Martin-Löf *à la* Coq

Mechanized normalisation for a dependently typed language

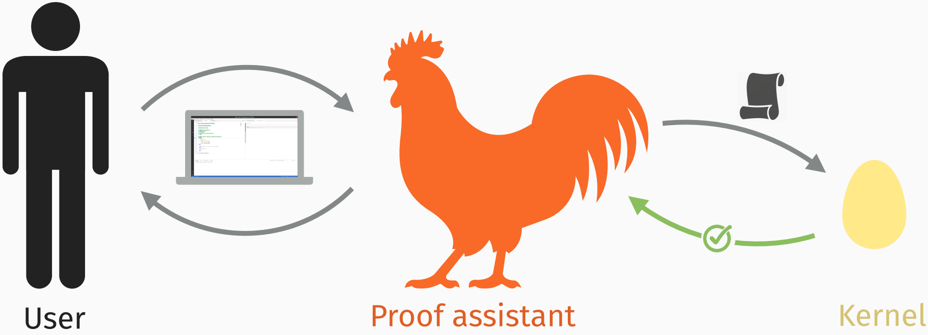
Arthur Adjedj[1] **Meven Lennon-Bertrand[2]** Kenji Maillard[3] Pierre-Marie Pédrot[3] Loïc Pujet[4]

[1]ENS Paris-Saclay [2]University of Cambridge [3]Inria [4]Stockholm University

Journée Reciprog – 03 juin 2024

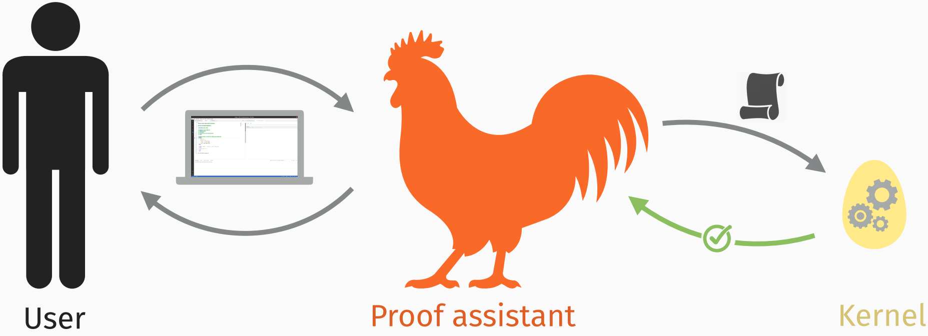
A bit of context

The grail



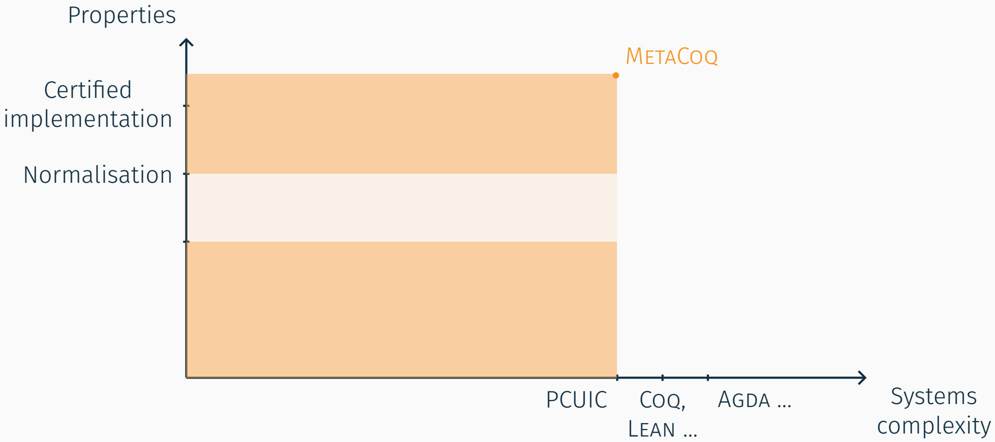
The de Bruijn architecture

The grail



The de Bruijn architecture: a perfect target for certification!

