Trocq: Proof Transfer for Free, With or Without Univalence

Cyril Cohen[1], Enzo Crance[2,3], Assia Mahboubi[3]

[1]Université Côte d'Azur, Inria, France
[2]Mitsubishi Electric R&D Centre Europe, France
[3]Nantes Université, École Centrale Nantes, CNRS, INRIA, LS2N, UMR 6004, France

Liber Abaci October 17^{nth}, 2024

Context

I am working with a proof assistant with a trusted kernel in DTT. (E.g. Coq, Lean, Agda, ...)

- I want to get a concrete value **within the proof assistants**, e.g. withing Coq/MATHCOMP [11]:
  - $\begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = -2$, but computation is locked

Context

I am working with a proof assistant with a trusted kernel in DTT. (E.g. Coq, Lean, Agda, ...)

- I want to get a concrete value **within the proof assistants**, e.g. withing Coq/MathComp [11]:
  - $\begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = -2$, but computation is locked

    $\longrightarrow$ I want to "unlock"

Context

I am working with a proof assistant with a trusted kernel in DTT. (E.g. Coq, Lean, Agda, ...)

- I want to get a concrete value **within the proof assistants**, e.g. withing Coq/$\textsc{MathComp}$ [11]:
  - $\begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = -2$, but computation is locked
    $\longrightarrow$ I want to "unlock"
  - $\text{prime}(29986577) = \top$, but computation takes > 2min

Context

I am working with a proof assistant with a trusted kernel in DTT. (E.g. Coq, Lean, Agda, ...)

- I want to get a concrete value **within the proof assistants**, e.g. withing Coq/$\textsc{Mathcomp}$ [11]:
  - $\begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = -2$, but computation is locked
    $\longrightarrow$ I want to "unlock"
  - $\text{prime}(29986577) = \top$, but computation takes $> 2\text{min}$
    $\longrightarrow$ I want to run an optimized algorithm

Context

I am working with a proof assistant with a trusted kernel in DTT. (E.g. Coq, Lean, Agda, ...)

- I want to get a concrete value **within the proof assistants**, e.g. withing Coq/MathComp [11]:
  - $\begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = -2$, but computation is locked
    $\longrightarrow$ I want to "unlock"
  - $\text{prime}(29986577) = \top$, but computation takes $>$ 2min
    $\longrightarrow$ I want to run an optimized algorithm
- My goal is about a given type, but my lemma is about an equivalent one.
  E.g. I want $\forall x : \mathbb{B}, x \oplus x = \bot$

Context

I am working with a proof assistant with a trusted kernel in DTT. (E.g. Coq, Lean, Agda, ...)

- I want to get a concrete value **within the proof assistants**, e.g. withing Coq/MATHCOMP [11]:
  - $\begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = -2$, but computation is locked
    $\longrightarrow$ I want to "unlock"
  - $\text{prime}(29986577) = \top$, but computation takes $>$ 2min
    $\longrightarrow$ I want to run an optimized algorithm
- My goal is about a given type, but my lemma is about an equivalent one.
  E.g. I want $\forall x : \mathbb{B}, x \oplus x = \bot$
  $\longrightarrow$ but I have $\forall x : \mathbb{Z}/2\mathbb{Z}, x + x = 0$

Context

I am working with a proof assistant with a trusted kernel in DTT. (E.g. Coq, Lean, Agda, ...)

- I want to get a concrete value **within the proof assistants**, e.g. withing Coq/MATHCOMP [11]:
  - $\begin{vmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = -2$, but computation is locked
    $\longrightarrow$ I want to "unlock"
  - $\mathrm{prime}(29986577) = \top$, but computation takes $> 2\mathrm{min}$
    $\longrightarrow$ I want to run an optimized algorithm
- My goal is about a given type, but my lemma is about an equivalent one.
  E.g. I want $\forall x : \mathbb{B}, x \oplus x = \bot$
  $\longrightarrow$ but I have $\forall x : \mathbb{Z}/2\mathbb{Z}, x + x = 0$

Context and motivation

Some previous work answering exactly the questions above

- CoqEAL [5] (Cano, me, Dénès, Martin-Dorel, Mörtberg, Rouhling, Roux, Siles), does **data** transfer.
- Univalent Parametricity [13, 14] (Sozeau, Tabareau, Tanter), changes representation **using univalence**.

Context and motivation

Some previous work answering exactly the questions above

- CoqEAL [5] (Cano, me, Dénès, Martin-Dorel, Mörtberg, Rouhling, Roux, Siles), does **data** transfer.
- Univalent Parametricity [13, 14] (Sozeau, Tabareau, Tanter), changes representation **using univalence**.

This work generalizes both. Indeed we may

- change representation **without univalence** in some cases,
- change representation **with partial isos** in some cases.

Comparison to other previous work in the paper.

What is Troc(q)?

Troc [tʁɔk] subst. masc.
Échange direct de biens sans intervention de monnaie.

∼ "A direct exchange of goods without the use of money" CNRTL

What is Troc(q)?

Troc [tʁɔk] subst. masc.
Échange direct de biens sans intervention de monnaie.

∼ "A direct exchange of goods without the use of money" CNRTL

(Proof) **T**ransfer for **Rocq**

Assia, Cyril, Enzo

What is Troc(q)?

Troc [tʁɔk] subst. masc.
Échange direct de biens sans intervention de monnaie.

~ "A direct exchange of goods without the use of money" CNRTL

(Proof) **T**ransfer for **Rocq**

Assia, Cyril, Enzo

It's a calculus, an Elpi implementation of it and a **prototype** associated tactic

What is Troc(q)?

Troc [tʁɔk] subst. masc.
Échange direct de biens sans intervention de monnaie.

∼ "A direct exchange of goods without the use of money" CNRTL

(Proof) **T**ransfer for **Rocq**

Assia, Cyril, Enzo

It's a calculus, an Elpi implementation of it and a **prototype** associated tactic

[Cyril Cohen, Enzo Crance, and Assia Mahboubi. "Artifact Report: Trocq: Proof Transfer for Free, With or Without Univalence". In: *ESOP 2024 - 33rd European Symposium on Programming*. Vol. LNCS-14576. Programming Languages and Systems 33rd European Symposium on Programming, ESOP 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6–11, 2024, Proceedings, Part I. Luxembourg, Luxembourg: Springer Nature Switzerland, Apr. 2024, pp. 269–274. DOI: 10.1007/978-3-031-57262-3_11. URL: https://inria.hal.science/hal-04623207]

1

Old and new examples

The canonical example

We have the standard definition of Peano natural numbers (stdlib)

$$\texttt{Inductive}\ \mathbb{N} := \texttt{O}_{\mathbb{N}} : \mathbb{N} \mid \texttt{S}_{\mathbb{N}}\ (\texttt{n} : \mathbb{N}) : \mathbb{N}.$$

For which, we have:

$$\mathbb{N}_{\text{ind}} : \forall\ \texttt{P} : \mathbb{N} \longrightarrow \square,\ \texttt{P}\ \texttt{O}_{\mathbb{N}} \longrightarrow (\forall\ \texttt{n} : \mathbb{N},\ \texttt{P}\ \texttt{n} \longrightarrow \texttt{P}\ (\texttt{S}\ \texttt{n})) \longrightarrow \forall\ \texttt{n} : \mathbb{N},\ \texttt{P}\ \texttt{n}$$

The canonical example

We have the standard definition of Peano natural numbers (stdlib)

```
Inductive ℕ := 0ℕ : ℕ | Sℕ (n : ℕ) : ℕ.
```

For which, we have:

```
ℕind : ∀ P : ℕ → □, P 0ℕ → (∀ n : ℕ, P n → P (S n)) → ∀ n : ℕ, P n
```

Here is an alternative binary representation (stdlib)

```
Inductive pos := xI : pos → pos | xO : pos → pos | xH : pos.
Inductive N := 0N : N | Npos : pos → N.

Fixpoint Spos (p : pos) : pos := match p with
  xH ⇒ xO xH | xO p ⇒ xI p | xI p ⇒ xO (Spos p) end.

Definition SN (n : N) :=
  match n with Npos p ⇒ Npos (Spos p) | _ ⇒ Npos xH end.
```

