

Continuity in dependent type theory

joint work with Martin Baillon, Pierre-Marie Pédro

Assia Mahboubi

Leeds-Ghent seminar – April 27th 2022

Inria, LS2N, Université de Nantes, Vrije Universiteit Amsterdam

Theorem (folklore?)

Every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ definable in Gödel's system T is continuous.

Theorem (folklore?)

Every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ definable in Gödel's system T is continuous.

with Gödel's system $T =$ simply typed λ -calculus + \mathbb{N} + recursor.

Theorem (folklore?)

Every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ definable in Gödel's system T is continuous.

with Gödel's system $T =$ simply typed λ -calculus $+ \mathbb{N} +$ recursor.

[Foundations of Constructive Mathematics. M.J. Beeson. Springer, 1985]

Theorem

Every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ definable in S is continuous.

Theorem

Every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ definable in S is continuous.

with S a dependent type theory.

Theorem

Every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ definable in S is continuous.

with S a dependent type theory.

Generalizing [Continuity of Gödel's system T functionals via effectful forcing. M. Escardó. MFPS'2013.]

A function $f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ is continuous at $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ if:

$$\forall W \in \mathcal{V}_{\mathbb{N}}(f \alpha), \exists V \in \mathcal{V}_{\mathbb{N} \rightarrow \mathbb{N}}(\alpha), \forall \alpha' : \mathbb{N} \rightarrow \mathbb{N}, \alpha' \in V \Rightarrow (f \alpha') \in W.$$

A function $f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ is continuous at $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ if:

$$\forall W \in \mathcal{V}_{\mathbb{N}}(f \alpha), \exists V \in \mathcal{V}_{\mathbb{N} \rightarrow \mathbb{N}}(\alpha), \forall \alpha' : \mathbb{N} \rightarrow \mathbb{N}, \alpha' \in V \Rightarrow (f \alpha') \in W.$$

With:

- Discrete topology on \mathbb{N}
- Product topology on $\mathbb{N} \rightarrow \mathbb{N}$

This becomes:

$$\exists w : \text{list } \mathbb{N}, \forall \alpha' : \mathbb{N} \rightarrow \mathbb{N}, (\forall i, \alpha'(w_i) = \alpha(w_i)) \Rightarrow (f \alpha') = (f \alpha).$$

Core:

- CC_ω : predicative variant of the Calculus of Constructions (CC), with dependent pairs

Core:

- CC_ω : predicative variant of the Calculus of Constructions (CC), with dependent pairs

Extensions:

- Identity types, à la Martin L of (MLTT)
- **Inductive types** with parameters and indices

CC_ω with dependent pairs: typing rules

$$A, B, M, N ::= \square_i \mid x \mid M N \mid \lambda x : A. M \mid \Pi x : A. M \mid \Sigma x : A. B \mid M.\pi_1 \mid M.\pi_2 \mid (M, N)$$
$$\Gamma, \Delta ::= \cdot \mid \Gamma, x : A$$

CC_ω with dependent pairs: typing rules

$A, B, M, N ::= \square_i \mid x \mid MN \mid \lambda x : A. M \mid \Pi x : A. M \mid \Sigma x : A. B \mid M.\pi_1 \mid M.\pi_2 \mid (M, N)$

$\Gamma, \Delta ::= \cdot \mid \Gamma, x : A$

$$\frac{}{\vdash \cdot} \quad \frac{\Gamma \vdash A : \square_i}{\vdash \Gamma, x : A} \quad \frac{\vdash \Gamma \quad (x : A) \in \Gamma}{\Gamma \vdash x : A} \quad \frac{\vdash \Gamma \quad i < j}{\Gamma \vdash \square_i : \square_j}$$

CC_ω with dependent pairs: typing rules

$A, B, M, N ::= \square_i \mid x \mid MN \mid \lambda x : A. M \mid \Pi x : A. M \mid \Sigma x : A. B \mid M.\pi_1 \mid M.\pi_2 \mid (M, N)$

$\Gamma, \Delta ::= \cdot \mid \Gamma, x : A$

$$\frac{}{\vdash \cdot} \quad \frac{\Gamma \vdash A : \square_i}{\vdash \Gamma, x : A} \quad \frac{\vdash \Gamma \quad (x : A) \in \Gamma}{\Gamma \vdash x : A} \quad \frac{\vdash \Gamma \quad i < j}{\Gamma \vdash \square_i : \square_j}$$

$$\frac{\Gamma \vdash A : \square_i \quad \Gamma \vdash M : B}{\Gamma, x : A \vdash M : B} \quad \frac{\Gamma \vdash A : \square_i \quad \Gamma, x : A \vdash B : \square_j}{\Gamma \vdash \Pi x : A. B : \square_{\max(i,j)}}$$

$$\frac{\Gamma \vdash M : \Pi x : A. B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B\{x := N\}} \quad \frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash \Pi x : A. B : \square_i}{\Gamma \vdash \lambda x : A. M : \Pi x : A. B}$$

CC_ω with dependent pairs: typing rules

$A, B, M, N ::= \square_i \mid x \mid MN \mid \lambda x : A. M \mid \Pi x : A. M \mid \Sigma x : A. B \mid M.\pi_1 \mid M.\pi_2 \mid (M, N)$

