Canonicité et Normalisation

Thierry Coquand

Résumé de la leçon précédente

J'ai présenté une définition précise d'une version de la théorie des types avec $\Pi, \Sigma,$ Id et hiérarchie d'univers; exemples de *modèles* de cette théorie : modèle ensembliste et modèles de préfaisceaux.

Deux méthodes pour construire des modèles : modèle interne, et modèle de recollement le long de morphisme exact à gauche.

Le modèle des termes est spécifié comme le modèle *initial* pour la notion de morphisme strict.

Un des buts de cette leçon sera de montrer que l'égalité est *décidable* pour ce modèle initial, et que l'on a $(A, B) = (A', B')$ si $\Pi\ A\ B = \Pi\ A'\ B'$ et e.g. $\Pi\ A\ B$ n'est pas égal à U_n; un corollaire est que le judgement de typage $a : A$ est *décidable* et un autre corollaire est que l'opération de β-réduction préserve le type.

Résumé de la leçon précédente

Morphisme de modèle exact à gauche $\psi : M \to M_0$

$\psi(\mathsf{id}) = \mathsf{id} \qquad \psi(\sigma\delta) = \psi(\sigma)\psi(\delta) \qquad \psi(A\sigma) = \psi(A)\psi(\sigma) \qquad \psi(t\sigma) = \psi(t)\psi(\sigma)$

Le morphisme est donc *strict* par rapport aux substitutions.

On peut alors définir $\langle \psi\mathsf{p}, \psi\mathsf{q} \rangle : \psi(\Gamma.A) \to \psi(\Gamma).\psi(A)$

On introduit $\downarrow : 1 \to \psi(1)$ et $\downarrow : \psi(\Gamma).\psi(A) \to \psi(\Gamma.A)$ avec

$\downarrow \langle\rangle = \mathsf{id} \qquad \downarrow \langle \psi\mathsf{p}, \psi\mathsf{q} \rangle = \mathsf{id} \qquad \langle \psi\mathsf{p}, \psi\mathsf{q} \rangle \downarrow = \mathsf{id}$

Cela entraine en particulier $(\psi\mathsf{p}) \downarrow = \mathsf{p}$ et $(\psi\mathsf{q}) \downarrow = \mathsf{q}$.

Résumé de la leçon précédente

Cette notion de morphisme exact à gauche est *différente* de celle de pseudo-morphisme, introduite par P. Clairambault et P. Dybjer
The Biequivalence of Locally Cartesian Closed Categories and Martin-Löf Type Theories (2012).

Pour cette notion, on a un isomorphisme $\psi(A\sigma) \simeq \psi(A)\psi(\sigma)$.

Pour un morphisme exact à gauche, on impose une égalité $\psi(A\sigma) = \psi(A)\psi(\sigma)$ et de même $\psi(a\sigma) = \psi(a)\psi(\sigma)$.

Cette leçon

Un des buts de cette leçon est d'expliquer comment montrer

Théorème : *L'égalité dans le modèle initial est décidable pour chaque sorte; si* A *est dans* $\mathsf{Type}(\Gamma)$ *et* a *est un terme au premier ordre construit avec les constantes, on peut décider si on peut déduire* $\Gamma \vdash a : A$ *ou non.*

Ceci est une propriété désirable : le terme a peut représenter une preuve potentielle de A, et on doit pouvoir décider si une preuve potentielle est *correcte* ou non.

Kreisel 《dictum》 *we can recognize a proof when we see one*

On va d'abord présenter la preuve d'une propriété plus faible, la propriété de *canonicité*, qui entraine la cohérence.

Hilbert et la programmation fonctionnelle

Hilbert 1923-30 introduit un système de calcul fonctionnel avec deux types de base pour les entiers et pour les ordinaux.

Le type des entiers est construit à partir des opèrations 0 et $n+1$

Dans ce système, on a une représentation symbolique de chaque entier k par un terme de la forme $\overline{k} = (\dots((0+1)+1)\dots)+1$.

Le type des ordinaux est construit à partir des opèrations 0 et $n+1$ et $\mathsf{lim}(u)$ pour $u : N \to Ord$

Il définit le schéma de récursion primitive

$$f(0, x_1, \dots, x_n) = a(x_1, \dots, x_n) \qquad f(n+1, x_1, \dots, x_n) = g(n, f(n, x_1, \dots, x_n), x_1, \dots, x_n)$$

et remarque que l'on semble obtenir plus de fonctions si on permet des paramètres fonctionnels

Hilbert et la programmation fonctionnelle

$$\iota(0, f, a) = a \qquad \iota(n+1, f, a) = f(a, \iota(n, f, a))$$

$$\varphi(0, a, b) = a + b \qquad \varphi(n+1, a, b) = \iota(b, \varphi(n), a)$$

ce qui peut se reécrire en

$$\varphi(0, a, b) = a + b \quad \varphi(n+1, a, 0) = a \quad \varphi(n+1, a, b+1) = \varphi(n, a, \varphi(n+1, a, b))$$

Il demande à son étudiant Ackermann de montrer que l'on ne peut pas définir une telle fonction sans utiliser des paramètres fonctionnels (i.e. de manière primitive récursive); ce peut être vue comme une justification de la programmation fonctionnelle.

