Normalisation et relation ensembles/types

Thierry Coquand

Canonicité

Dans la leçon précédente, nous avons montré le

Théorème: *Si* $\vdash t : N$ *il existe un entier* k *tel que* $\vdash t = \overline{k} : N$

La preuve peut être faite sans introduire une relation de réduction sur les termes, et peut être vue comme la construction d'un modèle de recollement le long du morphisme exact à gauche $\psi : M \to \mathrm{Ens}$ avec M modèle initial et $\psi\Gamma = 1 \to \Gamma$.

L'essence de la preuve est que la projection $Gl(\psi) \to M$ qui est un morphisme strict, a pour section le morphisme initial $M \to Gl(\psi)$.

Canonicité

Cette preuve peut être vue comme une formulation de la preuve de canonicité de Tait/Shoenfield où l'on remplace la *propriété* de calculabilité par une *structure* de calculabilité

$A'\rho\rho' u$ famille d'ensemble pour $\Gamma \vdash A$ et $\rho : 1 \to \Gamma$ et $\rho' \in \Gamma'\rho$ et $\vdash u : A\rho$

$1'\langle\rangle = \{0\}$ $\qquad$ $(\Gamma.A)'\langle\rho, u\rangle = \Sigma_{\rho':\Gamma'\rho} A'\rho\rho' u$

$(\Pi_A B)'\rho\rho' w = \Pi_{u:\mathsf{Elem}(A\rho)} \Pi_{u':A'\rho\rho' u} B'(\rho, u)(\rho', u')(w\ u)$

$(\Sigma_A B)'\rho\rho' w = \Sigma_{u':A'\rho\rho'(w.1)} B'(\rho, w.1)(\rho', u')(w.2)$

$(\lambda b)'\rho\rho' u u' = b'(\rho, u)(\rho', u')$ $\qquad$ $(c\ a)'\rho\rho' = c'\rho\rho'(a\rho)(a'\rho\rho')$

Si $\Gamma \vdash a : A$ alors $a'\rho\rho' : A'\rho\rho'(a\rho)$.

Canonicité

Il est intéressant que l'idée pour ce modèle est essentiellement celle de la preuve de *Intuitionistic Type Theory: Predicative Part* (1973), introduite pour des raisons philosophiques. *On a remplacé la preuve de confluence par une sémantique, et le fait que la forme canonique peut être extraite de la sémantique.*

C'est aussi une traduction directe des intuitions de Gödel

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.

Canonicité

Le fait que l'on peut voir cet argument comme un modèle de recollement me semble répondre aux objections contre β, η-conversion discutées dans le papier
About Models for Intuitionistic Type Theories and the notion of Definitional Equality, Martin-Löf (1975)

Si on a $a = b : A$ alors on aura $a' = b' : A'(a)$ et la preuve de cette égalité ne fait intervenir dans la métathéorie que des β, η-conversions (sans faire appel à une égalité extensionnelle).

Canonicité

Cette preuve est formellement analogue à la transformation de paramétricité.

$A' : A \to U$ si $A : U$ et $a' : A'a$ si $a : A$

Il est remarquable que cette transformation marche avec $U : U$

$U'X = X \to U$ et on a $U' : U'U = U \to U$.

L'espoir de Martin-Löf était d'exploiter cette circularité pour représenter la notion de catégorie.

In a forthcoming paper I plan to show that, unlike set theory, the present theory of types is adequate for a straightforward formalization of category theory. This is so because the basic axiom that there is a type of types introduces precisely that kind of selfreference which is needed in order to construct, for instance, functor categories and the category of all categories. (1971)

Canonicité et univers

À cause du paradoxe de Girard, en 1973-79, Martin-Löf remplace $U : U$ par une stratification des univers. Ceci correspond à la stratification des propositions introduites par Russell.

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 et univers

En mathématiques, la notion d'univers n'est plus mentionnée explicitement, e.g.

Pour déduire (i) des deux autres propriétés, il faut supposer au depart que les catégories fibrés sont ⟪ *petites* ⟫ *relativement'a des choix d'univers que les auteurs n'ont pas crû bon de préciser au debut de cet ouvrage. Ils justifient ce déplorable laxisme (dont ce n'est pas le dernier exemple) par l'espoir que ledit ouvrage gagne en lisibilité ce qu'il perd peut-être en rigueur.* dans *Champs algébriques* G. Laumon, L. Moret-Bailly.

M. Shulman donne un exemple d'un argument incorrect si on oublie cette hiérarchie et conclut

Ever since this experience, I've been unconvinced by any argument along the lines of "type-in-type is okay as long as you don't do anything stupid". I'm willing to grant that what I did was stupid; the point is that it's possible to do stupid things by accident.

Canonicité et univers

Stacks fait appel à une technique pour ne pas utiliser un système plus fort que ZF *Contrary to what happens in [SGA4], we do not want to choose a universe. Instead we pick a "partial universe" (which is a suitably large set as in Sets, Section 3.5), and consider all schemes contained in this set.*

En théorie des types, la notion d'univers est utilisée pour représenter les propositions, surtout dans un cadre prédicatif.

Pour la preuve des 4 couleurs par exemple, l'utilisation d'un type des propositions permet d'éviter d'avoir à introduire un système d'univers. Même dans ce cas cependant, si on un type des propositions, comme dans *Rocq* ou *Lean*, la notion d'univers est utilisée pour former des types de structures.

La catégorie des groupes dans un univers donné dépend de l'univers, résultat formalisé dans Agda par M. Escardo, alors que la catégorie des groupes de présentation finie n'en dépend pas (avec univalence).