We are not alone



Normalisation

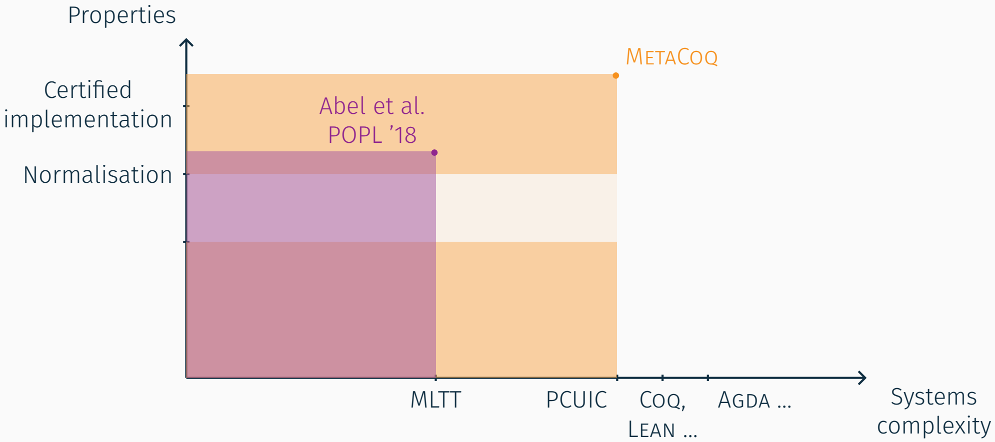
- every reduction path $t_0 \rightsquigarrow t_1 \rightsquigarrow t_2 \rightsquigarrow \ldots$ is finite
- there is exactly one normal form $\bar{t} \in \mathsf{Nf}$ in each equivalence class for $\cong$
- …

- every reduction path $t_0 \rightsquigarrow t_1 \rightsquigarrow t_2 \rightsquigarrow \ldots$ is finite
- there is exactly one normal form $\bar{t} \in \mathrm{Nf}$ in each equivalence class for $\cong$
- …

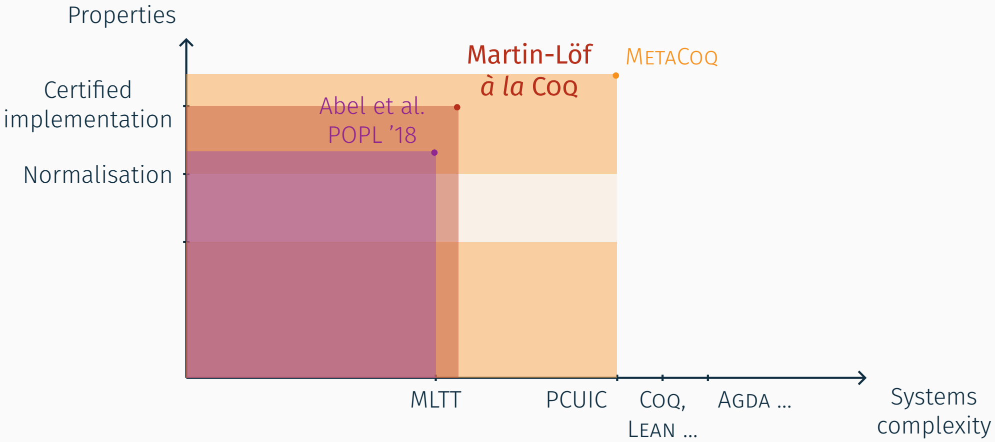
The mother of all properties:

- decidability of conversion
- canonicity
- consistency ⚠

We are not alone



We are not alone



Martin-Löf Type Theory

Martin-Löf's logical framework

$$\Gamma \vdash A \qquad \Gamma \vdash A \cong B \qquad \Gamma \vdash t : A \qquad \Gamma \vdash t \cong u : A$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash t : B} \qquad \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash t \cong u : B}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t \cong t : A} \qquad \frac{\Gamma \vdash t \cong u : A}{\Gamma \vdash u \cong t : A} \qquad \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A}$$

+ for types

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x A}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash t : B} \qquad \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash t \cong u : B}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t \cong t : A} \qquad \frac{\Gamma \vdash t \cong u : A}{\Gamma \vdash u \cong t : A} \qquad \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A}$$

+ for types

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x A}$$

Derivations are not unique!

$$\frac{\Gamma \vdash A \qquad \Gamma, x : A \vdash B}{\Gamma \vdash \Pi\, x : A.B}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda\, x : A.t : \Pi\, x : A.B}$$

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

$$\frac{\Gamma \vdash t \cong t' : \Pi\, x : A.B \qquad \Gamma \vdash u \cong u' : A}{\Gamma \vdash t\ u \cong t'\ u' : B[u]}$$

+ other congruences

$$\beta\ \frac{\Gamma \vdash A \qquad \Gamma, x : A \vdash B \qquad \Gamma, x : A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda\, x : A.t)\, u \cong t[u] : B[u]}$$

$$\eta\ \frac{\Gamma \vdash f : \Pi\, x : A.B}{\Gamma \vdash f \cong \lambda x : A.f\ x : \Pi\, x : A.B}$$

Large elimination

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathrm{N}} \qquad \frac{\vdash \Gamma}{\Gamma \vdash 0 : \mathrm{N}} \qquad \frac{\Gamma \vdash n : \mathrm{N}}{\Gamma \vdash \mathrm{S}(n) : \mathrm{N}}$$

$$\frac{\Gamma \vdash s : \mathrm{N} \qquad \Gamma, z{:}\,\mathrm{N} \vdash P \qquad \Gamma \vdash b_0 : P[0] \qquad \Gamma \vdash b_\mathrm{S} : \Pi\, y{:}\,\mathrm{N}\,.P[y] \to P[\mathrm{S}(y)]}{\Gamma \vdash \mathrm{ind}(s; z.P; b_0 \mid b_\mathrm{S}) : P[s]}$$

$$\frac{\Gamma, z{:}\,\mathrm{N} \vdash P \qquad \Gamma \vdash b_0 : P[0] \qquad \Gamma \vdash b_\mathrm{S} : \Pi\, y{:}\,\mathrm{N}\,.P[y] \to P[\mathrm{S}(y)]}{\Gamma \vdash \mathrm{ind}(0; z.P; b_0 \mid b_\mathrm{S}) \cong b_0 : P[0]}$$

I spare you the successor

Universe(s)

$$\frac{\vdash \Gamma}{\Gamma \vdash \square}$$

$$\frac{\Gamma \vdash A : \square}{\Gamma \vdash A}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathbf{N} : \square}$$

$$\frac{\Gamma \vdash A : \square \qquad \Gamma, x\colon A \vdash B : \square}{\Gamma \vdash \Pi\, x\colon A.B : \square}$$

With this + $\mathbf{ind}$, you can start doing *nasty* things

Normal and neutral forms

$$\frac{}{\mathrm{nf}\ \lambda\, x{:}\, A.t} \qquad \frac{}{\mathrm{nf}\ 0} \qquad \frac{}{\mathrm{nf}\ \mathrm{S}(n)} \qquad \frac{}{\mathrm{nf}\ \mathbf{N}} \qquad \frac{}{\mathrm{nf}\ \Pi\, x{:}\, A.B} \qquad \frac{}{\mathrm{nf}\ \square} \qquad \frac{\mathrm{ne}\ n}{\mathrm{nf}\ n}$$

$$\frac{}{\mathrm{ne}\ x} \qquad \frac{\mathrm{ne}\ f}{\mathrm{ne}\ f\ u} \qquad \frac{\mathrm{ne}\ s}{\mathrm{ne}\ \mathrm{ind}(s; z.P; b_0 \mid b_{\mathrm{S}})}$$