The canonical example

We have the standard definition of Peano natural numbers (stdlib)

```
Inductive ℕ := 0ℕ : ℕ | Sℕ (n : ℕ) : ℕ.
```

For which, we have:

```
ℕind : ∀ P : ℕ → □, P 0ℕ → (∀ n : ℕ, P n → P (S n)) → ∀ n : ℕ, P n
```

Here is an alternative binary representation (stdlib)

```
Inductive pos := xI : pos → pos | xO : pos → pos | xH : pos.
Inductive N := 0N : N | Npos : pos → N.

Fixpoint Spos (p : pos) : pos := match p with
  xH ⇒ xO xH | xO p ⇒ xI p | xI p ⇒ xO (Spos p) end.

Definition SN (n : N) :=
  match n with Npos p ⇒ Npos (Spos p) | _ ⇒ Npos xH end.
```

We want

```
Nind : ∀ P : N → □, P 0N → (∀ n : N, P n → P (S n)) → ∀ n : N, P n
```

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.
2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,

Examples of frustration collected during various events

A non exhaustive list

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies n = 0 \text{ or } 1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

Examples of frustration collected during various events

A non exhaustive list

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.
2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.
3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?
   - $\deg : \mathbb{R}[X] \to \mathbb{N}$

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?
   - $\deg : \mathbb{R}[X] \to \mathbb{N}$
   - or $\deg : \mathbb{R}[X] \to \mathbb{N} \cup \{-\infty\}$.

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?
   - $\deg : \mathbb{R}[X] \longrightarrow \mathbb{N}$
   - or $\deg : \mathbb{R}[X] \longrightarrow \mathbb{N} \cup \{-\infty\}$.

   Mathlib has .

.

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?
   - $\deg : \mathbb{R}[X] \longrightarrow \mathbb{N}$
   - or $\deg : \mathbb{R}[X] \longrightarrow \mathbb{N} \cup \{-\infty\}$.

   Mathlib has both: `Polynomial.degree` and `Polynomial.natDegree`.

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.
2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.
3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?
   - $\deg : \mathbb{R}[X] \longrightarrow \mathbb{N}$
   - or $\deg : \mathbb{R}[X] \longrightarrow \mathbb{N} \cup \{-\infty\}$.

   Mathlib has both: `Polynomial.degree` and `Polynomial.natDegree`.
   Mathcomp has .

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?
   - $\deg : \mathbb{R}[X] \longrightarrow \mathbb{N}$
   - or $\deg : \mathbb{R}[X] \longrightarrow \mathbb{N} \cup \{-\infty\}$.

   Mathlib has both: `Polynomial.degree` and `Polynomial.natDegree`.
   Mathcomp has neither, uses $\mathrm{size} : \mathbb{R}[X] \longrightarrow \mathbb{N}$.

Examples of frustration collected during various events

A non exhaustive list

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?
   - $\deg : \mathbb{R}[X] \to \mathbb{N}$
   - or $\deg : \mathbb{R}[X] \to \mathbb{N} \cup \{-\infty\}$.

   Mathlib has both: `Polynomial.degree` and `Polynomial.natDegree`.
   Mathcomp has neither, uses size : $\mathbb{R}[X] \to \mathbb{N}$.
   In set theory $\mathbb{N} \subseteq \mathbb{N} \cup \{-\infty\}$, but in type theory $\mathbb{N} \hookrightarrow \mathbb{N} \cup \{-\infty\}$.

Examples of frustration collected during various events

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9. $\Rightarrow$ Proof of Concept Demo

3. Problems "created by the use of type theory", e.g. what is the type of $\deg(P)$ for $P \in \mathbb{R}[X]$?
   - $\deg : \mathbb{R}[X] \to \mathbb{N}$
   - or $\deg : \mathbb{R}[X] \to \mathbb{N} \cup \{-\infty\}$.

   Mathlib has both: `Polynomial.degree` and `Polynomial.natDegree`.
   Mathcomp has neither, uses size : $\mathbb{R}[X] \to \mathbb{N}$.
   In set theory $\mathbb{N} \subseteq \mathbb{N} \cup \{-\infty\}$, but in type theory $\mathbb{N} \hookrightarrow \mathbb{N} \cup \{-\infty\}$.

Planned integration of Trocqto Coq

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

```
decide  (* := by rewrite computable; vm_compute. *)
```

(decide is the name of the equivalent lean tactic)

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

```
by rewrite (@mod 9); decide. (* where mod : ∀{n}, Z -> Z/nZ *)
```

3. Problems "created by type theory", e.g. in set theory $\mathbb{N} \subseteq \mathbb{N} \cup \{-\infty\}$, but in type theory $\mathbb{N} \hookrightarrow \mathbb{N} \cup \{-\infty\}$.
   - either use `rewrite -Fin (* where Fin : N ->\bar N *)`

Planned integration of Trocqto Coq

1. Computing the degree of a polynomial, e.g. $\deg((2X + X^5 + 1)X^2) = 7$.

```
decide  (* := by rewrite computable; vm_compute. *)
```

(decide is the name of the equivalent lean tactic)

2. Reduction modulo, e.g.
   - n is a square and a cube $\implies$ $n = 0$ or $1 \mod 7$,
   - $3 \nmid abc \implies a^3 + b^3 \neq c^3$ by reduction modulo 9.

```
by rewrite (@mod 9); decide. (* where mod : ∀{n}, Z -> Z/nZ *)
```

3. Problems "created by type theory", e.g. in set theory $\mathbb{N} \subseteq \mathbb{N} \cup \{-\infty\}$, but in type theory $\mathbb{N} \hookrightarrow \mathbb{N} \cup \{-\infty\}$.
   - either use `rewrite -Fin (* where Fin : N ->\bar N *)`
   - or ... use directly $\mathbb{N}$ lemmas on $\mathbb{N} \cup \{-\infty\}$

2

Revisiting parametricity and univalent parametricity

Parametricity: standard version [2]

- Context translation:

$$\llbracket \langle\rangle \rrbracket = \langle\rangle \tag{1}$$

$$\llbracket \Gamma, x : A \rrbracket = \llbracket \Gamma \rrbracket, x : A, x' : A', x_R : \llbracket A \rrbracket\ x\ x' \tag{2}$$

- Term translation:

$$\llbracket \Box_i \rrbracket = \lambda A\, A'.\, A \to A' \to \Box_i \tag{3}$$

$$\llbracket x \rrbracket = x_R \tag{4}$$

$$\llbracket A\ B \rrbracket = \llbracket A \rrbracket\ B\ B'\ \llbracket B \rrbracket \tag{5}$$

$$\llbracket \lambda x : A.\, t \rrbracket = \lambda(x : A)(x' : A')(x_R : \llbracket A \rrbracket\ x\ x').\, \llbracket t \rrbracket \tag{6}$$

$$\llbracket \Pi x : A.\, B \rrbracket = \lambda f\ f'.\, \Pi(x : A)(x' : A')(x_R : \llbracket A \rrbracket\ x\ x').\ \llbracket B \rrbracket (f\ x)(f'\ x') \tag{7}$$

Parametricity: standard version [2]

- Context translation:

$$[\![\langle\rangle]\!] = \langle\rangle \tag{1}$$

$$[\![\Gamma, x : A]\!] = [\![\Gamma]\!], x : A, x' : A', x_R : [\![A]\!]\ x\ x' \tag{2}$$

- Term translation:

$$[\![\square_i]\!] = \lambda A\, A'.\, A \to A' \to \square_i \tag{3}$$

$$[\![x]\!] = x_R \tag{4}$$

$$[\![A\ B]\!] = [\![A]\!]\ B\ B'\ [\![B]\!] \tag{5}$$

$$[\![\lambda x : A.\, t]\!] = \lambda (x : A)(x' : A')(x_R : [\![A]\!]\ x\ x').\, [\![t]\!] \tag{6}$$

$$[\![\Pi x : A.\, B]\!] = \lambda f\, f'.\, \Pi (x : A)(x' : A')(x_R : [\![A]\!]\ x\ x').\ [\![B]\!](f\ x)(f'\ x') \tag{7}$$

- Abstraction theorem: If $\Gamma \vdash t : T$ then $[\![\Gamma]\!] \vdash t : T$, $[\![\Gamma]\!] \vdash t' : T'$ and $[\![\Gamma]\!] \vdash [\![t]\!] : [\![T]\!]\ t\ t'$.