Normalisation

La propriété de canonicité ne suffit pas pour montrer que l'égalité est décidable dans le modèle initial M; pour cela on doit construire un nouveau modèle, le *modèle de normalisation*.

Pour montrer la normalisation, Tait (et Girard) considèrent des termes 0_A. À la place, on peut utiliser des termes *neutres*, décrits par la grammaire

$$k \ ::= \ x \mid k \ u$$

et *normaux* par la grammaire

$$u \ ::= \ k \mid \lambda_x.u.$$

Pour une présentation claire de cette preuve, voir par exemple
The Expressiveness of Simple and Second-Order Type Structures, S. Fortune, D. Leivant, M. O'Donnell (1983)

Normalisation

Voici par exemple comment la preuve de Tait peut être formulée : on définit $N'(t)$ comme le fait que t est normalisable et $(A \to B)'(t)$ comme $A'(u) \to B'(t\ u)$. La preuve que tous les termes sont normalisables reposent alors sur le Lemme suivant, montré par induction sur le type A

(1) $A'(t)$ entraine que t est normalisable

(2) si k est neutre de type A, alors $A'(k)$.

On montre alors que chaque terme est calculable, et donc normalisable par (1), par induction sur le terme.

Normalisation

Avec les types dépendants, il n'est pas si clair de comment définir les termes neutres indépendamment du contexte dans lequel ils sont valides. Martin-Löf 1972 suggère d'ajouter des *constantes* pour chaque type.

Techniquement, on doit itérer cette addition, car si ajoute des constantes, on crée de nouveaux types et on doit ajouter de nouvelles constantes; c'est semblable à ce qui se passe dans la preuve du théorème de complétude par Henkin.

Mais cette approche ne semble pas marcher avec une version purement équationnelle de la théorie des types, sans relation de réduction : comment prouver que l'addition de constantes est une extension conservative ?

Préfaisceaux

En 1988, j'ai suggéré de considérer le modèle de préfaisceaux sur les contextes avec *renommage* ou *inclusion* pour résoudre ce problème, cf.
An algorithm for testing conversion in type theory, Th. C., (1991) et
A proof of Strong Normalisation for the Theory of Constructions Using a Kripke-Like Interpretation, Th.C. et J. Gallier (1990).

La motivation de base était que la preuve de canonicité/normalisation est constructive et donc doit pouvoir s'interpréter dans un modèle de préfaisceau arbitraire.

Mais cette approche résout aussi (ce qui m'a surpris) le problème de conservativité. La connection avec la notion d'ensemble nominal est détaillée dans la note (non publiée)
Reduction-free normalisation for system F (1996), par Altenkirch, Hofmann et Streicher, cf. *Higher-Order Abstract Syntax for Coq* (1995), Despeyroux, Felty et Hirschowitz.

Préfaisceaux

Cette méthode d'utiliser les contextes comme ⟪mondes possibles⟫ pour un modèle de Kripke/préfaisceaux est aussi utilisée pour une autre preuve du résultat de complétude par Hindley *The completness for typing λ-terms* (1983) dans *The completeness of typing for context-semantics* Th. C. (2005). Cf. aussi *Constructing possible worlds*, A. Ranta (1991).

Soit M le modèle des termes; on peut construire une catégorie $\mathcal{C}$: les objets sont les contextes de M et on redéfinit $\Delta \to_S \Gamma$ comme un ensemble de *codes* pour les renommages ou pour les affaiblissements. Comme pour la théorie des ensembles nominaux, on a le choix entre inclusion et renommage. Cf. *Nominal Renaming Sets*, M.J. Gabbay et M. Hofmann (2008).

On choisit ici les renommages, mais le même argument marche avec affaiblissements.

Le même choix intervient pour les modèles effectifs de l'univalence.

Préfaisceaux

On peut donc former un modèle de la théorie des types sur cette catégorie $\mathcal{C}$

Un contexte sera un préfaisceau, i.e. une collection d'ensembles $F(\Delta)$ avec flèches de restriction $F(\Delta) \to F(\Delta_1),\ \rho \mapsto \rho\alpha$ pour $\alpha : \Delta_1 \to_S \Delta$

Un type sur F sera une collection d'ensembles $A(\Delta, \rho)$ pour $\rho : F(\Delta)$ avec flèches de restriction $A(\Delta, \rho) \to A(\Delta_1, \rho\alpha),\ u \mapsto u\alpha$

Un élément de A sera une section $a(\Delta, \rho) : A(\Delta, \rho)$ avec $a(\Delta, \rho)\alpha = a(\Delta_1, \rho\alpha)$.

On a une hiérarchie d'univers $\mathcal{U}_n$, avec $\mathcal{U}_n(\Delta)$ défini comme l'ensemble des familles d'ensembles U_n-petits $A(\Delta_1, \alpha : \Delta_1 \to_S \Delta)$ avec flèches de restrictions.

Préfaisceaux: définition de $\Delta_1 \to_S \Delta$

$v_0 \in \mathsf{Var}(\Gamma.A, B)$ si $B = A.\mathsf{p}$ avec $\overline{v_0} = \mathsf{q}$ dans $\mathsf{Elem}(\Gamma.A, B)$.

$v_{n+1} \in \mathsf{Var}(\Gamma.A, B)$ si $v_n \in \mathsf{Var}(\Gamma, B)$ avec $\overline{v_{n+1}} = \overline{v_n}\mathsf{p}$ dans $\mathsf{Elem}(\Gamma.A, B)$.

$\langle\rangle_S : \Gamma \to_S 1$ avec $\overline{\langle\rangle_S} = \langle\rangle : \Gamma \to 1$.

