Soutenance – Thèse de master

Règles de réécriture dans le $\lambda\Pi$-calcul modulo théorie

Thomas Traversié

Encadrants : Valentin Blot et Gilles Dowek

Qu'est ce qu'une preuve ?

- Preuve écrite **à la main**
  - En apparence rigoureuse
  - Peut contenir des erreurs
  - Divers niveaux de fiabilité

- Preuve **formelle**
  - Formalisme mathématique
  - Basée sur des déductions logiques

Vérification de preuves

- **Vérification formelle**
  - Vérifier automatiquement une preuve formelle
  - Certification de résultats mathématiques (théorème des quatre couleurs) ou de systèmes critiques (métros automatiques)

- De nombreux systèmes de preuve

Un système de preuve « universel »

- $\lambda\Pi$-calcul modulo théorie (aussi appelé Dedukti)
  - Cadre logique dans lequel on peut exprimer plusieurs théories
  - En particulier les théories d'autres systèmes

- Lambdapi
  - Assistant de preuve
  - Basé sur le $\lambda\Pi$-calcul modulo théorie

Le $\lambda\Pi$-calcul modulo théorie

λ **λ-calcul simplement typé**

$\mathsf{nat} : \mathtt{TYPE} \qquad 0 : \mathsf{nat} \qquad \mathsf{succ} : \mathsf{nat} \to \mathsf{nat}$

$+ : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \qquad \mathsf{list} : \mathsf{nat} \to \mathtt{TYPE}$

Π **types dépendants**

$\mathsf{concat} : \Pi x, y : \mathsf{nat}.\ \mathsf{list}\ x \to \mathsf{list}\ y \to \mathsf{list}\ (x + y)$

$\mathcal{R}$ **règles de réécriture $\ell \hookrightarrow r$**

$x + 0 \hookrightarrow x$

Le $\lambda\Pi$-calcul modulo théorie

- **Théorie** $\mathcal{T} = (\Sigma, \mathcal{R})$:
  - Signature $\Sigma = \{c_1 : A_1,\ \ldots,\ c_n : A_n\}$
  - Système de réécriture $\mathcal{R}$: $\ell \hookrightarrow r$

- **Conversion** $\equiv_{\beta\mathcal{R}}$ générée
  - par les règles $\mathcal{R}$
  - par β-réduction $(\lambda x : A.\ t)\ u \hookrightarrow t[x \mapsto u]$

Ex : $(\lambda x : \mathsf{nat}.\ \mathsf{succ}\ x + x)\ 0 \equiv_{\beta\mathcal{R}} \mathsf{succ}\ 0 + 0 \equiv_{\beta\mathcal{R}} \mathsf{succ}\ 0$

Règles de typage

$$\frac{\Gamma \vdash A : \texttt{TYPE} \qquad \Gamma, x : A \vdash B : s \qquad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.\ t : \Pi x : A.\ B}\ [\textsc{Abs}]$$

$$\frac{\Gamma \vdash A : \texttt{TYPE} \qquad \Gamma, x : A \vdash B : s}{\Gamma \vdash \Pi x : A.\ B : s}\ [\textsc{Prod}] \qquad \frac{\Gamma \vdash t : \Pi x : A.\ B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B[x \mapsto u]}\ [\textsc{App}]$$

$$\frac{\Gamma \vdash t : A \qquad \vdash B : s}{\Gamma \vdash t : B}\ [\textsc{Conv}]\ A \equiv_{\beta\mathcal{R}} B$$

Exemple : entiers naturels et listes

$$\mathsf{nat} : \mathtt{TYPE} \qquad 0 : \mathsf{nat} \qquad \mathsf{succ} : \mathsf{nat} \to \mathsf{nat} \qquad + : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \qquad x + 0 \hookrightarrow x$$

$$x + \mathsf{succ}\ y \hookrightarrow \mathsf{succ}\ (x + y) \qquad \mathsf{list} : \mathsf{nat} \to \mathtt{TYPE} \qquad \mathsf{nil} : \mathsf{list}\ 0$$

$$\mathsf{isRev} : \Pi x : \mathsf{nat}.\ \mathsf{list}\ x \to \mathsf{list}\ x \to \mathtt{TYPE} \qquad \mathsf{concat} : \Pi x, y : \mathsf{nat}.\ \mathsf{list}\ x \to \mathsf{list}\ y \to \mathsf{list}\ (x + y)$$