Parametricity: sequent style

Parametricity contexts:

$$\Xi ::= \varepsilon \mid \Xi,\; x \sim x' \;\because\; x_R.$$

Parametricity rules:

$$\frac{}{\Xi \vdash \square_i \sim \square_i \;\because\; \lambda(A\,B : \square_i).\, A \to B \to \square_i}\ (\textsc{ParamSort})$$

$$\frac{(x, x', x_R) \in \Xi \qquad \Xi \vdash}{\Xi \vdash x \sim x' \;\because\; x_R}\ (\textsc{ParamVar})$$

$$\frac{\Xi \vdash M \sim M' \;\because\; M_R \qquad \Xi \vdash N \sim N' \;\because\; N_R}{\Xi \vdash M\,N \sim M'\,N' \;\because\; M_R\,N\,N'\,N_R}\ (\textsc{ParamApp})$$

$$\frac{\Xi, x \sim x' \;\because\; x_R \vdash M \sim M' \;\because\; M_R}{\Xi \vdash \lambda x : A.\, M \sim \lambda x' : A'.\, M' \;\because\; \lambda x\,x'\,x_R.\, M_R}\ (\textsc{ParamLam})$$

$$\frac{\Xi \vdash A \sim A' \;\because\; A_R \qquad \Xi, x \sim x' \;\because\; x_R \vdash B \sim B' \;\because\; B_R \qquad x, x' \notin \mathrm{Var}(\Xi)}{\Xi \vdash \Pi x : A.\, B \sim \Pi x' : A'.\, B' \;\because\; \lambda f\,g.\, \Pi x\,x'\,x_R.\, B_R\ (f\ x)\ (g\ x')}\ (\textsc{ParamPi})$$

Parametricity: sequent abstraction theorem

We say Ξ is admissible for Γ if

$$\Gamma \rhd \Xi \triangleq \frac{\Xi \vdash \Gamma(x) \sim A' \because A_R}{\Gamma(x') = A' \wedge \Gamma(x_R) = A_R\ x\ x'}$$

We rephrase the abstraction theorem:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash M : A \qquad \Gamma \rhd \Xi \qquad \Xi \vdash M \sim M' \because M_R \qquad \Xi \vdash A \sim A' \because A_R}{\Gamma \vdash M' : A' \qquad \text{and} \qquad \Gamma \vdash M_R : A_R\ M\ M'}$$

In particular, by applying it to $\Gamma \vdash A : \square_i$ instead, we get:

$$\frac{\Gamma \rhd \Xi \qquad \Xi \vdash A \sim A' \because A_R}{\Gamma \vdash A_R : A \to A' \to \square_i}$$

Example: motivating raw paramericity

Assume we have a derivation

$$\dfrac{\vdots}{\mathsf{fold}\ \mathbb{N}\ 0_{\mathbb{N}}\ (+_{\mathbb{N}})[1_{\mathbb{N}}, 2_{\mathbb{N}}, 3_{\mathbb{N}}] \sim \mathsf{fold}\ \mathrm{N}\ 0_{\mathrm{N}}\ (+_{\mathrm{N}})[1_{\mathrm{N}}, 2_{\mathrm{N}}, 3_{\mathrm{N}}]\ \because\ w}$$

Example: motivating raw paramericity

Assume $\phi : \mathbb{N} \to \mathrm{N}$ and

$$\begin{aligned} \mathbb{N} \sim \mathrm{N} &\because \lambda(m : \mathbb{N})(n : \mathrm{N}).\phi(m) = n \\ 0_{\mathbb{N}} \sim 0_{\mathrm{N}} &\because (0_R : \phi(0_{\mathbb{N}}) = 0_{\mathrm{N}}) \\ S_{\mathbb{N}} \sim S_{\mathrm{N}} &\because (S_R : \forall m \forall n, \phi(m) = n \to \phi(S_{\mathbb{N}} m) = S_{\mathrm{N}} n) \end{aligned}$$

Assume we have a derivation

$$\frac{\vdots}{\text{fold } \mathbb{N}\ 0_{\mathbb{N}}\ (+_{\mathbb{N}})[1_{\mathbb{N}}, 2_{\mathbb{N}}, 3_{\mathbb{N}}] \sim \text{fold } \mathrm{N}\ 0_{\mathrm{N}}\ (+_{\mathrm{N}})[1_{\mathrm{N}}, 2_{\mathrm{N}}, 3_{\mathrm{N}}] \because w}$$

Example: motivating raw paramericity

Assume $\phi : \mathbb{N} \to \mathrm{N}$ and

$$\begin{aligned}
\mathbb{N} &\sim \mathrm{N} \because \lambda(m : \mathbb{N})(n : \mathrm{N}).\phi(m) = n \\
0_{\mathbb{N}} &\sim 0_{\mathrm{N}} \because (0_R : \phi(0_{\mathbb{N}}) = 0_{\mathrm{N}}) \\
S_{\mathbb{N}} &\sim S_{\mathrm{N}} \because (S_R : \forall m \forall n, \phi(m) = n \to \phi(S_{\mathbb{N}} m) = S_{\mathrm{N}} n)
\end{aligned}$$

Assume we have a derivation

$$\frac{\vdots}{\mathrm{fold}\ \mathbb{N}\ 0_{\mathbb{N}}\ (+_{\mathbb{N}})[1_{\mathbb{N}}, 2_{\mathbb{N}}, 3_{\mathbb{N}}] \sim \mathrm{fold}\ \mathrm{N}\ 0_{\mathrm{N}}\ (+_{\mathrm{N}})[1_{\mathrm{N}}, 2_{\mathrm{N}}, 3_{\mathrm{N}}] \because w}$$

Then w has type

$$\phi\,(\mathrm{fold}\ \mathbb{N}\ 0_{\mathbb{N}}\ (+_{\mathbb{N}})[1_{\mathbb{N}}, 2_{\mathbb{N}}, 3_{\mathbb{N}}]) = \mathrm{fold}\ \mathrm{N}\ 0_{\mathrm{N}}\ (+_{\mathrm{N}})[1_{\mathrm{N}}, 2_{\mathrm{N}}, 3_{\mathrm{N}}]$$

Example: motivating univalent paramericity

Assume $P : \square_i \longrightarrow \square_j$ is a closed term.

$$P\mathbb{N} \sim P\mathbf{N} \because w$$

The witness w has type

$$[\![\square_j]\!] \, (P\mathbb{N}) \, (P\mathbf{N})$$

i.e.

$$(P\mathbb{N}) \longrightarrow (P\mathbf{N}) \longrightarrow \square_j$$

Example: motivating univalent paramericity

Assume $P : \square_i \longrightarrow \square_j$ is a closed term.

$$P\mathbb{N} \sim P\mathbf{N} \because w$$

The witness w has type

$$[\![\square_j]\!] \; (P\mathbb{N}) \; (P\mathbf{N})$$

i.e.

$$(P\mathbb{N}) \longrightarrow (P\mathbf{N}) \longrightarrow \square_j$$

We want

$$(P\mathbb{N}) \leftrightarrow (P\mathbf{N})$$

Univalent parametricity: standard version

- Term translation:

$$[\,\square_i\,] = p_{\square_i}$$

$$[\,x\,] = x_R$$

$$[\,A\ B\,] = [\,A\,]\ B\ B'\ [\,B\,]$$

$$[\,\lambda x : A.\,t\,] = \Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\,t\,]$$

$$[\,\Pi x : A.\,B\,] = p_{\Pi}\ [\,A\,]\ [\,B\,]$$

- Type translation: $[\![\,A\,]\!] = [\,A\,].1 \qquad [\![\,A\,]\!]^{\text{eq}} = [\,A\,].2 \qquad [\![\,A\,]\!]^{\text{coh}} = [\,A\,].3$