$\langle\alpha, u\rangle_S : \Delta \to_S \Gamma.A$ si $\alpha : \Delta \to_S \Gamma$ et $u \in \mathsf{Var}(\Delta, A\overline{\alpha})$ avec $\overline{\langle\alpha, u\rangle_S} = \langle\overline{\alpha}, \overline{u}\rangle : \Delta \to \Gamma.A$.

En particulier on peut définir $\mathsf{id}_S : \Gamma \to_S \Gamma$ et $\mathsf{p}_S : \Gamma.A \to_S \Gamma$ par induction sur Γ.

Préfaisceaux

La collection d'ensembles $\mathsf{Type}(\Delta)$ définit alors un préfaisceau en définissant $A\alpha = A\overline{\alpha}$

Dans ce modèle $\hat{\mathcal{C}}$ on a donc un type Type.

Sur ce type on a une famille de types $\mathsf{Elem}(A)$ pour $A : \mathsf{Type}$

On définit $\mathsf{Elem}(\Delta, A) \to \mathsf{Elem}(\Delta_1, A\alpha)$ par $a\alpha = a\overline{\alpha}$.

On a aussi la famille de types $\mathsf{Var}(A)$ pour $A : \mathsf{Type}$.

Mais on peut aussi définir un type Type_S des expressions syntaxiques en forme normale et $\mathsf{Elem}_S(A)$.

Préfaisceaux

Si $v : \mathsf{Var}(\Gamma, A)$ alors $v : \mathsf{Neut}(\Gamma, A)$

Si $X : \mathsf{Type}_S(\Gamma)$ et $Y : \mathsf{Type}_S(\Gamma.\overline{X})$ alors $\Pi_S\ X\ Y : \mathsf{Type}_S(\Gamma)$.

Si $X : \mathsf{Type}_S(\Gamma)$ et $Y : \mathsf{Type}_S(\Gamma.\overline{X})$ et $w : \mathsf{Neut}(\Gamma, \Pi_{\overline{X}}\overline{Y})$ et $u : \mathsf{Elem}_S(\Gamma, \overline{X})$ alors $\mathsf{app}_S\ X\ Y\ w\ u : \mathsf{Neut}(\Gamma, \overline{Y}[\overline{u}])$.

Si $X : \mathsf{Type}_S(\Gamma)$ et $Y : \mathsf{Type}_S(\Gamma.\overline{X})$ et $v : \mathsf{Elem}_S(\Gamma.\overline{X}, \overline{Y})$ alors $\lambda_S\ X\ Y\ v : \mathsf{Type}_S(\Gamma, \Pi_{\overline{X}}\overline{Y})$

Si $X : \mathsf{Neut}(\Gamma, U_n)$ et $t : \mathsf{Neut}(\Gamma, T_n\overline{X})$ alors $X : \mathsf{Elem}_S(\Gamma, U_n)$ et $t : \mathsf{Elem}_S(\Gamma, T_n\overline{X})$

$\overline{\Pi_S\ X\ Y} = \Pi\ \overline{X}\ \overline{Y}$ $\qquad$ $\overline{\lambda_S\ X\ Y\ v} = \lambda\ \overline{v}$ $\qquad$ $\overline{\mathsf{app}_S\ X\ Y\ w\ u} = \overline{w}\ \overline{u}$

Recollement avec un modèle de préfaisceau

Si $\alpha : \Delta \to_S \Gamma$ on a $\overline{\alpha} : \Delta \to \Gamma$ dans M. On a $\beta \mathsf{id}_S = \beta$ et $\mathsf{id}_S \alpha = \alpha$ et $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ avec des égalités *strictes* et décidables.

Pour A dans $\mathsf{Type}(\Gamma)$, l'ensemble $\mathsf{Elem}_S(\Gamma, A)$ est l'ensemble des *expressions syntaxiques* de type A en forme normale η-*longue*. L'ensemble $\mathsf{Type}_S(\Gamma)$ est l'ensemble des expressions syntaxiques pour les types en forme normale. On doit aussi définir simultanément $\mathsf{Neut}(\Gamma, A)$.

Chaque ensemble $\mathsf{Elem}_S(\Gamma, A), \mathsf{Type}_S(\Gamma), \mathsf{Var}(\Gamma, A), \mathsf{Neut}(\Gamma, A)$ a une égalité *décidable*, alors qu'il n'est pas clair a priori que l'égalité pour $\mathsf{Elem}(\Gamma, A)$ (et pour $\mathsf{Type}(\Gamma)$) soit décidable.

Comme on ne considère que des renommages on peut définir des opérations de substitutions $X\alpha : \mathsf{Type}_S(\Delta)$ si $\alpha : \Delta \to_S \Gamma$ si $X : \mathsf{Type}_S(\Gamma)$ avec $\overline{X\alpha} = \overline{X}\overline{\alpha}$. On peut donc voir $\mathsf{Type}, \mathsf{Type}_S$ comme des préfaisceaux sur $\mathcal{C}$ et $\mathsf{Elem}, \mathsf{Elem}_S, \mathsf{Var}, \mathsf{Neut}$ comme des préfaisceaux dépendants sur Type.

Catégorie syntaxique

À toute expression syntaxique e dans $\mathsf{Elem}_S(\Gamma, A)$, $\mathsf{Var}(\Gamma, A)$ ou $\mathsf{Neut}(\Gamma, A)$, on peut associer sa ⟪valeur⟫ $\overline{e}$ élément de $\mathsf{Elem}(\Gamma, A)$ dans le modèle initial.