- Si $\ell : \mathsf{list}\ (\mathsf{succ}\ 0)$, on a $\mathsf{concat}\ (\mathsf{succ}\ 0)\ 0\ \ell\ \mathsf{nil} : \mathsf{list}\ (\mathsf{succ}\ 0 + 0)$
- On a $\mathsf{list}\ (\mathsf{succ}\ 0 + 0) \equiv_{\beta\mathcal{R}} \mathsf{list}\ (\mathsf{succ}\ 0)$

Contributions

1. Implémentation de la théorie des ensembles dans le $\lambda\Pi$-calcul modulo théorie et dans LAMBDAPI (1A et 2A)

2. Preuve que des résultats prouvables dans le $\lambda\Pi$-calcul modulo théorie avec des règles de réécriture sont aussi prouvables avec des axiomes (3A)

Partie 1 :

Implémentation de la théorie des ensembles

Théorie des ensembles

- Théorie de **référence** en mathématiques
  - « Paradis pour les mathématiciens » (Hilbert)
  - A la base de plusieurs systèmes de preuve (Mizar, Isabelle/ZF et Atelier B)

- Définie par des **axiomes**
  - Extensionnalité : deux ensembles sont égaux s'ils ont les mêmes éléments
  - Paire : pour x, y deux ensembles, il existe un ensemble $\{x, y\}$ qui contient x et y
  - Et d'autres

Comment l'implémenter dans LAMBDAPI ?

- Déclarer chaque **axiome**
  ↪ pas de propriété d'élimination des coupures ✗

- Déclarer une **règle de réécriture** à la place de chaque axiome
  ↪ pas de propriété d'élimination des coupures [Crabbé, 1974] ✗

- **Exprimer** les ensembles avec une notion de *graphes pointés* [Dowek et Miquel, 2007]
  ↪ propriété d'élimination des coupures ✓

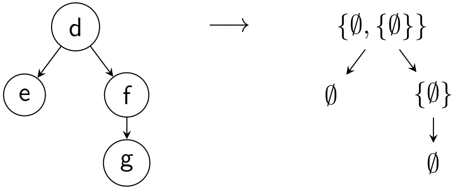
Contributions

- **Implémentation** de la théorie des ensembles avec graphes pointés dans Lambdapi
- Adaptation de la Déduction modulo au $\lambda\Pi$-calcul modulo théorie
- Passage de preuves à la main à des preuves **formelles**

Théorie des graphes pointés

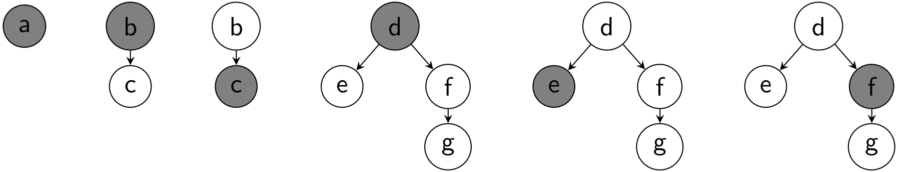
- **Graphe pointé** : graphe orienté avec une racine [Aczel, 1988]

- Interprétation dépend de la **localisation de la racine**



Racine en e ou g : $\emptyset$ Racine en f : $\{\emptyset\}$ Racine en d : $\{\emptyset, \{\emptyset\}\}$

- Quelques graphes pointés :



$\emptyset$ $\quad\{\emptyset\}$ $\quad\emptyset$ $\quad\{\emptyset, \{\emptyset\}\}$ $\quad\emptyset$ $\quad\{\emptyset\}$

- Un même graphe peut représenter différents ensembles
  Un même ensemble peut être représentés par différents graphes

