Introduction à la théorie homotopique des types

Thierry Coquand

GDR Informatique Mathématique, Journées nationales, 29 Janvier 2014

Contexte de ce travail

Un but est de mettre au point un bon système formel pour représenter et vérifier les preuves mathématiques sur ordinateur

Un tel système sera aussi utile pour spécifier/vérifier des programmes

Contexte de ce travail

Il est naturel que ce genre de travaux ait des liens forts avec les travaux sur les fondements des mathématiques

Jusqu'à récemment, cette interaction allait seulement dans un sens

Je vais présenter *une vision nouvelle sur les fondements des mathématiques* suggérée par la recherche d'un bon système pour représenter les preuves

En particulier un éclairage nouveau sur une des notions de base des mathématiques : la notion d'*égalité*

Fondements des mathématiques

1908 Zermelo *Untersuchungen über die Grundlagen der Mengenlehre*

ZFC système de référence pour les mathématiciens

→ Système MIZAR

1908 Russell *Mathematical Logic as Based on the Theory of Types*

Surtout connu par certains informaticiens et logiciens

pratiquement inconnu parmi les mathématiciens

(quelques exceptions : A. Turing, T. Hales, V. Voevodsky)

→ Systèmes HOL, Isabelle, Idris, Epigram, Coq, Agda

Fondements des mathématiques

1940 Church *A Formulation of the Simple Theory of Types*

Utilise la notation du λ-calcul

Notation $\lambda x.x^2 + 1$ pour la fonction $f(x) = x^2 + 1$

Cette notation est à la base de la programmation fonctionnelle (LISP, Scheme, Miranda, O'Caml, Haskell) et de la sémantique dénotationelle

Les travaux de P.J. Landin (1964) montrent que ce calcul est très approprié pour décrire les langages de programmation

Fondements des mathématiques et informatique

En théorie des ensembles, une fonction est esentiellement un *graphe fonctionnel*, objet "statique"

La notation du λ-calcul suggère un moyen d'incorporer les calculs dans les raisonnements

Un exemple est donné par la représentation de la preuve de décidabilité de la théorie monadique du second ordre sur les mots finis (MSO) par D. Traytel et T. Nipkow dans le système Isabelle

On obtient un algorithme prouvé correct que l'on peut faire tourner, et une présentation élégante de cette preuve qui utilise des notions de programmation fonctionnelle

Fondements des mathématiques et informatique

Autre contribution importante du système de Church : une notion de type extrêmement simple et naturel

Un type o qui représente le type des "propositions" (l'introduction du type des booleens figure comme un des aspects nouveaux et importants du langage Algol 60 dans une note d'E.W. Dijkstra)

Un type ι qui représente un type d'"individus"

Un type de fonctions $\alpha \to \beta$, ou $(\beta)\alpha$ dans la notation de Church

Par exemple $\lambda x.x : \alpha \to \alpha$

Un tel système de type est utilisé dans O'Caml et Haskell

Fonctions en mathématique et en informatique

On obtient deux notions de fonction, comme *graphe fonctionnel* ou comme *programme*

En théorie des ensembles une fonction est essentiellement un graphe fonctionnel

Comment relier ces deux notions de fonction ?

Tout graphe fonctionnel doit déterminer un programme

Church introduit un opérateur $\iota x.\varphi$ et l'"axiome de description"

Si $\exists! x : \alpha.\varphi$ alors $\varphi(\iota x.\varphi)$

Fonctions en mathématique et en informatique

On a alors l'"axiome du choix unique"

$\forall x.\exists! y.\psi(x,y) \rightarrow \exists f.\forall x.\psi(x,f(x))$

en définissant $f(x) = \iota y.\psi(x,y)$

(Cet axiome est impliqué par l'opérateur $\epsilon x.\varphi$ de Hilbert qui vérifie

si $\exists x : \alpha.\varphi$ alors $\varphi(\epsilon x.\varphi)$)

Utiliser $\exists! x : \alpha.\varphi$ suppose que l'on a une notion d'égalité sur le type α

Égalité en mathématique

Le premier axiome de la théorie des ensembles est l'axiome d'extensionalité qui dit que deux ensembles qui ont les mêmes éléments sont égaux

Dans le sytème de Church on a deux formes de l'axiome d'extensionalité

10^o deux propositions équivalentes sont égales

$$(\varphi \equiv \psi) \rightarrow \varphi = \psi$$

$10^{\alpha\beta}$ deux fonctions égales en chaque point sont égales

$$(\forall x^{\alpha}.f(x) = g(x)) \rightarrow f = g$$

Égalité en mathématique

On peut justifier ces axiomes en *redéfinissant* l'égalité par induction sur le type et en montrant que l'on obtient bien une relation satisfaisant les lois de l'égalité

C'est possible aussi en théorie des ensembles et c'est ce que l'on doit faire par exemple quand on construit des modèles par la technique du "forcing"

Peut-on justifier/expliquer l'opérateur $\iota x.\varphi$?