Tout contexte Γ de M définit un préfaisceau $\psi\Gamma$ de $\hat{\mathcal{C}}$ avec $\psi\Gamma(\Delta) = \Delta \to \Gamma$ et flèche de restriction pour $\alpha : \Delta_1 \to_S \Delta$ qui est $\rho \mapsto \rho\overline{\alpha}$.

ψ définit alors un *morphisme exact à gauche* $M \to \hat{\mathcal{C}}$: si A dans $\mathsf{Type}(\Gamma)$ on définit ψA comme $\psi A(\Delta, \rho) = \mathsf{Elem}(\Delta, A\rho)$ et $\psi a(\Delta, \rho) = a\rho$.

Intuitivement, on permet l'introduction d'environnements ⟪ouverts⟫ $\Delta \to \Gamma$ au lieu de seulement considérer des environnements clos $1 \to \Gamma$, et, à chaque moment dans le raisonnement, on peut introduire de nouvelles variables.

Catégorie syntaxique

Dans ce modèle, on peut considérer Type, Type_S et on a des types dépendants $\mathsf{Elem}(A)$ et $\mathsf{Elem}_S(A)$ sur $A : \mathsf{Type}$.

Chaque $\mathsf{Type}_S(\Gamma)$ et chaque $\mathsf{Elem}_S(\Gamma, A)$ ont une égalité décidable. Si on prend les renommages comme morphisme de $\mathcal{C}$, en général, l'égalité sur Type_S n'est pas décidable de manière interne.

Théorème: *Les applications* $\mathsf{Type}_S \to \mathsf{Type},\ X \mapsto \overline{X}$ *et* $\mathsf{Elem}_S(A) \to \mathsf{Elem}(A),\ e \mapsto \overline{e}$ *sont des isomorphismes.*

Corollaire: *Dans le modèle des termes, les ensembles* $\mathsf{Type}(\Gamma)$ *et* $\mathsf{Elem}(\Gamma, A)$ *ont une égalité décidable.*

Catégorie syntaxique

Utiliser ce modèle résout le problème des types vides.

Proposition: Si $\vdash f : \Pi\ A\ B$ et $\vdash g : \Pi\ A\ B$ et $\forall_{x:Var(A)} f\ \overline{x} = g\ \overline{x}$ dans le modèle $\hat{\mathcal{C}}$, alors $\vdash f = g : \Pi\ A\ B$. De plus, f et g définissent des éléments globaux de $\mathsf{Elem}(\Pi\ A\ B)$ et $\vdash f = g : \Pi\ A\ B$ si, et seulement si, ces éléments sont égaux.

Ceci est valide même si $\mathsf{Elem}(1, A)$ est vide, e.g. pour $A = \Pi_U T$.

Le modèle de normalisation

Comme pour la preuve de canonicité, on peut définir une transformation ⟪syntaxique⟫ où l'on associe à chaque $\Gamma \vdash$ une famille d'ensembles/de préfaisceau $\Gamma'(\rho)$ sur $\rho : \psi(\Gamma)$ et à chaque $\Gamma \vdash A$ une famille $A'\rho\rho' u$ pour u dans $(\psi A)\rho$ et à chaque $\Gamma \vdash a : A$ une section $a'\rho\rho'$ dans $A'\rho\rho'((\psi a)\rho)$.

On va considérer une variation de cette transformation; cette variation doit être comprise comme une version de la preuve de la normalisation de Tait, avec une *structure* de calculabilité au lieu d'avoir un *prédicat* de calculabilité.

Cela donne un nouveau modèle de la théorie des types : le modèle de *normalisation*.

Catégorie syntaxique

Si A dans $\mathsf{Type}(\Gamma)$ et a dans $\mathsf{Elem}(\Gamma, A)$ on peut définir $\mathsf{Elem}_S(\Gamma, A)|a$ comme l'ensemble des expressions e dans $\mathsf{Elem}_S(\Gamma, A)$ telles que $\overline{e} = a$ dans $\mathsf{Elem}(\Gamma, A)$, et $\mathsf{Type}_S(\Gamma)|A$ comme l'ensemble des expressions X dans Type_S telles que $\overline{X} = A$.

On travaille avec le modèle de la théorie des types de préfaisceaux sur $\mathcal{C}$.

Dans cette variation, à chaque type A du modèle des termes on associe un *quadruplet* $A', A_0, \alpha_A, \beta_A$.

Le modèle de normalisation

Dans cette variation, on définit en plus de $A'\rho\rho'$

(0) $A_0\rho\rho'$ dans $\mathsf{Type}_S|(\psi A)\rho$

(1) $\alpha_A\rho\rho'$ dans $\Pi_{u:(\psi A)\rho}A'\rho\rho' u \to \mathsf{Elem}_S((\psi A)\rho)|u$

(2) $\beta_A\rho\rho'$ dans $\Pi_{k:\mathsf{Neut}((\psi A)\rho)}A'\rho\rho'\overline{k}$.

Les conditions (1) et (2) correspondent au lemme de la preuve de normalisation de Tait

(1) Tout terme calculable est normalisable

(2) Tout terme neutre (pour Tait, seulement 0_A) est calculable

Si $a : \mathsf{Elem}(\Gamma, A)$ alors on a $a'\rho\rho' : A'\rho\rho'((\psi a)\rho)$.

Le modèle de normalisation

On définit $U'_n X$ pour $X : \mathsf{Elem}(U_n)$ comme l'ensemble des quadruplets $(X', X_0, \alpha_X, \beta_X)$ avec $X_0 : \mathsf{Elem}_S(U_n)|X$ et $X' : \mathsf{Elem}(T_n X) \to \mathcal{U}_n$ et $\alpha_X u u' : \mathsf{Elem}_S(T_n X)|u$ pour $u : \mathsf{Elem}(T_n X)$ et $u' : X' u$ et $\beta_X k : X' \overline{k}$ pour $k : \mathsf{Neut}(T_n X)$.