En 1988, Loïc Colson pourra montrer un résultat analogue, mais de nature ⟪intensionnelle⟫ : on n'a pas de ⟪bonne⟫ fonction primitive récursive pour calculer le minimum de deux entiers mais il y a un bon algorithme si on permet des paramètres fonctionnels.

Hilbert et la programmation fonctionnelle

La motivation du travail de Hilbert est de montrer la cohérence d'un système avec des formules atomiques $t = u$, avec t, u de types N, et d'introduire $\epsilon : (N \to N) \to N$ avec l'axiome

$$f(x) = 0 \to f(\epsilon(f)) = 0.$$

La méthode est de montrer comment *éliminer* le symbole ϵ de toute preuve *donnée*, par ⟪essais-erreurs⟫ (on essaie d'abord avec $\epsilon(f) = 0$, puis on modifie la valeur si cela ne marche pas).

Pour cela, il est essentiel de pouvoir calculer la valeur de tout terme clos, sans ϵ, de type N. C'est le résultat de *canonicité* : tout terme de type N est convertible à un terme de la forme $\overline{k}$.

Gödel et Dialectica

Ce système fonctionnel sera repris par Gödel en 1940, 1958, 1970 (sans le type des ordinaux); ceci deviendra le *système* T. Un but est d'étudier la cohérence de l'arithmétique.

Il introduit un calcul avec une égalité $t = u$ sur un type quelconque, et il pense à cette égalité comme un égalité *définitionnelle*, qui doit être *décidable*; pour cela on a besoin d'un théorème de *normalisation*; on associe à chaque terme t un terme $norm(t)$ tel que t et u sont convertibles si, et seulement si, $norm(t)$ et $norm(u)$ sont identiques.

The mathematicians will probably raise objections against that, because contemporary mathematics is thoroughly extensional and hence no clear notions of intensions have been developed. But it is nevertheless certain that, at least within the framework of a particular language, completely precise concepts of this kind can be defined, Gödel, lettre à Bernays (1970)

Cette notion sera l'objet de nombreuses discussions entre Howard et Tait, cf. *Conversations with Bill about Functionals and Terms*, Howard (2019).

Gödel et Dialectica

Gödel publie *Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes* (1958)

Le papier de Tait *Intensional interpretations of functionals of finite type, part I* (1967) essaie de préciser certains points de ce papier et présente une preuve de canonicité et normalisation pour le système de Gödel.

Tait essaie de motiver que l'on doit considérer β, η-conversion comme égalité définitionnelle. Il donne l'exemple de $t = \lambda_x x$ et $u = \lambda_x(x, x).1$ qui sont β-convertibles mais pas faiblement β-convertible. Comme $t\ x$ et $u\ x$ sont convertibles pour une variable x, il écrit
So it seems reasonable, even compelling, to regard t and u as definitionally equal.

Prédicat de calculabilité

Hilbert, Gödel n'utilisent pas la notation du λ-calcul introduite par Church, 1930.

On introduit des constantes, qui peuvent être définies par équation explicite

$$f\ x_1\ \ldots\ x_n = t$$

ou par récursion

$$f\ 0 = a \qquad f\ (n+1) = g\ n\ (f\ n)$$

(un peu comme dans le langage *Miranda* de D. Turner)

Les termes sont : constantes, variables et application $t\ u$ si $t : A \to B$ et $u : A$

Le livre de Shoenfield *Mathematical logic* (1967) présente une preuve claire de canonicité (qui va inspirer Girard dans sa preuve de normalisation pour le system F).

Prédicat de calculabilité

$$\frac{}{t = t} \qquad \frac{t = u \qquad t = v}{u = v} \qquad \frac{t = t_1 : A \to B \qquad u = u_1 : A}{t\ u = t_1\ u_1 : B}$$

$f\ a_1\ \ldots\ a_n = t(a_1/x_1, \ldots, a_n/x_n)$ si $f\ x_1\ \ldots\ x_n = t$

$f\ 0 = a \qquad f\ (n+1) = g\ n\ (f\ n)$

Ce calcul peut être vue comme une version simple du calcul purement équationnel que j'ai présenté dans ma leçon précédente.

Prédicat de calculabilité

On définit *être calculable* par induction sur le type.

$N'(t)$ signifie que t est *convertible* à un terme de la forme $\overline{k}$.

$(A \to B)'(t)$ signifie $\forall_u\ A'(u) \to B'(t\ u)$.

