Replacing Rewrite Rules by Equational Axioms in the $\lambda\Pi$-Calculus Modulo Theory

LMF PhD Seminar

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

May 28th 2024

A proof of $2 + 2 = 4$

- Axioms

$$x + \text{succ } y = \text{succ } (x + y)$$

$$x + 0 = x$$

- Deduction

$$\begin{aligned} 2 + 2 &:= \text{succ}^2\ 0 + \text{succ}^2\ 0 \\ &= \text{succ } (\text{succ}^2\ 0 + \text{succ } 0) \\ &= \text{succ}^2\ (\text{succ}^2\ 0 + 0) \\ &= \text{succ}^2\ (\text{succ}^2\ 0) \\ &:= 4 \end{aligned}$$

Another proof of $2 + 2 = 4$

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

- Rewrite rules

$$x + \mathsf{succ}\ y \hookrightarrow \mathsf{succ}\ (x + y)$$
$$x + 0 \hookrightarrow x$$

- Computation

$$\begin{array}{rcl} 2 + 2 & := & \mathsf{succ}^2\ 0 + \mathsf{succ}^2\ 0 \\ & \equiv & \mathsf{succ}\ (\mathsf{succ}^2\ 0 + \mathsf{succ}\ 0) \\ & \equiv & \mathsf{succ}^2\ (\mathsf{succ}^2\ 0 + 0) \\ & \equiv & \mathsf{succ}^2\ (\mathsf{succ}^2\ 0) \\ & := & 4 \end{array}$$

so $2 + 2 = 4$ using the reflexivity of equality

Equational axioms or rewrite rules?

Logical systems
with equational axioms

$$x + \mathsf{succ}\ y = \mathsf{succ}\ (x + y)$$
$$x + 0 = x$$

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

Logical systems
with rewrite rules

$$x + \mathsf{succ}\ y \hookrightarrow \mathsf{succ}\ (x + y)$$
$$x + 0 \hookrightarrow x$$

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

Equational axioms or rewrite rules?

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

Equational axioms or rewrite rules?

Logical systems with equational axioms

$$x + \mathrm{succ}\ y = \mathrm{succ}\ (x + y)$$
$$x + 0 = x$$

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

If $\ell : \mathit{list}\ (2 + 2)$
then $\mathrm{transp}\ e\ \ell : \mathit{list}\ 4$
with $e : 2 + 2 = 4$ and
$\mathrm{transp} : (2 + 2 = 4) \to \mathit{list}\ (2 + 2) \to \mathit{list}\ 4$

Logical systems with rewrite rules

$$x + \mathrm{succ}\ y \hookrightarrow \mathrm{succ}\ (x + y)$$
$$x + 0 \hookrightarrow x$$

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

If $\ell : \mathit{list}\ (2 + 2)$
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

- **Logical framework**
  – Possible to express many theories
  – Application: proof interoperability
  – Implemented in DEDUKTI [Assaf et al, 2016]

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

In this work

- Theoretical motivation: Is a result **provable** with rewrite rules also provable with axioms?
- Practical motivation: **Interoperability** between proof systems via DEDUKTI
- Contribution [FoSSaCS 2024]

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]

  – In extensional type theory, $\ell = r$ entails $\ell \equiv r$
  – In the $\lambda\Pi$-calculus modulo theory, $\ell \hookrightarrow r$ entails $\ell \equiv r$

Outline

Outline

Outline

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	$\Sigma ::= \langle\rangle \mid \Sigma, c : A \mid \Sigma, \ell \hookrightarrow r$
Contexts	$\Gamma ::= \langle\rangle \mid \Gamma, x : A$

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

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

Theories of the $\lambda\Pi$-calculus modulo theory

- $\hookrightarrow_{\beta\Sigma}$ is generated by β-reduction and the rewrite rules of Σ

- Theory $\mathcal{T}$ defined by a signature Σ such that:
  – for each $\ell \hookrightarrow r \in \Sigma$, the constants that occur in ℓ and r belong to Σ
  – the relation $\hookrightarrow_{\beta\Sigma}$ is confluent
  – each rule of Σ preserves typing

Typing rules

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

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

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

Convertibility rules

- **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

Outline

Encoding of the notions of proposition and proof

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

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

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

Rewrite rules of the encoding

- Desired behaviour:
  - Functionality $El\ (a \rightsquigarrow b) \hookrightarrow El\ a \to El\ b$
  - Implication $Prf\ (a \Rightarrow b) \hookrightarrow Prf\ a \to Prf\ b$
  - Universal quantifier $Prf\ (\forall\ a\ b) \hookrightarrow \Pi z : El\ a.\ Prf\ (b\ z)$