Univalent parametricity: standard version

- Term translation:

$$[\,\square_i\,] = \begin{pmatrix} \lambda A\,B.\,\Sigma(R : \mathsf{rel}_i\ A\ B)(e : A \simeq B).\Pi ab.(R\ a\ b) \simeq (a = e^{-1} b) \\ \mathsf{id}_{\square_i} \\ \mathsf{univ}_{\square_i} \end{pmatrix}$$

$$[\,x\,] = x_R$$

$$[\,A\ B\,] = [\,A\,]\ B\ B'\ [\,B\,]$$

$$[\,\lambda x : A.\,t\,] = \Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\,t\,]$$

$$[\,\Pi x : A.\,B\,] = \begin{pmatrix} \lambda f\ f'.\,\Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\![\,B\,]\!](f\ x)(f'\ x') \\ \mathsf{Equiv}_{\Pi}[\![\,A\,]\!]^{\mathsf{eq}}[\![\,B\,]\!]^{\mathsf{eq}} \\ \mathsf{univ}_{\Pi} \end{pmatrix}$$

- Type translation: $[\![\,A\,]\!] = [\,A\,].1 \qquad [\![\,A\,]\!]^{\mathsf{eq}} = [\,A\,].2 \qquad [\![\,A\,]\!]^{\mathsf{coh}} = [\,A\,].3$

Univalent parametricity: standard version

- Term translation:

$$[\,\square_i\,] = \begin{pmatrix} \lambda A\,B.\,\Sigma(R : \mathsf{rel}_i\ A\ B)(e : A \simeq B).\Pi ab.(R\ a\ b) \simeq (a = e^{-1} b) \\ \mathsf{id}_{\square_i} \\ \mathsf{univ}_{\square_i} \end{pmatrix}$$

$$[\,x\,] = x_R$$

$$[\,A\ B\,] = [\,A\,]\ B\ B'\ [\,B\,]$$

$$[\,\lambda x : A.\,t\,] = \Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\,t\,]$$

$$[\,\Pi x : A.\,B\,] = \begin{pmatrix} \lambda f\,f'.\,\Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\![\,B\,]\!](f\ x)(f'\ x') \\ \mathsf{Equiv}_{\Pi}[\![\,A\,]\!]^{\mathsf{eq}}[\![\,B\,]\!]^{\mathsf{eq}} \\ \mathsf{univ}_{\Pi} \end{pmatrix}$$

- Type translation: $[\![\,A\,]\!] = [\,A\,].1 \qquad [\![\,A\,]\!]^{\mathsf{eq}} = [\,A\,].2 \qquad [\![\,A\,]\!]^{\mathsf{coh}} = [\,A\,].3$
- Abstraction theorem: If $\Gamma \vdash t : T$ then $[\![\,\Gamma\,]\!] \vdash [\,t\,] : [\![\,T\,]\!]\ t\ t'$.

Univalent parametricity: standard version

- Term translation:

$$[\,\square_i\,] = \begin{pmatrix} \lambda A\,B.\,\Sigma(R : \mathrm{rel}_i\ A\ B)(e : A \simeq B).\Pi ab.(R\ a\ b) \simeq (a = e^{-1} b) \\ \mathrm{id}_{\square_i} \\ \mathrm{univ}_{\square_i} \end{pmatrix}$$

$$[\,x\,] = x_R$$

$$[\,A\ B\,] = [\,A\,]\ B\ B'\ [\,B\,]$$

$$[\,\lambda x : A.\,t\,] = \Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\,t\,]$$

$$[\,\Pi x : A.\,B\,] = \begin{pmatrix} \lambda f\,f'.\,\Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\![\,B\,]\!](f\ x)(f'\ x') \\ \mathrm{Equiv}_{\Pi}[\![\,A\,]\!]^{\mathrm{eq}}[\![\,B\,]\!]^{\mathrm{eq}} \\ \mathrm{univ}_{\Pi} \end{pmatrix}$$

- Type translation: $[\![\,A\,]\!] = [\,A\,].1 \qquad [\![\,A\,]\!]^{\mathrm{eq}} = [\,A\,].2 \qquad [\![\,A\,]\!]^{\mathrm{coh}} = [\,A\,].3$
- Abstraction theorem: If $\Gamma \vdash t : T$ then $[\![\,\Gamma\,]\!] \vdash [\,t\,] : [\![\,T\,]\!]\ t\ t'$.
- Remark A: If $\Gamma \vdash A : \square_i$ then $[\![\,\Gamma\,]\!] \vdash [\,A\,] : [\![\,\square_i\,]\!]\ A\ A'$.

Univalent parametricity: standard version

- Term translation:

$$[\,\square_i\,] = \begin{pmatrix} \lambda A\,B.\,\Sigma(R : \mathsf{rel}_i\ A\ B)(e : A \simeq B).\Pi ab.(R\ a\ b) \simeq (a = e^{-1} b) \\ \mathsf{id}_{\square_i} \\ \mathsf{univ}_{\square_i} \end{pmatrix}$$

$$[\,x\,] = x_R$$

$$[\,A\ B\,] = [\,A\,]\ B\ B'\ [\,B\,]$$

$$[\,\lambda x : A.\,t\,] = \Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\,t\,]$$

$$[\,\Pi x : A.\,B\,] = \begin{pmatrix} \lambda f\,f'.\,\Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\![\,B\,]\!](f\ x)(f'\ x') \\ \mathsf{Equiv}_{\Pi}[\![\,A\,]\!]^{\mathsf{eq}}[\![\,B\,]\!]^{\mathsf{eq}} \\ \mathsf{univ}_{\Pi} \end{pmatrix}$$

- Type translation: $[\![\,A\,]\!] = [\,A\,].1 \qquad [\![\,A\,]\!]^{\mathsf{eq}} = [\,A\,].2 \qquad [\![\,A\,]\!]^{\mathsf{coh}} = [\,A\,].3$
- Abstraction theorem: If $\Gamma \vdash t : T$ then $[\![\,\Gamma\,]\!] \vdash [\,t\,] : [\![\,T\,]\!]\ t\ t'$.
- Remark A: If $\Gamma \vdash A : \square_i$ then $[\![\,\Gamma\,]\!] \vdash [\,A\,] : [\![\,\square_i\,]\!]\ A\ A'$.
- Remark B: $\vdash [\,\square_i\,] : [\![\,\square_{i+1}\,]\!]\ \square_i\ \square_i$.

Univalent parametricity: standard version

- Term translation:

$$[\,\square_i\,] = p_{\square_i}$$

$$\begin{aligned}[\,x\,] &= x_R \\ [\,A\ B\,] &= [\,A\,]\ B\ B'\ [\,B\,] \\ [\,\lambda x : A.\,t\,] &= \Pi(x : A)(x' : A')(x_R : [\![\,A\,]\!]\ x\ x').\,[\,t\,]\end{aligned}$$

$$[\,\Pi x : A.\,B\,] = p_{\Pi}\ [\,A\,]\ [\,B\,]$$

- Type translation: $[\![\,A\,]\!] = [\,A\,].1 \qquad [\![\,A\,]\!]^{\mathrm{eq}} = [\,A\,].2 \qquad [\![\,A\,]\!]^{\mathrm{coh}} = [\,A\,].3$
- Abstraction theorem: If $\Gamma \vdash t : T$ then $[\![\,\Gamma\,]\!] \vdash [\,t\,] : [\![\,T\,]\!]\ t\ t'$.
- Remark A: If $\Gamma \vdash A : \square_i$ then $[\![\,\Gamma\,]\!] \vdash [\,A\,] : [\![\,\square_i\,]\!]\ A\ A'$.
- Remark B: $\vdash [\,\square_i\,] : [\![\,\square_{i+1}\,]\!]\ \square_i\ \square_i$.