Un système où “tout est justifié”

1908 Russell *Mathematical Logic as Based on the Theory of Types*

1940 Church *A Formulation of the Simple Theory of Types*

1973 Martin-Löf *An Intuitionistic Theory of Types: Predicative Part*

(Curry, de Bruijn, Howard, Tait, Scott, Martin-Löf, Girard, . . .)

Idée nouvelle : penser les *propositions* comme des *types*

Un système où “tout est justifié”

Ceci “explique” certaines règles logiques, par exemple

$\lambda x.t$ est de type $A \to B$ si t est de type B où x est de type A

$f(u)$ est de type B si f est de type $A \to B$ et u de type A

Un système où "tout est justifié"

de Bruijn (1967) reconnait que cette vue des propositions comme types est très appropriée pour representer les preuves mathématiques sur ordinateur (système AUTOMATH)

Ceci réduit le problème de la vérification des preuves au problème de la vérification des types

Par raison d'uniformité, de Bruijn introduit la notion de *type dépendant* et l'opération de produit $(\Pi x : A)B$ qui est le type des opérations f telles que $f(a)$ est de type $B(x/a)$ si a est de type A

Par dualité, il est naturel d'introduire le type $(\Sigma x : A)B$ qui est le type des paires (a, b) où a est de type A et b de type $B(x/a)$

Un système où “tout est justifié”

Il est aussi naturel d'introduire un type “somme disjointe” $A + B$

Les élément seront de la forme $\mathsf{inl}(a)$ avec a dans A ou $\mathsf{inr}(b)$ avec b dans B

On peut définir des fonctions sur $A + B$ par analyse de cas

Un système où "tout est justifié"

Le système de type est plus complexe et il contient une notion de type dépendant $B(x),\ x : A$

Le rapprochement des mathématiques et de l'informatique est encore plus prononcé puisque, maintenant, même les *preuves* sont représentées comme des *programmes*

Un exemple est fourni par la preuve formelle du théorème des 4 couleurs par G. Gonthier et B. Werner

Une preuve *nouvelle* de ce théorème directement inspirée par des notions informatiques

Un système où “tout est justifié”

Le type des propositions o doit être considéré comme un “univers” U

Univers : un type dont les éléments sont des types

On doit avoir une égalité sur le type U

Quand est-ce que deux types sont égaux ?

De nouvelles lois pour l'égalité

P. Martin-Löf introduit (1973) une notion d'égalité primitive en théorie des types dépendants

La "proposition" exprimant que deux éléments a_0 et a_1 d'un type A sont égaux devient une famille de type $\mathsf{Id}_A(a_0, a_1)$

Quelles sont les lois de l'égalité dans ce système ?

Tout élément est égal à lui-même $1_a : \mathsf{Id}_A(a, a)$

La loi de Leibnitz qui dit que $C(a)$ implique $C(x)$ si on a $p : \mathsf{Id}_A(a, x)$

Un système où “tout est justifié”

La loi *nouvelle* mise en évidence par P. Martin-Löf (1973) est que dans le type $(\Sigma x : A)\mathsf{Id}_A(a, x)$, qui contient l'élément $(a, 1_a)$, tout élément (x, ω) est en fait *égal* à cet élément $(a, 1_a)$

Difficile d'avoir une intuition ?

Est-ce que cette égalité doit satisfaire les axiomes d'extensionalité ?

En fait, que deviennent les axiomes d'extensionalité dans ce contexte ?

