EuroProofNet

Encoding logics in $\lambda\Pi/\mathcal{R}$

Frédéric Blanqui

Deducteam

Encoding logics in $\lambda\Pi/\mathcal{R}$

we have seen what is a theory in the $\lambda\Pi$-calculus modulo rewriting:

- a signature mapping a number of symbols to their types
- a set of rewrite rules on those symbols

we are now going to see how to encode logics as $\lambda\Pi/\mathcal{R}$ theories

First-order logic

- the set of terms
  - built from a set of function symbols equipped with an arity
- the set of propositions
  - built from a set of predicate symbols equipped with an arity
  - and the logical connectives $\top$, $\bot$, $\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$, $\forall$, $\exists$
- the set of axioms (the actual theory)
- the subset of provable propositions
  - using deduction rules (e.g. natural deduction)

Natural deduction

provability, $\vdash$, is a relation between a sequence of propositions Γ (the assumptions) and a proposition B (the conclusion) inductively defined from introduction and elimination rules for each connective:

$$(\Rightarrow\text{-intro})\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad (\Rightarrow\text{-elim})\ \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$(\forall\text{-intro})\ \frac{\Gamma \vdash A \quad x \notin \Gamma}{\Gamma \vdash \forall x, A} \qquad (\forall\text{-elim})\ \frac{\Gamma \vdash \forall x, A}{\Gamma \vdash A\{(x, u)\}}$$

...

Encoding of first-order logic

- the set of terms ι : TYPE
  - built from a set of function symbols equipped with an arity

function symbol: $\iota \to \ldots \to \iota \to \iota$

Encoding of first-order logic

- the set of terms $\qquad I : \texttt{TYPE}$
  - built from a set of function symbols equipped with an arity

    function symbol: $I \to \ldots \to I \to I$

- the set of propositions $\qquad Prop : \texttt{TYPE}$
  - built from a set of predicate symbols equipped with an arity

    predicate symbol: $I \to \ldots \to I \to Prop$

Encoding of first-order logic

- the set of terms $I : \texttt{TYPE}$
  - built from a set of function symbols equipped with an arity
    function symbol: $I \to \ldots \to I \to I$

- the set of propositions $Prop : \texttt{TYPE}$
  - built from a set of predicate symbols equipped with an arity
    predicate symbol: $I \to \ldots \to I \to Prop$
  - and the logical connectives $\top$, $\bot$, $\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$, $\forall$, $\exists$
    $\top : Prop$, $\neg : Prop \to Prop$, $\forall : (I \to Prop) \to Prop$, ...
    we use λ-calculus to encode quantifiers:
    we encode $\forall x, A$ as $\forall(\lambda x : I, A)$

Encoding of first-order logic

- the set of terms I : TYPE
  - built from a set of function symbols equipped with an arity
    function symbol: $I \to \ldots \to I \to I$
- the set of propositions $Prop$: TYPE
  - built from a set of predicate symbols equipped with an arity
    predicate symbol: $I \to \ldots \to I \to Prop$
  - and the logical connectives $\top$, $\bot$, $\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$, $\forall$, $\exists$
    $\top : Prop$, $\neg : Prop \to Prop$, $\forall : (I \to Prop) \to Prop$, ...
    we use λ-calculus to encode quantifiers:
    we encode $\forall x, A$ as $\forall(\lambda x : I, A)$
- the set of axioms (the actual theory)
- the subset of provable propositions
  - using deduction rules (e.g. natural deduction)

but how to encode proofs?

Using λ-terms to represent proofs
(Curry-de Bruijn-Howard isomorphism)

by interpreting propositions as types ($\Rightarrow/\rightarrow$, $\forall/\Pi$)

the natural deduction rules

$$(\Rightarrow\text{-intro})\ \frac{\Gamma, \quad A \vdash \quad B}{\Gamma \vdash \qquad\qquad A \Rightarrow B}$$

$$(\Rightarrow\text{-elim})\ \frac{\Gamma \vdash \quad A \Rightarrow B \quad \Gamma \vdash \quad A}{\Gamma \vdash \quad B}$$

$$(\forall\text{-intro})\ \frac{\Gamma \vdash \quad A \qquad x \notin \Gamma}{\Gamma \vdash \qquad \forall x, A}$$

$$(\forall\text{-elim})\ \frac{\Gamma \vdash \quad \forall x, A}{\Gamma \vdash \qquad A\{(x, u)\}}$$