$$\frac{}{\mathrm{nf}\ \lambda\, x\colon A.t} \qquad \frac{}{\mathrm{nf}\ 0} \qquad \frac{}{\mathrm{nf}\ \mathrm{S}(n)} \qquad \frac{}{\mathrm{nf}\ \mathbf{N}} \qquad \frac{}{\mathrm{nf}\ \Pi\, x\colon A.B} \qquad \frac{}{\mathrm{nf}\ \square} \qquad \frac{\mathrm{ne}\ n}{\mathrm{nf}\ n}$$

$$\frac{}{\mathrm{ne}\ x} \qquad \frac{\mathrm{ne}\ f}{\mathrm{ne}\ f\ u} \qquad \frac{\mathrm{ne}\ s}{\mathrm{ne}\ \mathrm{ind}(s; z.P; b_0 \mid b_{\mathrm{S}})}$$

Idea: things that have “finished computing”

$$\frac{\Gamma \vdash A \qquad \Gamma, x\colon A \vdash B \qquad \Gamma, x\colon A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda\, x\colon A.t)\, u \rightsquigarrow^\star t[u] : B[u]}$$

+ other β rules (no η)

$$\frac{\Gamma \vdash t \rightsquigarrow^\star t' : \Pi\, x\colon A.B}{\Gamma \vdash t\ u \rightsquigarrow^\star t'\ u : B[u]}$$

+ other **head** congruences

$$\frac{\Gamma \vdash A \rightsquigarrow^\star A' : \square}{\Gamma \vdash A \rightsquigarrow^\star A'}$$

The logical relation

A standard recipe, with a twist

Usually, for logical relations, one

1. defines a suitable predicate by induction on types
2. shows that each typing rule is sound for this predicate
3. enjoys

A standard recipe, with a twist

Usually, for logical relations, one

1. defines a suitable predicate by induction on types
2. shows that each typing rule is sound for this predicate
3. enjoys

Dependent types are more complicated:

- up to conversion
- might not be a nice constructor (not all types are $\mathbb{N}$, Π or $\square$!)

The logical relation: reducibility

A predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

A predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

Natural numbers:

$$\frac{\mathcal{T} :: \Gamma \vdash T \rightsquigarrow^{\star} \mathbf{N}}{\mathrm{red}_{\mathbf{N}}(\mathcal{T}) :: \Gamma \Vdash T}$$

A predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

Given $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

Natural numbers:

$$\frac{\mathcal{T} :: \Gamma \vdash T \leadsto^{\star} \mathbf{N}}{\mathrm{red}_{\mathbf{N}}(\mathcal{T}) :: \Gamma \Vdash T}$$

A predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

Given $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

Natural numbers:

$$\frac{\Gamma \vdash t \rightsquigarrow^{\star} 0 : \mathbf{N}}{\Gamma \Vdash_{\mathrm{red}_{\mathbf{N}}(\mathcal{T})} t : T} \qquad \frac{\Gamma \vdash t \rightsquigarrow^{\star} \mathrm{S}(t') : \mathbf{N} \quad \Gamma \Vdash t' : \mathbf{N}}{\Gamma \Vdash_{\mathrm{red}_{\mathbf{N}}(\mathcal{T})} t : T}$$

The logical relation: reducibility

A predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

Given $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

Natural numbers:

neutral-specific conversion

$$\frac{\Gamma \vdash t \rightsquigarrow^{\star} 0 : \mathrm{N}}{\Gamma \Vdash_{\mathrm{red_N}(\mathcal{T})} t : T} \qquad \frac{\Gamma \vdash t \rightsquigarrow^{\star} \mathrm{S}(t') : \mathrm{N} \quad \Gamma \Vdash t' : \mathrm{N}}{\Gamma \Vdash_{\mathrm{red_N}(\mathcal{T})} t : T} \qquad \frac{\Gamma \vdash t \rightsquigarrow^{\star} n : \mathrm{N} \quad \Gamma \vdash n \approx n : \mathrm{N}}{\Gamma \Vdash_{\mathrm{red_N}(\mathcal{T})} t : T}$$

A predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

Given $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

$$\frac{\Gamma \vdash B \rightsquigarrow^{\star} \mathbf{N}}{\Gamma \Vdash_{\mathrm{red}_{\mathrm{N}}(\mathcal{T})} A \cong B}$$

$\Gamma \Vdash_{\mathrm{red}_{\mathrm{N}}(\mathcal{T})} t \cong u : A$ is essentially similar to $\Gamma \Vdash_{\mathrm{red}_{\mathrm{N}}(\mathcal{T})} t : T$

A Kripke logical relation:

$$\frac{\Gamma \vdash T \rightsquigarrow^{\star} \Pi\, x\colon A.B \qquad \mathcal{A} :: \forall \rho :: \Delta \leq \Gamma . \Delta \Vdash A[\rho] \qquad \mathcal{B} :: \forall (\rho :: \Delta \leq \Gamma)\, a.\;\; \Delta \Vdash_{\mathcal{A}\,\rho} a : A[\rho] \Rightarrow \Delta \Vdash B[\rho, a]}{\mathrm{red}_{\Pi}(\mathcal{A}, \mathcal{B}) :: \Gamma \Vdash T}$$

$$\frac{\forall (\rho :: \Delta \leq \Gamma)\, a\, (h :: \Delta \Vdash_{\mathcal{A}\,\rho} a : A[\rho]).\;\; \Delta \Vdash_{\mathcal{B}\,\rho\,h} t[\rho]\, a : B[\rho, a]}{\Gamma \Vdash_{\mathrm{red}_{\Pi}(\mathcal{A},\mathcal{B})} f : T}$$

A Kripke logical relation:

$$\frac{\Gamma \vdash T \leadsto^\star \Pi\, x\colon A.B \qquad \mathcal{A} :: \forall \rho :: \Delta \leq \Gamma. \Delta \Vdash A[\rho] \qquad \mathcal{B} :: \forall (\rho :: \Delta \leq \Gamma)\, a.\quad \Delta \Vdash_{\mathcal{A}\,\rho} a : A[\rho] \Rightarrow \Delta \Vdash B[\rho, a]}{\mathrm{red}_\Pi(\mathcal{A}, \mathcal{B}) :: \Gamma \Vdash T}$$

$$\frac{\forall (\rho :: \Delta \leq \Gamma)\, a\, (h :: \Delta \Vdash_{\mathcal{A}\,\rho} a : A[\rho]).\quad \Delta \Vdash_{\mathcal{B}\,\rho\,h} t[\rho]\, a : B[\rho, a]}{\Gamma \Vdash_{\mathrm{red}_\Pi(\mathcal{A},\mathcal{B})} f : T}$$

Reducibility at the universe is reducibility of types:

$$\frac{\Gamma \Vdash A}{\Gamma \Vdash_{\mathrm{red}_\Box(\mathcal{T})} A : T}$$

- Mutual definition of $\Gamma \Vdash A$ and $\Gamma \Vdash t : A$ via the Π-type
- reducibility at the universe $\Gamma \Vdash A : \square$ calls itself?
- ...

- Mutual definition of $\Gamma \Vdash A$ and $\Gamma \Vdash t : A$ via the Π-type
- reducibility at the universe $\Gamma \Vdash A : \square$ calls itself?
- ...

Induction-recursion + stratified definitions

- Mutual definition of $\Gamma \Vdash A$ and $\Gamma \Vdash t : A$ via the Π-type
- reducibility at the universe $\Gamma \Vdash A : \square$ calls itself?
- ...

