Du calcul des constructions à l'axiome d'univalence

Thierry Coquand

Collège de France/ Sciences du logiciel, Janvier 2019

La correspondance de Curry-Howard aujourd'hui

Cette correspondance a juste 50 ans !

de Bruijn *The mathematical language AUTOMATH, its usage and some of its extensions.* Symposium of Automatic Demonstration, IRIA, Versailles, Dec. 1968

Howard *The formulae-as-types notion of construction.* Manuscript 1969.

Le calcul des constructions

Synthèse (1984) des travaux de de Bruijn, Girard et Martin-Löf

de Bruijn, papiers disponibles sur l'archive https://www.win.tue.nl/automath/ en particulier

N.G. de Bruijn, *AUTOMATH, a language for mathematics*, 1973

L.S. van Benthem Jutting, *The development of a text in AUT-QE*, 1973

AUTOMATH

Treating propositions as types is definitely not in the way of thinking of ordinary mathematician, yet it is very close to what he actually does

A survey of the project AUTOMATH
N.G. de Bruijn, 1980, dans le livre
To H.B. Curry, essays on combinary logic, lambda-calculus and formalism

AUTOMATH

Représentation d'un énoncé tel que

Théorème 1: *Soit* x *un nombre réel tel que* $f(x) > 1$ *et soit* n *un entier. Si on a* $g(x) > x^n$ *alors* $f(x) > n$.

Si un mathématicien veut utiliser cet énoncé plus tard, avec $x = p$ et $n = 5$, il doit fournir une preuve (1) de $f(p) > 1$ et après une preuve (2) de $g(p) > p^5$

Il peut alors énoncer $f(p) > 5$ en *appliquant* le théorème 1 et en donnant *dans cet ordre*

p, la preuve (1), 5 et la preuve (2)

AUTOMATH

Dans le système AUTOMATH cela sera représenté par

Corollaire = Théorème 1(p, (1), 5, (2)) : A

où A est l'énoncé $f(p) > 5$

AUTOMATH

Une notion cruciale (inspirée par la notion de *structure de blocs* du langage ALGOL 60) est la notion de *contexte*: suite de déclarations de variables d'objet (avec leur type) et de noms d'hypothèses dans un ordre *arbitraire*

$x : R,\ h_1 : f(x) > 1,\ n : N,\ h_2 : g(x) > x^n$

AUTOMATH avait une "sorte" primitive type

$R : \text{type},\ N : \text{type},\ x : R,\ h_1 : f(x) > 1,\ n : N,\ h_2 : g(x) > x^n$

On pouvait aussi introduire une sorte primitive prop ou avoir prop = type

AUTOMATH

AUTOMATH utilisait la même notation pour l'abstraction typée $[x : A]M$ et pour le produit dépendant $[x : A]B$

On obtient un calcul assez minimal

$M, A \ ::= \ x \mid M \ M \mid [x : A]M \mid \text{type}$

AUTOMATH

Un des premiers exemples était la notion d'égalité sur A : type

eq : $[x : A][y : A]$type

refl : $[x : A]$eq x x

eucl : $[x : A][y : A][z : A]$eq x $z \to$ eq y $z \to$ eq x y

Preuve de la symmétrie

$[x : A][y : A][h :$ eq x $y]$eucl y x y (refl y) h

est de type $[x : A][y : A]$ (eq x $y) \to$ eq y x

AUTOMATH et système F

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash [x : A]M : [x : A]B} \qquad \frac{\Gamma \vdash N : [x : A]B \qquad \Gamma \vdash M : A}{\Gamma \vdash N\ M : B(M/x)}$$

AUTOMATH et système F

$$C \ ::= \ \mathsf{type} \mid [x : A]C$$

$$\frac{\Gamma, x : A \vdash B : \mathsf{type}}{\Gamma \vdash [x : A]B : \mathsf{type}} \qquad \frac{\Gamma, x : A \vdash C}{\Gamma \vdash [x : A]C}$$

$$\frac{}{\vdash \mathsf{type}} \qquad \frac{\Gamma \vdash A : \mathsf{type}}{\Gamma, x : A \vdash \mathsf{type}} \qquad \frac{\Gamma \vdash C}{\Gamma, x : C \vdash \mathsf{type}}$$

