X$ est une variable de type, un terme t de type A est réductible s'il est fortement

normalisant. ie $[\![X]\!] = \mathscr{SN}_X$
Implication : Si $A = C \to D$, un terme t de type A est réductible si pour tout terme u réductible de type C le terme tu est réductible de type D. i.e. $[\![C \to D]\!] = [\![C]\!] \to [\![D]\!]$.
Quantification : Si $A = \forall X.C$, un terme t de type A est réductible si pout tout type B et tout candidat de réductibilité $\mathscr{B}$ de type B, alors tB est réductible de type $C^{[B/X]}$ où la réductibilité pour ce type est définie par induction comme précédemment mais en prenant $\mathscr{B}$ comme définition de la réductibilité pour B (i.e. avec $[\![B]\!] = \mathscr{B}$).

C'est une définition compliquée de la réductibilité, mais elle est nécessaire. On ne pouvait pas dire qu'un terme t de type $A = \forall X.C$ est réductible si pour tout type B, tB est réductible de type $C^{[B/X]}$ car le type $C^{[B/X]}$ peut-être beaucoup plus long que $\forall X.C$. Il suffit de prendre $A = B = \forall X.X$ pour se convaincre que l'on tourne en rond, on a besoin de la définition de la réductibilité de A pour définir celle de A...

On remarque que si A n'est pas une variable de type, la définition de $[\![A]\!]$ introduit une quantification et donc $t \in [\![A]\!]$ ne peut pas être exprimé au premier ordre. De plus si A contient une quantificateur, la définition de $[\![A]\!]$ comporte une quantification du second ordre et la formule $t \in [\![A]\!]$ ne peut plus être exprimé dans $\mathscr{PA}_2$.

Définition : Réductibilité paramétrée

Soit ϕ une application qui à chaque variable associe un candidat de réductibilité. On dit que ϕ est un *contexte de candidats*. On utilisera les notations suivantes : si A est un type, A^ϕ est le type obtenu en substituant chaque variable libre X de A par le type de $\phi(X)$. De même, $\phi[\mathscr{D}/X]$ est l'application ψ définie par $\psi(X) = \mathscr{D}$ et $\psi(Y) = \phi(Y)$ si $X \neq Y$.

La *réductibilité paramétrée* $[\![A]\!]^\phi$ est alors l'ensemble de termes de type A^ϕ défini de la manière suivante :
Variable : Si $A = X$ est une variable de type, alors $[\![X]\!]^\phi = \phi(X)$.
Implication : Si $A = B \to C$, alors $[\![B \to C]\!]^\phi = [\![B]\!]^\phi \to [\![C]\!]^\phi$.
Quantification : Si $A = \forall X.C$ alors $[\![\forall X.C]\!]^\phi$ est l'ensemble des termes t tels que pour tout type D et tout candidat de réductibilité $\mathscr{D}$ de ce type alors $tD \in [\![C]\!]^{\phi[\mathscr{D}/X]}$.

Proposition :

La réductibilité paramétrée $[\![A]\!]^\phi$ est un candidat de réductibilité de type A^ϕ.

On retrouve la définition de la réductibilité en prenant ϕ l'application qui à X associe $\mathscr{SN}_X$ l'ensemble des termes fortement normalisants de type X. Et donc la proposition permet de dire que pour tout type A, $[\![A]\!]$ est un candidat de réductibilité. Par (CR1), un terme réductible est fortement normalisant.

Preuve :

On raisonne par induction sur le type A :
Variable : Si $A = X$ est une variable de type, alors $[\![X]\!]^\phi = \phi(X)$ est un candidat de réductibilité.
Implication : C'est le lemme précédent.
Quantification : Si $A = \forall X.C$:

- **CR1 :** Soient $t \in [\![A]\!]^\phi$, D un type et $\mathscr{D}$ un candidat de réductibilité de ce type. Par définition, $tD \in [\![C]\!]^{\phi[\mathscr{D}/X]}$ et par l'hypothèse d'induction (CR1) tD est fortement normalisant donc t aussi.
- **CR2 :** Soit $t \in [\![A]\!]^\phi$ tel que $t \rightsquigarrow t'$. Pour tout type D et tout candidat de réductibilité $\mathscr{D}$, on a $tD \in [\![C]\!]^{\phi[\mathscr{D}/X]}$ et $tD \rightsquigarrow t'D$. Par l'hypothèse d'induction (CR2), $t'D \in [\![C]\!]^{\phi[\mathscr{D}/X]}$. D'où $t' \in [\![A]\!]^\phi$.
- **CR3 :** Supposons t de type A simple et tel que tous ses réduits immédiats sont dans $[\![A]\!]^\phi$. Fixons un type D et $\mathscr{D}$. Un réduit immédiat de tD est nécessairement de la forme $t'D$ où $t \to_1 t'$. Or $t'D \in [\![C]\!]^{\phi[\mathscr{D}/X]}$. Par l'hypothèse d'induction (CR3), $tD \in [\![C]\!]^{\phi[\mathscr{D}/X]}$ et donc $t \in [\![A]\!]^\phi$.