Lemme 1 : *Un terme est construit à partir de constantes calculables est calculable*

Lemme 2 : *Un terme convertible à un terme calculable est calculable*

Lemme 3 : *Si $t\ \overline{k}$ est calculable pour tout k, alors t est calculable*

Lemme 4 : *Toute constante est calculable*

Théorème : *Tout terme clos de type N est convertible à un terme de la forme $\overline{k}$*

Prédicat de calculabilité

La preuve est élégante, et utilise dans la méta-théorie une logique avec $\rightarrow$ et $\forall$ Comme elle est effective, elle donne un procédé de calculer la valeur d'un terme t de type N.

Gödel avait déjà un argument similaire en 1940

The computable functions of type 0 are the natural numbers

If the concepts of "computable function of type t_0", "computable function of type t_1",..., "computable function of type t_k" (for some $k \geqslant 1$) have been defined, then a computable function of type $(t_0, t_1, \ldots, t_k)$ is defined to be an operation which is always effective (and constructively recognisable as such), and which assigns a computable function of type t_0 to each k-tuple of computable functions of types $t_1, t_2, \ldots, t_k$.

We must regard this concept as being immediately intelligible.

Recollement et canonicité

Mais il n'inclut pas dans son texte la preuve de canonicité.

In the Princeton notes, Gödel observes that it is not difficult to prove calculability of the functionals in the following sense: a functional F *of type* $\sigma_1, \ldots, \sigma_n \to o$ *is said to be calculable if for arbitrary calculable* $t_1, \ldots, t_n$ *of types* $\sigma_1, \ldots, \sigma_n$ *respectively,* $Ft_1 \ldots t_n$ *can be proved to be equal to a numeral. Gödel goes on to say "I don't want to give this proof in more detail because it is of no great value for our purpose for the following reason: if you analyse this proof it turns out that it makes use of logical axioms, also for expressions containing quantifiers and it is exactly these axioms which we want to deduce from the system* Σ*."* Troesltra, Gödel Collected Work 2 (1995).

Prédicat de calculabilité

Tait (1967), pour montrer la *normalisation*, introduit une relation de réduction sur les termes; comme dans Girard (1971), c'est la réduction des redex en priorité les plus internes.

Il redéfinit $N'(t)$ comme : t se réduit sur un terme $\overline{k}$ et montre alors que tout terme t est normalisable pour un type quelconque. Il introduit pour cela une fonction 0_A par induction sur A : $0_N = 0$ et $0_{A\to B}x = 0_B$

Lemme : *Le term 0_A est calculable et si $t : A$ est calculable alors t est normalisable*

Théorème : *Tout terme est normalisable*

Ceci permet, avec une propriété de confluence, de montrer que la conversion de termes est *décidable* pour chaque type, comme l'exigeait Gödel.

On s'attend à ce que la preuve de canonicité soit comme une version plus simple de la preuve de normalisation

Prédicat de calculabilité pour système F

En 1971, Girard introduit un système avec quantification sur les types; par exemple le type $T = \Pi_X X \to X$ qui a pour élément la fonction $\lambda_X \lambda_{x:X} x$

Qu'est-ce que $T'(t)$ calculable pour le type A veut dire?

Si on essaie : pour tout A on doit avoir $t\ A$ calculable au type $A \to A$ can A peut être un type qui contient T et la définition est circulaire

Prédicat de calculabilité pour système F

Girard a l'idée d'introduire la notion de *candidat de calculabilité*

Un candidat de calculabilité au type A est juste un prédicat sur l'ensemble des termes de type A (clos par conversion)

On peut alors définir $T'(t)$ comme : pour tout type A et pour *tout* candidat C_A on a

$$C_A(u) \to C_A(t\ A\ u)$$

et cette définition n'est plus circulaire.

Par exemple on a bien $T'(t)$ pour $t = \lambda_X \lambda_{x:X} x$ car on a $t\ A\ u$ convertible à u.

Prédicat de calculabilité pour système F

La même idée marche pour la normalisation mais il faut cette fois introduire une *constante* 0_A avec de nouvelles lois de réduction $0_{A\to B}\ t \to 0_A$ et $0_{\forall_X T(X)}\ A \to 0_{T(A)}$ et imposer au candidat C_A de vérifier les conditions

(1) $C_A(t)$ entraine que t est normalisable

(2) On a $C_A(0_A)$

(3) Si $t \to u$ et $C_A(u)$ alors on a $C_A(t)$

Girard généralise cette preuve au système $F\omega$ et c'est cette preuve que Martin-Löf essaie d'adapter à type : type

L'introduction de ces constantes 0_A est nécessaire par la présence de types ⟪vides⟫ comme $\forall_X X$. Pour le système T, on peut définir 0_A par induction sur A en partant de 0 pour $A = N$.

Prédicat de calculabilité

Cette notion de ⟪candidat⟫ de calculabilité va être importante, même pour un calcul prédicatif.