Univalent parametricity: sequent style

$$\frac{}{\Xi \vdash_u \square_i \sim \square_i \because p_{\square_i}} \text{(UParamSort)}$$

$$\frac{(x, x', x_R) \in \Xi \qquad \Xi \vdash}{\Xi \vdash_u x \sim x' \because x_R} \text{(UParamVar)}$$

$$\frac{\Xi \vdash_u M \sim M' \because M_R \qquad \Xi \vdash_u N \sim N' \because N_R}{\Xi \vdash_u M\ N \sim M'\ N' \because M_R\ N\ N'\ N_R} \text{(UParamApp)}$$

$$\frac{\Xi \vdash_u A \sim A' \because A_R \qquad \Xi, x \sim x' \because x_R \vdash_u M \sim M' \because M_R}{\Xi \vdash_u \lambda x : A.\, M \sim \lambda x' : A'.\, M' \because \lambda x\, x'\, x_R.\, M_R} \text{(UParamLam)}$$

$$\frac{\Xi \vdash_u A \sim A' \because A_R \qquad \Xi, x \sim x' \because x_R \vdash_u B \sim B' \because B_R}{\Xi \vdash_u \Pi x : A.\, B \sim \Pi x' : A'.\, B' \because p_\Pi\ A_R\ B_R} \text{(UParamPi)}$$

Univalent parametricity: sequent abstraction

We rephrase the Univalent parametricity abstraction theorem:

$$\frac{\mathit{\Gamma} \vdash \qquad \mathit{\Gamma} \vdash M : A \qquad \mathit{\Gamma} \rhd \mathit{\Xi} \qquad \mathit{\Xi} \vdash_u M \sim M' \because M_R \qquad \mathit{\Xi} \vdash_u A \sim A' \because A_R}{\mathit{\Gamma} \vdash M' : A' \qquad \text{and} \qquad \mathit{\Xi} \vdash_u M_R : \left(A_R\ M\ M'\right).1}$$

Remark A:

$$\frac{\mathit{\Gamma} \vdash A : \square_i \qquad \mathit{\Xi} \vdash_u A \sim A' \because A_R \qquad \mathit{\Gamma} \rhd \mathit{\Xi}}{\mathit{\Gamma} \vdash_u A_R : \left(p_{\square_i}\ A\ A'\right).1}$$

Remark B:

$$\vdash_u p_{\square_i} : \left(p_{\square_{i+1}}\ \square_i\ \square_i\right).1$$

3

Type equivalence in a kit

Observation

The key datastructure in univalent parametricity is the one of **relations which are the graph of equivalences**

$$\boxdot^{u}\ A\ B \triangleq \left(\Sigma(R : A \to B \to \square)(e : A \simeq B).\Pi ab.(R\ a\ b) \simeq (a = e^{-1} b)\right).$$

Observation

The key datastructure in univalent parametricity is the one of **relations which are the graph of equivalences**

$$\boxdot^{u}\ A\ B \triangleq \left(\Sigma(R : A \to B \to \square)(e : A \simeq B).\Pi ab.(R\ a\ b) \simeq (a = e^{-1} b)\right).$$

The two following observations

- "inhabiting this structure triggers uses of univalence",
- "it is not symmetric (one direction is privileged in e)",

Observation

The key datastructure in univalent parametricity is the one of **relations which are the graph of equivalences**

$$\boxdot^{u}\ A\ B \triangleq \left(\Sigma(R : A \to B \to \Box)(e : A \simeq B).\Pi ab.(R\ a\ b) \simeq (a = e^{-1} b)\right).$$

The two following observations

- "inhabiting this structure triggers uses of univalence",
- "it is not symmetric (one direction is privileged in e)",

correspond exactly to the two achievements:

- change representation **without univalence** in some cases,
- change representation **with partial isos** in some cases.

Disassembling type equivalence

- We use a variation of (exercise in the HoTT Book):

$$(A \simeq B) \;\simeq\; \Sigma R : A \to B \to \square.\, \mathsf{IsFun}(R) \times \mathsf{IsFun}(R^{-1})$$

$$\text{with} \quad \mathsf{IsFun}(R) \triangleq \Pi a : A.\, \mathsf{IsContr}(\Sigma b : B.\, R\ a\ b)$$

$$R^{-1} \triangleq \lambda a\ b.\, R\ b\ a$$

Disassembling type equivalence

- We use a variation of (exercise in the HoTT Book):

$$(A \simeq B) \;\simeq\; \Sigma R : A \to B \to \square.\, \mathsf{IsFun}(R) \times \mathsf{IsFun}(R^{-1})$$

$$\text{with} \quad \mathsf{IsFun}(R) \triangleq \Pi a : A.\, \mathsf{IsContr}(\Sigma b : B.\, R\; a\; b)$$

$$R^{-1} \triangleq \lambda a\; b.\, R\; b\; a$$

- Then we remark $\mathsf{IsFun}(R) \simeq \mathsf{IsUmap}(R)$, where

$$\begin{aligned}\mathsf{IsUmap}(R) \triangleq\; & \Sigma(m : A \to B). \\ & \Sigma(g_1 : \Pi(a : A)(b : B).\, m\; a = b \to R\; a\; b). \\ & \Sigma(g_2 : \Pi(a : A)(b : B).\, R\; a\; b \to m\; a = b). \\ & \Pi(a : A)(b : B).\, (g_1\; a\; b) \circ (g_2\; a\; b) \doteqdot id.\end{aligned}$$

Disassembling type equivalence

- We use a variation of (exercise in the HoTT Book):

$$(A \simeq B) \;\simeq\; \Sigma R : A \to B \to \square.\, \mathsf{IsFun}(R) \times \mathsf{IsFun}(R^{-1})$$

$$\text{with} \quad \mathsf{IsFun}(R) \triangleq \Pi a : A.\, \mathsf{IsContr}(\Sigma b : B.\, R\ a\ b)$$

$$R^{-1} \triangleq \lambda a\ b.\, R\ b\ a$$

- Then we remark $\mathsf{IsFun}(R) \simeq \mathsf{IsUmap}(R)$, where

$$\begin{aligned} \mathsf{IsUmap}(R) \triangleq\ & \Sigma(m : A \to B). \\ & \Sigma(g_1 : \Pi(a : A)(b : B).\, m\ a = b \to R\ a\ b). \\ & \Sigma(g_2 : \Pi(a : A)(b : B).\, R\ a\ b \to m\ a = b). \\ & \Pi(a : A)(b : B).\, (g_1\ a\ b) \circ (g_2\ a\ b) \doteqdot id. \end{aligned}$$

- We pose

$$\boxdot^{\top}\ A\ B \;\triangleq\; \Sigma R : A \to B \to \square.\, \mathsf{IsUmap}(R) \times \mathsf{IsUmap}(R^{-1})$$