□

4.2.3 Le lemme de substitution

Lemme : Substitution
$[\![A^{[B/Y]}]\!]^{\phi} = [\![A]\!]^{\phi[[\![B]\!]^{\phi}/Y]}$

Bien que sa démonstration semble être une trivialité, ce lemme est capital pour la démonstration du théorème de normalisation. On utilise un schéma de compréhension plus général pour le formuler car $[\![B]\!]^{\phi}$ n'est pas un ensemble exprimable par une formule du second ordre. C'est ici que se manifeste l'incomplétude de $\mathscr{PA}_2$.

Preuve :
On raisonne par induction sur le type :
Variable : Si A est une variable, on a deux cas :
– Si $A = Y$, $[\![A^{[B/Y]}]\!]^{\phi} = [\![B]\!]^{\phi} = [\![A]\!]^{\phi[[\![B]\!]^{\phi}/Y]}$.
– Si $A \neq Y$, $[\![A^{[B/Y]}]\!]^{\phi} = [\![A]\!]^{\phi} = [\![A]\!]^{\phi[[\![B]\!]^{\phi}/Y]}$.
Implication : Si $A = C \to D$,

$$\begin{aligned}
[\![(C \to D)^{[B/Y]}]\!]^{\phi} &= [\![C^{[B/Y]} \to D^{[B/Y]}]\!]^{\phi} \\
&= [\![C^{[B/Y]}]\!]^{\phi} \to [\![D^{[B/Y]}]\!]^{\phi} \\
&= [\![C]\!]^{\phi[[\![B]\!]^{\phi}/Y]} \to [\![D]\!]^{\phi[[\![B]\!]^{\phi}/Y]} \\
&= [\![C \to D]\!]^{\phi[[\![B]\!]^{\phi}/Y]}.
\end{aligned}$$

Quantification : Si $A = \forall X.C$, on a deux cas,
– Si $X = Y$:

$$\begin{aligned}
t \in [\![(\forall X.C)^{[B/X]}]\!]^{\phi} &\Leftrightarrow t \in [\![\forall X.C]\!]^{\phi} \\
&\Leftrightarrow \forall (D, \mathscr{D}), tD \in [\![C]\!]^{\phi[\mathscr{D}/X]} \\
&\Leftrightarrow \forall (D, \mathscr{D}), tD \in [\![C]\!]^{\phi[[\![B]\!]^{\phi}/X][\mathscr{D}/X]} \\
&\Leftrightarrow t \in [\![\forall X.C]\!]^{\phi[[\![B]\!]^{\phi}/X]}.
\end{aligned}$$

– Si $X \neq Y$:

$$\begin{aligned}
t \in [\![(\forall X.C)^{[B/Y]}]\!]^{\phi} &\Leftrightarrow t \in [\![\forall X.C^{[B/Y]}]\!]^{\phi} \\
&\Leftrightarrow \forall (D, \mathscr{D}), tD \in [\![C^{[B/Y]}]\!]^{\phi[\mathscr{D}/X]} \\
&\Leftrightarrow \forall (D, \mathscr{D}), tD \in [\![C]\!]^{\phi[\mathscr{D}/X][[\![B]\!]^{\phi}/Y]} \\
&\Leftrightarrow t \in [\![\forall X.C]\!]^{\phi[[\![B]\!]^{\phi}/Y]}.
\end{aligned}$$

□

4.2.4 Le théorème

Théorème : Normalisation forte
Tout terme du système F est réductible, donc en particulier tout terme est fortement normalisant.

La preuve consiste en une induction sur les termes, et nous aurons besoin des quatre propositions suivantes pour montrer l'hérédité.

Proposition : Application
Supposons $t \in [\![C \to D]\!]^{\phi}$ et $u \in [\![C]\!]^{\phi}$, alors $tu \in [\![D]\!]^{\phi}$.

Preuve :
C'est juste la définition de $[\![C \to D]\!]^{\phi} = [\![C]\!]^{\phi} \to [\![D]\!]^{\phi}$. □

Proposition : Abstraction

Si t est un terme de type D^ϕ et si pour tout terme $u \in [\![C]\!]^\phi$, $t[u/x] \in [\![D]\!]^\phi$, alors $\lambda x.t \in [\![C \to D]\!]^\phi$.