On peut alors définir $\alpha_{U_n} X(X', X_0, \alpha_X, \beta_X) = X_0$ et $\beta_{U_n} K = (K', K, \alpha_K, \beta_K)$ avec $K' u = \mathsf{Neut}(K)|u$ et $\alpha_K u u' = u'$ et $\beta_K k = k$.

On peut voir l'essence de cette définition dans la preuve de normalisation pour type : type dans *Intuitionistic Type Theory*, Martin-Löf, (1971).

Cette définition provient de *Canonicity and Normalisation for Dependent Type Theory*, Th.C. (2018).

Le modèle de normalisation

De manière ⟪externe⟫ chaque Γ' est une famille d'ensemble $\Gamma'(\Delta, \rho)$ pour $\rho : \Delta \to \Gamma$ avec flèche de restriction $\Gamma'(\Delta, \rho) \to \Gamma'(\Delta_1, \rho\overline{\alpha})$ pour $\alpha : \Delta_1 \to_S \Delta$.

Par exemple, de manière externe, A_0 est une fonction qui associe à $\rho : \Delta \to \Gamma$ et ρ' dans $\Gamma'(\Delta, \rho)$ un élément $A_0(\Delta, \rho)\rho'$ de $\mathsf{Type}_S(\Delta)$.

Et ceci est naturel en Δ : si on a $\alpha : \Delta_1 \to_S \Delta$ alors $A_0(\Delta, \rho)\rho'\alpha = A_0(\Delta_1, \rho\overline{\alpha})(\rho'\alpha)$

Le modèle de normalisation

On construit alors $\mathsf{id}' : \Gamma'(\Gamma, \mathsf{id})$ par induction sur Γ, et on peut définir une fonction

$$\mathsf{nf} : \mathsf{Elem}(\Gamma, A) \to \mathsf{Elem}_S(\Gamma, A)$$

par $\mathsf{nf}(a) = \alpha_A(\Gamma, \mathsf{id})\mathsf{id}'(a'(\Gamma, \mathsf{id})\mathsf{id}')$. Cette fonction vérifie $a = \overline{\mathsf{nf}(a)}$.

On définit aussi $\mathsf{nf}(A) : \mathsf{Type}_S(\Gamma)$ par $\mathsf{nf}(A) = A_0(\Gamma, \mathsf{id})\mathsf{id}'$, qui vérifie

$A = \overline{\mathsf{nf}(A)}$

$\mathsf{nf}(\Pi\ A\ B) = \Pi_S\ \mathsf{nf}(A)\ \mathsf{nf}(B)$.

Le modèle de normalisation

On obtient alors que chaque $\mathsf{Elem}(\Gamma, A)$ a une égalité décidable, car nf est injective :

-si $a = b : \mathsf{Elem}(\Gamma, A)$ alors $\mathsf{nf}(a) = \mathsf{nf}(b) : \mathsf{Elem}_S(\Gamma, A)$.

-réciproquement, si $\mathsf{nf}(a) = \mathsf{nf}(b) : \mathsf{Elem}_S(\Gamma, A)$ alors

$$a = \overline{\mathsf{nf}(a)} = \overline{\mathsf{nf}(b)} = b$$

En fait, on peut montrer le résultat suivant.

Théorème: nf *est un inverse de la fonction* $\mathsf{Elem}_S(\Gamma, A) \to \mathsf{Elem}(\Gamma, A),\ e \mapsto \overline{e}$.

Le modèle de normalisation

Voici l'argument de P. Hancock dans *Intuitionistic Type Theory: Predicative Part* (1973) pour montrer que Π est une opération injective.

Si $\Pi\ A\ B = \Pi\ A_1\ B_1$ alors $\mathsf{nf}(\Pi\ A\ B) = \mathsf{nf}(\Pi\ A_1\ B_1)$ et donc

$\Pi_S\ \mathsf{nf}(A)\ \mathsf{nf}(B) = \Pi_S\ \mathsf{nf}(A_1)\ \mathsf{nf}(B_1)$

ceci entraine $A = \overline{\mathsf{nf}(A)} = \overline{\mathsf{nf}(A_1)} = A_1$ et $B = \overline{\mathsf{nf}(B)} = \overline{\mathsf{nf}(B_1)} = B_1$.

Also, because of the type restrictions on the rules of conversion and reduction, the method for proving the Church-Rosser property developped in combinatory logic apparently no longer works. Instead, the uniqueness of normal form and the Church-Rosser propery are proved, almost without effort, as corollaries to the construction of the term model by a new method, due to Peter Hancock. Martin-Löf (1973)

Le modèle de normalisation

Ce modèle est à rapprocher des implémentations d'évaluation des termes ouverts qui reposent sur une machine avec environnements, qui remontent aux travaux de Landin *A Mechanical Evaluation of Expressions* (1964), telle que celle décrite dans
A Compiled Implementation of Strong Reduction, B. Grégoire et X. Leroy (2002).

C'était aussi remarqué dans le papier *Reduction-free normalisation for system F* (1996)

The ultimate goal would be to derive implementations of these systems which would be more efficient than the existing ones because the reduction-free normalisation algorithm can employ the interpreter of the underlying functional programming language,

Le modèle de normalisation

Ceci est à rapprocher du commentaire de Martin-Löf (1972) après sa preuve de normalisation.

