From Rewrite Rules to Axioms in the $\lambda\Pi$-Calculus Modulo Theory

FoSSaCS 2024

Thomas Traversié
joint work with Valentin Blot, Gilles Dowek and Théo Winterhalter

Equational axioms or rewrite rules?

- For Poincaré, deriving $2+2=4$ is not a meaningful proof, but a simple verification

- Two families of logical systems

With equational axioms

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

We **prove** that $2+2=4$

With rewrite rules

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

We **compute** that $(2+2=4) \equiv (4=4)$

Equational axioms or rewrite rules?

- For Poincaré, deriving $2+2=4$ is not a meaningful proof, but a simple verification

- Two families of logical systems

With equational axioms	**With rewrite rules**
$x + \mathit{succ}\ y = \mathit{succ}\ (x+y)$	$x + \mathit{succ}\ y \hookrightarrow \mathit{succ}\ (x+y)$
$x + 0 = x$	$x + 0 \hookrightarrow x$
We **prove** that $2+2=4$	We **compute** that $(2+2=4) \equiv (4=4)$
If $\ell : \mathit{list}\ (2+2)$	If $\ell : \mathit{list}\ (2+2)$
but not necessarily $\ell : \mathit{list}\ 4$	then $\ell : \mathit{list}\ 4$

The $\lambda\Pi$-calculus modulo theory

- The $\lambda\Pi$-calculus modulo theory [Cousineau and Dowek, 2007]
  - λ-calculus + dependent types + rewrite rules
  - Implemented in DEDUKTI [Assaf et al, 2016]

- **Logical framework**
  - Possible to express many theories
  - Application: proof interoperability via DEDUKTI

- User-friendly framework
  - **Deduction** $\rightarrow$ user
  - **Computation** $\rightarrow$ system

In this paper

- Theoretical motivation: Is a result **provable** with rewrite rules also provable with axioms?
- Practical motivation: **Interoperability** between proof systems via DEDUKTI
- Contribution:

Rewrite rules **can be replaced** by equational axioms
in the $\lambda\Pi$-calculus modulo theory with a prelude encoding

Related work

- **Deduction modulo theory** = first-order **predicate logic** + rewrite rules
  $\hookrightarrow$ Rewrite rules can be replaced by axioms [Dowek et al, 2003]

- Translations of **extensional** type theory into **intensional** type theory [Oury, 2005, Winterhalter et al, 2019]

Outline

1. The $\lambda\Pi$-calculus modulo theory
2. Equality in the $\lambda\Pi$-calculus modulo theory
3. Replacement of user-defined rewrite rules by equational axioms

The $\lambda\Pi$-calculus modulo theory

The $\lambda\Pi$-calculus modulo theory

- Syntax

Sorts	$s ::= \mathtt{TYPE} \mid \mathtt{KIND}$
Terms	$t, u, A, B ::= c \mid x \mid s \mid \Pi x : A.\ B \mid \lambda x : A.\ t \mid t\ u$
Signatures	$\Sigma ::= \langle\rangle \mid \Sigma, c : A \mid \Sigma, \ell \hookrightarrow r$

$\Pi x : A.\ B$ written $A \to B$ if x not in B

- Theory $\mathcal{T}$ defined by a well-formed signature Σ

- Careful!
  - **No identity types**
  - **Finite hierarchy of sorts** $\mathtt{TYPE} : \mathtt{KIND}$

Type system

- Typing rules for dependent λ-calculus

- Conversion rule

$$\frac{\Gamma \vdash t : A \qquad (\Gamma \vdash A : s) \equiv (\Gamma \vdash B : s)}{\Gamma \vdash t : B}\ [\textsc{Conv}]$$

- **Convertibility rules** for building $(\Gamma \vdash u : A) \equiv (\Delta \vdash v : B)$
  - Generated by β-reduction and the rewrite rules of Σ
  - Closed by context, reflexive, symmetric and transitive

Prelude encoding Σ_{pre}

- Encoding of the notions of **proposition** and **proof** [Blanqui et al, 2023]
  $\hookrightarrow$ Always used in practice

- Universe of **sorts** Set with injection $El : Set \to \texttt{TYPE}$
  $\hookrightarrow$ Sort of propositions o, proposition P of type $El\ o$

- Universe of **propositions** $El\ o$ with injection $Prf : El\ o \to \texttt{TYPE}$
  $\hookrightarrow$ A proof of P is of type $Prf\ \ P$

- Arrows and quantifiers

$$El\ (a \leadsto_d b) \hookrightarrow \Pi z : El\ a.\ El\ (b\ z) \qquad El\ (\pi\ a\ b) \hookrightarrow \Pi z : Prf\ a.\ El\ (b\ z)$$
$$Prf\ (a \Rightarrow_d b) \hookrightarrow \Pi z : Prf\ a.\ Prf\ (b\ z) \qquad Prf\ (\forall\ a\ b) \hookrightarrow \Pi z : El\ a.\ Prf\ (b\ z)$$