AUTOMATH et système F

On note $A \to B$ pour $[x : A]B$ si x n'est pas libre dans B

$[A : \mathsf{type}]A \to A$ est le type de l'application identité "polymorphe"

$$[A : \mathsf{type}][x : A]x \;\; : \;\; [A : \mathsf{type}]A \to A$$

AUTOMATH et système F

Codage de Russell-Prawitz

$\bot = [A : \mathsf{type}]A \qquad A \wedge B = [X : \mathsf{type}](A \to B \to X) \to X$

$\mathsf{eq} \quad : [A : \mathsf{type}]A \to A \to \mathsf{type}$

$\mathsf{eq}\ A\ x\ y = [P : A \to \mathsf{type}]P\ x \to P\ y$

Codage de Church, Martin-Löf, Böhm-Berarducci

$\mathsf{bool} = [A : \mathsf{type}]A \to A \to A \qquad \mathsf{nat} = [A : \mathsf{type}]A \to (A \to A) \to A$

Le calcul des constructions: AUTOMATH et système F

Un moyen uniforme de représenter le système Fω de Girard

Mais aussi la logique d'ordre supérieure de Church (1940)

Traitement uniforme des termes et des lois logiques

La vérification des preuves se réduit à un problème de vérification de types

Peut être représenté sur machine

Le calcul des constructions: AUTOMATH et système F

Un des premiers exemples représenté dans ce système était le résultat prouvé dans le livre *Idéographie*, Frege 1879

Frege introduit non seulement les quantificateurs, mais aussi la logique d'ordre supérieure !

$\varphi \to \psi \to \varphi$ $\qquad$ $(\varphi \to \psi \to \delta) \to (\varphi \to \psi) \to \varphi \to \delta$

Frege: on a $\varphi \to \forall x\ \psi(x)$ si $\varphi \to \psi(x)$ *et* x n'est pas libre dans φ

AUTOMATH

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash [x : A]M\ \ : [x : A]B}$$

Le calcul des constructions: AUTOMATH et système F

Frege prouve que la clôture transitive d'une relation fonctionnelle définit une relation d'ordre linéaire

Un autre exemple était la preuve de Gérard Huet du lemme de Newman qui utilise un principe d'induction noethérienne

$$[P : A \to \mathsf{type}]([x : A]([y : A]R\ x\ y \to P\ y) \to P\ x) \to [x : A]P\ x$$

Inductively Defined Types in the Calculus of Constructions,
Ch. Paulin-Mohring, F. Pfenning, MFPS 1989

Codage du système F_2 dans le système F_3

Le calcul des constructions: AUTOMATH et système F

AUTOMATH introduit une nouvelle notion d'égalité *définitionnelle*

Notion introduite en λ-calcul et en logique combinatoire

Elle est distincte de l'égalité introduite comme type *book equality*

La notion d'égalité définitionelle n'est pas présente dans Frege qui a une égalité primitive utilisée pour exprimer les définitions

On obtient un système de notations pour les preuves

Par exemple, une des premières preuves de Frege utilise le même lemme de deux manières différentes

Cohérence et expressivité

Est-ce que le calcul est *cohérent* ?

Martin-Löf avait introduit un calcul très similaire avec un axiome type : type

Le système devient incohérent si on introduit

$$\frac{\Gamma, x : A \vdash B : \mathsf{type}}{\Gamma \vdash \Sigma(x : A)B : \mathsf{type}}$$

On peut définir $\exists(x : A)B = [X : \mathsf{type}]([x : A]B \to X) \to X$ et une projection $M.1 : A$ si $M : \exists(x : A)B$ mais pas $M.2 : B(M.1/x)$

On peut définir 0 et $1 : \mathsf{bool}$ mais il n'est pas possible de prouver $\neg(\mathsf{eq\ bool}\ 0\ 1)$

Cohérence et expressivité

Girard: le calcul des constructions avec deux niveaux est incohérent

de Bruijn: paradoxe des arbres

Cohérence et expressivité

Le paradoxe de Girard s'applique à l'extension $\mathsf{type} : \mathsf{type}_1$ avec les lois

$$\frac{\Gamma, x : A \vdash B : \mathsf{type}}{\Gamma \vdash [x : A]B : \mathsf{type}} \qquad \frac{\Gamma, x : A \vdash B : \mathsf{type}_1}{\Gamma \vdash [x : A]B : \mathsf{type}_1}$$