Using λ-terms to represent proofs
(Curry-de Bruijn-Howard isomorphism)

by interpreting propositions as types ($\Rightarrow/\rightarrow$, $\forall/\Pi$)

the natural deduction rules corresponds to the typing rules of $\lambda\Pi$:

$$(\Rightarrow\text{-intro})\ \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A, t : A \Rightarrow B}$$

$$(\Rightarrow\text{-elim})\ \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$$

$$(\forall\text{-intro})\ \frac{\Gamma \vdash t : A \qquad x \notin \Gamma}{\Gamma \vdash \lambda x, t : \forall x, A}$$

$$(\forall\text{-elim})\ \frac{\Gamma \vdash t : \forall x, A}{\Gamma \vdash tu : A\{(x, u)\}}$$

and proof checking is reduced to type checking

Expliciting the Brouwer-Heyting-Kolmogorov interpretation

terms of type *Prop* are not types. . .

but we can interpret a proposition as a type by applying:

$$\boxed{\mathit{Prf} : \mathit{Prop} \to \mathtt{TYPE}}$$

Prf A is the type of proofs of proposition A

Expliciting the Brouwer-Heyting-Kolmogorov interpretation

terms of type *Prop* are not types...

but we can interpret a proposition as a type by applying:

$$\boxed{Prf : Prop \to \mathtt{TYPE}}$$

Prf A is the type of proofs of proposition A

but

$$\lambda x : Prf\ A, x \quad : \quad Prf\ A \to Prf\ A$$

and

$$\lambda x : Prf\ A, x \quad \not: \quad Prf\,(A \Rightarrow A)$$

Expliciting the Brouwer-Heyting-Kolmogorov interpretation

terms of type *Prop* are not types...

but we can interpret a proposition as a type by applying:

$$\boxed{Prf : Prop \to \texttt{TYPE}}$$

$Prf\ A$ is the type of proofs of proposition A

but

$$\lambda x : Prf\ A, x \quad : \quad Prf\ A \to Prf\ A$$

and

$$\lambda x : Prf\ A, x \quad \not: \quad Prf(A \Rightarrow A)$$

unless we add the <u>rewrite rule</u>:

$$\boxed{Prf(A \Rightarrow B) \quad \hookrightarrow \quad Prf\ A \to Prf\ B}$$

Encoding $\Rightarrow$

because $Prf\,(A \Rightarrow B) \hookrightarrow Prf\ A \rightarrow Prf\ B$

the introduction rule for $\Rightarrow$ is the abstraction:

$$(\Rightarrow\text{-intro})\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \text{(conv)}\ \dfrac{\text{(abs)}\ \dfrac{\Gamma, x : Prf\ A \vdash t : Prf\ B}{\Gamma \vdash \lambda x : A, t : Prf\ A \rightarrow Prf\ B}}{\Gamma \vdash \lambda x : A, t : Prf\,(A \Rightarrow B)}$$

Encoding $\Rightarrow$

because $Prf\,(A \Rightarrow B) \hookrightarrow Prf\ A \rightarrow Prf\ B$

the introduction rule for $\Rightarrow$ is the abstraction:

$$(\Rightarrow\text{-intro})\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \begin{array}{l}(\text{abs})\\(\text{conv})\end{array} \frac{\dfrac{\Gamma, x : Prf\ A \vdash t : Prf\ B}{\Gamma \vdash \lambda x : A, t : Prf\ A \rightarrow Prf\ B}}{\Gamma \vdash \lambda x : A, t : Prf\,(A \Rightarrow B)}$$

the elimination rule for $\Rightarrow$ is the application:

$$(\Rightarrow\text{-elim})\ \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$(\text{app})\ \frac{(\text{conv})\ \dfrac{\Gamma \vdash t : Prf\,(A \Rightarrow B)}{\Gamma \vdash t : Prf\ A \rightarrow Prf\ B} \qquad \Gamma \vdash u : Prf\ A}{\Gamma \vdash tu : Prf\ B}$$