Théorie des graphes pointés

- Définitions

$x\ \eta_a\ y$	arête de y à x dans le graphe pointé a
a/x	change la racine du graphe pointé a comme étant le nœud x
$\mathrm{root}(a)$	retourne la racine du graphe pointé a

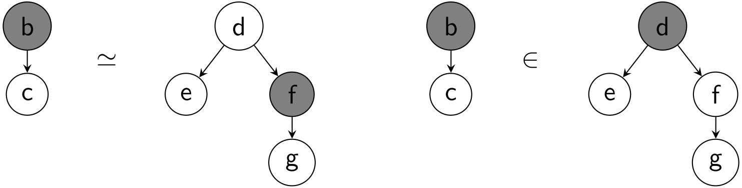
- Règles de réécriture

$$x\ \eta_{a/z}\ y \hookrightarrow x\ \eta_a\ y$$

$$\mathrm{root}(a/x) \hookrightarrow x$$

$$(a/x)/y \hookrightarrow a/y$$

Relations entre graphes pointés



- Bisimilarité

$$
\begin{aligned}
a \simeq b \hookrightarrow\ & \exists r,\ r\ \mathrm{root}(a)\ \mathrm{root}(b) \\
& \wedge\ \forall x\ \forall x'\ \forall y\ (x'\ \eta_a\ x\ \wedge\ r\ x\ y\ \Rightarrow\ \exists y'\ (y'\ \eta_b\ y\ \wedge\ r\ x'\ y')) \\
& \wedge\ \forall y\ \forall y'\ \forall x\ (y'\ \eta_b\ y\ \wedge\ r\ x\ y\ \Rightarrow\ \exists x'\ (x'\ \eta_a\ x\ \wedge\ r\ x'\ y'))
\end{aligned}
$$

- Appartenance

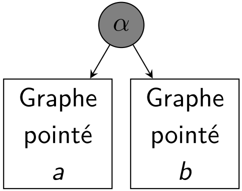
$$a \in b \hookrightarrow \exists x\ (x\ \eta_b\ \mathrm{root}(b)\ \wedge\ a \simeq (b/x))$$

Constructions

- Pour chaque axiome de la théorie des ensembles, on a un **constructeur** défini par des règles de réécriture

- Paire : $\forall a\ \forall b\ \exists c\ \forall x\ (x \in c \Leftrightarrow (x \simeq a \vee x \simeq b))$

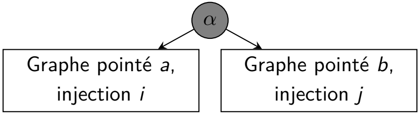
  Construction de $c = \{a, b\}$



  Nœuds $a \neq$ nœuds de $b \neq \alpha$

Constructions

- Injections **disjointes** i, j telles que α n'est **pas** dans leurs images



- Paire : $$\mathrm{root}(\{a, b\}) \hookrightarrow \alpha$$

$$\begin{aligned} x\ \eta_{\{a,b\}}\ x' \hookrightarrow\ & (\exists y \exists y'\ (x = i(y) \wedge x' = i(y') \wedge y\ \eta_a\ y')) \\ & \vee (\exists y \exists y'\ (x = j(y) \wedge x' = j(y') \wedge y\ \eta_b\ y')) \\ & \vee (x = i(\mathrm{root}(a)) \wedge x' = \alpha) \\ & \vee (x = j(\mathrm{root}(b)) \wedge x' = \alpha) \end{aligned}$$

- Constructions similaires pour les autres axiomes

Graphes pointés et théorie des ensembles

- La théorie des graphes pointés **valide** les axiomes de la théorie des ensembles [Dowek et Miquel, 2007, preuves informelles]

- Chaque axiome est un théorème dans la théorie des graphes pointés
  $+$ Lemmes intermédiaires sur le structure des graphes pointés
  $=$ **53 lemmes nécessaires**

Paire (lemma 43) : $\forall x\ (x \in \{a, b\} \Leftrightarrow (x \simeq a \vee x \simeq b))$
$\llcorner$ Lemme 36 : $(\{a, b\}/i(\mathrm{root}(a))) \simeq a$
$\llcorner$ Lemme 37 : $(\{a, b\}/j(\mathrm{root}(b))) \simeq b$