Types et homotopie

1908 Russell *Mathematical Logic as Based on the Theory of Types*

1940 Church *A Formulation of the Simple Theory of Types*

1973 Martin-Löf *An Intuitionistic Theory of Types: Predicative Part*

2009 Voevodsky, *Axiome de l'Univalence*

Stratification des types

Un type A est une *proposition*

$(\Pi x_0 : A)(\Pi x_1 : A)\mathsf{Id}_A(x_0, x_1)$

Un type est un *ensemble*

$(\Pi x_0 : A)(\Pi x_1 : A)\mathsf{prop}(\mathsf{Id}_A(x_0, x_1))$

Un type est un *groupoide*

$(\Pi x_0 : A)(\Pi x_1 : A)\mathsf{set}(\mathsf{Id}_A(x_0, x_1))$

La théorie des types apparait alors comme une généralisation *de la théorie des ensembles*

Théorème d'Hedberg

Soit N_k le type fini avec k éléments

Soit N le type des entiers naturels

Le type N_0 représente le type vide/la proposition absurde

Soit $\neg A$ le type $A \to N_0$

Un type A est "décidable" $A + \neg\, A$

On peut montrer que les types N_k, N ont un égalité décidable

Si un type a une égalité décidable ce type est un ensemble

(Hedberg, 1996)

Équivalence

Voevodsky donne une définition très simple et uniforme d'une notion d'équivalence pour $f : A \to B$

Si A et B sont des *ensembles* on retrouve la notion de *bijection* entre ensembles

Si A et B sont des *propositions* on retrouve la notion d'équivalence logique entre propositions

Si A et B sont des *groupoides* on retrouve la notion d'équivalence catégorique entre groupoides

Équivalence

Si $f : A \to B$ la *fibre* de f en $b : B$ est le type

$F(b) = (\Sigma x : A)\mathsf{Id}_B(b, f(x))$

f est une *équivalence* si cette fibre est *contractible* en chaque point b

$F(b) \times \mathsf{prop}(F(b))$

$A \simeq B$ sera $(\Sigma f : A \to B)\mathsf{Equiv}(f)$

Par exemple, l'application identique est une équivalence en utilisant la nouvelle loi de l'égalité découverte par P. Martin-Löf et donc on a $A \simeq A$

L'Axiome d'Univalence

L'*Axiome d'Univalence* dit en gros que si $f : A \to B$ est une equivalence alors A et B sont égaux

Plus précisément, l'application $\mathsf{Id}_U(A, B) \to A \simeq B$ est une équivalence

Ceci généralise directement l'axiome d'extensionalité pour les *propositions* dans le système de Church

Voevodsky a montré que cet axiome entraine l'axiome d'extensionalité pour les *fonctions*

L'Axiome d'Univalence

Cet axiome entraine aussi

-des ensembles isomorphes sont égaux

-des structures algébriques isomorphes sont égales

-des groupoides équivalents (au sens catégoriques) sont égaux

-des catégories équivalentes sont égales

Types et homotopie

Cette stratification est motivée par un modèle où un *type* est interprété par un "espace topologique à homotopie près"

Techniquement un type est interprété par un ensemble simplicial qui vérifie la condition de Kan

Une famille de type $B(x),\ x : A$ est interprétée par une fibration de Kan

Le type $\mathsf{Id}_A(a_0, a_1)$ est l'espace des *chemins* entre a_0 et a_1

Kan *A Combinatorial Definition of Homotopy Groups*, 1958

Types et homotopie

Que devient la nouvelle loi sur l'égalité découverte par P. Martin-Löf dans cette interprétation ?

Elle dit que l'espace total de la fibration définie par l'espace des chemins est *contractible*

C'est exactement ce fait qui, d'après J.P. Serre, est à l'origine de sa méthode de l'espace des chemins en topologie algébrique !

Ensembles ordonnés et catégorie

Dans cette approche

la notion de groupoide est plus fondamentale que la notion de catégorie

Une catégorie est un "ensemble ordonné au niveau des groupoides"

Un *ensemble ordonné* est donné par un *ensemble* A avec une relation $R(x, y)$ proposition qui est transitive et reflexive et telle que l'implication

$\mathsf{Id}_A(x, y) \to R(x, y) \times R(y, x)$

est une équivalence logique

Ensembles ordonnés et catégorie

Une *catégorie* est donné par un *groupoide* A avec une relation $\mathsf{Hom}(x, y)$ *ensemble*

Cette famille d'ensemble est "transitive" (on a une opération de composition associative) et "reflexive" (on a un élément neutre pour la composition)