Encoding $\forall$

we can do something similar for $\forall : (I \to Prop) \to Prop$ by taking:

$$\boxed{Prf(\forall A) \quad \hookrightarrow \quad \Pi x : I, Prf(A\,x)}$$

Encoding the other connectives

the other connectives can be defined by using a meta-level quantification on propositions:

$$Prf(A \wedge B) \quad \hookrightarrow \quad \Pi C : Prop, (Prf\ A \rightarrow Prf\ B \rightarrow Prf\ C) \rightarrow Prf\ C$$

Encoding the other connectives

the other connectives can be defined by using a meta-level quantification on propositions:

$$Prf(A \wedge B) \quad \hookrightarrow \quad \Pi C : Prop, (Prf\ A \to Prf\ B \to Prf\ C) \to Prf\ C$$

introduction and elimination rules can be derived:

($\wedge$-intro):

$$\lambda a : Prf\ A, \lambda b : Prf\ B, \lambda C : Prop, \lambda h : Prf\ A \to Prf\ B \to Prf\ C, hab$$

is of type

$$Prf\ A \to Prf\ B \to Prf(A \wedge B)$$

Encoding the other connectives

the other connectives can be defined by using a meta-level quantification on propositions:

$$\boxed{\mathit{Prf}(A \wedge B) \quad \hookrightarrow \quad \Pi C : \mathit{Prop}, (\mathit{Prf}\ A \to \mathit{Prf}\ B \to \mathit{Prf}\ C) \to \mathit{Prf}\ C}$$

introduction and elimination rules can be derived:

($\wedge$-intro):

$$\lambda a : \mathit{Prf}\ A, \lambda b : \mathit{Prf}\ B, \lambda C : \mathit{Prop}, \lambda h : \mathit{Prf}\ A \to \mathit{Prf}\ B \to \mathit{Prf}\ C, hab$$

is of type

$$\mathit{Prf}\ A \to \mathit{Prf}\ B \to \mathit{Prf}(A \wedge B)$$

($\wedge$-elim1):

$$\lambda c : \mathit{Prf}(A \wedge B), c\, A\, (\lambda a : \mathit{Prf}\ A, \lambda b : \mathit{Prf}\ B, a)$$

is of type

$$\mathit{Prf}(A \wedge B) \to \mathit{Prf}\ A$$

To summarize: $\lambda\Pi/\mathcal{R}$-theory *FOL* for first-order logic

signature Σ_{FOL}:

$I : \mathtt{TYPE}$

$f : I \to \ldots \to I \to I$ for each function symbol f of arity n

$Prop : \mathtt{TYPE}$

$P : I \to \ldots \to I \to Prop$ for each predicate symbol P of arity n

$\top : Prop,\ \neg : Prop \to Prop,\ \forall : (I \to Prop) \to Prop, \ldots$

$Prf : Prop \to \mathtt{TYPE}$

$a : Prf\ A$ for each axiom A

rules $\mathcal{R}_{FOL}$:

$$
\begin{aligned}
Prf(A \Rightarrow B) &\hookrightarrow Prf\ A \to Prf\ B \\
Prf(\forall A) &\hookrightarrow \Pi x : I, Prf(A\,x) \\
Prf(A \wedge B) &\hookrightarrow \Pi C : Prop, (Prf\ A \to Prf\ B \to Prf\ C) \to Prf\ C \\
Prf\,\bot &\hookrightarrow \Pi C : Prop, Prf\ C \\
Prf(\neg A) &\hookrightarrow Prf\,A \to Prf\,\bot
\end{aligned}
$$

...