$\Gamma, \Delta ::= \cdot \mid \Gamma, x : A$

$$\frac{}{\vdash \cdot} \quad \frac{\Gamma \vdash A : \square_i}{\vdash \Gamma, x : A} \quad \frac{\vdash \Gamma \quad (x : A) \in \Gamma}{\Gamma \vdash x : A} \quad \frac{\vdash \Gamma \quad i < j}{\Gamma \vdash \square_i : \square_j}$$

$$\frac{\Gamma \vdash A : \square_i \quad \Gamma \vdash M : B}{\Gamma, x : A \vdash M : B} \quad \frac{\Gamma \vdash A : \square_i \quad \Gamma, x : A \vdash B : \square_j}{\Gamma \vdash \Pi x : A. B : \square_{\max(i,j)}}$$

$$\frac{\Gamma \vdash M : \Pi x : A. B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B\{x := N\}} \quad \frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash \Pi x : A. B : \square_i}{\Gamma \vdash \lambda x : A. M : \Pi x : A. B}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : \square_i \quad \Gamma \vdash A \equiv B}{\Gamma \vdash M : B}$$

$A, B, M, N ::= \square_i \mid x \mid MN \mid \lambda x : A. M \mid \Pi x : A. M \mid \Sigma x : A. B \mid M.\pi_1 \mid M.\pi_2 \mid (M, N)$

$\Gamma, \Delta ::= \cdot \mid \Gamma, x : A$

$$\begin{array}{c}
 \frac{}{\vdash \cdot} \quad \frac{\Gamma \vdash A : \square_i}{\vdash \Gamma, x : A} \quad \frac{\vdash \Gamma \quad (x : A) \in \Gamma}{\Gamma \vdash x : A} \quad \frac{\vdash \Gamma \quad i < j}{\Gamma \vdash \square_i : \square_j} \\
 \\
 \frac{\Gamma \vdash A : \square_i \quad \Gamma \vdash M : B}{\Gamma, x : A \vdash M : B} \quad \frac{\Gamma \vdash A : \square_i \quad \Gamma, x : A \vdash B : \square_j}{\Gamma \vdash \Pi x : A. B : \square_{\max(i,j)}} \\
 \\
 \frac{\Gamma \vdash M : \Pi x : A. B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B\{x := N\}} \quad \frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash \Pi x : A. B : \square_i}{\Gamma \vdash \lambda x : A. M : \Pi x : A. B} \\
 \\
 \frac{\Gamma \vdash M : A \quad \Gamma \vdash B : \square_i \quad \Gamma \vdash A \equiv B}{\Gamma \vdash M : B} \\
 \\
 \frac{\Gamma \vdash A : \square_i \quad \Gamma, x : A \vdash B : \square_j}{\Gamma \vdash \Sigma x : A. B : \square_{\max(i,j)}} \quad \frac{\Gamma \vdash M : \Sigma x : A. B}{\Gamma \vdash M.\pi_1 : A} \\
 \\
 \frac{\Gamma \vdash M : \Sigma x : A. B}{\Gamma \vdash M.\pi_2 : B\{x := M.\pi_1\}}
 \end{array}$$

Inductive $\mathbb{N} := \text{O} : \mathbb{N} \mid \text{S} : \mathbb{N} \rightarrow \mathbb{N}$

Inductive $\mathbb{N} := \text{O} : \mathbb{N} \mid \text{S} : \mathbb{N} \rightarrow \mathbb{N}$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbb{N} : \square_i}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \text{O} : \mathbb{N}}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \text{S} : \mathbb{N} \rightarrow \mathbb{N}}$$

Inductive $\mathbb{N} := \text{O} : \mathbb{N} \mid \text{S} : \mathbb{N} \rightarrow \mathbb{N}$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbb{N} : \square_i} \quad \frac{\Gamma \vdash}{\Gamma \vdash \text{O} : \mathbb{N}} \quad \frac{\Gamma \vdash}{\Gamma \vdash \text{S} : \mathbb{N} \rightarrow \mathbb{N}}$$
$$\frac{\Gamma \vdash P : \mathbb{N} \rightarrow \square_i \quad \Gamma \vdash t_{\text{O}} : P \text{ O} \quad \Gamma \vdash t_{\text{S}} : \prod n : \mathbb{N}. P n \rightarrow P (\text{S } n)}{\Gamma \vdash \mathbb{N}_{\text{ind}} P t_{\text{O}} t_{\text{S}} : \prod n : \mathbb{N}. P n}$$

Inductive types in the Calculus of Inductive Constructions

Inductive $\mathbb{N} := \text{O} : \mathbb{N} \mid \text{S} : \mathbb{N} \rightarrow \mathbb{N}$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbb{N} : \square_i} \quad \frac{\Gamma \vdash}{\Gamma \vdash \text{O} : \mathbb{N}} \quad \frac{\Gamma \vdash}{\Gamma \vdash \text{S} : \mathbb{N} \rightarrow \mathbb{N}}$$
$$\frac{\Gamma \vdash P : \mathbb{N} \rightarrow \square_i \quad \Gamma \vdash t_O : P \text{ O} \quad \Gamma \vdash t_S : \prod n : \mathbb{N}. P n \rightarrow P (\text{S } n)}{\Gamma \vdash \mathbb{N}_{\text{ind}} P t_O t_S : \prod n : \mathbb{N}. P n}$$

$$\mathbb{N}_{\text{ind}} P t_O t_S \text{ O} \equiv t_O$$

$$\mathbb{N}_{\text{ind}} P t_O t_S (\text{S } n) \equiv t_S n (\mathbb{N}_{\text{ind}} P t_O t_S n)$$