"Type" is not a type
A. Meyer, M.B. Reinhold, POPL 86

Pour avoir un système cohérent, il faut imposer

$$\frac{\Gamma \vdash A : \mathsf{type}_1 \qquad \Gamma, x : A \vdash B : \mathsf{type}_1}{\Gamma \vdash [x : A]B : \mathsf{type}_1}$$

Cohérence et expressivité

On arrive ainsi à un système avec $\mathsf{type}_i : \mathsf{type}_{i+1}$ et la loi

$$\frac{\Gamma \vdash A : \mathsf{type}_i \qquad \Gamma, x : A \vdash B : \mathsf{type}_j}{\Gamma \vdash [x : A]B : \mathsf{type}_{max(i,j)}}$$

et il est naturel d'utiliser la notation prop pour la sorte imprédicative type

$$\frac{\Gamma, x : A \vdash B : \mathsf{prop}}{\Gamma \vdash [x : A]B : \mathsf{prop}}$$

Th. C. *An analysis of Girard's paradox*, LICS 1986

Cohérence et expressivité

Question : comment ce système se compare avec la théorie des ensembles ?

Conjecture : le système est plus fort que le système Zermelo

Cohérence et expressivité

On peut définir $\mathsf{nat} : \mathsf{type}_2$ par $\mathsf{nat} = [A : \mathsf{type}_1]A \to (A \to A) \to A$

On peut définir $0 : \mathsf{nat}$ et $\mathsf{S} : \mathsf{nat} \to \mathsf{nat}$ et montrer l'axiome d'infinité !

$[x : \mathsf{nat}]\neg\mathsf{eq}\ \mathsf{nat}\ 0\ (\mathsf{S}\ x)$

$[x : \mathsf{nat}][y : \mathsf{nat}]\mathsf{eq}\ \mathsf{nat}\ (\mathsf{S}\ x)\ (\mathsf{S}\ y) \to \mathsf{eq}\ \mathsf{nat}\ x\ y$

Cohérence et expressivité

Le calcul des constructions implicite : syntaxe et sémantique
A. Miquel, thèse, 2001

On peut représenter les *graphes pointés* et la relation de *bissimilarité*

Un *ensemble* peut alors être codé comme un graphe à bissimilarité près

Arbres (ensembles) non nécessairement bien fondés

Tous les axiomes de Zermelo sont vérifiés !

Cohérence et expressivité

Problèmes avec la représentation des types de données

-on doit relativiser N : nat $\to$ prop pour obtenir l'axiome d'induction

-fonctions versus relations fonctionnelles

-la représentation de la récursion primitive récursive

$f\ 0 = a \qquad f\ (\mathsf{S}\ n) = g(n, f\ n)$

est possible, mais pas très naturelle

Cela suggère l'introduction des types de données de manière primitive avec des lois de calcul (récursion primitive récursive) en suivant Martin-Löf

Cohérence et expressivité

Synthèse entre AUTOMATH et le système de Martin-Löf

AUTOMATH : notion de *contexte* et réduction de la vérification des *preuves* à un problème (décidable) de vérification de *typage*

Martin-Löf : types de données primitifs, correspondance entre la notion de constructeurs et les lois d'introduction en logique (typage non décidable 79-86)

Bishop : preuve constructive = algorithme

Une approche au problème de la correction de programme

Cohérence et expressivité

Inductive Definition in Type Theory
N. Mendler, PhD thesis, 1987

Inductively defined types
Th. C., Ch. Paulin-Mohring, COLOG-88

Système plus faible que ZF (travaux de M. Rathjen)

A compiled implementation of strong reduction.
B. Grégoire, X. Leroy, ICFP 2002

G. Gonthier, B. Werner preuve du théorème des 4 couleurs

Preuve du théorème de Feit et Thompson

Types dépendants et lois logiques

$\mathsf{inl} : A \to A + B \qquad \mathsf{inr} : B \to A + B$

On peut définir $h : A + B \to C$ par cas

$h\ (\mathsf{inl}\ x) = f\ x \qquad h\ (\mathsf{inr}\ y) = g\ y$

$h : (A + B) \to C$ si $f : A \to C$ et $g : B \to C$

Types dépendants et lois logiques

$h : \Pi(z : A + B)C(z)$ si $f : \Pi(x : A)C(\mathsf{inl}\ x)$ et $g : \Pi(y : B)C(\mathsf{inr}\ y)$