Encoding of first-order logic in $\lambda\Pi/FOL$

encoding of terms:

$$|x| = x$$
$$|ft_1 \dots t_n| = f|t_1| \dots |t_n|$$

encoding of propositions:

$$|Pt_1 \dots t_n| = P|t_1| \dots |t_n|$$
$$|\top| = \top$$
$$|A \wedge B| = |A| \wedge |B|$$
$$|\forall x, A| = \forall(\lambda x : I, |A|)$$
$$\dots$$
$$|\Gamma, A| = |\Gamma|, x_{\|\Gamma\|+1} : A$$

encoding of proofs:

$$\left| \frac{\pi_{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B} (\Rightarrow_i) \right| = \lambda x_{\|\Gamma\|+1} : Prf\ |A|, |\pi_{\Gamma, A \vdash B}|$$

$$\left| \frac{\pi_{\Gamma \vdash A \Rightarrow B} \quad \pi_{\Gamma \vdash A}}{\Gamma \vdash B} (\Rightarrow_e) \right| = |\pi_{\Gamma \vdash A \Rightarrow B}|\, |\pi_{\Gamma \vdash A}|$$

$\dots$

Properties of the encoding in $\lambda\Pi/FOL$

- a term is mapped to a term of type I
- a proposition is mapped to a term of type $Prop$
- a proof of A is mapped to a term of type $Prf\ |A|$

Properties of the encoding in $\lambda\Pi/FOL$

- a term is mapped to a term of type I
- a proposition is mapped to a term of type $Prop$
- a proof of A is mapped to a term of type $Prf\ |A|$

if we find t of type $Prf\ |A|$, can we deduce that A is provable ?

Properties of the encoding in $\lambda\Pi/FOL$

- a term is mapped to a term of type I
- a proposition is mapped to a term of type $Prop$
- a proof of A is mapped to a term of type $Prf\ |A|$

if we find t of type $Prf\ |A|$, can we deduce that A is provable ?

- yes, the encoding is **conservative:**
  if $Prf\ |A|$ is inhabited then A is provable

proof sketch: because $\hookrightarrow_{\beta\mathcal{R}}$ terminates and is confluent, t has a normal form, and terms in normal form can be easily translated back in first-order logic and natural deduction

Multi-sorted first-order logic

for each sort I_k (e.g. point, line, circle), add:

$I_k : \texttt{TYPE}$
$\forall_k : (I_k \to Prop) \to Prop$

$Prf(\forall_k A) \hookrightarrow \Pi x : I_k, Prf(Ax)$

Polymorphic first-order logic

same trick as for the BHK interpretation of propositions:

$Set : \mathtt{TYPE}$ — type of sorts
$El : Set \to \mathtt{TYPE}$ — interpretation of sorts as types
$\iota : Set$ — for each sort ι

$\forall : \Pi a : Set, (El\, a \to Prop) \to Prop$

$Prf\,(\forall a p) \hookrightarrow \Pi x : El\, a, Prf\,(p\, x)$

Higher-order logic

order	quantification on
1	elements
2	sets of elements
3	sets of sets of elements
...	...
ω	any set

Higher-order logic

order	quantification on
1	elements
2	sets of elements
3	sets of sets of elements
...	...
ω	any set

quantification on functions:

$\rightsquigarrow : Set \to Set \to Set$

$El(a \rightsquigarrow b) \hookrightarrow El\ a \to El\ b$

Higher-order logic

order	quantification on
1	elements
2	sets of elements
3	sets of sets of elements
...	...
ω	any set

quantification on functions:

$\leadsto\ :\ Set \to Set \to Set$

$El(a \leadsto b) \hookrightarrow El\ a \to El\ b$

quantification on propositions (e.g. $\forall p, p \Rightarrow p$):

$o\ :\ Set$

$El\ o \hookrightarrow Prop$

Encoding dependent types

dependent implication:

$\Rightarrow_d : \Pi a : \mathit{Prop}, (\mathit{Prf}\ a \rightarrow \mathit{Prop}) \rightarrow \mathit{Prop}$

$\mathit{Prf}\,(a \Rightarrow_d b) \hookrightarrow \Pi x : \mathit{Prf}\ a, \mathit{Prf}\,(b\,x)$

Encoding dependent types

dependent implication:

$\Rightarrow_d : \Pi a : \mathit{Prop}, (\mathit{Prf}\ a \to \mathit{Prop}) \to \mathit{Prop}$

$\mathit{Prf}(a \Rightarrow_d b) \hookrightarrow \Pi x : \mathit{Prf}\ a, \mathit{Prf}(b\, x)$

dependent types:

$\leadsto_d : \Pi a : \mathit{Set}, (\mathit{El}\ a \to \mathit{Set}) \to \mathit{Set}$

$\mathit{El}(a \leadsto_d b) \hookrightarrow \Pi x : \mathit{El}\ a, \mathit{El}(b\, x)$

Encoding dependent types

dependent implication:

$\Rightarrow_d : \Pi a : Prop, (Prf\ a \to Prop) \to Prop$

$Prf(a \Rightarrow_d b) \hookrightarrow \Pi x : Prf\ a, Prf(b\,x)$

dependent types:

$\leadsto_d : \Pi a : Set, (El\ a \to Set) \to Set$

$El(a \leadsto_d b) \hookrightarrow \Pi x : El\ a, El(b\,x)$

proofs in object-terms:

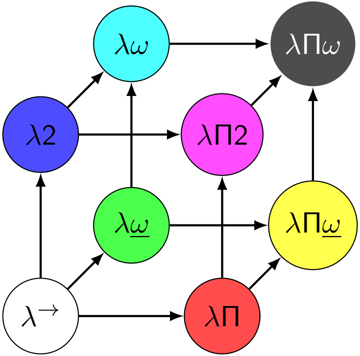
$\pi : \Pi p : Prop, (Prf\ p \to Set) \to Set$

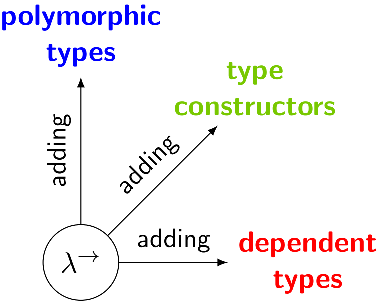
$El(\pi\ p\ a) \hookrightarrow \Pi h : Prf\ p, El(a\,h)$

example: $div : El(\iota \leadsto_d \lambda y : El\,\iota, \pi(y > 0)(\lambda h, \iota))$
takes 3 arguments: $x : El\,\iota$, $y : El\,\iota$, $h : Prf(y > 0)$
and returns a term of type $El\,\iota$

Encoding the systems of Barendregt's λ-cube

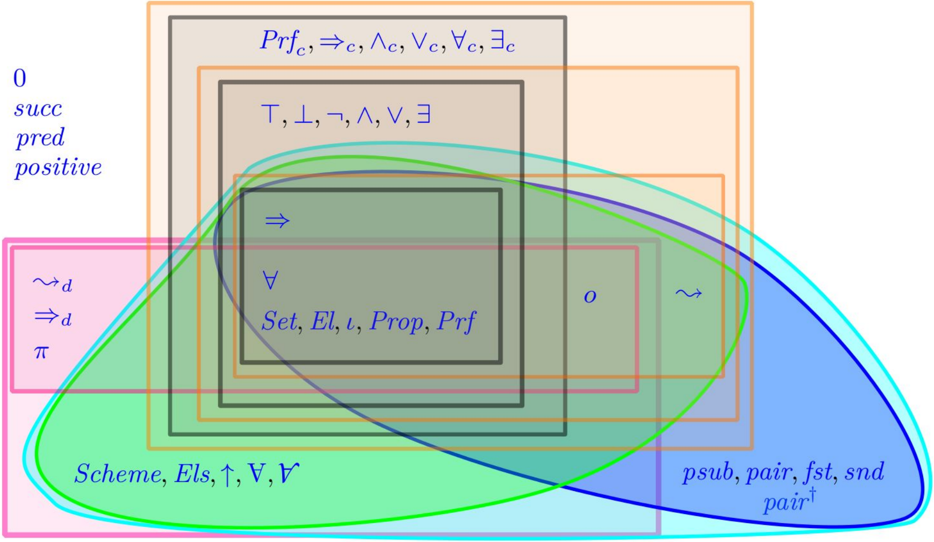
feature	**PTS rule**	$\lambda\Pi/\mathcal{R}$ **rule**
simple types	$\mathtt{TYPE}, \mathtt{TYPE}$	$\mathit{Prf}\,(a \Rightarrow_d b) \hookrightarrow \Pi x : \mathit{Prf}\ a, \mathit{Prf}\,(b\,x)$
polymorphic types	$\mathtt{KIND}, \mathtt{TYPE}$	$\mathit{Prf}\,(\forall\ a\ b) \hookrightarrow \Pi x : \mathit{El}\ a, \mathit{Prf}\,(b\,x)$
dependent types	$\mathtt{TYPE}, \mathtt{KIND}$	$\mathit{El}(\pi\ a\ b) \hookrightarrow \Pi x : \mathit{Prf}\ a, \mathit{El}(b\,x)$
type constructors	$\mathtt{KIND}, \mathtt{KIND}$	$\mathit{El}(a \leadsto_d b) \hookrightarrow \Pi x : \mathit{El}\ a, \mathit{El}(b\,x)$