- Compréhension : $\forall b\ \exists c\ \forall a\ [a \in c \Leftrightarrow (a \in b\ \wedge\ P(x \leftarrow a))]\ (*)$
- Domaine des propositions **restreint** : $(*)$ P formule dans le langage $\{\simeq,\ \in\}$ et avec des quantifications uniquement sur des graphes pointés
- Classe de formules $\xrightarrow{\text{interprétation}}$ classe des propositions
  - Formules : par induction
  - Interprétation : par des règles de réécriture

Implémentation

- Implémentation dans LAMBDAPI : preuves **formelles** des 53 lemmes
- Plusieurs **simplifications** par rapport à la Déduction modulo
- Pour un utilisateur : importer les fichiers et utiliser les « axiomes » normalement
- Conjecture : propriété d'élimination des coupures

Partie 2 :

Remplacement des règles de réécriture par des axiomes

Axiome ou réécriture ?

- Deux types de systèmes de preuve

Avec axiomes

$$x + succ\ y = succ\ (x + y)$$
$$x + 0 = x$$

On **prouve** $2 + 2 = 4$

Avec règles de réécriture

$$x + succ\ y \hookrightarrow succ\ (x + y)$$
$$x + 0 \hookrightarrow x$$

On **calcule** $(2 + 2 = 4) \equiv (4 = 4)$

- Un résultat démontrable avec règles de réécriture l'est-il aussi avec des axiomes ?
  Ex : Si on a ℓ : list $(2 + 2)$, on n'a (ou pas) ℓ : list 4

Méthode

- **Remplacer** chaque règle de réécriture $\ell \hookrightarrow r$ par un axiome $\ell = r$
- Remplacer chaque utilisation de la règle [Conv] pour passer de

$$t : A \text{ à } t : B \text{ avec } A \equiv_{\beta\mathcal{R}} B$$

par un **transport** pour passer de

$$t : A \text{ à } \mathsf{transp}\ p\ t : B \text{ avec } p : A = B$$

[Oury, 2005, Winterhalter *et al.*, 2019]

Contributions

- Formalisation d'une **traduction** d'un système avec règles de réécriture vers un système sans règles

- Définition de deux **égalités**
  – pour comparer des termes
  – pour comparer des types

- Construction d'une égalité $p : A = B$ pour chaque conversion $A \equiv_{\beta\mathcal{R}} B$

Encodage prélude

- Encodage des notions de proposition et preuve [Blanqui *et al.*, 2023]

- **Univers des types** $Set : \mathtt{TYPE}$, injection $El : Set \to \mathtt{TYPE}$
  Ex : nat : Set, 0 : El nat

- **Univers des propositions** $El\ o$, injection $Prf : El\ o \to \mathtt{TYPE}$
  Ex : isRev : $\Pi x : El$ nat. El list $x \to El$ list $x \to El\ o$

- **Proposition** $P : El\ o$, **preuve** de P de type $Prf\ \ P$

Encodage prélude

- Signature Σ_{pre} contient Set, El, o, Prf, $\leadsto_d$, $\Rightarrow_d$, π, $\forall$

- Règles de réécriture $\mathcal{R}_{pre}$

$$El\ (x \leadsto_d y) \hookrightarrow \Pi z : El\ x.\ El\ (y\ z)$$

$$El\ (\pi\ x\ y) \hookrightarrow \Pi z : Prf\ x.\ El\ (y\ z)$$

$$Prf\ (x \Rightarrow_d y) \hookrightarrow \Pi z : Prf\ x.\ Prf\ (y\ z)$$

$$Prf\ (\forall\ x\ y) \hookrightarrow \Pi z : El\ x.\ Prf\ (y\ z)$$

Petits types

- **Petits types** : convertibles par $\mathcal{R}_{pre}$ à des types de la forme

$$\mathcal{S} ::= Set \mid \mathcal{S} \to \mathcal{S}$$

$$\mathcal{P} ::= Prf\ \ a \mid \mathcal{P} \to \mathcal{S} \mid \Pi z : \mathcal{S}.\ \mathcal{P}$$

$$\mathcal{E} ::= El\ \ b \mid \mathcal{E} \to \mathcal{S} \mid \Pi z : \mathcal{S}.\ \mathcal{E}$$