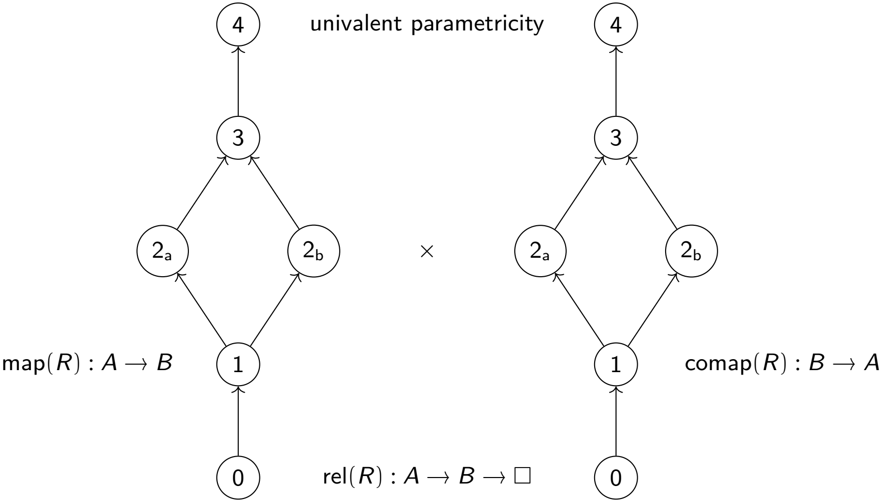
Reassembling type equivalence

For $\alpha = (n, k) \in \mathcal{A} \triangleq \{0, 1, 2_a, 2_b, 3, 4\}^2$, we pose:

$$\boxdot^{\alpha} \triangleq \lambda(A\ B\ :\ \square).\Sigma(R\ :\ A \to B \to \square).\mathrm{Class}_{\alpha}\ R$$

$$\begin{aligned}
\mathrm{Class}_{\alpha}\ R &\triangleq (\mathrm{M}_n\ R) \times (\mathrm{M}_k\ R^{-1}) \\
\mathrm{M}_0\ R &\triangleq . \\
\mathrm{M}_1\ R &\triangleq (A \to B) \\
\mathrm{M}_{2_a}\ R &\triangleq \Sigma m : A \to B.\, G_{2_a}\ m\ R \\
G_{2_a}\ m\ R &\triangleq \Pi a\, b.\, m\ a = b \to R\ a\ b \\
\mathrm{M}_{2_b}\ R &\triangleq \Sigma m : A \to B.\, G_{2_b}\ m\ R \\
G_{2_b}\ m\ R &\triangleq \Pi a\, b.\, R\ a\ b \to m\ a = b \\
\mathrm{M}_3\ R &\triangleq \Sigma m : A \to B.\, (G_{2_a}\ m\ R) \times (G_{2_b}\ m\ R) \\
\mathrm{M}_4\ R &\triangleq \Sigma m : A \to B.\, \Sigma(g_1 : G_{2_a}\ m\ R).\, \Sigma(g_2 : G_{2_b}\ m\ R). \\
&\qquad \Pi a\, b.\, (g_1\ a\ b) \circ (g_1\ a\ b) \doteqdot id
\end{aligned}$$

The lattice of annotations $\mathcal{A}$



Fun facts about $\boxdot^{\cdot}$

- Noting $\bot = (0,0)$, $\boxdot^{\bot}$ is equivalent to the data of a relation.
- Noting $\top = (4,4)$, the definitions of $\boxdot^{\top}$ and $\boxdot^{(4,4)}$ coincide.
- $\boxdot^{(4,0)}\ A\ B$ is the same as a function $A \longrightarrow B$
- $\boxdot^{(0,4)}\ A\ B$ is the same as a function $B \longrightarrow A$
- $\boxdot^{(4,2_a)}\ A\ B$ is the same as a split epi $A \twoheadrightarrow B$
- $\boxdot^{(4,2_b)}\ A\ B$ is the same as a split mono $A \rightarrowtail B$

The elements $p_{\square}^{\alpha,\beta}$ of $\boxdot^{\beta} \square \square$

Let

$$\mathcal{D}_{\square} = \{(\alpha, \beta) \in \mathcal{A}^2 \mid \alpha = \top \vee \beta \in \{0, 1, 2_a\}^2\}$$

For all $(\alpha, \beta) \in \mathcal{D}_{\square}$ we can define $p_{\square}^{\alpha,\beta}$ such that

$$\vdash_u p_{\square}^{\alpha,\beta} \ : \ \boxdot^{\beta} \square \ \square \quad \text{and} \quad \mathsf{rel}(p_{\square}^{\alpha,\beta}) \equiv \boxdot^{\alpha}$$

The elements $p_{\square}^{\alpha,\beta}$ of $\boxdot^{\beta} \square \square$

Let

$$\mathcal{D}_{\square} = \{(\alpha, \beta) \in \mathcal{A}^2 \mid \alpha = \top \vee \beta \in \{0, 1, 2_a\}^2\}$$

For all $(\alpha, \beta) \in \mathcal{D}_{\square}$ we can define $p_{\square}^{\alpha,\beta}$ such that

$$\vdash_u p_{\square}^{\alpha,\beta} \; : \; \boxdot^{\beta} \square \square \quad \text{and} \quad \mathsf{rel}(p_{\square}^{\alpha,\beta}) \equiv \boxdot^{\alpha}$$

$\boxdot^{\beta} \square \square$ may have several inhabitants
A translation must explain which one to target.
We need to annotate $\square$ everywhere!

4

Trocq

Annotating

$$M, N, A, B \in \mathcal{T}_{CC_\omega^+} ::= \Box_i^\alpha \mid x \mid M\ N \mid \lambda x : A.\ M \mid \Pi x : A.\ B$$

$$\frac{\Gamma \vdash_+ M : A \qquad \Gamma \vdash_+ A \preccurlyeq B}{\Gamma \vdash_+ M : B}\ (\textsc{Conv}^+)$$

$$\frac{(\alpha, \beta) \in \mathcal{D}_\Box}{\Gamma \vdash_+ \Box_i^\alpha : \Box_{i+1}^\beta}\ (\textsc{Sort}^+) \qquad \frac{(x, A) \in \Gamma \qquad \Gamma \vdash_+}{\Gamma \vdash_+ x : A}\ (\textsc{Var}^+) \qquad \frac{\Gamma \vdash_+ A : \Box_i \qquad x \notin \mathrm{Var}(\Gamma)}{\Gamma,\ x : A \vdash_+}\ (\textsc{Context}^+)$$

$$\frac{\Gamma \vdash_+ M : \Pi x : A.\ B \qquad \Gamma \vdash_+ N : A}{\Gamma \vdash_+ M\ N : B[x := N]}\ (\textsc{App}^+) \qquad \frac{\Gamma, x : A \vdash_+ M : B}{\Gamma \vdash_+ \lambda x : A.\ M : \Pi x : A.\ B}\ (\textsc{Lam}^+)$$

$$\frac{\Gamma \vdash_+ A : \Box_i^\alpha \qquad \Gamma \vdash_+ B : \Box_i^\beta \qquad \mathcal{D}_\rightarrow(\gamma) = (\alpha, \beta)}{\Gamma \vdash_+ A \rightarrow B : \Box_i^\gamma}\ (\textsc{Arrow}^+) \qquad \frac{\Gamma \vdash_+ A : \Box_i^\alpha \qquad \Gamma, x : A \vdash_+ B : \Box_i^\beta \qquad \mathcal{D}_\Pi(\gamma) = (\alpha, \beta)}{\Gamma \vdash_+ \Pi x : A.\ B : \Box_i^\gamma}\ (\textsc{Pi}^+)$$

Subtyping

$$\frac{\Gamma \vdash_{+} A : K \qquad \Gamma \vdash_{+} B : K \qquad A \equiv B}{\Gamma \vdash_{+} A \preccurlyeq B}\ (\textsc{SubConv})$$

$$\frac{\alpha \geq \beta \qquad i \leq j}{\Gamma \vdash_{+} \Box_i^{\alpha} \preccurlyeq \Box_j^{\beta}}\ (\textsc{SubSort})$$

$$\frac{\Gamma \vdash_{+} M'\ N : K \qquad \Gamma \vdash_{+} M \preccurlyeq M'}{\Gamma \vdash_{+} M\ N \preccurlyeq M'\ N}\ (\textsc{SubApp})$$

$$\frac{\Gamma, x : A \vdash_{+} M \preccurlyeq M'}{\Gamma \vdash_{+} \lambda x : A.\, M \preccurlyeq \lambda x : A.\, M'}\ (\textsc{SubLam})$$

$$\frac{\Gamma \vdash_{+} \Pi x : A.\, B : \Box_i \qquad \Gamma \vdash_{+} A' \preccurlyeq A \qquad \Gamma, x : A' \vdash_{+} B \preccurlyeq B'}{\Gamma \vdash_{+} \Pi x : A.\, B \preccurlyeq \Pi x : A'.\, B'}\ (\textsc{SubPi})$$