With parameters:

Inductive list ($A : \square$) : $\square :=$

$\text{nil} : \text{list } A \mid \text{cons} : A \rightarrow \text{list } A \rightarrow \text{list } A$

With parameters:

`Inductive list (A : \square) : \square :=`

`nil : list A | cons : A \rightarrow list A \rightarrow list A`

With indices:

`Inductive vect (A : \square) : \mathbb{N} \rightarrow \square :=`

`vnil : vect A 0 | vcons : $\forall n : \mathbb{N}$, A \rightarrow vect A n \rightarrow vect A (S n)`

$$\mathcal{C} : ((\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}) \rightarrow \square$$

$$\mathcal{C} f := \Pi(\alpha : \mathbb{N} \rightarrow \mathbb{N}). \Sigma(l : \text{list } \mathbb{N}). \Pi(\beta : \mathbb{N} \rightarrow \mathbb{N}). \alpha \approx_l \beta \rightarrow f \alpha = f \beta.$$

with

$$\alpha \approx_l \beta := \text{map } l \alpha = \text{map } l \beta$$

Theorem

For any $\vdash_S f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$, there exists a proof $\vdash_T p : \mathcal{C} f$.

with \mathcal{T} and \mathcal{S} two “appropriate” dependent type theories.

See:

$$\vdash f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$$

as a natural number computed using calls to a fixed oracle α :

$$\alpha : \mathbb{N} \rightarrow \mathbb{N} \vdash n : \mathbb{N}$$

See:

$$\vdash f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$$

as a natural number computed using calls to a fixed oracle α :

$$\alpha : \mathbb{N} \rightarrow \mathbb{N} \vdash n : \mathbb{N}$$

More generally, we will study:

$$\alpha : \prod i : \mathbb{I}, O\ i \vdash a : A$$

for fixed arbitrary types:

- of questions to the oracle: $\vdash \mathbb{I} : \square_0$
- of answers from the oracle: $\vdash O : \mathbb{I} \rightarrow \square_0$

See:

$$\vdash f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$$

as a natural number computed using calls to a fixed oracle α :

$$\alpha : \mathbb{N} \rightarrow \mathbb{N} \vdash n : \mathbb{N}$$

More generally, we will study:

$$\alpha : \prod i : \mathbb{I}, O\ i \vdash a : A$$

for fixed arbitrary types:

- of questions to the oracle: $\vdash \mathbb{I} : \square_0$
- of answers from the oracle: $\vdash O : \mathbb{I} \rightarrow \square_0$

Denote $\mathbb{Q} := \prod i : \mathbb{I}, O\ i$ the type of oracles.

Inductive $\mathcal{D} (A : \square) : \square := \eta : A \rightarrow \mathcal{D} A \mid \beta : \Pi(i : \mathbb{I}). (\mathbb{O} i \rightarrow \mathcal{D} A) \rightarrow \mathcal{D} A.$

Inductive $\mathcal{D} (A : \square) : \square := \eta : A \rightarrow \mathcal{D} A \mid \beta : \prod(i : \mathbb{I}). (\mathbb{O} i \rightarrow \mathcal{D} A) \rightarrow \mathcal{D} A.$

A term $\vdash d : \mathcal{D} A$ represents a dialogue tree with:

- labels on inner nodes in \mathbb{I} ;
- labels on leaves in A ;
- arcs from inner node i indexed by the inhabitants of type $(\mathbb{O} i)$.

$\vdash d : \mathfrak{D} A$ computes a term in A using an oracle $\alpha : \prod i : \mathbb{I}, O i$

$$\begin{aligned} \partial & : \quad \prod \{A : \square\} (\alpha : \mathbb{Q}) (d : \mathfrak{D} A). A \\ \partial \alpha (\eta x) & := x \\ \partial \alpha (\beta i k) & := \partial \alpha (k (\alpha i)). \end{aligned}$$

From dialogue trees to functions

$\vdash d : \mathfrak{D} A$ computes a term in A using an oracle $\alpha : \prod i : \mathbb{I}, O i$

$$\begin{aligned} \partial & : \quad \prod \{A : \square\} (\alpha : \mathbb{Q}) (d : \mathfrak{D} A). A \\ \partial \alpha (\eta x) & := x \\ \partial \alpha (\beta i k) & := \partial \alpha (k (\alpha i)). \end{aligned}$$

Definition

A function $f : \mathbb{Q} \rightarrow A$ is said to be *eloquent* if there is a dialogue tree $d : \mathfrak{D} A$ and a proof that $\prod \alpha : \mathbb{Q}. f \alpha = \partial \alpha d$.

From dialogue trees to functions

$\vdash d : \mathfrak{D} A$ computes a term in A using an oracle $\alpha : \prod i : \mathbb{I}, O i$

$$\begin{aligned} \partial & : \quad \prod \{A : \square\} (\alpha : \mathbb{Q}) (d : \mathfrak{D} A). A \\ \partial \alpha (\eta x) & := x \\ \partial \alpha (\beta i k) & := \partial \alpha (k (\alpha i)). \end{aligned}$$

Definition

A function $f : \mathbb{Q} \rightarrow A$ is said to be *eloquent* if there is a dialogue tree $d : \mathfrak{D} A$ and a proof that $\prod \alpha : \mathbb{Q}. f \alpha = \partial \alpha d$.