Girard introduisait ce système pour généraliser l'interprétation Dialectica de Gödel (pour l'arithmétique) à l'analyse (arithmétique du second-ordre).

De manière peut-être surprenante, la même idée permettait de montrer que le processus de normalisation pour l'analyse formulée en déduction naturelle termine toujours, malgré son caractère circulaire. Cela donnait une nouvelle preuve ⟪effective⟫ de la conjecture de Takeuti (1951).

Évolution de la théorie des types

71-72 : *An Intuitionistic Theory of Types* conversion définie au niveau des termes non typés, preuve de normalisation avec seulement β-conversion

73 : *An Intuitionistic Theory of Types: Predicative Part* introduction du jugement de conversion, preuve de normalisation, mais pour une version ⟪combinateur⟫ (β-réduction ⟪faible⟫)

79 : *Constructive Mathematics and Computer Programming* jugement de conversion, canonicité, β et η conversion, mais la normalisation n'est *pas* valide, et le typage n'est *pas* décidable

84 : *Intuitionistic Type Theory*, même système, représentation des univers comme types avec fonction de décodage

Évolution de la théorie des types

Un changement important a été d'introduire un jugement de conversion explicite.

Pour les versions 71-73, on définit $A = B$ au niveau de termes syntaxiques comme la conversion telle que définie dans le λ-calcul, qui, par le théorème de confluence, revient au fait que A et B peuvent se réduire sur le même terme.

Il en résulte alors que, si l'on a $\Pi_{x:A}B = \Pi_{x:A'}B'$, alors $A = A'$ et $B = B'$ $(x : A)$.

C'est un lemme clef pour montrer que l'opération de β-réduction préserve les types, propriété essentielle si on veut utiliser ce système comme un langage de programmation fonctionnelle.

Évolution de la théorie des types, version 1973

Si on veut par exemple définir un modèle ensembliste, on doit avoir un jugement de conversion, cf. *On Relating Type Theories and Set Theories* (1998) P. Aczel.

Mais pour cette version avec un jugement de conversion, il n'est pas si simple de montrer que la β-réduction préserve les types; on a besoin de l'*injectivité* de l'opération du produit dépendant.

L'idée, due à Peter Hancock, pour montrer cette propriété apparait dans la version 1975; la propriété de confluence est vue comme une conséquence de la construction d'un modèle de recollement.

Avec un jugement de conversion, il n'est pas si simple de résoudre le problème des types vides : si on ajoute des constantes, comment être sûr que cette extension est conservative ?

Évolution de la théorie des types, version 1979

Pour la version 1979, Martin-Löf ajoute la loi

$$\frac{\Gamma \vdash p : \mathsf{Id}\ A\ a\ b}{\Gamma \vdash a = b : A}$$

On peut alors montrer l'*extensionalité*, en particulier $(\forall_{n:N} \mathsf{Id}\ N\ (f\ n)\ (g\ n)) \to \mathsf{Id}\ (N \to N)\ f\ g$

Pour cela, on utilise la loi suivante

$$\frac{\Gamma, x : N \vdash f\ x = g\ x : N}{\Gamma \vdash f = g : N \to N}$$

Ceci entraine que le jugement $f = g : N \to N$ n'est *pas* décidable, et on peut en déduire que le jugement $a : A$ n'est pas décidable.

Évolution de la théorie des types, version 1979

Avec des univers, ce système n'est pas normalisable; le terme suivant

$$\lambda_{X:U}\lambda_{e:\mathsf{Id}\ U\ X\ (X\to X)}(\lambda_{x:X}x\ x)\ (\lambda_{x:X}x\ x) : \Pi_{X:U}(\mathsf{Id}\ U\ X\ (X \to X)) \to X$$

n'a pas de forme normale.

Howard écrira dans sa revue *A discussion of the use of the equality relation in the book is beyond the scope of this review. Though the author's treatment of equality is clearly valid for his computational semantics (if I have understood the latter properly), there are various issues that need to be explored or clarified.* (1986)

Un autre caractère surprenant de ce système est qu'il prouve la négation de la thèse de Church (comme remarqué par Troelstra). Mais ceci vient en fait de l'interprétation forte de l'existence : M. Escardo (2013) a montré la négation du principe de continuité avec une telle interprétation de l'existence.

Évolution de la théorie des types, version 1979

On peut toutefois montrer la canonicité, en donnant une sémantique en terme de ⟪Partial Equivalence Relation⟫

Ceci est possible grâce à une définition ⟪inductive-récursive⟫ complexe, justifiée par exemple dans le papier *A Non-Type-Theoretic Definition of Martin-Lof's Types* S. Allen (1987).

Le système NuPrl utilise cette loi; il ne peut pas donc pas utiliser un algorithme de typage, et à la place utilise l'approche ⟪LCF⟫, due à Robin Milner, avec un type abstrait de preuves.