Induction-recursion + stratified definitions

$\cdot \nVdash^{0} \square$ but $\cdot \Vdash^{1} \square$, and

$$\frac{\cdot \Vdash^{0} A}{\cdot \Vdash^{1}_{\mathrm{red}_{\square}(\dots)} A : \square}$$

Induction-Recursion

Inductive domain (U),
recursive function (El)

```
data U : Set
El : U → Set

data U where
  b : U
  π : (A : U)
      (B : El A → U)
      → U

El b = bool
El (π A B) =
  (a : El A) → El (B a)
```

Induction-Recursion

Inductive domain (U), recursive function (El)

```
data U : Set
El : U → Set

data U where
  b : U
  π : (A : U)
      (B : El A → U)
      → U

El b = bool
El (π A B) =
  (a : El A) → El (B a)
```

Small Induction-Recursion

Inductively defined image of the function (ImEl)

```
data ImEl : Set → Set₁ where
  isb : ImEl bool
  isπ : {A : Set}
        {B : A → Set}
        (iA : ImEl A)
        (iB : (a : A) → ImEl (B a))
        → ImEl ((a : A) → B a)

record U : Set₁ where
  field
    el : Set
    imEl : ImEl el
```

```
El : U → Set
El x = x .el

b : U
b .el = bool
b .imEl = isb

π : (A : U) (B : El A → U) → U
(π A B) .el =
  (a : A .el) → (B a) .el
(π A B) .imEl =
  isπ (A .imEl)
      (λ a → (B a) .imEl)
```

Induction-Recursion

Inductive domain (U), recursive function (El)

```
data U : Set
El : U → Set

data U where
  b : U
  π : (A : U)
      (B : El A → U)
      → U

El b = bool
El (π A B) =
  (a : El A) → El (B a)
```

Small Induction-Recursion

Inductively defined image of the function (ImEl)

```
data ImEl : Set → Set₁ where
  isb : ImEl bool
  isπ : {A : Set}
        {B : A → Set}
        (iA : ImEl A)
        (iB : (a : A) → ImEl (B a))
        → ImEl ((a : A) → B a)

record U : Set₁ where
  field
    el : Set
    imEl : ImEl el
```

```
El : U → Set
El x = x .el

b : U
b .el = bool
b .imEl = isb

π : (A : U) (B : El A → U) → U
(π A B) .el =
  (a : A .el) → (B a) .el
(π A B) .imEl =
  isπ (A .imEl)
      (λ a → (B a) .imEl)
```

- Need to re-encapsulate
- Typically raises universe levels

Inducti

Inductive c
recursive f

```
data U :
El : U →

data U wh
  b : U
  π : (A
      (B
      → U

El b = bo
El (π A B
  (a : El
```

A → U) → U

a) .el

.imEl)

```
Inductive LR@{i j k} {l : TypeLevel} (rec : forall l', l' << l -> RedRel@{i j})
: RedRel@{j k} :=
  | LRU {Γ A} (H : [Γ ||-U<l> A]) :
      LR rec Γ A
      (fun B   => [Γ ||-U≅ B ])
      (fun t   => [ rec | Γ ||-U t      : A | H ])
      (fun t u => [ rec | Γ ||-U t ≅ u : A | H ])
  | LRne {Γ A} (neA : [ Γ ||-ne A ]) :
      LR rec Γ A
      (fun B   =>  [ Γ ||-ne A ≅ B      | neA])
      (fun t   =>  [ Γ ||-ne t       : A | neA])
      (fun t u =>  [ Γ ||-ne t ≅ u : A | neA])
  | LRPi {Γ : context} {A : term} (ΠA : PiRedTy@{j} Γ A) (ΠAad : PiRedTyAdequat
    LR rec Γ A
      (fun B   => [ Γ ||-Π A ≅ B      | ΠA ])
      (fun t   => [ Γ ||-Π t       : A | ΠA ])
      (fun t u => [ Γ ||-Π t ≅ u : A | ΠA ])
  | LRNat {Γ A} (NA : [Γ ||-Nat A]) :
    LR rec Γ A (NatRedTyEq NA) (NatRedTm NA) (NatRedTmEq NA)
  | LREmpty {Γ A} (NA : [Γ ||-Empty A]) :
    LR rec Γ A (EmptyRedTyEq NA) (EmptyRedTm NA) (EmptyRedTmEq NA)
  | LRSig {Γ : context} {A : term} (ΣA : SigRedTy@{j} Γ A) (ΣAad : SigRedTyAdeq
    LR rec Γ A (SigRedTyEq ΣA) (SigRedTm ΣA) (SigRedTmEq ΣA)
  | LRList {Γ : context} {A : term}
      (LA : ListRedTyPack@{j} Γ A) (LAAd : ListRedTyAdequate@{j k} (LR rec) LA)
    LR rec Γ A
```