Remark: Every **eloquent** function f is **continuous**.

Proof. Let d_f be a dialogue tree associated with f , and $\alpha : \mathbb{Q}$. Now α selects a path in d_f from the root to a leaf. Consider $\ell_\alpha : \text{list } \mathbb{I}$ the corresponding list of labels.

Theorem

For any $\vdash_S f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$, there exist a proof $\vdash_T p : \mathcal{C} f$.

Proof. Construct a model of \mathcal{S} for which every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ is eloquent.

Theorem

For any $\vdash_S f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$, there exist a proof $\vdash_T p : \mathcal{C} f$.

Proof. Construct a model of \mathcal{S} in \mathcal{T} , for which which every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ is eloquent.

For \mathcal{S} and \mathcal{T} two dependent type theory, a **syntactic model** of \mathcal{S} in \mathcal{T} is:

- a translation $[_]$ of terms of \mathcal{S} into terms of \mathcal{T} ;
- a translation $\llbracket _ \rrbracket$ of types of \mathcal{S} into types of \mathcal{T} ;
- a translation $\llbracket _ \rrbracket$ of contexts of \mathcal{S} into contexts of \mathcal{T} ;

For \mathcal{S} and \mathcal{T} two dependent type theory, a **syntactic model** of \mathcal{S} in \mathcal{T} is:

- a translation $[_]$ of terms of \mathcal{S} into terms of \mathcal{T} ;
- a translation $\llbracket _ \rrbracket$ of types of \mathcal{S} into types of \mathcal{T} ;
- a translation $\llbracket _ \rrbracket$ of contexts of \mathcal{S} into contexts of \mathcal{T} ;

Translations are typically defined by induction on the syntax of their argument.

Expected properties:

- Computational soundness: $M \equiv N$ implies $\llbracket M \rrbracket \equiv \llbracket N \rrbracket$
- Typing soundness: $\Gamma \vdash_{\mathcal{S}} M : A$ implies $\llbracket \Gamma \rrbracket \vdash_{\mathcal{T}} \llbracket M \rrbracket : \llbracket A \rrbracket$
- Consistency preservation: $\llbracket \prod A : \square_i . A \rrbracket$ is not inhabited.

Remark: Consistency of the source \mathcal{S} follows from consistency of the target \mathcal{T} .

Example: independence of funext from CC_ω

Take \mathcal{S} as $CC_\omega + Eq$ and \mathcal{T} as $CC_\omega + Eq + \mathbb{B}$.

$$\begin{aligned} [\Box_i]_f &:= \Box_i \\ [x]_f &:= x \\ [\lambda x : A. M]_f &:= (\lambda x : \llbracket A \rrbracket_f. \llbracket M \rrbracket_f, \mathbf{true}) \\ [M N]_f &:= \pi_1 \llbracket M \rrbracket_f \llbracket N \rrbracket_f \\ [\Pi x : A. B]_f &:= (\Pi x : \llbracket A \rrbracket_f. \llbracket B \rrbracket_f) \times \mathbb{B} \\ \llbracket A \rrbracket_f &:= \llbracket A \rrbracket_f \end{aligned}$$

[The next 700 syntactical models of type theory. S. Boulier, P.-M. Pédro, N. Tabareau. Procs. of CPP'17]

Now define:

$$\text{funext} := \Pi(A : \square_i)(B : \square_i)(f g : A \rightarrow B).(\Pi x : A.(f x =_B g x)) \rightarrow f =_{A \rightarrow B} g$$

Theorem

There exists a closed proof of $\llbracket \text{funext} \rightarrow \perp \rrbracket_f$

Example: independence of funext from CC_ω

Now define:

$$\text{funext} := \prod(A : \square_i)(B : \square_i)(f\ g : A \rightarrow B).(\prod x : A.(f\ x =_B\ g\ x)) \rightarrow f =_{A \rightarrow B}\ g$$

Theorem

There exists a closed proof of $\llbracket \text{funext} \rightarrow \perp \rrbracket_f$

Proof. Define $f := (\lambda x : \mathbb{B}.x, \text{true})$ and $g := (\lambda x : \mathbb{B}.x, \text{false})$.

Example: independence of funext from CC_ω

Now define:

$$\text{funext} := \Pi(A : \square_i)(B : \square_i)(f g : A \rightarrow B).(\Pi x : A.(f x =_B g x)) \rightarrow f =_{A \rightarrow B} g$$

Theorem

There exists a closed proof of $\llbracket \text{funext} \rightarrow \perp \rrbracket_f$

Proof. Define $f := (\lambda x : \mathbb{B}.x, \text{true})$ and $g := (\lambda x : \mathbb{B}.x, \text{false})$.

We have $f, g : \llbracket \mathbb{B} \rightarrow \mathbb{B} \rrbracket_f$ and $(f =_{\mathbb{B} \rightarrow \mathbb{B}} g) \rightarrow \perp$.

Example: independence of funext from CC_ω

Now define:

$$\text{funext} := \prod(A : \square_i)(B : \square_i)(f g : A \rightarrow B).(\prod x : A.(f x =_B g x)) \rightarrow f =_{A \rightarrow B} g$$

Theorem

There exists a closed proof of $\llbracket \text{funext} \rightarrow \perp \rrbracket_f$

Proof. Define $f := (\lambda x : \mathbb{B}.x, \text{true})$ and $g := (\lambda x : \mathbb{B}.x, \text{false})$.