$$K ::= \Box_i \mid \Pi x : A.\, K$$

Calculus for Trocq

$$\frac{(\alpha,\beta) \in \mathcal{D}_{\square}}{\Delta \vdash_t \square_i^{\alpha} \ @\ \square_{i+1}^{\beta} \sim \square_i^{\alpha} \because p_{\square_i}^{\alpha,\beta}} \ (\textsc{TrocqSort}) \qquad \frac{(x, A, x', x_R) \in \Delta \quad \ldots}{\Delta \vdash_t x \ @\ A \sim x' \because x_R} \ (\textsc{TrocqVar})$$

$$\frac{\Delta \vdash_t M \ @\ \Pi x : A.\, B \sim M' \because M_R \qquad \Delta \vdash_t N \ @\ A \sim N' \because N_R}{\Delta \vdash_t M\ N \ @\ B[x := N] \sim M'\ N' \because M_R\ N\ N'\ N_R} \ (\textsc{TrocqApp})$$

$$\frac{\Delta \vdash_t A \ @\ \square_i^{\alpha} \sim A' \because A_R \qquad \Delta, x \ @\ A \sim x' \because x_R \vdash_t M \ @\ B \sim M' \because M_R}{\Delta \vdash_t \lambda x : A.\, M \ @\ \Pi x : A.\, B \sim \lambda x' : A'.\, M' \because \lambda x\, x'\, x_R.\, M_R} \ (\textsc{TrocqLam})$$

$$\frac{\Delta \vdash_t A \ @\ \square_i^{\alpha} \sim A' \because A_R \qquad \Delta \vdash_t B \ @\ \square_i^{\beta} \sim B' \because B_R \qquad (\alpha,\beta) = \mathcal{D}_{\rightarrow}(\delta)}{\Delta \vdash_t A \rightarrow B \ @\ \square_i^{\delta} \sim A' \rightarrow B' \because p_{\rightarrow}^{\delta}\ A_R\ B_R} \ (\textsc{TrocqArrow})$$

$$\frac{\Delta \vdash_t A \ @\ \square_i^{\alpha} \sim A' \because A_R \qquad \Delta, x \ @\ A \sim x' \because x_R \vdash_t B \ @\ \square_i^{\beta} \sim B' \because B_R \qquad (\alpha,\beta) = \mathcal{D}_{\Pi}(\delta)}{\Delta \vdash_t \Pi x : A.\, B \ @\ \square_i^{\delta} \sim \Pi x' : A'.\, B' \because p_{\Pi}^{\delta}\ A_R\ B_R} \ (\textsc{TrocqPi})$$

$$\frac{\Delta \vdash_t M \ @\ A \sim M' \because M_R \qquad \gamma(\Delta) \vdash_{+} A \preccurlyeq B}{\Delta \vdash_t M \ @\ B \sim M' \because \Downarrow_B^A M_R} \ (\textsc{TrocqConv})$$

Abstraction Theorem for Trocq

We have:

$$\frac{\gamma(\Delta) \vdash_+ \qquad \gamma(\Delta) \vdash_+ M : A \qquad \Delta \vdash_t M \mathbin{@} A \sim M' \because M_R \qquad \Delta \vdash_t A \mathbin{@} \square_i^\alpha \sim A' \because A_R}{\gamma(\Delta) \vdash_+ M' : A' \qquad \text{and} \qquad \gamma(\Delta) \vdash_+ M_R : \mathrm{rel}(A_R)\ M\ M'}$$

Remark A:

$$\frac{\gamma(\Delta) \vdash_+ A : \square^\alpha \qquad \Delta \vdash_t A \mathbin{@} \square^\alpha \sim A' \because A_R}{\gamma(\Delta) \vdash_+ A_R : \boxdot^\alpha\ A\ A'}.$$

Remark B:

$$\vdash_+ p_\square^{\alpha,\beta} : \boxdot^\beta\ \square^\alpha\ \square^\alpha$$

5

Extra material

The elements p_{Π}^{δ} of $\boxdot^{\delta}\ (\Pi A.B)\ (\Pi A'.B')$

We need to identify the triples $(\alpha, \beta, \delta) \in \mathcal{A}^3$ for which it is possible to construct a term p_{Π}^{δ} such that:

$$\frac{\delta \vdash A_R\ :\ \boxdot^{\alpha}\ A\ A' \qquad \delta,\ x : A,\ x' : A',\ x_R : A_R\ x\ x' \vdash B_R\ :\ \boxdot^{\beta}\ B\ B'}{\delta\ \vdash\ p_{\Pi}^{\delta}\ A_R\ B_R\ :\ \boxdot^{\delta}\ (\Pi x : A.\, B)\ (\Pi x' : A'.\, B')} \quad \text{and}$$

$$\mathsf{rel}(p_{\Pi}^{\delta}\ A_R\ B_R) \equiv \lambda f.\lambda f'.\Pi(x\ :\ A)(x'\ :\ A')(x_R : \mathsf{rel}(A_R)\ x\ x').$$
$$\mathsf{rel}(B_R)\ (f\ x)\ (f\ x')$$

The elements p_{Π}^{δ} of $\boxdot^{\delta}\ (\Pi A.B)\ (\Pi A'.B')$

We need to identify the triples $(\alpha, \beta, \delta) \in \mathcal{A}^3$ for which it is possible to construct a term p_{Π}^{δ} such that:

$$\frac{\delta \vdash A_R\ :\ \boxdot^{\alpha}\ A\ A' \qquad \delta,\ x : A,\ x' : A',\ x_R : A_R\ x\ x' \vdash B_R\ :\ \boxdot^{\beta}\ B\ B'}{\delta\ \vdash\ p_{\Pi}^{\delta}\ A_R\ B_R\ :\ \boxdot^{\delta}\ (\Pi x : A.\ B)\ (\Pi x' : A'.\ B')} \quad \text{and}$$

$$\mathrm{rel}(p_{\Pi}^{\delta}\ A_R\ B_R) \equiv \lambda f.\lambda f'.\Pi(x\ :\ A)(x'\ :\ A')(x_R : \mathrm{rel}(A_R)\ x\ x').$$
$$\mathrm{rel}(B_R)\ (f\ x)\ (f\ x')$$

We prove that p_{Π}^{δ} exists for all $(\alpha, \beta) \in \mathcal{D}_{\pi}(\delta)$, where ...

Definition of $\mathcal{D}_{\Pi}(\delta)$

For any $\delta \in \mathcal{A}$:

$$\mathcal{D}_{\Pi}(\delta) = \mathcal{D}_{\Pi}(\delta_1, 0) \vee \mathcal{D}_{\Pi}(\delta_2, 0)^{-1}$$

Where for all $\alpha, \beta \in \mathcal{A}$

$$\begin{aligned}
(\alpha, \beta)^{-1} &\triangleq (\alpha^{-1}, \beta^{-1}) \\
\alpha^{-1} &\triangleq (\alpha_2, \alpha_1) \\
(\alpha, \beta) \vee (\alpha', \beta') &\triangleq (\alpha \vee \alpha', \beta \vee \beta') \\
\alpha \vee \beta &\triangleq (\alpha_1 \vee \beta_1, \alpha_2 \vee \beta_2)
\end{aligned}$$

Definition of $\mathcal{D}_{\Pi}(\delta)$

For any $\delta \in \mathcal{A}$:

$$\mathcal{D}_{\Pi}(\delta) = \mathcal{D}_{\Pi}(\delta_1, 0) \vee \mathcal{D}_{\Pi}(\delta_2, 0)^{-1}$$

Where for all $\alpha, \beta \in \mathcal{A}$

$$(\alpha, \beta)^{-1} \triangleq (\alpha^{-1}, \beta^{-1})$$
$$\alpha^{-1} \triangleq (\alpha_2, \alpha_1)$$
$$(\alpha, \beta) \vee (\alpha', \beta') \triangleq (\alpha \vee \alpha', \beta \vee \beta')$$
$$\alpha \vee \beta \triangleq (\alpha_1 \vee \beta_1, \alpha_2 \vee \beta_2)$$

Thus, it suffices to define $\mathcal{D}_{\Pi}(m, 0)$ for all $m \in \{0, 1, 2_a, 2_b, 3, 4\}$
The same holds for $\mathcal{D}_{\rightarrow}(\delta)$.