Example: natural numbers and lists

$$\begin{array}{lll} \mathsf{nat} : \mathit{Set} & + : \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} & \mathsf{list} : \mathit{El}\ \mathsf{nat} \to \mathit{Set} \\ 0 : \mathit{El}\ \mathsf{nat} & x + 0 \hookrightarrow x & \mathsf{nil} : \mathit{El}\ (\mathsf{list}\ 0) \\ \mathsf{succ} : \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} & x + \mathsf{succ}\ y \hookrightarrow \mathsf{succ}\ (x + y) & \end{array}$$

$$\mathsf{cons} : \Pi x : \mathit{El}\ \mathsf{nat}.\ \mathit{El}\ \mathsf{list}\ x \to \mathit{El}\ \mathsf{nat} \to \mathit{El}\ (\mathsf{list}\ (\mathsf{succ}\ x))$$

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

- We have $\ell : \mathit{El}\ \mathsf{list}\ (\mathsf{succ}\ 0) \vdash \mathsf{concat}\ (\mathsf{succ}\ 0)\ 0\ \ell\ \mathsf{nil} : \mathit{El}\ \mathsf{list}\ (\mathsf{succ}\ 0 + 0)$
- We have $[\vdash \mathsf{succ}\ 0 + 0 : \mathit{El}\ \mathsf{nat}] \equiv [\vdash \mathsf{succ}\ 0 : \mathit{El}\ \mathsf{nat}]$

Example: natural numbers and lists

$$\begin{array}{lll} \mathsf{nat} : \mathit{Set} & + : \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} & \mathsf{list} : \mathit{El}\ \mathsf{nat} \to \mathit{Set} \\ 0 : \mathit{El}\ \mathsf{nat} & x + 0 \hookrightarrow x & \mathsf{nil} : \mathit{El}\ (\mathsf{list}\ 0) \\ \mathsf{succ} : \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} & x + \mathsf{succ}\ y \hookrightarrow \mathsf{succ}\ (x + y) & \end{array}$$

$$\mathsf{cons} : \Pi x : \mathit{El}\ \mathsf{nat}.\ \mathit{El}\ \mathsf{list}\ x \to \mathit{El}\ \mathsf{nat} \to \mathit{El}\ (\mathsf{list}\ (\mathsf{succ}\ x))$$

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

- We have $\ell : \mathit{El}\ \mathsf{list}\ (\mathsf{succ}\ 0) \vdash \mathsf{concat}\ (\mathsf{succ}\ 0)\ 0\ \ell\ \mathsf{nil} : \mathit{El}\ \mathsf{list}\ (\mathsf{succ}\ 0 + 0)$
- We have $[\vdash \mathsf{list}\ (\mathsf{succ}\ 0 + 0) : \mathit{Set}] \equiv [\vdash \mathsf{list}\ (\mathsf{succ}\ 0) : \mathit{Set}]$

Example: natural numbers and lists

$$\begin{array}{lll}
\mathsf{nat} : \mathit{Set} & + : \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} & \mathsf{list} : \mathit{El}\ \mathsf{nat} \to \mathit{Set} \\
0 : \mathit{El}\ \mathsf{nat} & x + 0 \hookrightarrow x & \mathsf{nil} : \mathit{El}\ (\mathsf{list}\ 0) \\
\mathsf{succ} : \mathit{El}\ \mathsf{nat} \to \mathit{El}\ \mathsf{nat} & x + \mathsf{succ}\ y \hookrightarrow \mathsf{succ}\ (x + y) &
\end{array}$$

$$\mathsf{cons} : \Pi x : \mathit{El}\ \mathsf{nat}.\ \mathit{El}\ \mathsf{list}\ x \to \mathit{El}\ \mathsf{nat} \to \mathit{El}\ (\mathsf{list}\ (\mathsf{succ}\ x))$$

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

- We have $\ell : \mathit{El}\ \mathsf{list}\ (\mathsf{succ}\ 0) \vdash \mathsf{concat}\ (\mathsf{succ}\ 0)\ 0\ \ell\ \mathsf{nil} : \mathit{El}\ \mathsf{list}\ (\mathsf{succ}\ 0 + 0)$
- We have $[\vdash \mathit{El}\ (\mathsf{list}\ (\mathsf{succ}\ 0 + 0)) : \mathtt{TYPE}] \equiv [\vdash \mathit{El}\ (\mathsf{list}\ (\mathsf{succ}\ 0)) : \mathtt{TYPE}]$

Method

- Goal: replace user-defined rewrite rules by equational axioms

- In the signature: replace each user-defined **rewrite rule** $\ell \hookrightarrow r$ by an **equational axiom** $\ell = r$

- In the derivations: replace each use of the **conversion rule**

  "from $t : A$ we get $t : B$ with $A \equiv B$"

  by the insertion of a **transport**

  "from $t : A$ we get $\mathsf{transp}\ p\ t : B$ with $p : A = B$"

Equality

Two equalities

- In the $\lambda\Pi$-calculus modulo theory, we have a hierarchy between
  - objects $u : A$
  - types $A : \mathtt{TYPE}$
- Two equalities: one for **objects**, one for **types**