We have $f, g : \llbracket \mathbb{B} \rightarrow \mathbb{B} \rrbracket_f$ and $(f =_{\mathbb{B} \rightarrow \mathbb{B}} g) \rightarrow \perp$.

But for any $x : \mathbb{B}$, $\llbracket f x =_{\mathbb{B}} g x \rrbracket_f$ is $x =_{\mathbb{B}} x$.

Theorem

For any $\vdash_S f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$, there exist a proof $\vdash_{CIC} p : \mathcal{C} f$.

Proof. Construct a model of S in CIC, for which which every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ is eloquent.

Theorem

For any $\vdash_S f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$, there exist a proof $\vdash_{CIC} p : \mathcal{C} f$.

Proof. Construct a model of S in CIC, for which which every function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ is eloquent.

The construction comes in three stages.

First model: branching translation

Remember that we fixed two parameters $\mathbb{I} : \square_0$ and $O : \mathbb{I} \rightarrow \square_0$.

Definition

For any type $\vdash A : \square$, a *pythia* is term:

$$\beta_A : \Pi i : \mathbb{I}. (O\ i \rightarrow A) \rightarrow A$$

First model: branching translation

Remember that we fixed two parameters $\mathbb{I} : \square_0$ and $O : \mathbb{I} \rightarrow \square_0$.

Definition

For any type $\vdash A : \square$, a *pythia* is term:

$$\beta_A : \prod i : \mathbb{I}. (O\ i \rightarrow A) \rightarrow A$$

This first model equips every type $\vdash_S A : \square$ in the source S with a pythia.

For every type $\vdash_S A : \square$, define:

$$[A]_b := ([A]_b : \square, \beta_A) \quad \text{with } \beta_A \text{ a pythia}$$

First model: branching translation, for \mathbb{N}

Define $\llbracket \mathbb{N} \rrbracket_b$ as \mathbb{N}_b with:

Inductive $\mathbb{N}_b : \square :=$

$O_b : \mathbb{N}_b \mid S_b : \mathbb{N}_b \rightarrow \mathbb{N}_b \mid \beta_{\mathbb{N}} : \prod (i : \mathbb{I}). (\odot i \rightarrow \mathbb{N}_b) \rightarrow \mathbb{N}_b.$

First model: branching translation, for \mathbb{N}

Define $\llbracket \mathbb{N} \rrbracket_b$ as \mathbb{N}_b with:

Inductive $\mathbb{N}_b : \square :=$

$O_b : \mathbb{N}_b \mid S_b : \mathbb{N}_b \rightarrow \mathbb{N}_b \mid \beta_{\mathbb{N}} : \prod (i : \mathbb{I}). (\odot i \rightarrow \mathbb{N}_b) \rightarrow \mathbb{N}_b.$

The non-dependent eliminator is:

$$\begin{aligned} \llbracket \mathbb{N}_{\text{cse}} \rrbracket_b & : \quad \prod P : \llbracket \square \rrbracket_b. \llbracket P \rrbracket_b \rightarrow (\mathbb{N}_b \rightarrow \llbracket P \rrbracket_b \rightarrow \llbracket P \rrbracket_b) \rightarrow \mathbb{N}_b \rightarrow \llbracket P \rrbracket_b \\ \llbracket \mathbb{N}_{\text{cse}} \rrbracket_b P \rho_O \rho_S O_b & : = \rho_O \\ \llbracket \mathbb{N}_{\text{cse}} \rrbracket_b P \rho_O \rho_S (S_b n) & : = \rho_S n (\llbracket \mathbb{N}_{\text{cse}} \rrbracket_b P \rho_O \rho_S n) \end{aligned}$$

First model: branching translation, for \mathbb{N}

Define $\llbracket \mathbb{N} \rrbracket_b$ as \mathbb{N}_b with:

Inductive $\mathbb{N}_b : \square :=$

$O_b : \mathbb{N}_b \mid S_b : \mathbb{N}_b \rightarrow \mathbb{N}_b \mid \beta_{\mathbb{N}} : \prod(i : \mathbb{I}). (\mathbb{O} i \rightarrow \mathbb{N}_b) \rightarrow \mathbb{N}_b.$

The non-dependent eliminator is:

$$\begin{aligned} \llbracket \mathbb{N}_{\text{cse}} \rrbracket_b & : \quad \prod P : \llbracket \square \rrbracket_b. \llbracket P \rrbracket_b \rightarrow (\mathbb{N}_b \rightarrow \llbracket P \rrbracket_b \rightarrow \llbracket P \rrbracket_b) \rightarrow \mathbb{N}_b \rightarrow \llbracket P \rrbracket_b \\ \llbracket \mathbb{N}_{\text{cse}} \rrbracket_b P \rho_O \rho_S O_b & := \rho_O \\ \llbracket \mathbb{N}_{\text{cse}} \rrbracket_b P \rho_O \rho_S (S_b n) & := \rho_S n (\llbracket \mathbb{N}_{\text{cse}} \rrbracket_b P \rho_O \rho_S n) \\ \llbracket \mathbb{N}_{\text{cse}} \rrbracket_b P \rho_O \rho_S (\beta_{\mathbb{N}} i k) & := \beta_P i (\lambda(o : \mathbb{O} i). \llbracket \mathbb{N}_{\text{cse}} \rrbracket_b P \rho_O \rho_S (k o)) \end{aligned}$$

Agenda of the dependent eliminator:

- For $P : \mathbb{N}_b \rightarrow \llbracket \square \rrbracket_b$;
- From p_O and p_S ;
- Produce a term of type $(P (\beta_{\mathbb{N}} i k)).\pi_1$

Agenda of the dependent eliminator:

- For $P : \mathbb{N}_b \rightarrow \llbracket \square \rrbracket_b$;
- From p_O and p_S ;
- Produce a term of type $(P (\beta_{\mathbb{N}} i k)).\pi_1$

But this is impossible.