La construction du modèle de préfaisceau, et un modèle de l'univalence, construit autour des ensembles cubiques, a été formalisé par Mark Bickford dans ce système (2017). Les énoncés sont élégants, mais certains arguments informels étaient surprenamment complexes à représenter.

Il est en tout cas intéressant d'avoir une notation *explicite* des preuves, ce qui n'est pas le cas dans cette approche LCF. De plus, en théorie des types telle que je l'ai présenté, ces termes de preuve ont une structure riche et permettent des opérations de réduction et de substitution.

Évolution de la théorie des types

Le système *Lean* ne vérifie pas la propriété de normalisation et de décidabilité de typage, cf. *Failure of Normalization in Impredicative Type Theory with Proof-Irrelevant Propositional Equality*, A. Abel et Th. C. (2019).

Le typage est semi-décidable.

Pour le système *Rocq*, ce n'est pas clair; la loi de *Constructive Epsilon* est prouvée à partir d'un principe dont la normalisation n'a pas été étudiée.

Dans les deux cas, on a la propriété de canonicité.

Canonicité

Si on essaie de généraliser la preuve de Shoenfield pour la théorie des types comme présentée dans la leçon précédente, comme théorie équationnelle, on obtient le problème suivant

Pour $T = N$, calculable signifie convertible à un term $\overline{k}$

Pour $T = \Pi_A B$ on définira $T'(t)$ de la manière suivante : si u calculable au type A alors $t\ u$ calculable au type $B[u]$

Mais pour $T = U_n$ il n'est pas clair quelle doit être la définition

On voudrait définir $T'(X)$ comme : X est convertible à $\Pi\ A\ B$ ou à $\Sigma\ A\ B$ ou à $U_m,\ m < n$

Mais on a peut-être $\Pi\ A\ B = U_m$ ou Π n'est peut-être pas injectif.

Il semble donc difficile de définir a priori la calculabilité par induction sur le type.

Canonicité

On peut résoudre ces problèmes en introduisant une *relation* de réduction et en considérant une relation de calculabilité à la place d'un prédicat de calculabilité

La preuve devient assez complexe. (Cet argument a pu toutefois être vérifié dans le système Agda.)

Mais il est de toute manière étrange d'introduire une relation pour un système qui ne vérifie pas l'extensionalité, et d'avoir un argument si complexe pour la *canonicité*.

Canonicité

Une clef est de définir une *structure* de calculabilité au lieu d'un *prédicat* de calculabilité

Tous les problèmes de confluence disparaissent alors comme par magie, et en fait, on peut montrer la confluence par cette méthode.

De plus, ceci peut être présenté comme une application de la technique de recollement le long d'un morphisme exact à gauche.

Reformulation comme modèle de recollement

Soit M un modèle quelconque, et Ens le modèle ensembliste, on peut toujours définir un morphisme *exact à gauche* $\psi : M \to \mathsf{Ens}$

$$\psi(\Gamma) = 1 \to \Gamma \qquad \psi(\sigma)\nu = \sigma\nu \qquad \psi(A)\rho = \mathsf{Elem}(1, A\rho) \qquad \psi(a)\rho = a\rho$$

A priori $\psi(1)$ n'est pas égal à 1 et $\psi(\Gamma.A)$ n'est pas égal à $\psi(\Gamma).\psi(A)$ mais on peut définir

$$\downarrow : \psi(\Gamma).\psi(A) \to \psi(\Gamma.A) \qquad \downarrow(\rho, a) = \langle \rho, a \rangle.$$

$\mathsf{Elem}(1, (A \to B)\rho)$ n'est pas isomorphe à $\mathsf{Elem}(1, A\rho) \to \mathsf{Elem}(1, B\rho)$ en général

On a un morphisme strict de projection $Gl(\psi) \to M$ et si M est le modèle initial on a une section, qui donne le résultat de canonicité.

Reformulation comme modèle de recollement

On rappelle que pour la paramétricité on considérait le morphisme strict pour le modèle des termes M

id : $M \to M$

Le morphisme que l'on considère ici n'est *pas* le morphisme strict initial $M \to$ Ens.

Ce qui intervient dans la preuve de canonicité est le morphisme strict initial $M \to Gl(\psi)$.

Le fait que la paramétricité *et* la propriété de canonicité peuvent être capturer de manière analogue par un recollement le long d'un morphisme exact à gauche a été observé par A. Kaposi
Gluing for Type Theory, S. Huber, A. Kaposi et Ch. Sattler (2019).