A number theoretic function which can be constructed in the theory of types is mechanically computable

Of course, the fact that there is a not necessarily mechanical procedure for computing every function in the present theory of types does not require any proof at all for us, intelligent beings, who can understand the meaning of the types and the terms to recognize that the axioms and rules of inference of the theory are consonant with the intuitionistic notion of function according to which a function is the same as a rule or method.

Variations

Cette méthode d'utiliser des *structures* de calculabilité est modulaire.

Extension avec types de données et définitions inductives indexées.

Extension avec un type des propositions ⟪proof-irrelevant⟫, qui est la théorie de *Lean Reduction Free Normalisation for a proof-irrelevant type of propositions*, Th. C. (2021)

Extension avec un type des propositions ⟪proof-relevant⟫
Normalization for Type Theory with an Impredicative Universe, Z. Xie (2024)

Version pour théorie des types avec un interval
Normalization for Cubical Type Theory, J. Sterling et C. Angiuli (2021)

First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory. J. Sterling (2022).

Variations

B. Werner *On the strength of proof-irrelevant type theories* (2008) considère une égalité 《proof-irrelevant》 Id $A\ a\ b : o$ avec une loi

$$J : \Pi_{A:U}\Pi_{P:A\to U}\Pi_{a\ b:A} P\ a \to \mathsf{Id}\ A\ a\ b \to P\ b \qquad J\ A\ P\ a\ a\ x\ e = x$$

ce qui est à rapprocher de l'opération δ de Church avec $\delta\ M\ M = 1$ et

$$\delta\ M\ N = 0$$

si M et N ont une forme normale distincte.

Une loi similaire est ajoutée dans le système Lean. On peut montrer la canonicité pour cette extension, mais le système n'est *pas* normalisable, cf.
Failure of Normalization in Impredicative Type Theory with proof-irrelevant propositional equality, A. Abel et Th.C. (2019).

Le paradoxe des arbres de de Bruijn

de Bruijn a formulé la version concrète suivante du paradoxe de Russell :

qu'est-ce qu'un arbre ? C'est construit à partir d'une famille d'arbre $(t_i,\ i \in I)$, que l'on rassemble avec une nouvelle racine, on obtient alors un arbre $t = \mathsf{sup}(t_i,\ i \in I)$

Disons qu'un arbre t est *bon* s'il est différent de chacun de ses sous-arbres immédiats t_i. Alors l'arbre obtenu en rassemblant la collection des arbres qui sont bons doit être bon, et on a une contradiction.

Il est direct de représenter ce paradoxe dans un système avec un type de tous les types U, et Π, Σ et un type de donnée V avec pour seul constructeur $\mathsf{sup} : \Pi_{X:U}(X \to V) \to V$

Pour représenter ce paradoxe, on n'a pas besoin de la récursion sur ce type V, on a juste besoin d'un opérateur d'analyse de cas : $\mathsf{sup}\ X\ f = X$ et $\mathsf{sup}\ X\ f = f$

Le paradoxe des arbres de de Bruijn

On a en fait deux versions

(1) si on ne considère que des arbres bien fondés, i.e. on ajoute un principe d'induction/de récursion, alors on peut montrer que tout arbre est différent de ses sous-arbres immédiats par induction, et on a une contradiction en considérant l'arbre sup V id; c'est comme le paradoxe de Burali-Forti

(2) sans principe d'induction, juste avec définition par cas, on peut former le type des bons arbres $B = \Sigma_{t:V} G(t)$ et on a une contradiction en considérant l'arbre sup B π_1; c'est comme le paradoxe de Russell.

En représentant ce paradoxe en théorie des types, je me suis rendu compte que ce type V des arbres était un cas particulier du type W A B, introduit par Martin-Löf en 1979. Mais en fait, ce type était précisément introduit pour représenter les *ensembles* comme des *arbres* en théorie des types.

Le paradoxe des arbres de de Bruijn

Ce type V a été introduit par G. Leversha, étudiant de P. Aczel, pour indexer le process de représenter les ensembles dans sa thèse *Formal System for Constructive Mathematics* (1976). C'est une généralisation naturelle du type des ordinaux, introduits par Hilbert en 1925, et en théorie des types par Martin-Löf (1971); c'est le type des arbres bien fondés avec un branchement arbitraire dans un univers donné U.

P. Aczel a remarqué que l'on pouvait *directement* utiliser ce type des arbres pour représenter des ensembles dans *The Type Theoretic Interpretation of Constructive Set Theory* (1978). Il écrit *Unfortunately it was not realised at that time that the type itself gave a constructive notion of set. Instead it was only used as an appropriate constructive notion of ordinal to use in indexing the stages of Leversha's construction.*

Les ensembles en théorie des types

La relation d'égalité $\|\alpha = \beta\|$ pour les ensembles est alors la relation de bissimulation

$$\|\mathsf{sup}\ A\ f = \mathsf{sup}\ B\ g\| = (\Pi_{a:A}\Sigma_{b:B}\,\|f\ a = g\ b\|) \times (\Pi_{b:B}\Sigma_{a:A}\,\|f\ a = g\ b\|)$$

Ceci capture exactement l'intuition que dans la notion $\{f(a) \mid a : A\}$ l'ordre et la duplication ne doivent pas jouer un rôle

On définit alors $\|x\ \epsilon\ \mathsf{sup}\ A\ f\|$ par $\Sigma_{a:A}\,\|x = f\ a\|$

L'ensemble vide est $\mathsf{sup}\ \bot\ f$ et $\{a_0, \ldots, a_{k-1}\}$ est $\mathsf{sup}\ N_k\ f$ avec $f\ i = a_i$.