$(A + B) \to C$ est remplacé par $\Pi(z : A + B)C(z)$

h est définie par cas $h\ (\mathsf{inl}\ x) = f\ x \qquad h\ (\mathsf{inr}\ y) = g\ y$

$h : \Pi(z :\bot)C(z)$

$h : \Pi(n : \mathsf{nat})C(n)$ par les lois $h\ 0 : C(0) = a$ et $h\ (\mathsf{S}\ n) : C(\mathsf{S}\ n) = g\ n\ (f\ n)$

$a : C(0)$ et $g : \Pi(n : N)C(n) \to C(\mathsf{S}\ n)$

on obtient de manière naturelle la récursion primitive récursive (Hilbert 1925)

Égalité comme type de données

refl x : Id $A\ x\ x$

$f : \Pi(x\ y : A)\mathsf{Id}\ A\ x\ y \to C(x, y)$ défini par

$f\ \ x\ x\ (\mathsf{refl}\ x) : C(x, x) = d(x)$

Plus petite relation réflexive

$f : \Pi(x\ y : A)(p : \mathsf{Id}\ A\ x\ y)C(x, y, p)$ défini par

$f\ x\ x\ (\mathsf{refl}\ x) : C(x, x, \mathsf{refl}\ x) = d(x)$

Égalité comme type de données

C'est une loi *nouvelle* pour la notion d'*identifications* !

Contexte $x : A,\ y : A,\ p : \mathsf{Id}\ A\ x\ y$

$\mathsf{Id}\ A\ x\ y$ est un type

Plusieurs manières d'identifier deux éléments de A en général

Cette loi a des conséquences remarquables

Égalité comme type de données

Tout élément de $S = \Sigma(x : A)\mathsf{Id}\ A\ a\ x$ est égal à $(a, \mathsf{refl}\ a)$

$\Pi(a : A)(x : A)(p : \mathsf{Id}\ A\ a\ x)C(a, x, p)$

$C(a, x, p) = \mathsf{Id}\ S\ (a, \mathsf{refl}\ a)\ (x, p)$

$C(a, a, \mathsf{refl}\ a) = \mathsf{Id}\ S\ (a, \mathsf{refl}\ a)\ (a, \mathsf{refl}\ a)$

Tout type définit un (∞)-groupoïde
F. Lamarche, M. Hofmann, Th. Streicher 1993)

Dans un autre domaine, J.-P. Serre avait formulé une loi tout à fait similaire pour les espaces de lacets dans sa thèse (1951)

Collections et identifications

Curry-Howard : Id $A\ a\ b$ peut être vu comme un type

Est-il naturel de considérer que ce type peut avoir plusieurs éléments ?

Oui, si on pense à ce type comme le type des *identifications* de a et b

Quelles sont les notions d'identification que l'on rencontre en mathématique ?

Lois d'*identification* de deux *collections* d'objets mathématiques ?

Structures algébriques et ordonnées

Ensemble muni de lois de compositions ou de relations qui satisfont certaines propriétés

E.g. groupes, anneaux, treilies

On veut identifier des structures isomorphes

Il peut y avoir plusieurs identifications possibles

Notion de *structure*

Structures algébriques et ordonnées

C'est ce qui est étudié par Bourbaki dans le chapitre *théorie des structures*

Très utile pour l'*abstraction* et donc la *modularisation*

Le même raisonnement abstrait peut être instancié pour des structures concrêtes différentes

Cf. *Types, abstraction and parametric polymorphism*, J.R. Reynolds, 1983

travaux de Morris, Liskov and Zilles, ... sur l'importance de l'indépendence des représentations pour le développement modulaire des programmes

Structures algébriques et ordonnées

La réalisation que deux structures sont isomorphes est souvent importante

On peut alors transporter les intuitions d'une structure sur une autre

On peut résoudre un problème en le transformant en un problème sur une structure isomorphe

Exemple : la correspondance de Galois entre les sous-extensions normales et les sous-groupes distingués

Description des objets mathématiques

Deux groupes isomorphes vérifient les mêmes propriétés “structurales”