Structure de calculabilité

Comme pour la paramétricité, ceci peut être décrit syntaxiquement; cf.
Canonicity and normalisation for dependent type theory, Th.C. 2018

Si $B(x)$ famille de type sur A alors $B'(u,u')(v)$ est une famille d'ensembles pour v terme clos (modulo conversion) si u terme clos de type A et u' élément de l'ensemble $A'(u)$

$B'(u,u')$ dépend de u *et* de u'

Structure de calculabilité

$T = N$, $T'(t)$ est $\{k \mid t \text{ conv } \overline{k}\}$

$T = \Pi_{x:A} B$, $T'(t)$ est $\Pi_{u \in \mathsf{Elem}(A)} \Pi_{u' \in A'(u)} B'(u, u')(t\ u)$

$T = \Sigma_{x:A} B$, $T'(t)$ est $\Sigma_{u' \in A'(t.1)} B'(t.1, u')(t.2)$

$T = U_k$, $T'(X)$ is $\mathsf{Elem}(X) \to \mathcal{U}_k$

$\mathsf{Elem}(A)$ ensemble des termes clos de type A (modulo conversion)

Structure de calculabilité

$T = N$, $T'(t)$ is $\{k \mid t \text{ conv } \overline{k}\}$

$N'(t)$ est un *ensemble* qui *a priori* peut avoir plusieurs éléments : peut-être que 0 et 1 sont égaux dans le modèle des termes M

$U'_k(X) = \mathsf{Elem}(X) \to \mathcal{U}_k$ est un ensemble; en fait, c'est une version de l'ensemble des candidats de calculabilité sur le type X.

Structure de calculabilité

$(t\ u)'$ est $t'\ u\ u'$

$(\lambda_{x:A} t)'$ est $\lambda_{u \in \mathsf{Elem}(A)} \lambda_{u' \in A'(u)} t'(u, u')$

$(t.i)'$ est $t'.i$

$(u, v)'$ est (u', v')

On obtient un argument semblable à celui pour le système de Gödel, mais où la *propriété* de calculabilité est remplacée par une *structure* de calculabilité.

Structure de calculabilité

On a un traitement *uniforme* des termes et des types

Pour un type A on a une famille d'ensembles $A'(u)$ pour u terme clos de type A type A *et* on définit u' qui est un élément de type $A'(u)$

On peut montrer que si $t : T$ alors t' est un élément de $T'(t)$

Si t_0 conv $t_1 : T$ alors $t'_0 = t'_1$ dans l'ensemble $T'(t_0) = T'(t_1)$

En particulier si $t : N$ on a t' qui est un terme de la form $\overline{k}$ tel que t conv $\overline{k}$, ce qui est exactement la canonicité.

Structure de calculabilité

On construit un modèle où les contextes sont des couples Γ, Γ' avec Γ contexte syntaxique et $\Gamma'(\rho)$ famille d'ensemble pour $\rho : 1 \to \Gamma$

Si $\sigma : \Delta \to \Gamma$ on a $\sigma'\ \nu\ \nu' \in \Gamma'(\sigma\nu)$ pour $\nu : 1 \to \Delta$ et $\nu' \in \Delta'(\nu)$.

Si $\Gamma \vdash A$ on a une famille d'ensembles $A'\ \rho\ \rho'\ u$ pour $\rho : 1 \to \Gamma$ et $\rho' \in \Gamma'(\rho)$ et $\Gamma \vdash u : A$.

Si $\Gamma \vdash a : A$ on a $a'\ \rho\ \rho' \in A'\ \rho\ \rho'\ u$.

Si $\Gamma \vdash a_0 = a_1 : A$ on a $a'_0\ \rho\ \rho' = a'_1\ \rho\ \rho'$.

Recollement et canonicité

Cette preuve peut être vue comme un exemple de la technique de recollement.

$\psi : M \to$ Ens on a un morphisme *strict* $Gl(\psi) \to M$ et si on part du modèle initial, ce morphisme a une section $M \to Gl(\psi)$.

L'analogie entre preuve de canonicité et recollement pour 1-topos était clair dans les années 80s, e.g. *A Note on Freyd Cover and Friedman Slash*, A. Scedrov and P.J. Scott, 1982

L'idée est due à P. Freyd, cf.
Intuitionist type theory and the free topos, J. Lambek et P.J. Scott (1980).

Toutefois, comme nous allons le voir plus tard, ceci est plutôt lié à une autre forme du résultat de canonicité, la canonicité *homotopique*; en effet, la théorie étudiée par Lambek et Scott ont des axiomes d'extensionnalité.

Recollement et canonicité

Ce qui est remarquable dans cette preuve, c'est que ce qui est utilisé dans la méta-théorie est ⟪proche⟫ du système que l'on étudie. (Ce n'est pas le cas pour une preuve qui utilise une définition ⟪inductive-récursive⟫.) La méta-théorie dans laquelle on fait cette preuve n'a pas à être la théorie des ensembles.

C'est ce qui est implicite dans le papier de Martin-Löf 73
An Intuitionistic Theory of Types: Predicative Part

Il utilise comme méta-théorie une théorie des types.

On peut voir le résultat sur le recollement de modèle comme une justification de cette approche : ce résultat est valable pour des modèles arbitraire de la théorie des types.