[An Effectful Way to Eliminate Addiction to Dependence, P.-M. Pédrot, N, Tabareau. Procs. of LICS 2017.]

Agenda of the dependent eliminator:

- For $P : \mathbb{N}_b \rightarrow \llbracket \square \rrbracket_b$;
- From p_0 and p_S ;
- Produce a term of type $(P (\beta_{\mathbb{N}} i k)).\pi_1$

But this is impossible.

Way out: restrict elimination in the source, and take $\mathcal{S} := \text{BTT}$.

[An Effectful Way to Eliminate Addiction to Dependence, P.-M. Pédrot, N. Tabareau. Procs. of LICS 2017.]

$$\frac{\Gamma \vdash P : \square \quad \Gamma \vdash t_0 : P \quad \Gamma \vdash t_S : \mathbb{N} \rightarrow P \rightarrow P}{\Gamma \vdash \mathbb{N}_{\text{cse}} P t_0 t_S : \mathbb{N} \rightarrow P}$$

$$\frac{\Gamma \vdash P : \mathbb{N} \rightarrow \square \quad \Gamma \vdash t_0 : \mathbb{N}_{\text{str}} \text{O} P \quad \Gamma \vdash t_S : \Pi(n : \mathbb{N}). \mathbb{N}_{\text{str}} n P \rightarrow \mathbb{N}_{\text{str}} (S n) P}{\Gamma \vdash \mathbb{N}_{\text{rec}} P t_0 t_S : \Pi(n : \mathbb{N}). \mathbb{N}_{\text{str}} n P}$$

$$\frac{\Gamma \vdash P : \square \quad \Gamma \vdash t_O : P \quad \Gamma \vdash t_S : \mathbb{N} \rightarrow P \rightarrow P}{\Gamma \vdash \mathbb{N}_{\text{cse}} P t_O t_S : \mathbb{N} \rightarrow P}$$

$$\frac{\Gamma \vdash P : \mathbb{N} \rightarrow \square \quad \Gamma \vdash t_O : \mathbb{N}_{\text{str}} O P \quad \Gamma \vdash t_S : \Pi(n : \mathbb{N}). \mathbb{N}_{\text{str}} n P \rightarrow \mathbb{N}_{\text{str}} (S n) P}{\Gamma \vdash \mathbb{N}_{\text{rec}} P t_O t_S : \Pi(n : \mathbb{N}). \mathbb{N}_{\text{str}} n P}$$

where

$$\begin{aligned} \mathbb{N}_{\text{str}} (n : \mathbb{N}) (P : \mathbb{N} \rightarrow \square) : \square &:= \\ \mathbb{N}_{\text{cse}} ((\mathbb{N} \rightarrow \square) \rightarrow \square) (\lambda(Q : \mathbb{N} \rightarrow \square). Q O) & \\ (\lambda(m : \mathbb{N}) (_ : (\mathbb{N} \rightarrow \square) \rightarrow \square) (Q : \mathbb{N} \rightarrow \square). Q (S m)) n P. & \end{aligned}$$

Theorem

The branching translation $[_]_b$ defines a syntactic model from BTT to CIC.

Dialogue in the branching model

Inductive $\mathbb{N}_b : \square :=$

$\mathbf{O}_b : \mathbb{N}_b \mid \mathbf{S}_b : \mathbb{N}_b \rightarrow \mathbb{N}_b \mid \beta_{\mathbb{N}} : \prod (i : \mathbb{I}). (\odot i \rightarrow \mathbb{N}_b) \rightarrow \mathbb{N}_b.$

$$\begin{aligned} \partial^{\mathbb{N}} & : \quad \mathbb{Q} \rightarrow \mathbb{N}_b \rightarrow \mathbb{N} \\ \partial^{\mathbb{N}} \alpha \mathbf{O}_b & := \quad \mathbf{O} \\ \partial^{\mathbb{N}} \alpha (\mathbf{S}_b n_b) & := \quad \mathbf{S} (\partial^{\mathbb{N}} \alpha n_b) \\ \partial^{\mathbb{N}} \alpha (\beta_{\mathbb{N}} i k) & := \quad \partial^{\mathbb{N}} \alpha (k (\alpha i)). \end{aligned}$$

Relate:

$$\alpha : (\mathbb{N} \rightarrow \mathbb{N}) \vdash n : \mathbb{N}$$

with:

$$\vdash n_b : \mathbb{N}_b$$

Second model: axiom translation from BTT to CIC

This second model forces the presence a reserved variable $\alpha : \mathbb{Q}$ in the context.