$(\mathsf{sup}\ A\ f) \cup (\mathsf{sup}\ B\ g)$ peut être défini comme $\mathsf{sup}\ (A+B)\ h$ avec $h\ (\mathsf{inl}\ a) = f\ a$ et $h\ (\mathsf{inr}\ b) = g\ b$.

Voici comment représenter l'ensemble ω

$$\Delta(0) = \emptyset \qquad \Delta(n+1) = \Delta(n) \cup \{\Delta(n)\} \qquad \omega = \{\Delta(n) \mid n : N\}$$

Les ensembles en théorie des types

L'interprétation des formules de la théorie des ensembles s'étend de la manière suivante

$$\|\varphi \to \psi\| = \|\varphi\| \to \|\psi\| \qquad \|\varphi \wedge \psi\| = \|\varphi\| \times \|\psi\| \qquad \|\varphi \vee \psi\| = \|\varphi\| + \|\psi\|$$

$$\|\forall_{x \epsilon \alpha} \varphi(x)\| = \Pi_{x:\overline{\alpha}} \|\varphi(\widetilde{\alpha}(x))\|$$

$$\|\exists_{x \epsilon \alpha} \varphi(x)\| = \Sigma_{x:\overline{\alpha}} \|\varphi(\widetilde{\alpha}(x))\|$$

$$\|\forall_x \varphi(x)\| = \Pi_{x:V} \|\varphi(x)\|$$

$$\|\exists_x \varphi(x)\| = \Sigma_{x:V} \|\varphi(x)\|$$

Les ensembles en théorie des types

P. Aczel peut alors vérifier les axiomes d'une version constructive CZF de ZF, qui est strictement plus faible que ZF (en fait, plus faible que PA_2), mais telle que CZF + EM est aussi forte que ZF.

W. Veldman, dans son rapport sur le papier de P. Aczel, remarque qu'il y a une analogie claire avec les modèles Booléens, où l'on définit les valeurs de vérité $\|\alpha \in \beta\|$ et $\|\alpha = \beta\|$ de manière similaire. Ceci montre aussi le caractère spécial de ces modèles (ou des modèles de faisceaux), indexés par les ordinaux.

Les ensembles en théorie des types

Pour cette interprétation, on n'a pas besoin du type Id $A\ a\ b$ dans la théorie des types.

Les ensembles sont interprétés par les éléments du type V, muni de la relation $\simeq$, que l'on peut montrer être une relation d'équivalence.

Je trouve que cette interprétation montre que la notion de *types* est plus primitive que la notion d'*ensembles*. La notion d'ensemble, telle que comprise dans ZF, est une notion complexe, qui fait intervenir cette relation $\simeq$ de bissimulation.

Cette interprétation justifie naturellement l'axiome de *fondation*, puisqu'un ensemble est interprété comme un arbre bien fondé.

Les ensembles en théorie des types

Comme je l'ai rappelé dans ma leçon inaugurale, les ensembles peuvent être obtenus par itération de l'opération $V_{\alpha+1} = Pow(V_\alpha) = V_\alpha \to o$ le long des ordinaux.

En théorie des types, on peut redéfinir $Pow(A)$ comme $\Sigma_{X:U} A^X$.

Ceci donne une autre motivation pour considerer $V = W\ U\ T$ qui est un point fixe de $V \simeq \Sigma_{X:U} V^X$.

La thèse de G. Leversha représentait cette itération en utilisant le type des arbres (et introduisait pour la première fois ce type) comme type des ordinaux.

Les ensembles en théorie des types avec un type des propositions

En présence d'un type de proposition o, cette interprétation a été étudiée par B. Werner *Sets in Types, Types in Sets* (1997).

En particulier, il a pu montrer que si l'on ajoute à la théorie des types une forme de l'axiome du choix, cette interprétation justifie les axiomes de ZF.

Ceci s'applique au système *Lean*, qui est ainsi prouvablement équivalent à ZFC + hiérarchie de cardinaux inaccessibles.

M. Carneiro utilise une telle interprétation pour traduire une preuve du théorème de Dirichlet (sur les nombres premiers dans une progression arithmétique) du système *MetaMath* (avec les axiomes de ZFC) dans *Lean*. Il obtient alors une preuve d'abord pour la notion d'entiers codée en théorie des ensembles, mais peut en déduire la version avec le type des entiers N.

Les ensembles en théorie des types avec un type des propositions

A. Miquel a remarqué que l'on peut traduire directement les ensembles comme des graphes pointés, sans introduire un type des arbres bien fondés.

Un ensemble est interprété par un triplet A, R, a avec $R : A \to A \to o$ et $a : A$.

On peut définir imprédicativement une relation de bissimulation, comme *plus grand* point fixe.

$(A, R, a) \simeq (B, S, b)$ est la conjonction de

(1) $\Pi_{x:A} R\ a\ x \to \exists_{y:B} S\ b\ y \times (A, R, x) \simeq (B, S, y)$

(2) $\Pi_{y:B} S\ b\ y \to \exists_{x:A} R\ a\ x \times (A, R, x) \simeq (B, S, y)$

Les ensembles en théorie des types avec un type des propositions

$\simeq$ est définie comme la réunion de toutes les relations $\sim$ telles que $(A, R, a) \sim (B, S, b)$ entraine

(1) $\Pi_{x:A} R\ a\ x \to \exists_{y:B} S\ b\ y \times (A, R, x) \sim (B, S, y)$

(2) $\Pi_{y:B} S\ b\ y \to \exists_{x:A} R\ a\ x \times (A, R, x) \sim (B, S, y)$

On obtient ainsi un modèle d'une théorie des ensembles, *non nécessairement bien fondés*, sans utiliser de types de données comme le type des arbres, en interprétant $(A, R, a)\ \epsilon\ (B, S, b)$ comme $\exists_{y:B} S\ y\ b \times (A, R, a) \simeq (B, S, y)$.