Recollement et canonicité

Type des entiers

$$\frac{a : C(0) \qquad `g : \Pi_{n:N} C(n) \to C(S(n))}{f : \Pi_{n:N} C(n)} \qquad f\ 0 = a \qquad f\ S(n) = g\ n\ (f\ n)$$

Type des ordinaux

$$\frac{a : C(0) \qquad `g : \Pi_{n:Ord} C(n) \to C(S(n)) \qquad h : \Pi_{u:N\to Ord}(\Pi_{n:N} C(u\ n)) \to C(\mathsf{lim}(u))}{f : \Pi_{x:Ord} C(x)}$$

$$f\ 0 = a \qquad f\ S(x) = g\ n\ (f\ x) \qquad f\ \mathsf{lim}(u) = h\ u\ (\lambda_n f\ (u\ n))$$

Recollement et canonicité

$N'(t)$ est un *ensemble* qui peut avoir plusieurs éléments

Si on a un type des ordinaux, avec constructeur $0,\ S(x),\ \mathsf{lim}(u)$, un objet de type $Ord'(t)$ est un arbre bien fondé, qui peut être

-$0'$ si $t = 0$

-$S'(a, a')$ si $t = S(a)$ et a' élément de $Ord'(a)$

-$\mathsf{lim}'(u, u')$ si $t = \mathsf{lim}(u)$ et u' est une famille d'éléments $u'\ n\ n'$ dans $Ord'(u\ n)$ pour $n : \mathsf{Elem}(N)$ et n' dans $N'(n)$.

Recollement et canonicité

En général, si $\psi : M \to M_0$ morphisme exact à gauche et si M a un type des entiers N, il ne suffit pas en général d'avoir un type des entiers dans M_0. On a besoin d'une *famille inductive indexée*.

Le prédicat de calculabilité pour N est le prédicat défini par les lois

$$\frac{}{0' : N'(0)} \qquad \frac{n' : N'(n)}{S'(n, n') : N'(S(n))}$$

On a une situation analogue pour le type des ordinaux.

$$\frac{}{0' : Ord'(0)} \qquad \frac{n' : Ord'(n)}{S'(n, n') : Ord'(S(n))} \qquad \frac{u' : \forall_{n:\mathsf{Elem}(N)} N'(n) \to Ord'(u\ n)}{\mathsf{lim}'\ u\ u' : Ord'(\mathsf{lim}(u))}$$

Types de données

La méthode marche en général pour tout type de données. Par exemple, $(A + B)'(z)$ sera défini par les lois

$$\frac{a' : A'(a)}{\mathsf{inl}'(a, a') : (A + B)'(\mathsf{inl}(a))} \qquad \frac{b' : B'(b)}{\mathsf{inr}'(b, b') : (A + B)'(\mathsf{inr}(b))}$$

Pour le type $\mathsf{Id}\ A\ a\ b$, Martin-Löf 1973 construit le prédicat de calculabilité comme famille inductive indexée.

Types de données

En particulier pour le type $\bot$, sans constructeur : $\bot'\,(x)$ est l'ensemble vide.

Dans ce cas, le résultat de canonicité dit $\neg\mathsf{Elem}(\bot)$: le type $\mathsf{Elem}(\bot)$ est vide.

On obtient donc une preuve de *cohérence*.

Recollement et canonicité

Ceci explique aussi pourquoi Martin-Löf pouvait avoir une preuve de canonicité/normalisation pour un système avec type : type, car le résultat de recollement est possible pour un tel système.

$\mathsf{type}'(A)$ est $\mathsf{Elem}(A) \to \mathcal{U}$

La preuve de canonicité est alors valide.

Mais comme elle est conduite dans une méta-théorie contradictoire, elle ne conduit pas à la cohérence et on peut construire un terme de type $\bot$, et l'ensemble $\mathsf{Elem}(\bot)$ n'est pas vide.

Recollement et canonicité

L'argument ne peut pas être utilisé tel quel pour être sur de la cohérence du système. Martin-Löf (1979) présente une justification ⟪intuitive⟫ à la place d'une preuve de normalisation; et le livre 1984 discute de ce changement.

Il oppose *Metamathematical consistency* (programme de Hilbert) à *Simple minded consistency*

This method is the only one applicable when, like Hilbert, we give up the hope of a semantical justification of the axioms and rules of inference; it could be followed, with success, also for intuitionistic type theory, but, since we have been as meticulous about its semantics as about its syntax, we have no need of it. Instead, we convince ourselves directly of its consistency in the following simple minded way.

Tout élément doit avoir une forme canonique et le type $\bot$ n'a pas de forme canonique, donc il ne peut pas avoir d'élément. Le système NuPrl est construit en suivant cette approche : chaque loi est justifiée par une justification sémantique qui va garantir l'existence d'une forme canonique; cf.
A Non-Type-Theoretic Definition of Martin-Lof's Types S. Allen (1987).