$$\begin{aligned} [\Box_i]_a &:= \Box_i \\ [A]_a &:= [A]_a \\ [\cdot]_a &:= \alpha : \mathbb{Q} \\ [\Gamma, x : A]_a &:= [\Gamma]_a, x_a : [A]_a \\ \dots &:= \dots \end{aligned}$$

Second model: axiom translation from BTT to CIC

This second model forces the presence a reserved variable $\alpha : \mathbb{Q}$ in the context.

$$\begin{aligned} [\Box_i]_a &:= \Box_i \\ [A]_a &:= [A]_a \\ [\cdot]_a &:= \alpha : \mathbb{Q} \\ [\Gamma, x : A]_a &:= [\Gamma]_a, x_a : [A]_a \\ \dots &:= \dots \end{aligned}$$

Theorem

The branching translation $[_]_a$ defines a syntactic model from BTT to CIC.

Note: $[_]_a$ should really be denoted $[_]_a^\alpha$, for a given name α .

Third model: algebraic parametricity translation

Given $\alpha : \mathbb{N} \rightarrow \mathbb{N}, \Gamma \vdash t : A$, relate:

$$\llbracket \Gamma \rrbracket_a \vdash \llbracket t \rrbracket_a : \llbracket A \rrbracket_a$$

with:

$$\llbracket \Gamma \rrbracket_b \vdash \llbracket t \rrbracket_b : \llbracket A \rrbracket_b$$

using an **internal** logical relation:

$$\llbracket \Gamma \rrbracket_\varepsilon \vdash \llbracket t \rrbracket_\varepsilon : \llbracket A \rrbracket_\varepsilon \llbracket t \rrbracket_a \llbracket t \rrbracket_b$$

Third model: algebraic parametricity translation

Given $\alpha : \mathbb{N} \rightarrow \mathbb{N}, \Gamma \vdash t : A$, relate:

$$\llbracket \Gamma \rrbracket_a \vdash \llbracket t \rrbracket_a : \llbracket A \rrbracket_a$$

with:

$$\llbracket \Gamma \rrbracket_b \vdash \llbracket t \rrbracket_b : \llbracket A \rrbracket_b$$

using an **internal** logical relation:

$$\llbracket \Gamma \rrbracket_\varepsilon \vdash \llbracket t \rrbracket_\varepsilon : \llbracket A \rrbracket_\varepsilon \llbracket t \rrbracket_a \llbracket t \rrbracket_b$$

In fact, the parametricity predicate $\llbracket A \rrbracket_\varepsilon$ has to be **algebraic** as well.

Third model: algebraic parametricity translation

For every type $\vdash_S A : \square$:

$$[A]_\varepsilon := ([A]_\varepsilon, \beta_A^\varepsilon)$$

with:

- $[A]_\varepsilon : [A]_a \rightarrow [A]_b \rightarrow \square$
- $\beta_A^\varepsilon : \prod (x_a : [A]_a) (i : \mathbb{I}) (k : \mathbb{O} \ i \rightarrow [A]_b). [A]_\varepsilon \ x_a \ (k \ (\alpha \ i)) \rightarrow [A]_\varepsilon \ x_a \ (\beta_A \ i \ k)$

Third model: algebraic parametricity translation

Inductive $\mathbb{N}_\varepsilon (\alpha : \mathbb{Q}) : \mathbb{N} \rightarrow \mathbb{N}_b \rightarrow \square :=$

| $\mathbf{O}_\varepsilon : \mathbb{N}_\varepsilon \alpha \mathbf{O} \mathbf{O}_b$

| $\mathbf{S}_\varepsilon : \prod (n_a : \mathbb{N}) (n_b : \mathbb{N}_b) (n_\varepsilon : \mathbb{N}_\varepsilon \alpha n_a n_b). \mathbb{N}_\varepsilon \alpha (\mathbf{S} n_a) (\mathbf{S}_b n_b)$

Third model: algebraic parametricity translation

Inductive $\mathbb{N}_\varepsilon (\alpha : \mathbb{Q}) : \mathbb{N} \rightarrow \mathbb{N}_b \rightarrow \square :=$

| $\mathbb{O}_\varepsilon : \mathbb{N}_\varepsilon \alpha \mathbb{O} \mathbb{O}_b$

| $\mathbb{S}_\varepsilon : \prod (n_a : \mathbb{N}) (n_b : \mathbb{N}_b) (n_\varepsilon : \mathbb{N}_\varepsilon \alpha n_a n_b). \mathbb{N}_\varepsilon \alpha (\mathbb{S} n_a) (\mathbb{S}_b n_b)$

| $\beta_{\mathbb{N}}^\varepsilon : \prod (n_a : \mathbb{N}) (i : \mathbb{I}) (k : \mathbb{O} i \rightarrow \mathbb{N}_b) (n_\varepsilon : \mathbb{N}_\varepsilon \alpha n_a (k (\alpha i))). \mathbb{N}_\varepsilon \alpha n_a (\beta_{\mathbb{N}} i k)$

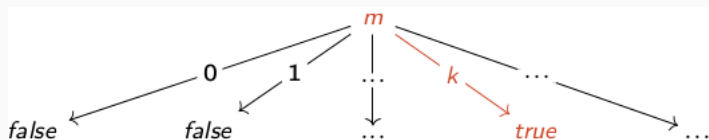
Third model: algebraic parametricity translation

$$\beta_A^\varepsilon : \Pi(x_a : \llbracket A \rrbracket_a) (i : \mathbb{I}) (k : \mathbb{O} i \rightarrow \llbracket A \rrbracket_b). \llbracket A \rrbracket_\varepsilon x_a (k (\alpha i)) \rightarrow \llbracket A \rrbracket_\varepsilon x_a (\beta_A i k)$$