- $Set \to (Set \to Set)$ ✓
  $Prf\ \ a \to Prf\ \ b$ convertible à $Prf\ \ (a \Rightarrow_d (\lambda z : Prf\ \ a.\ b))$ ✓
  $Prf\ \ a \to Set \to Prf\ \ b$ ✗

Théorie avec encodage prélude

- Théorie $\mathcal{T} = (\Sigma, \mathcal{R})$ avec encodage prélude lorsque :
  – $\Sigma = \Sigma_{pre} \cup \Sigma_{\mathcal{T}}$
  – $\mathcal{R} = \mathcal{R}_{pre} \cup \mathcal{R}_{\mathcal{T}}$
  – pour chaque $c : A \in \Sigma_{\mathcal{T}}$, A est un petit type
  – pour chaque $\ell \hookrightarrow r \in \mathcal{R}_{\mathcal{T}}$, on a $\ell : A$ et $r : A$ avec A un petit type

- Contrainte **en général respectée** : théorie des ensembles, logique des prédicats, etc

Égalité(s)

- Dans le $\lambda\Pi$-calcul modulo théorie, on a une hiérarchie entre
  - les termes $u : A$
  - les types $A : \mathtt{TYPE}$

- Deux égalités : une pour les **termes**, une pour les **types**

Égalité entre termes

- **Hétérogène** : pour comparer des termes de types différents [McBride, 1999]

- Notation : $u \;{}_A{\approx}_B\; v$ avec $u : A$, $v : B$, $A : \texttt{TYPE}$ et $B : \texttt{TYPE}$

- Réflexive, symétrique, transitive

- Si les deux termes ont le même type, on a une égalité de **Leibniz**

$$\mathrm{leib}_A^{\mathrm{Prf}} : \Pi u, v : A.\ \Pi p : u \;{}_A{\approx}_A\; v.\ \Pi P : A \rightarrow El\ o.\ Prf\ (P\ u) \rightarrow Prf\ (P\ v)$$

Égalité entre petits types

- On ne peut **pas** définir d'égalité entre types dans le $\lambda\Pi$-calcul modulo théorie car $\mathtt{TYPE} \to \mathtt{TYPE} \to \mathtt{TYPE}$ n'est pas bien typé

- Égalité $\kappa(A, B)$ entre **petits types** A et B

$$\mathit{Prf}\ \ a \approx \mathit{Prf}\ \ b\ ✗\ \text{mais}\ a \approx b\ ✓$$

$$\mathit{El}\ \ a \approx \mathit{El}\ \ b\ ✗\ \text{mais}\ a \approx b\ ✓$$

Égalité entre petits types

- **Extensionnalité fonctionnelle** avec domaines différents

$$
\begin{aligned}
\mathrm{fun}_{A_1,A_2,B_1,B_2} \quad : \quad & \Pi f_1 : (\Pi x : A_1.\ B_1).\ \Pi f_2 : (\Pi y : A_2.\ B_2).\ \kappa(A_1, A_2) \\
& \to (\Pi x : A_1.\ \Pi y : A_2.\ x \approx y \to f_1\ x \approx f_2\ y) \\
& \to f_1 \approx f_2
\end{aligned}
$$

Remplacement des règles par des axiomes

- Pour toute **règle** $\ell \hookrightarrow r \in \mathcal{R}_{\mathcal{T}}$ (avec $\ell, r : A$ et les variables libres $x_1 : B_1, \ldots, x_n : B_n$), on prend l'**axiome**

$$\mathsf{eq}_{\ell r} : \Pi x_1 : B_1.\ \ldots.\ \Pi x_n : B_n.\ \ell\ {}_A{\approx}_A\ r$$

- Pour chaque **conversion** $A \equiv_{\beta\mathcal{R}} B$, on doit construire une **égalité** $p : \kappa(A, B)$

Typage de la conversion

- On explicite $A \equiv_{\beta\mathcal{R}} B$ en introduisant son **typage**
  - Règles de base : $\mathcal{R}$ et β-réduction
  - Règles pour la réflexivité, symétrie, transitivité
  - Règles de clôture (application, abstraction, produit)

- $p : \kappa(A, B)$ est construite grâce au typage de $A \equiv_{\beta\mathcal{R}} B$

- Soit $t : A$ et $p : \kappa(A, B)$ avec A et B des petits types.