Properties of the logical relation

- Escape: $\Gamma \Vdash A \Rightarrow \Gamma \vdash A$
- Irrelevance
- Equivalence: reflexivity, symmetry, transitivity
- Weakening
- Neutral reflection
- Closure by anti-reduction

- Escape: $\Gamma \Vdash A \Rightarrow \Gamma \vdash A$
- Irrelevance
- Equivalence: reflexivity, symmetry, transitivity
- Weakening
- Neutral reflection
- Closure by anti-reduction

Fundamental lemma: if $\Gamma \vdash t : A$ then $\mathcal{A} :: \Gamma \Vdash A$ and $\Gamma \Vdash_{\mathcal{A}} t : A$

$\Rightarrow$ all types/terms have a whnf

Algorithmic conversion

Declarative conversion

Arbitrarily mixing:

- Transitivity, symmetry, reflexivity,
- Congruences (arbitrary),
- Computation steps (β),
- Extensionality steps (η).

Declarative vs algorithmic conversion

Declarative conversion

Arbitrarily mixing:

- Transitivity, symmetry, reflexivity,
- Congruences (arbitrary),
- Computation steps (β),
- Extensionality steps (η).

Algorithmic conversion

Term/type-directed alternation of:

- Reduction to weak-head normal form,
- Type-directed extensionality rules,
- Selected congruences.

$\Rightarrow$ Implementable

Declarative vs algorithmic conversion

Declarative conversion

Arbitrarily mixing:

- Transitivity, symmetry, reflexivity,
- Congruences (arbitrary),
- Computation steps (β),
- Extensionality steps (η).

Algorithmic conversion

Term/type-directed alternation of:

- Reduction to weak-head normal form,
- Type-directed extensionality rules,
- Selected congruences.

⇒ Implementable

How can we compare the two presentations?

Declarative conversion

Arbitrarily mixing:

- Transitivity, symmetry, reflexivity,
- Congruences (arbitrary),
- Computation steps (β),
- Extensionality steps (η).

Algorithmic conversion

Term/type-directed alternation of:

- Reduction to weak-head normal form,
- Type-directed extensionality rules,
- Selected congruences.

$\Rightarrow$ Implementable

How can we compare the two presentations?

Algorithmic $\longrightarrow$ Declarative: Admissibility of algorithmic rules

Declarative vs algorithmic conversion

Declarative conversion

Arbitrarily mixing:

- Transitivity, symmetry, reflexivity,
- Congruences (arbitrary),
- Computation steps (β),
- Extensionality steps (η).