Les ensembles en théorie des types avec un type des propositions

Si on a $U : U$, on peut construire de cette manière un modèle d'une théorie des ensembles avec un ensemble de tous les ensembles.

Ceci permet de construire une preuve de $\bot$ dans ce système qui est une traduction du paradoxe de Russell, cf. *Le calcul des constructions implicite: syntaxe et sémantique*, A. Miquel (2001).

On peut définir un type ⟪universel⟫ des graphes, qui correspond à un ensemble de tous les ensembles

$$(\Pi_{X:U} X \to (X \to X \to o) \to o) \to o$$

qui simplifie le type existentiel

$$\exists_{X:U} X \times (X \to X \to U) = \Pi_{Y:U}(\Pi_{X:U} X \to (X \to X \to o) \to Y) \to Y$$

Les ensembles en théorie des types avec un type des propositions

A. Miquel donne une autre application de cette interprétation

Théorème: *La théorie des ensembles de Zermelo peut être interprétée dans une théorie des types avec un type des propositions et 3 univers et produits dépendants*

J'insiste sur le fait que cette théorie des types n'a pas besoin de types Σ ou Id ou types de données.

La preuve se trouve dans sa thèse, et une version améliorée est dans le papier
λ-*Z: Zermelo's Set Theory as a PTS with 4 Sorts*, (2004).

Une version pour la théorie des topos avec des relations bien fondées se trouvent dans
Categorical set theory: A characterisation of the category of sets, G. Osius (1974).

Les ensembles en théorie des types avec l'univalence

La deuxième partie du cours va porter sur l'axiome d'univalence. Je vais expliquer ce que devient l'interprétation des ensembles comme arbres bien fondés dans ce cadre.

Voevodsky introduit une stratification des types suivant la complexité de leur égalité.

En particulier isContr $A = \Sigma_{a:A}\Pi_{x:A}\mathsf{Id}\ A\ a\ x$ et isProp $A = \Pi_{a\ b:A}\mathsf{isContr}\ (\mathsf{Id}\ A\ a\ b)$ et isSet $A = \Pi_{a\ b:A}\mathsf{isProp}\ (\mathsf{Id}\ A\ a\ b)$ et plus généralement la notion de n-type pour chaque n.

Le type des ⟪ensembles⟫ $S = \Sigma_{X:U}\mathsf{isSet}\ X$ n'est pas un ensemble (0-type), mais un groupoïde (1-type) et on a $\Pi_{a\ b:S}\mathsf{isSet}\ (\mathsf{Id}\ S\ a\ b)$.

Le type U n'est pas un n-type pour chaque n, et le type des arbres bien fondés V aussi n'est pas un n-type pour chaque n.

On peut introduire aussi le type des propositions $\mathsf{hProp}(U) = \Sigma_{X:U}\mathsf{isProp}\ X$ qui est un *ensemble*.

Les ensembles en théorie des types avec l'univalence

Dans ce cadre on a une équivalence remarquable pour chaque type A

$$U^A \simeq \Sigma_{X:U} A^X,$$

et on a une notion naturelle de *plongement* $f : A \to B$ qui est que chaque fibre de f doit être une proposition. On note $A \hookrightarrow B$ le type des plongements $\Sigma_{f:A\to B}\mathsf{isEmb}\ f$. On a alors une équivalence

$$\mathsf{hProp}(U)^A \simeq \Sigma_{X:U} X \hookrightarrow A$$

qui est le type des prédicats sur A.

Les ensembles en théorie des types avec l'univalence

On a alors pour x et y dans V l'équivalence $\mathsf{Id}\ V\ x\ y \simeq \Pi_{z:V} z \epsilon x \simeq z \epsilon y$

On définit alors la notion d'ensembles ⟪itératifs⟫ $\mathsf{isItSet}\ (\sup\ A\ f) = \mathsf{isEmb}\ f \times \Pi_{a:A} \mathsf{isItSet}\ (f\ a)$ et on obtient $V^0 = \Sigma_{x:V} \mathsf{isItSet}\ x$.

Ce type V^0 est un *ensemble*, au sens ⟪homotopique⟫ de Voevodsky, et $x\ \epsilon\ \sup\ A\ f = \Sigma_{a:A} \mathsf{Id}\ V^0\ x\ (f\ a)$ est une *proposition*.

De plus, $\mathsf{Id}\ V^0\ x\ y$ est équivalent à dire que x et y sont *bissimilaires*.

Ce raffinement est dû à H. Gylterud *From multisets to sets in Homotopy Type Theory* (2018).

Cette représentation est utilisée pour formaliser le modèle ensembliste de la théorie des types *The Category of Iterative Sets in Homotopy Type Theory and Univalent Foundations*, D. Gratzer, H. Gylterud, A. Mörtberg, E. Stenholm (2024). De manière intéressante, la vérification de la sémantique de l'opération $\Sigma_A B$ semble non triviale techniquement, et n'est pas complète.

Les ensembles en théorie des types avec l'univalence

Ceci est un bon exemple de ce qui se passe avec l'axiome d'univalence.

Chaque type A est muni d'une relation d'égalité $\mathsf{Id}\ A\ a\ b$.

Avec l'axiome d'univalence, cette égalité va devenir l'égalité ⟪attendue⟫ sur le type A, e.g. sur les fonctions ce sera l'égalité extensionnelle, sur les structures algébriques, ce sera l'isomorphisme, ...

Ici, l'égalité sur V^0 est automatiquement la bissimulation.