Preuve :

Il s'agit de montrer que pour tout terme $u \in [\![C]\!]^\phi$, $(\lambda x.t)u \in [\![D]\!]^\phi$. Par hypothèse, $t = t[x/x] \in [\![D]\!]^\phi$ donc par (CR1) est fortement normalisant. On peut donc définir $\nu(t)$ et $\nu(u)$ et raisonner par récurrence sur $\nu(t)+\nu(u)$. Un réduit immédiat de ce terme est de la forme $t[u/x]$ ou $(\lambda x.t')u$ avec $t \to_1 t'$ ou encore $(\lambda x.t)u'$ avec $u \to_1 u'$. Le premier appartient à $[\![D]\!]^\phi$ par hypothèse. Dans les autres cas on utilise l'hypothèse de récurrence car $\nu(t') < \nu(t)$ et $\nu(u') < \nu(u)$. Ainsi, tous les réduits immédiats sont dans $[\![D]\!]^\phi$, et par (CR3), $(\lambda x.t)u \in [\![D]\!]^\phi$. □

Proposition : Abstraction universelle

Si t est un terme de type A^ϕ et si pour tout type D et tout candidat de réductibilité $\mathscr{D}$ de ce type $t^{[D/Y]} \in [\![A]\!]^{\phi[\mathscr{D}/Y]}$, alors $\Lambda Y.t \in [\![\forall Y.A]\!]^\phi$.

Preuve :

Il s'agit de montrer que pour tout type D et tout candidat de réductibilité $\mathscr{D}$ de ce type $(\Lambda Y.t)D \in [\![A]\!]^{\phi[\mathscr{D}/Y]}$. Or $t = t^{[Y/Y]} \in [\![A]\!]^{\phi[\mathscr{D}/Y]}$ donc est fortement normalisant. Comme pour l'abstraction, nous allons raisonner par récurrence sur $\nu(t)$. Un réduit immédiat de $(\Lambda Y.t)D$ est de la forme $t^{[D/Y]}$ ou de la forme $(\Lambda Y.t')D$ avec $t \to_1 t'$. Le premier est appartient à $[\![A]\!]^{\phi[\mathscr{D}/Y]}$ par hypothèse, pour le second, c'est l'hypothèse de récurrence. Ainsi, tous les réduits immédiats sont dans $[\![A]\!]^{\phi[\mathscr{D}/Y]}$, et par (CR3), il en est de même pour $(\Lambda Y.t)D$. □

Proposition : Application universelle

Si $t \in [\![\forall X.C]\!]^\phi$, alors pour tout type D, si de plus pour toute variable libre Y de D, $\phi(Y)$ est de type Y, $tD \in [\![C^{[D/X]}]\!]^\phi$.

Preuve :

Par hypothèse, pour tout type D et tout candidat de réductibilité $\mathscr{D}$, $tD \in [\![C]\!]^{\phi[\mathscr{D}/X]}$, alors si D est comme dans l'énoncé, en prenant $\mathscr{D} = [\![D]\!]^\phi$, $tD \in [\![C]\!]^{\phi[[\![D]\!]^\phi/X]} = [\![C^{[D/X]}]\!]^\phi$ par le lemme de substitution. □

Proposition :

Soit t un terme de type T et supposons que les variables libres de t figurent parmi $x_1, x_2, \ldots, x_n$ de types $B_1, B_2, \ldots, B_n$, et soit ϕ un contexte de réductibilité. Si $u_1, ..., u_n$ sont des termes de type $B_1^\phi, \ldots, B_n^\phi$ qui sont dans $[\![B_1]\!]^\phi, \ldots, [\![B_n]\!]^\phi$ alors $t^\phi[u_1/x_1, \ldots, u_n/x_n] \in [\![T]\!]^\phi$

Preuve :

Par induction sur t : si $t = x_i$ est une variable, il s'agit de montrer que $u_i \in [\![B_i]\!]^\phi$ ce qui est l'hypothèse. Pour les quatre autres cas, il s'agit simplement de la proposition associée. □

Preuve du théorème

En prenant $\phi(X) = \mathscr{SN}_X$ et $u_i = x_i$, on obtient $t \in [\![T]\!]$, et donc tout terme est réductible. □

On remarque que l'on ne démontre pas directement que tout terme est réductible. D'une part, l'hypothèse d'induction est beaucoup plus forte que la normalisation forte du terme. D'autre part, la preuve passe par la définition de la réductibilité paramétrée dépendant d'un contexte de candidat, et la première notion de réductibilité n'en est qu'un cas particulier. Sans la quantification (par exemple dans le cas du λ-calcul simplement typé), la preuve est beaucoup plus directe : on montre que pour

tout type $A, [\![A]\!]$ vérifie les conditions (CR1),(CR2) et (CR3), puis on montre la dernière proposition où l'on n'a pas besoin d'effectuer de substitution pour les types et où l'on n'a pas besoin du lemme de substitution.