Algorithmic conversion

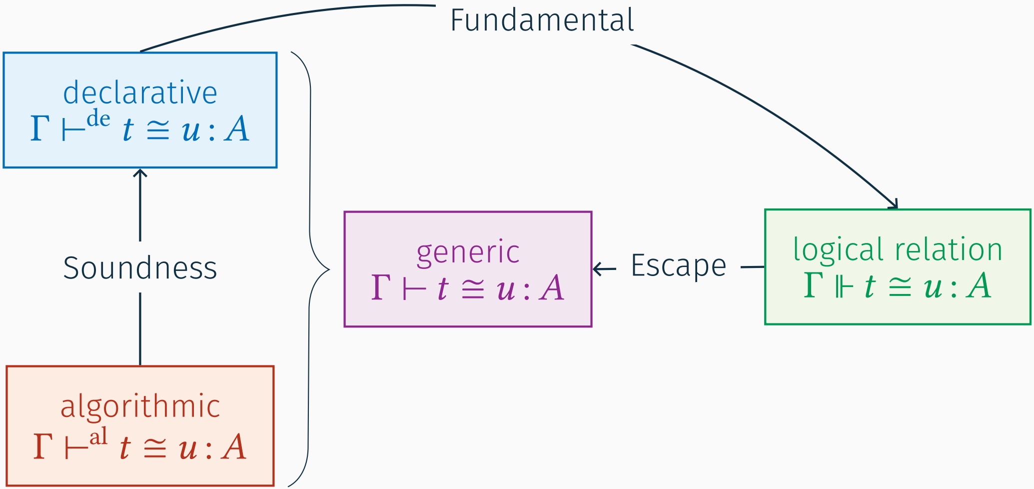
Term/type-directed alternation of:

- Reduction to weak-head normal form,
- Type-directed extensionality rules,
- Selected congruences.

$\Longrightarrow$ Implementable

How can we compare the two presentations?

Algorithmic $\longrightarrow$ Declarative: Admissibility of algorithmic rules
Declarative $\longrightarrow$ Algorithmic: Show that every derivation has a canonical form


Fundamental
declarative
$\Gamma \vdash^{\mathrm{de}} t \cong u : A$
Soundness
algorithmic
$\Gamma \vdash^{\mathrm{al}} t \cong u : A$
generic
$\Gamma \vdash t \cong u : A$
Escape
logical relation
$\Gamma \Vdash t \cong u : A$

Back to the big picture

Martin-Löf *à la* Coq

- Port to Coq
- More types (equality, lists...)

Martin-Löf *à la* Coq

- Port to Coq
- More types (equality, lists…)

- **Small** induction-recursion
- New insights from **bidirectional** typing
- A certified, **executable**, **type**-checker
- Proof engineering lessons?

Do we have to suffer this much?

- no fancy maths
- works in a rather barebone meta-theory
- strong result: we get a certified evaluator

Do we have to suffer this much?

- + no fancy maths
- + works in a rather barebone meta-theory
- + strong result: we get a certified evaluator
- - partial equivalence relations, by hand
- - Kripke quantification, by hand
- - no big picture

(ask Kenji about his suffering with W types) — referring to the two "by hand" items

Do we have to suffer this much?

+ no fancy maths
+ works in a rather barebone meta-theory
+ strong result: we get a certified evaluator
- partial equivalence relations, by hand
- Kripke quantification, by hand
- no big picture

(ask Kenji about his suffering with W types) — applies to the two "by hand" items above

A whole different approach "exists":

- syntax as a generalised algebraic theory (PER → equality)
- fancy categorical gluing in presheave toposes (Kripke quantification for free)
- normalisation by evaluation as a model (no reduction)

Is this the future?

And the guard condition?

Normalisation and the guard

- it *should* be possible to adapt to fixpoint + case + guard
- We probably want a less syntactic criterion? Sized types (Abel et al., 2017)?
- we might suffer quite a bit
- anyway, is this *really* what we want?

And the guard condition?

Normalisation and the guard

- it *should* be possible to adapt to fixpoint + case + guard
- We probably want a less syntactic criterion? Sized types (Abel et al., 2017)?
- we might suffer quite a bit
- anyway, is this *really* what we want?

Elaboration?

- how fancy a guard can we elaborate to a recursor?
- with which guarantees/properties? (is small IR a good example?)
- is normalisation easily transferable?

Thank you!