Third model: algebraic parametricity translation

$\beta_A^\varepsilon : \Pi(x_a : \llbracket A \rrbracket_a) (i : \mathbb{I}) (k : \mathbb{O} i \rightarrow \llbracket A \rrbracket_b). \llbracket A \rrbracket_\varepsilon x_a (k (\alpha i)) \rightarrow \llbracket A \rrbracket_\varepsilon x_a (\beta_A i k)$

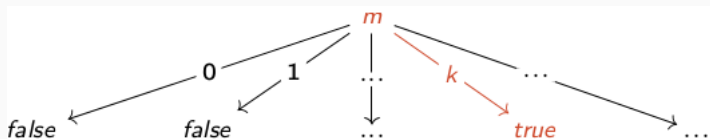
For example, consider the following branching boolean b , for $\mathbb{O} := _ : \mathbb{N} \mapsto \mathbb{N}$:



Third model: algebraic parametricity translation

$\beta_A^\varepsilon : \Pi(x_a : \llbracket A \rrbracket_a) (i : \mathbb{I}) (k : \odot i \rightarrow \llbracket A \rrbracket_b). \llbracket A \rrbracket_\varepsilon x_a (k (\alpha i)) \rightarrow \llbracket A \rrbracket_\varepsilon x_a (\beta_A i k)$

For example, consider the following branching boolean b , for $\odot := _ : \mathbb{N} \mapsto \mathbb{N}$:



As soon as:

$$\alpha m = k$$

We can prove that:

$$\llbracket B \rrbracket_\varepsilon \text{ true } b$$

Third model: algebraic parametricity translation

Theorem

The branching translation $[_]_\varepsilon$ defines a syntactic model from BTT to CIC.

Note: $[_]_\varepsilon$ is parameterized by the name α and by \mathbb{I} and \mathbb{O} .

Theorem

If $\vdash_{BTT} f : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ then $\vdash_{CIC} _ : \mathcal{C} \lambda\alpha.([f]_a^\alpha \alpha)$.

- Unicity

$$\vdash_{CIC} _ : \Pi(\alpha : \mathbb{Q}) (n_a : \llbracket \mathbb{N} \rrbracket_a)(n_b : \llbracket \mathbb{N} \rrbracket_b)(n_\epsilon : \llbracket \mathbb{N}_\epsilon \rrbracket_\epsilon n_a n_b). n_a = \partial^{\mathbb{N}} \alpha n_b$$

- Dialogue relation

$$\vdash_{CIC} _ : \Pi(\alpha : \mathbb{Q}) (n_b : \mathbb{N}_b). \mathbb{N}_\epsilon \alpha (\partial^{\mathbb{N}} \alpha n_b) n_b$$

- Generic element $\gamma_b : \mathbb{N}_b \rightarrow \mathbb{N}_b$

$$\vdash_{CIC} _ : \Pi(\alpha : \mathbb{N} \rightarrow \mathbb{N}) (n_b : \mathbb{N}_b). \partial^{\mathbb{N}} \alpha (\gamma_b n_b) = \alpha (\partial^{\mathbb{N}} \alpha n_b)$$

Theorem

$$\vdash_{CIC} _ : \Pi(\alpha : \mathbb{N} \rightarrow \mathbb{N}). [f]_a^\alpha \alpha = \partial^{\mathbb{N}} \alpha ([f]_b \gamma_b)$$

Proof: First construct:

$$\alpha : \mathbb{N} \rightarrow \mathbb{N} \vdash_{CIC} \gamma_\varepsilon : \llbracket \mathbb{N} \rightarrow \mathbb{N} \rrbracket_\varepsilon \alpha \gamma_b$$

Theorem

$$\vdash_{CIC} _ : \prod(\alpha : \mathbb{N} \rightarrow \mathbb{N}). [f]_a^\alpha \alpha = \partial^{\mathbb{N}} \alpha ([f]_b \gamma_b)$$

Proof: First construct:

$$\alpha : \mathbb{N} \rightarrow \mathbb{N} \vdash_{CIC} \gamma_\varepsilon : \llbracket \mathbb{N} \rightarrow \mathbb{N} \rrbracket_\varepsilon \alpha \gamma_b$$

Then by soundness:

$$\begin{array}{l} \alpha : \mathbb{N} \rightarrow \mathbb{N} \vdash_{CIC} [f]_a \alpha : \mathbb{N} \\ \vdash_{CIC} [f]_b \gamma_b : \mathbb{N}_b \\ \alpha : \mathbb{N} \rightarrow \mathbb{N} \vdash_{CIC} [f]_\varepsilon \alpha \gamma_b \gamma_\varepsilon : \mathbb{N}_\varepsilon \alpha ([f]_a \alpha) ([f]_b \gamma_b) \end{array}$$

And conclude using the previous properties.

- Transpose M. Escardó's proof to a dependently typed setting
- Formalized in Coq
- Internalization?
- Scope of the methodology?

[Gardening with the Pythia. Procs of CSL 2022]