Dès qu'on ajoute la quantification du second ordre, il faut considérer des candidats de réductibilité, c'est à dire des ensembles quelconques vérifiant (CR1),(CR2) et (CR3), et la condition pour être réductible porte sur *tous* les candidats, et parmi eux, il y a la *vraie* notion.

Qu'est ce qui fait que la preuve fonctionne lorsque l'on quantifie sur les candidats de réductibilité alors qu'elle ne peut pas fonctionner lorsque l'on ne s'interesse qu'au cas particulier de la vraie réductibilité ? En fait, lorsque l'on démontre un énoncé universel, on ne le démontre jamais pour chaque élément individuellement (sauf s'il n'y en a qu'un nombre fini). On le montre pour un élément *générique* qui ne possède comme propriété que son *type*. C'est ce qui se passe ici. L'abstraction universelle est tellement uniforme que la preuve peut fonctionner.

Ce caractère uniforme de l'abstraction universelle (et donc de la quantification du second ordre) est détaillée dans le théorème suivant de G. Longo. On se place dans le système Fc qui est le système F complété de l'axiome C :

Axiome C :
si t est un terme de type $\forall X.C$ et si X n'est pas libre dans C, alors pour tous types T et T', $tT = tT'$ et tT est de type C

Théorème : Généricité
Soient t, t' des termes de types $\forall X.C$ alors, si il existe un type T tel que $tT =_\beta t'T$ alors $t =_\beta t'$. Autrement dit, si deux fonctions définies pour tout type coïncident pour un seul type, alors elles sont égales.

5 Conclusion

Le théorème de normalisation prouvé, revenons un peu sur le premier théorème de Gödel. En effet, il pose a priori un problème de taille aux mathématiciens : “l’indécidabilité” de certaines propositions.

Ils ont heureusement trouvé depuis longtemps la solution. Comme on l’a vu pour la normalisation, une partie du travail avant d’arriver à une preuve mathématique consiste à choisir un système formel assez puissant où se jouera la démonstration que l’on imagine. Bien loin de l’horizon indépassable des propriétés “indémontrables” (Dans quel système ? Avec quels axiomes ?) le premier théorème d’incomplétude énonce avant tout l’incapacité intrinsèque des systèmes formels à capturer l’ensemble des mathématiques, et plus généralement la cognition humaine.

Suite à ce constat, ce théorème n’apparaît plus forcément comme un résultat négatif, mais comme une conscience – très bénéfique – des limites des outils que nous utilisons pour rendre les mathématiques communicables et intelligibles. Cela se manifestera par l’explosion récente de la théorie des modèles, ou plus modestement par notre indication concernant le cadre logique nécessaire pour la preuve du théorème de normalisation.

Références

[1] T. Fruchart et G. Longo. Carnap's remarks on impredicative definitions and the genericity theorem. *Logic and Foundations of Mathematics (Cantini et al. eds). Kluwer Academic Publishers*, 1999.

[2] J.-Y. Girard. Le point aveugle 1 : Vers la perfection. *Vision des sciences, Hermann*, 2006.

[3] J.-Y. Girard, P. Taylor, and Y. Lafont. Proofs and types. *Cambridge Univerity Press*, 1989.

[4] J. Goubault-Larrecq. Aspects logiques. *Notes de cours*, 1999.

[5] J. Goubault-Larrecq. Lambda-calcul et langages fonctionnels. *Notes de cours*, 1999.

[6] J.-L Krivine. Lambda-calcul : types et modèles. *Paris ; Milan ; Barcelone : Masson*, 1990.

[7] G. Longo. Introduction aux théorèmes d'incomplétude : de Gödel à Kruskal-Friedman. *Notes de cours*.

[8] G. Longo. Le système des types du second ordre et la cohérence de l'Arithmétique. *Notes de cours*.

[9] G. Longo. Prototype proofs in type theory. *Mathematical Logic Quaterly, vol. 46, n. 3*, 2000.

[10] G. Longo. Reflections on incompleteness. *Invited Lecture,Types for Proofs and Programs, Durham, (GB), Dec. 2000 ; Lecture Notes in Computer Science, vol 2277 (Callaghan et al. eds), pp. 160 - 180, Springer*, 2002.

[11] S. Shapiro. Foundation without foundationalism : a case for second-order logic. *Oxford : Clarendon Press*, 1991.

[12] Y. Villessuzanne V. Nesme. Cohérence d'une théorie imprédicative. *Mémoire de maîtrise*, 2001.