Definition of $\mathcal{D}_{\Pi}(m,0)$ and $\mathcal{D}_{\rightarrow}(m,0)$

m	$\mathcal{D}_{\Pi}(m,0)_1$	$\mathcal{D}_{\Pi}(m,0)_2$
0	$(0,0)$	$(0,0)$
1	$(0,2_a)$	$(1,0)$
2_a	$(0,4)$	$(2_a,0)$
2_b	$(0,2_a)$	$(2_b,0)$
3	$(0,4)$	$(3,0)$
4	$(0,4)$	$(4,0)$

m	$\mathcal{D}_{\rightarrow}(m,0)_1$	$\mathcal{D}_{\rightarrow}(m,0)_2$
0	$(0,0)$	$(0,0)$
1	$(0,1)$	$(1,0)$
2_a	$(0,2_b)$	$(2_a,0)$
2_b	$(0,2_a)$	$(2_b,0)$
3	$(0,3)$	$(3,0)$
4	$(0,4)$	$(4,0)$

6

Conclusion

Comparison

	Magaud [10]	Setoid rw. [12]	CoqEAL [5]	Transfer [6–9]	ZH [15]	TTS [14]	ACMZ [1]	Trackt [3]	Trocq
Heterogeneous relations	✓	✗	✓	✓	✓	✓	✓	✓	✓
Internal	✗	✓	✓	✓	✓	✓	✓	✓	✓
No anticipation	✓	✓	✓	✓	✓	✓	✗	✓	✓
Substitution under $\forall$	✓	✓	✗	✓	✓	✓	✓	✓	✓
Substitution in dep. types	✓	✗	✗	✗	✗	✓	✓	✗	✓
No univalence for ?	✓	✓	✓	✓	✓	✗	✗	✓	✓
Preorder relations	✗	✓	?	?	?	✗	?	?	✏
Subrelations	✗	✓	✗	✗	✗	✗	✗	✗	✏
QERs	✗	✏	➡	➡	➡	✗	✓	✗	➡
Subtyping relations	✗	✗	➡	➡	➡	✗	✗	➡	➡
System	Coq	Coq	Coq	Isab	Coq	Coq	(Cub	Coq	Coq

Bring home

- Change representation **without univalence** in some cases.
- Change representation **with partial isos** in some cases.

In our current version,

- univalence is required if and only if there is some $\square^\alpha$ such that $\alpha \geq (2_b, 0)$ or $\alpha \geq (0, 2_b)$ occurs in the derivation.
- reducing a goal G to an hypothesis H corresponds to finding an element $\boxdot^{(0,1)}\ G\ H$ (i.e. an arrow $H \rightarrow G$). If the body of G and H have the right variance, we might keep the invariant that nothing more than the partial isos $\square^{(4,2_a)}$, $\square^{(4,2_b)}$,$\square^{(2_a,4)}$ or $\square^{(2_b,4)}$ are required on given types.

In the future (with a bit more work), we may unify

- CoqEAL
- Univalent paramericity
- Generalized (Setoid) rewriting

7

Bibliography

bibliography I

[1] Carlo Angiuli et al. "Internalizing representation independence with univalence". In: *Proc. ACM Program. Lang.* 5.POPL (2021), pp. 1–30. DOI: 10.1145/3434293. URL: https://doi.org/10.1145/3434293.

[2] Jean-Philippe Bernardy and Marc Lasson. "Realizability and Parametricity in Pure Type Systems". In: *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*. Ed. by Martin Hofmann. Vol. 6604. Lecture Notes in Computer Science. Springer, 2011, pp. 108–122. DOI: 10.1007/978-3-642-19805-2_8. URL: https://doi.org/10.1007/978-3-642-19805-2_8.

[3] Valentin Blot et al. "Compositional Pre-processing for Automated Reasoning in Dependent Type Theory". In: *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2023, Boston, MA, USA, January 16-17, 2023*. Ed. by Robbert Krebbers et al. ACM, 2023, pp. 63–77. DOI: 10.1145/3573105.3575676. URL: https://doi.org/10.1145/3573105.3575676.

bibliography II

[4] Cyril Cohen, Enzo Crance, and Assia Mahboubi. "Artifact Report: Trocq: Proof Transfer for Free, With or Without Univalence". In: *ESOP 2024 - 33rd European Symposium on Programming*. Vol. LNCS-14576. Programming Languages and Systems 33rd European Symposium on Programming, ESOP 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6–11, 2024, Proceedings, Part I. Luxembourg, Luxembourg: Springer Nature Switzerland, Apr. 2024, pp. 269–274. DOI: 10.1007/978-3-031-57262-3_11. URL: https://inria.hal.science/hal-04623207.

[5] Cyril Cohen, Maxime Dénès, and Anders Mörtberg. "Refinements for Free!" In: *Certified Programs and Proofs - Third International Conference, CPP 2013, Melbourne, VIC, Australia, December 11-13, 2013, Proceedings*. Ed. by Georges Gonthier and Michael Norrish. Vol. 8307. Lecture Notes in Computer Science. Springer, 2013, pp. 147–162. DOI: 10.1007/978-3-319-03545-1_10. URL: https://doi.org/10.1007/978-3-319-03545-1_10.

[6] Florian Haftmann et al. "Data Refinement in Isabelle/HOL". In: *Interactive Theorem Proving*. Ed. by Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 100–115. ISBN: 978-3-642-39634-2.

bibliography III

[7] Brian Huffman and Ondřej Kunčar. "Lifting and Transfer: A Modular Design for Quotients in Isabelle/HOL". In: *Certified Programs and Proofs*. Ed. by Georges Gonthier and Michael Norrish. Cham: Springer International Publishing, 2013, pp. 131–146. ISBN: 978-3-319-03545-1.

[8] Peter Lammich. "Automatic Data Refinement". In: *Interactive Theorem Proving*. Ed. by Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 84–99. ISBN: 978-3-642-39634-2.

[9] Peter Lammich and Andreas Lochbihler. "Automatic Refinement to Efficient Data Structures: A Comparison of Two Approaches". In: *J. Autom. Reason.* 63.1 (2019), pp. 53–94. DOI: 10.1007/s10817-018-9461-9. URL: https://doi.org/10.1007/s10817-018-9461-9.

[10] Nicolas Magaud. "Changing Data Representation within the Coq System". In: *TPHOLs'2003*. Vol. 2758. © Springer-Verlag. LNCS, Springer-Verlag, 2003. URL: http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=2758&spage=87.

bibliography IV

[11] The Mathematical Components Team. *Mathematical Components Library*. https://github.com/math-comp/math-comp. Last stable version: 2.1 (2023). 2007.

[12] Matthieu Sozeau. "A New Look at Generalized Rewriting in Type Theory". In: *J. Formaliz. Reason.* 2.1 (2009), pp. 41–62. DOI: 10.6092/issn.1972-5787/1574. URL: https://doi.org/10.6092/issn.1972-5787/1574.

[13] Nicolas Tabareau, Éric Tanter, and Matthieu Sozeau. "Equivalences for free: univalent parametricity for effective transport". In: *Proceedings of the ACM on Programming Languages* 2.ICFP (2018), pp. 1–29.

[14] Nicolas Tabareau, Éric Tanter, and Matthieu Sozeau. "The marriage of univalence and parametricity". In: *Journal of the ACM (JACM)* 68.1 (2021), pp. 1–44.

[15] Théo Zimmermann and Hugo Herbelin. "Automatic and Transparent Transfer of Theorems along Isomorphisms in the Coq Proof Assistant". In: *Conference on Intelligent Computer Mathematics*. Washington, D.C., United States, 2015. URL: https://hal.science/hal-01152588.