Si un est commutatif l’autre est commutatif

Si un est résoluble l’autre est résoluble

Mais un des groupes peut contenir l’ensemble vide et pas l’autre

Bourbaki introduit la notion de propriétés *transportables*

En théorie des ensembles, toutes les notions ne sont pas transportables

Description des objets mathématiques

Le principe de Voevodsky pour un formalisme représentant les mathématiques

Toutes les notions doivent être invariantes par isomorphismes/équivalences

Forme très (trop ?) forte du principe d'extensionalité/modularité

Cf. exposé (2018) de P. Deligne *What do we mean by "equal"?*

Identification

Cette notion d'identification a été généralisée en mathématique

Collection de tous les groupes, ou de tous les ensembles (dans un univers fixé)

Exemple : soit B un ensemble donné

On peut identifier les deux collections ENS^B and ENS/B

Elles satisfont les mêmes propriétés "structurelles"

ENS^B collection des familles d'ensemble $X_b,\ b \in B$

ENS/B collection de "ensembles sur B", i.e. $Y, f : Y \to B$

Identification

$F : \mathrm{ENS}^B \to \mathrm{ENS}/B$ and $G : \mathrm{ENS}/B \to \mathrm{ENS}^B$

$F(X) = \Sigma(b : B)X_b, \pi_1$

$G(Y, f) = (f^{-1}(b))_{b \in X}$

$G(F(X))$ et X sont seulement *isomorphes* (et non égaux strictement)

$G(F(X))_b = \{b\} \times X_b$

Les deux collections (groupoïdes) ENS^B and ENS/B sont *équivalentes*

F et G ne définissent pas un isomorphisme

Équivalence comme identification

On obtient ainsi un nouveau moyen d'identifier des collections

Structure de catégorie cartésienne fermée sur ENS^B transportable sur ENS$/B$

Autre exemple: la collection $\mathbb{L}_5$ de tous les ordres linéaires avec 5 éléments

C'est une collection très large

Mais on a une *identification* avec le groupoïde trivial $\mathbb{L}_5 \simeq 1$!

Équivalence comme identification

Ces équivalences peuvent être utiles en calcul formel

Exemple : équivalence entre la catégorie des faisceaux cohérents sur l'espace projectif $\mathbb{P}^n$ et le quotient de Serre de la catégorie des modules gradués sur $k[X_0, \ldots, X_n]$

Système CAP (Categories, Algorithms, Programming), par M. Barakat

Des calculs doublement exponentiels peuvent être remplacés par des calculs polynomiaux avec un transport le long d'une équivalence

Description des objets mathématiques

Au niveau suivant, on a les 2-*groupoïdes*

On a de nouvelles lois d'identification

Composition horizontale et verticale, avec la loi d'échange

Puis n-groupoïdes, et ∞-groupoïdes

Les lois d'identification sont reliées aux groupes d'homotopie des sphères qui sont des objets mystérieux; par exemple le fait que la loi d'échange n'est pas stricte est relié à l'égalité $\pi_3(S^2) = \mathbb{Z}$

Description des objets mathématiques

« À ce moment apparait l'intuition que les ∞-groupoïdes doivent constituer des modèles, particulièrement adéquats, pour les types d'homotopie, les n-groupoïdes correspondant aux types d'homotopie tronqués (avec $\pi_i = 0$ pour $i > n$) »
(Grothendieck, Esquisses d'un programme, 1984)

Description des objets mathématiques

L'égalité définie inductivement ne vérifie pas le principe de "modularité"

$f = \lambda(x : \mathsf{nat})\ x + 0 \qquad g = \lambda(x : \mathsf{nat})\ 0 + x$

f et g doivent être "indiscernables" : même comportement "entrée-sortie"

Mais on a $C(f)$ et pas $C(g)$ si $C(u)$ est $\mathsf{Id}\ (\mathsf{nat} \to \mathsf{nat})\ f\ u$

L'axiome d'univalence

Une deuxième loi nouvelle pour les identifications

Version "Curry-Howard" de l'axiome d'extensionalité pour les propositions

$(p \equiv q) \equiv (p = q)$

$(A \simeq B) \simeq (\mathsf{Id}\ \mathsf{type}_n\ A\ B)$

L'axiome d'univalence

Il est possible de définir $A \simeq B$ de manière uniforme !

$$A \simeq B \;=\; \Sigma(f : A \to B)\mathsf{isEquiv}\ A\ B\ f$$

$$\mathsf{isEquiv}\ A\ B\ f \;=\; \Pi(y : B)\mathsf{isContr}\ (\Sigma(x : A)\mathsf{Id}\ B\ (f\ x)\ y)$$

$$\mathsf{isContr}\ T \;=\; \Sigma(t : T)\Pi(u : T)\mathsf{Id}\ T\ t\ u$$

An experimental library of formalized Mathematics based on the univalent foundations.