  Il existe transp tel que :
  - transp $p\ t : B$
  - transp $p\ t\ {}_B{\approx}_A\ t$

- Idée : au lieu de convertir un terme, on lui **applique** un transport

Traduction des termes

- Relation $\bar{t} \lhd t$ (« $\bar{t}$ est une traduction de t ») définie par

$$\frac{}{x \lhd x} \qquad \frac{}{c \lhd c} \qquad \frac{\bar{t} \lhd t \qquad \bar{u} \lhd u}{(\lambda x : \bar{t}.\ \bar{u}) \lhd (\lambda x : t.\ u)} \qquad \frac{\bar{t} \lhd t \qquad \bar{u} \lhd u}{(\Pi x : \bar{t}.\ \bar{u}) \lhd (\Pi x : t.\ u)}$$

$$\frac{\bar{t} \lhd t \qquad \bar{u} \lhd u}{(\bar{t}\ \bar{u}) \lhd (t\ u)} \qquad \frac{\bar{t} \lhd t}{(\mathrm{transp}\ p\ \bar{t}) \lhd t}$$

- **Lemme** : si on a deux traductions $\bar{t}$ et $\bar{t}'$ de t, alors $\bar{t} \approx \bar{t}'$

Résultat

- Soit une théorie $\mathcal{T} = (\Sigma, \mathcal{R})$ du $\lambda\Pi$-calcul modulo théorie telle que
  - $\mathcal{T}$ est une théorie avec encodage prélude
  - avec des petits types
- Alors il existe une théorie $\mathcal{T}^{ax}$ **sans règles** de réécriture $\mathcal{R}_{\mathcal{T}}$ et **avec axiomes** d'égalité
- Telle que pour tout $t : A$ dans $\mathcal{T}$, il existe $\bar{t} : \bar{A}$ dans $\mathcal{T}^{ax}$, avec A et $\bar{A}$ qui expriment la même idée

Conclusion

Conclusion

- Deux contributions, un thème commun : le pouvoir des règles de réécriture

- Des travaux de recherches **pratiques** et **théoriques**
  - Formalisation de preuves faites à la main
  - Des questionnements sur le fondement des systèmes de preuve

- Deux **articles**

Perspectives

- Implémentation du remplacement des règles de réécriture par des axiomes
  $\hookrightarrow$ **Interopérabilité** entre systèmes de preuves

- Interprétation d'une théorie dans une autre pour montrer des résultats de **terminaison relative**

Références

ACZEL, P. (1988). ***Non well-founded sets.*** Center for the Study of Language and Information, Stanford.

BLANQUI, F., DOWEK, G., GRIENENBERGER, É., HONDET, G. et THIRÉ, F. (2023). **A modular construction of type theories.** *Logical Methods in Computer Science*, Volume 19, Issue 1.

CRABBÉ, M. (1974). **Non-normalisation de la théorie de zermelo.** Manuscript.

DOWEK, G. et MIQUEL, A. (2007). **Cut elimination for zermelo set theory.** Manuscript.

MCBRIDE, C. (1999). ***Dependently Typed Functional Programs and their Proofs.*** Thèse de doctorat, University of Edinburgh.

OURY, N. (2005). **Extensionality in the Calculus of Constructions.** *In* HURD, J. et MELHAM, T., éditeurs : *Theorem Proving in Higher Order Logics*, pages 278–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

WINTERHALTER, T., SOZEAU, M. et TABAREAU, N. (2019). **Eliminating Reflection from Type Theory.** *In CPP 2019 - 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 91–103, Lisbonne, Portugal. ACM.