The $\lambda\Pi/\mathcal{R}$ theory U and its sub-theories

38 symbols, 28 rules, 13 sub-theories



Encoding functional Pure Type Systems

terms and types:

$$t := x \mid tt \mid \lambda x : t, t \mid \Pi x : t, t \mid s \in \mathcal{S}$$

typing rules:

$$\frac{}{\emptyset \vdash} \qquad \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash} \qquad \frac{\Gamma \vdash \quad (x, A) \in \Gamma}{\Gamma \vdash x : A}$$

$$(sort)\ \frac{\Gamma \vdash \quad (s_1, s_2) \in \mathcal{A}}{\Gamma \vdash s_1 : s_2}$$

$$(prod)\ \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2 \quad ((s_1, s_2), s_3) \in \mathcal{P}}{\Gamma \vdash \Pi x : A, B : s_3}$$

$$\frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash \Pi x : A, B : s}{\Gamma \vdash \lambda x : A, t : \Pi x : A, B} \qquad \frac{\Gamma \vdash t : \Pi x : A, B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B\{(x, u)\}}$$

$$\frac{\Gamma \vdash t : A \quad A \simeq_\beta A' \quad \Gamma \vdash A' : s}{\Gamma \vdash t : A'}$$

Encoding functional Pure Type Systems

(Cousineau & Dowek, 2007)

signature:

U_s : TYPE — for each sort $s \in \mathcal{S}$

$El_s : U_s \to \mathtt{TYPE}$

$s_1 : U_{s_2}$ — for every $(s_1, s_2) \in \mathcal{A}$

$\pi_{s_1,s_2} : \Pi a : U_{s_1}, (El_{s_1}\, a \to U_{s_2}) \to U_{s_3}$ — for every $(s_1, s_2, s_3) \in \mathcal{P}$

rules:

$El_{s_2}\, s_1 \hookrightarrow U_{s_1}$ — for every $(s_1, s_2) \in \mathcal{A}$

$El_{s_3}(\pi_{s_1,s_2}\, a\, b) \hookrightarrow \Pi x : El_{s_1}\, a, El_{s_2}(b\, x)$ — for every $(s_1, s_2, s_3) \in \mathcal{P}$

encoding:

$|x|_\Gamma = x$

$|s|_\Gamma = s$

$|\lambda x : A, t|_\Gamma = \lambda x : El_s |A|_\Gamma, |t|_{\Gamma, x:A}$ — if $\Gamma \vdash A : s$

$|tu|_\Gamma = |t|_\Gamma |u|_\Gamma$

$|\Pi x : A, B|_\Gamma = \pi_{s_1,s_2} |A|_\Gamma (\lambda x : El_{s_1} |A|_\Gamma, |B|_{\Gamma, x:A})$
if $\Gamma \vdash A : s_1$ and $\Gamma, x : A \vdash B : s_2$

Encoding other features

- recursive functions (Assaf 2015, Cauderlier 2016, Férey 2021)
  - different approaches, no general theory yet
  - encoding in recursors instead ? (cf. Sozeau, Cockx, ...)
- universe polymorphism (Genestier 2020)
  - requires rewriting with matching modulo AC
    or rewriting on AC canonical forms (Blanqui 2022)
- η-conversion on function types (Genestier 2020)
- predicate subtyping with proof irrelevance (Hondet 2020)
- co-inductive objects and co-recursion (Felicissimo 2021)