V. Voevodsky, MSCS, 2015.

L'axiome d'univalence

Le formalisme des types dépendants est bien approprié pour exprimer les lois de la notion d'identification entre deux collections

Les univers/sortes type_n ne sont pas des ensembles

bool peut être identifier à lui-même de deux manières différentes

Autre exemple naturel en mathématique : la collection des triangles (cf. exposé de P. Deligne et de D. Grayson); un triangle équilatéral s'identifie à lui-même de six manières différentes

Univers et identification

La collection de tous les faisceaux n'est pas un faisceau

On n'a pas recollement unique en général

Seulement unique à isomorphisme près

Notion de *champs*

Modalité exacte à gauche

Modalities in homotopy type theory
E. Rijke, M. Shulman, B. Spitters, 2017

Si $B(x)\ (x : A)$ est une famille de propositions $m_a^X : X \to X^{B(a)}$

$G\ X = \Pi(a : A)\mathsf{isEquiv}\ m_a^X$

$G : \mathsf{type}_n \to \mathsf{type}_n$

$G\ X$ est une proposition

On a $G\ (\Sigma(X : \mathsf{type}_n)G\ X)$!

Modalité exacte à gauche

Thèse de Kevin Quirin, Chapitre 4, 2016

Modèle (interne) de la théorie des types

$x_1 : [\![A_1]\!], \ldots, x_n : [\![A_n]\!] \vdash [M] : [\![A]\!]$

$[\![A]\!] = [A].1$

$[\mathsf{type}_k] = (\Sigma(X : \mathsf{type}_k)G(X), pf)$

$[\Pi(x : A)B] = (\Pi(x : [\![A]\!])[\![B]\!], pf)$

$[\Sigma(x : A)B] = (\Sigma(x : [\![A]\!])[\![B]\!], pf)$

Modalité exacte à gauche

$$[N\ M] = [N]\ [M]$$

$$[\lambda(x : A)M] = \lambda(x : [\![A]\!])[M]$$

$$[x] = x$$

$$[M.i] = [M].i$$

On obtient un modèle (interne) de la théorie des types

Interprétation logique comme transformation de programme

Ce modèle vérifie l'axiome d'univalence

Modalités et axiome de redimensionnement

Voevodsky définit $\|A\| = \Pi(X : \text{type}_n)\ \text{isProp}\ X \to (A \to X) \to X$

On a $\|A\| : \text{type}_{n+1}$ si $A : \text{type}_n$

Axiome de *redimensionnement* : $X : \text{type}_0$ dès que $X : \text{type}_n$ et $p : \text{isProp}\ X$

$\text{prop} = \Sigma(X : \text{type}_0)\ \text{isProp}\ X$

On obtient un type des “propositions”, comme pour le calcul des constructions, mais qui vérifie une forme très forte de l’“axiome du choix unique”

Symbole ι dans le système de Church

Cohérence ?

Conclusion

Le formalisme des types dépendants est bien approprié pour exprimer les lois de la notion d'identification entre deux collections

Curry-Howard : égalité et identification

Un bon mélange d'idées venant de considérations récentes en mathématiques et de la sémantique des langages de programmation

Bibliographie

Une explication (constructive) de l'axiome d'univalence

Cubical type theory: a constructive interpretation of univalence
C. Cohen, Th. C., S. Huber, A. Mörtberg, 2015.

Axioms for Modelling Cubical Type Theory in a Topos.
I. Orton and A. Pitts, EACSL, 2016.

Une explication des HITs

On Higher Inductive Types in cubical type theory
Th. C., S. Huber, A. Mörtberg, 2018

Bibliographie

Univalence et modularité

Equivalences for free: univalent parametricity for effective transport
N. Tabareau, É. Tanter, M. Sozeau, ICFP 2018

Modalités

Brouwer's fixed-point theorem in real-cohesive homotopy type theory
N. Shulamn, MSCS, 2018.

Interprétation logique comme transformation de programmes

Failure is not an option: an exceptional type theory
P.-M. Pédrot, N. Tabareau, ESOP 2018.