On peut définir $\mathsf{Iso}(x, y)$ qui est un *ensemble* et montrer que l'on a $\mathsf{Iso}(x, x)$ ce qui donne une application

$$\mathsf{Id}_A(x, y) \to \mathsf{Iso}(x, y)$$

On demande que cette application soit une équivalence entre les deux *ensembles* $\mathsf{Id}(x, y)$ et $\mathsf{Iso}(x, y)$

Transport de structures

Soit $\mathsf{Mon}(A)$ le type qui donne une structure de monoide sur A

$\mathsf{Mon}(A) = (\Sigma f : A \to A \to A)(\Sigma a : A) \ldots$

Si A et B sont des ensembles isomorphes, on a une preuve de

$\mathsf{Id}_U(A, B)$

et donc une preuve de

$\mathsf{Mon}(A) \to \mathsf{Mon}(B)$

Ceci réalise le *transport de structure* (Bourbaki) de monoide le long de l'isomorphisme entre A et B

Transport de structures

Par exemple N et $N + N_1$ sont isomorphes et donc on peut transporter toute structure sur N en une structure sur $N + N_1$ le long de cet isomorphisme

Le modèle des ensembles simpliciaux n'est pas effectif, et l'on ne peut pas l'utiliser pour "programmer" ce transport de structure

Avec M. Bezem et S. Huber, nous avons trouvé une version effective de ce modèle en utilisant des ensembles cubiques

Kan *Abstract Homotopy*, 1955

Ce modèle a été implémenté en Haskell
(avec C. Cohen, S. Huber et A. Mörtberg)

Graphes et fonctions

Dans ce modèle on peut définir un opérateur

$\mathsf{inh}(A)$

qui est une *proposition* exprimant que A a au moins un élément

Ceci permet de définir

$(\exists x : A)B$ comme étant $\mathsf{inh}((\Sigma x : A)B)$

Graphes et fonctions

$(\exists x : A)B$ satisfait les lois du quantificateur existentiel

Contrairement à $(\Sigma x : A)B$ on ne *peut pas* en général extraire un témoin d'une preuve de $(\exists x : A)B$

Toutefois cette extraction est possible dès que $(\Sigma x : A)B$ est une *proposition*

En particulier si $B(x)$ est une proposition et

$B(x_0) \to B(x_1) \to Id_A(x_0, x_1)$

Dans ce cas $(\Sigma x : A)B(x)$ est une proposition et on a

$(\exists x : A)B(x) \to (\Sigma x : A)B(x)$

Graphes et fonctions

Ceci *justifie* l'axiome de description de Church

Mais ceci s'applique aussi pour des situations où $B(x)$ est un ensemble

Graphes et fonctions

Par exemple on peut montrer *sans utiliser l'axiome du choix* qu'un foncteur pleinement fidèle et essentiellement surjectif est une équivalence de catégorie

On a $F : A \to B$ et pour chaque objet b de B le groupoide

$(\Sigma x : A)\mathsf{Iso}(F(x), b)$

est une *proposition* dès que F est pleinement fidèle

Si F est aussi essentiellement surjectif on peut donc définir (effectivement) son "inverse" $G : B \to A$

Directions nouvelles

Pour éviter les paradoxes (Girard) on introduit une hierarchie d'univers $U_0 : U_1 : U_2 : \ldots$

Un type $(\Sigma X : U_2)\mathsf{Id}_{U_2}(U_1, X)$ est intuitivement très "gros" (dans U_3)

Mais c'est une proposition

On ajoute le principe que si A est un type qui est une proposition alors A est dans U_0

Ceci n'introduit pas de contradictions (Voevodsky)

Directions nouvelles

De mẽme on peut introduire le principe que si $A : U_n$ et $B : U_m$ et $n < m$ et

$\mathsf{Id}_{U_m}(A, B)$

alors on a $B : U_n$

De cette manière la catégorie de tous les ensembles finis est dans U_0

Est-ce que tous les programmes terminent encore en ajoutant ces lois ?

Quelques pointeurs

S. Awodey and M. Warren *Homotopy theoretic model of identity types*, 2009

M. Hofmann and Th. Streicher *A groupoid model of type theory*, 1993

V. Voevodsky *Univalent foundation*, page web

Le livre HoTT, 2013

M. Bezem, Th. C., S Huber
A cubical set model of type theory, preprint, 2013