- Four constants and rewrite rules

$$El\ (a \rightsquigarrow_d b) \hookrightarrow \Pi z : El\ a.\ El\ (b\ z)$$

$$Prf\ (a \Rightarrow_d b) \hookrightarrow \Pi z : Prf\ a.\ Prf\ (b\ z)$$

$$Prf\ (\forall\ a\ b) \hookrightarrow \Pi z : El\ a.\ Prf\ (b\ z)$$

$$El\ (\pi\ a\ b) \hookrightarrow \Pi z : Prf\ a.\ El\ (b\ z)$$

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 $[\vdash \mathit{El}\ (\mathsf{list}\ (\mathsf{succ}\ 0 + 0)) : \mathtt{TYPE}] \equiv [\vdash \mathit{El}\ (\mathsf{list}\ (\mathsf{succ}\ 0)) : \mathtt{TYPE}]$

How to 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 $t : A$ and $A \equiv B$ we get $t : B$"

  by the insertion of a **transport**

  "from $t : A$ and $p : A = B$ we get $\mathrm{transp}\ p\ t : B$"

Outline

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

- We need two equalities:
  - one for **objects**
  - one for **types**

Outline

Equality between objects

- **Heterogeneous**: to compare objects of different types [McBride, 1999]

- Notation: $u \;{}_A{\approx}_B\; v$ with $u : A$, $v : B$, $A : \mathtt{TYPE}$ and $B : \mathtt{TYPE}$

- Axioms for reflexivity, symmetry, transitivity

$$\mathrm{refl}_A : \Pi u : A.\ u \;{}_A{\approx}_A\; u$$

$$\mathrm{sym}_{A,B} : \Pi u : A.\ \Pi v : B.\ u \;{}_A{\approx}_B\; v \rightarrow v \;{}_B{\approx}_A\; u$$

$$\mathrm{trans}_{A,B,C} : \Pi u : A.\ \Pi v : B.\ \Pi w : C.\ u \;{}_A{\approx}_B\; v \rightarrow v \;{}_B{\approx}_C\; w \rightarrow u \;{}_A{\approx}_C\; w$$

Axioms of equality between objects

- In the homogeneous case, it is a **Leibniz** equality

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

$$\begin{aligned}\mathrm{eqLeib}^{\mathrm{Prf}}_{A} \quad : \quad & \Pi u, v : A.\ \Pi p : u\ {}_{A}{\approx}_{A}\ v.\ \Pi P : A \to El\ o.\ \Pi t : Prf\ (P\ u). \\ & \mathrm{leib}^{\mathrm{Prf}}_{A}\ u\ v\ p\ P\ t\ {}_{Prf\ (P\ v)}{\approx}_{Prf\ (P\ u)}\ t\end{aligned}$$

- Congruence for application

$$\begin{aligned}\mathrm{app}_{A_1, A_2, B_1, B_2} \quad : \quad & \Pi t_1 : (\Pi x : A_1.\ B_1).\ \Pi t_2 : (\Pi x : A_2.\ B_2). \\ & \Pi u_1 : A_1.\ \Pi u_2 : A_2. \\ & t_1\ \approx\ t_2 \\ & \to u_1\ \approx\ u_2 \\ & \to t_1\ u_1\ {}_{B_1[x \mapsto u_1]}{\approx}_{B_2[x \mapsto u_2]}\ t_2\ u_2\end{aligned}$$

Outline

Equality between types

- We **cannot** define an equality between types
  $\hookrightarrow$ It would have type $\mathtt{TYPE} \to \mathtt{TYPE} \to \mathtt{TYPE}$, which is ill-typed

- But we **can** compare objects of type *Set* or *El o* using $\approx$

- Intuition:

$Prf\ a \approx Prf\ b$ ✗	but	$a \approx b$ ✓
$El\ a \approx El\ b$ ✗	but	$a \approx b$ ✓
$\Pi x : El\ a_1.\ Prf\ a_2 \approx \Pi x : El\ b_1.\ Prf\ b_2$ ✗	but	??
$Prf\ a_1 \to Prf\ a_2 \approx Prf\ b_1 \to Prf\ b_2$ ✗	but	??