Recollement et canonicité

Ceci est à rapprocher des commentaires de Gödel sur un essai (1969) de D. Scott de mettre au point un système de types dépendants

D. Scott venait de visiter de Bruijn et était inspiré par le système AUTOMATH; son approche est intéressante, car il essaie de mettre au point un système équationnel, et insiste sur le fait qu'il doit être *prédicatif*. Ceci va devenir le papier *Constructive Validity* (1970).

The point is that in a theory designed to give a satisfactory foundation (in the sense of complete evidence) of mathematics the primitive terms and axiomes are not *arbitrary; rather they must be* really *irreducible or, at any rate,* immediately *understandable, or* immediately *evident respectively.*
However, I think it is possible to remover the circularity by introducing a concept of "immediately evident," which might make your ":" decidable . . . The assumption that "immediately evident" is decidable is equivalent to the decidability of the proof predicate, and in both cases probably can be replaced by something weaker. Gödel, lettre à Scott (1969)

Recollement et canonicité

Le problème pour type : type est que l'on n'a même pas la canonicité, car un terme clos de type $\bot$ ne peut pas être convertible à un terme en forme canonique.

En 1973-79, Martin-Löf remplace type : type par une stratification des univers.

Mais ce système ne peut pas être utilisé en pratique : on ne veut pas dupliquer une preuve pour U_0 en une preuve sur U_1. (C'est ce que faisait Voevodsky à la main dans ses premiers essais de formalisation.)

On voudrait avoir des preuves ⟪polymorphes⟫ en les univers. Voevodsky propose un système dans *Universe polymorphic type system* (2014), que l'on a essayé de préciser dans *Type Theory with Explicit Universe Polymorphism*, (2024), M. Bezem, Th.C., P. Dybjer et M. Escardo.

Voir aussi *Universe polymorphism in Coq*, M. Sozeau et N. Tabareau (2014).

Canonicité homotopique

Si on ajoute l'univalence comme *un axiome/une constante*, on ne peut pas espérer avoir la canonicité.

Voevodsky a conjecturé (2010) le résultat suivant :

Si t *est un terme clos de type* N*, alors il existe une preuve close* e *et un terme de la forme* $\overline{k}$ *tel que* $e : \mathsf{Id}\ N\ t\ \overline{k}$

La preuve e peut utiliser l'univalence.

Cette conjecture a été résolue en 2019 par Ch. Sattler et Ch. Kapulkin, puis par une autre méthode par R. Bocquet (2023).

Ceci peut être vue comme une généralisation du résultat de P. Freyd présentée dans *Intuitionist type theory and the free topos*, J. Lambek et P.J. Scott (1980)

Canonicité homotopique

Dans les deux cas, on utilise

Théorème : *Si* M *et* M_0 *vérifient l'univalence,* et ψ envoie un type contractile sur un type contractile *alors la structure de modèle sur* $G(\psi)$ *vérifie l'univalence.*

On prend pour M_0 un modèle de l'univalence, et M le modèle des termes, avec une constante pour l'univalence.

Noter que si on prend pour M_0 le modèle simplicial, et $\psi(A)$ comme l'*ensemble* des termes clos de type A (vu comme ensemble simplicial), on ne vérifie *pas* cette condition.

Dans un cadre classique, étant donné un terme clos $t : N$, on énumère toutes les preuves et on est ⟪sûr⟫ d'obtenir au bout d'un moment une preuve d'un type de la forme $\mathsf{Id}\ N\ t\ \overline{k}$.

Pour une preuve constructive, on prend pour M_0 un modèle effectif, et on obtient un algorithme qui calcule $\overline{k}$ et cette preuve.

Importance et limitation du résultat de canonicité

Si on montre l'existence d'un entier vérifiant une propriété, ceci s'exprime comme de construire une terme de type $\Sigma_{x:N} P(x)$. Par le résultat de canonicité, on peut alors calculer un couple $\overline{k}, p$ avec p preuve de $P(\overline{k})$. Ceci n'est pas le cas dans un système qui utilise la logique classique.

Pour un exemple récent, cf.
Formalising and Computing the Fourth Homotopy Group of the 3-Sphere in Cubical Agda, Ljungström et Mörtberg, pour obtenir une preuve nouvelle par le calcul de $\pi_4(S_3) = \mathbb{Z}/2\mathbb{Z}$ (utilisant le travail de G. Brunerie 2014).

Toutefois, ce résultat ne suffit pas pour montrer la décidabilité de l'égalité dans le modèle initial, ou pour montrer que la β-réduction préserve les types, ou pour montrer que le jugement $a : A$ est décidable.

Pour cela, on n'a besoin de considérer des termes dans un contexte arbitraire, et pas seulement des termes clos. Une méthode est de raffiner la preuve de canonicité et d'utiliser un modèle de *préfaisceau*, et un modèle interne au modèle de recollement avec le modèle des termes.