Equality between objects

- **Heterogeneous**: to compare objects of different types [McBride, 1999]
- Notation: $u\ {}_A{\approx}_B\ v$ with $u : A$, $v : B$, $A : \texttt{TYPE}$ and $B : \texttt{TYPE}$
- Axioms: reflexivity, symmetry, transitivity
- Additional axiom: in the homogeneous case, it is a **Leibniz** equality

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

Equality between types

- We **cannot** define an equality between types since $\mathtt{TYPE} \to \mathtt{TYPE} \to \mathtt{TYPE}$ is ill-typed

- Intuition:

$$\begin{array}{ccc} Prf\ \ a \approx Prf\ \ b\ ✗ & \text{but} & a \approx b\ ✓ \\ El\ \ a \approx El\ \ b\ ✗ & \text{but} & a \approx b\ ✓ \end{array}$$

Small types

- **Small types**: types convertible using Σ_{pre} with types of the form

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

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

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

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

- In practice, all types are small

Equality between small types

- Equality $\kappa(A, B)$ between **small types** A et B

$$\kappa(Prf\ a_1, Prf\ a_2) \coloneqq a_1 \approx a_2 \qquad \kappa(El\ a_1, El\ a_2) \coloneqq a_1 \approx a_2 \qquad \kappa(S, S) \coloneqq \text{True if } S \in \mathcal{S}$$

$$\kappa(T_1 \to S, T_2 \to S) \coloneqq \kappa(T_1, T_2) \text{ if } S \in \mathcal{S}$$

$$\kappa(\Pi z : S.\ T_1, \Pi z : S.\ T_2) \coloneqq \Pi z : S.\ \kappa(T_1, T_2) \text{ if } S \in \mathcal{S}$$

- Axiom: **Functional extensionality** with different domains

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

Replacing rewrite rules by equational axioms

Transports

- Lemma: Let $t : A$ and $p : \kappa(A, B)$ with small types A and B.

  There exists a term transp p t such that:
  – transp p $t : B$
  – transp p t ${}_B{\approx}_A$ t

- Idea of the translation: **insert** transports in the terms

Translation of terms

- Relation $\bar{t} \triangleleft t$ ("$\bar{t}$ is a translation of t")

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

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

No more conversion rules!

- **Lemma**: if $\bar{t}$ and $\bar{t}'$ are two translations of t, then $\bar{t} \approx \bar{t}'$

Translation of signatures

$$\frac{}{\langle\rangle \ \triangleleft \ \langle\rangle} \qquad \frac{\bar{\bar{\Sigma}} \ \triangleleft \ \Sigma \qquad \bar{A} \ \triangleleft \ A}{(\bar{\bar{\Sigma}}, c : \bar{A}) \ \triangleleft \ (\Sigma, c : A)}$$

When $\ell, r : A$ with free variables $\boldsymbol{x} : \boldsymbol{B}$

$$\frac{\bar{\bar{\Sigma}} \ \triangleleft \ \Sigma \qquad \bar{\ell} \ \triangleleft \ \ell \qquad \bar{r} \ \triangleleft \ r \qquad \bar{\boldsymbol{B}} \ \triangleleft \ \boldsymbol{B} \qquad \bar{A} \ \triangleleft \ A}{(\bar{\bar{\Sigma}}, \mathrm{eq}_{\ell r} : \Pi \boldsymbol{x} : \bar{\boldsymbol{B}}.\ \bar{\ell} \ {}_{\bar{A}}{\approx}_{\bar{A}} \ \bar{r}) \ \triangleleft \ (\Sigma, \ell \hookrightarrow r)}$$

No more rewrite rules!

Main result

Let a theory $\mathcal{T} = (\Sigma_{pre} \cup \Sigma_{\mathcal{T}})$ such that all types are small.

- There exists a theory $\mathcal{T}^{ax} = (\Sigma_{pre} \cup \Sigma_{eq} \cup \bar{\Sigma}_{\mathcal{T}})$
  with Σ_{pre} the signature defining the equalities
- For every $A \equiv B$ in $\mathcal{T}$ with A and B small types, there exists some $p : \kappa(\bar{A}, \bar{B})$ in $\mathcal{T}^{ax}$
- For every $t : A$ in $\mathcal{T}$, we have $\bar{t} : \bar{A}$ in $\mathcal{T}^{ax}$

Axiomatized theory $\mathcal{T}^{ax}$

- Fully **axiomatized** user-defined signature $\bar{\Sigma}_{\mathcal{T}}$
  $\hookrightarrow$ Only the 4 rules of the prelude encoding in $\mathcal{T}^{ax}$
- Conservativity: $\mathcal{T}$ is **conservative** over $\mathcal{T}^{ax}$
- Relative consistency: if $\mathcal{T}^{ax}$ is **consistent** then $\mathcal{T}$ is also consistent

Conclusion

Takeaway message

- The $\lambda\Pi$-calculus modulo theory
  - **General** logical framework
  - Finite hierarchy of sorts and no identity types

- User-defined rewrite rules **can** be replaced by equational axioms
  $\hookrightarrow$ In practice, theories with prelude encoding and small types

- Application: interoperability *via* DEDUKTI

Check out the paper for more details!