Transforming types

- Representing dependent types with $\leadsto_d$, $\Rightarrow_d$, π or $\forall$ whenever possible

$$\nu(Set) := Set \qquad \nu(Prf\ a) := Prf\ a \qquad \nu(El\ a) := El\ a$$

$$\nu(\Pi x : A.\ B) := \begin{cases} Prf\ (a \Rightarrow_d (\lambda x : Prf\ a.\ b)) & \text{if } \nu(A) = Prf\ a \text{ and } \nu(B) = Prf\ b \\ El\ (a \leadsto_d (\lambda x : El\ a.\ b)) & \text{if } \nu(A) = El\ a \text{ and } \nu(B) = El\ b \\ Prf\ (\forall\ a\ (\lambda x : El\ a.\ b)) & \text{if } \nu(A) = El\ a \text{ and } \nu(B) = Prf\ b \\ El\ (\pi\ a\ (\lambda x : Prf\ a.\ b)) & \text{if } \nu(A) = Prf\ a \text{ and } \nu(B) = El\ b \\ \Pi x : \nu(A).\ \nu(B) & \text{otherwise} \end{cases}$$

- Using the four rewrite rules of Σ_{pre}, we have $A \equiv \nu(A)$

Small types

- **Small types**: types A such that $\nu(A)$ is defined and generated by

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

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

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

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

- In practice, all types are small

Equality between small types

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

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

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

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

- Example

$$\kappa(\Pi x : Set.\ Prf\ P \to Prf\ Q, \Pi x : Set.\ Prf\ R) := \Pi x : Set.\ (P \Rightarrow_d \lambda z : P.\ Q) \approx R$$

Axioms of equality between small types

- **Functional extensionality** with different domains

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

- If A is generated by $\mathcal{S}$, we simply have

$$
\begin{aligned}
\mathsf{fun}_{A,B_1,B_2} \quad : \quad & \Pi f_1 : (\Pi x : A.\ B_1).\ \Pi f_2 : (\Pi x : A.\ B_2).\\
& (\Pi x : A.\ f_1\ x \approx f_2\ x)\\
& \to f_1 \approx f_2
\end{aligned}
$$

Outline

Transports

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

  We can build a term transp $p\ t$ such that:
  – $\Gamma \vdash \text{transp}\ p\ t : B$
  – $\Gamma \vdash \text{transp}\ p\ t\ {}_{B}{\approx}_{A}\ t$

- Idea of the translation: **insert** transports in the terms each time $\textsc{Conv}$ is used

Outline

Translation

- Translation of terms $\bar{t} \triangleleft t$ ("$\bar{t}$ 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!

- Translation of contexts

$$\frac{}{\langle\rangle \triangleleft \langle\rangle} \qquad \frac{\bar{\Gamma} \triangleleft \Gamma \qquad \bar{A} \triangleleft A}{(\bar{\Gamma}, x : \bar{A}) \triangleleft (\Gamma, x : A)}$$

- **Equal translations**

  If $\bar{t}$ and $\bar{t}'$ are two translations of t, then $\bar{t} \approx \bar{t}'$

- **Switching translations**

  If $\bar{\Gamma} \vdash \bar{t} : \overline{A}$ and $\bar{\Gamma} \vdash \overline{A}' : \mathtt{TYPE}$, then there exists $\bar{t}' \lhd t$ such that $\bar{\Gamma} \vdash \bar{t}' : \overline{A}'$

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!

Outline

From rewrite rules to axioms

Let a theory $\mathcal{T} = (\Sigma_{pre} \cup \Sigma_{\mathcal{T}})$ a theory with prelude encoding.

Suppose that all types are small.

- There exists a signature $\bar{\bar{\Sigma}}_{\mathcal{T}} \triangleleft \Sigma_{\mathcal{T}}$ such that $\mathcal{T}^{ax} = (\Sigma_{pre} \cup \Sigma_{eq} \cup \bar{\bar{\Sigma}}_{\mathcal{T}})$ is a theory
  - Σ_{pre} remains unchanged
  - Σ_{eq} is 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 $\Gamma \vdash t : A$ in $\mathcal{T}$, we have $\bar{\Gamma} \vdash \bar{t} : \bar{A}$ in $\mathcal{T}^{ax}$

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

- $\bar{\Sigma}_{\mathcal{T}}$ is a fully **axiomatized** user-defined signature
  $\hookrightarrow$ The only rewrite rules in $\mathcal{T}^{ax}$ are the 4 of the prelude encoding

- Conservativity: $\mathcal{T}$ is **conservative** over $\mathcal{T}^{ax}$

- Relative consistency: if $\mathcal{T}^{ax}$ is **consistent** then $\mathcal{T}$ is also consistent

Outline

- Logical framework
  - Theories can be defined by users using typed constants and rewrite rules
  - Many theories **can** be expressed
  - Examples: Predicate Logic, Calculus of Constructions

- Minimal logical framework
  - Finite hierarchy of sorts and no identity types
  - Heterogeneous equality between objects
  - **Difficult** to define an equality between types

Replacing rewrite rules by equational axioms

- Considered theories
  - Prelude encoding
  - Small types

  $\hookrightarrow$ In practice, **always** the case

- User-defined rewrite rules **can** be replaced by equational axioms

- Application: interoperability between proof systems via DEDUKTI