Trocq parametricity translations for inductive types

Tomás Vallejos Parada
Team Gallinette, Inria, LS2N
2026-01-17

talk soon @ tvallejos.cl

Motivation

Proof transfer examples

Inductive nat := O | S : nat → nat.

Inductive Z := Pz : nat → Z | Nz : nat → Z.

Inductive binnat := Obin | NPos : positive → binnat.

You have:

(x + y = y + x) %Z

(foldl + [2; 3; 5] = 10) %list nat

ind_nat nat

You need:

(n + m = m + n) %N

(foldl + [2; 3; 5] = 10) %list Z

ind_nat %binnat

One would like to discharge them by pressing a button

Proof transfer frameworks via parametricity translations

- **"Raw" parametricity translation** (J.-P. Bernardy, P. Jansson, and R. Paterson)
- **Univalent parametricity translation** (N. Tabareau, É. Tanter, and M. Sozeau)
- **CoqEAL** (M. Dénès, A. Mörtberg, and V. Siles)

Latest: **Trocq** (C. Cohen, E. Crance, and A. Mahboubi)
Generalizes previous parametricity translations

Proof transfer requirements

In	User provides	Trocq translates
1. 5%N into 5%Z	a *relation* between A and B	type A into type B
2. list nat into list Z	a *proof* the container preserves the relation	through a *type parametric* container
3. vec nat into vec Z	a relation between A and A	type A into type A

For inductive types with *type parameters* and *recursion*, *no indices*
This work generates 2. and 3. **automatically**

Context

Trocq's hierarchy

$\mathrm{Param}_{n,k} AB := \Sigma R : A \to B \to \square . (\mathrm{M}_n R) \times (\mathrm{M}_k R^{\mathrm{op}})$

M_4 $\Pi a, b . (\mathrm{mR}(a,b)) \circ (\mathrm{Rm}(a,b)) \doteqdot \mathrm{id}$

M_3 $\mathrm{mR} \wedge \mathrm{Rm}$

M_{2_a} $\mathrm{mR} : \Pi a, b . m(a) = b \to R(a,b)$

M_{2_b} $\mathrm{Rm} : \Pi a, b . R(a,b) \to m(a) = b$

M_1 $\mathrm{m} : \mathrm{A} \longrightarrow \mathrm{B}$

M_0 $\mathrm{R} : \mathrm{A} \to \mathrm{B} \to \square$

Pros/cons of supporting weaker relations

Pros

- Supports weaker relations
- Enables avoiding the univalence axiom when possible

Cons

- Preserving relation across a container requires to prove

$$\Pi(AB : \square)\big(\mathrm{R} : \mathrm{Param}_{n,k} AB\big), \mathrm{Param}_{n,k}(\mathrm{list}\ A)(\mathrm{list}\ B)$$

- Not translating a type requires to prove

$$\mathrm{Param}_{n,k} AA$$

Derivation

A parametricity translation for inductive types

- Bottom-up derivation of the lattice
- Careful implementation of intermediate data requirements so coherence can be proven
- Strategy: postponing the elimination of identities

Implementation in ELPI built upon param2 (C. Cohen) and derive (E. Tassi) plugins, currently **automatically** generating:

1. $\mathrm{Param}_{n,0}\,AB$ preservation for parametric containers
2. $\mathrm{Param}_{n,0}\,AA$

For inductive types with type parameters and recursion, no indices
Find it at https://github.com/Tvallejos/trocq/tree/rocqpl-26/std/algo

An example derivation: nat

```
Inductive nR : nat → nat → Type :=
| OR : nR O O
| SR : ∀ n m, nR n m → nR (S n) (S m).
```

```
Fixpoint nM n := match n with
| O => O
| S n' => S (nM n') end.
```

```
generate projS : nat → nat and Sproj : ∀ n1 n2, S n1 = S n2 → projS n1 (S n1) = projS n1 (S n2)
```

```
Definition nmR n1 n2 :
  natM n1 = n2 → nR n1 n2 := by induction on
n1 and n2. On the diagonal, use nR Kth
constructor, Sproj and IH. Discriminate
outside it.
```

```
Definition nRm n1 n2 :
  nR n1 n2 → natM n1 = n2 :=
by induction on nR, f_equal and IH.
```

```
generate SprojSK: constructor cancels with the projection
```

```
Definition nat_coh n1 n2 (nR : nR n1 n2) :
   nmR _ _ (nRm n1 n2 nR) = nR := by induction on nR, rewrite under SprojSK and IH
```

Another example generation: list

```
Inductive listR (A B : Type) (R : A → B →
Type) : list A → list B → Type :=
| nilR : listR A B R (listR A B R nil nil)
| consR : ∀ a b, R a b
  → ∀ la lb, listR A B R la lb
    → (listR A B R (cons a la) (cons b lb)).
```

```
Fixpoint listM (A B : Type)
(R: Param10.Rel A B) l : list B :=
  match l with
| nil => nil
| cons a l' => cons (map R a) (listM _ _ R l')
end.
```

```
Definition list_mR (A B : Type) (R :
Param2a0.Rel A B) : ∀ la lb, listM A B R la = lb
→ List_R A B R la lb.
```

```
Definition List_Rm (A B : Type) (R :
Param2b0.Rel A B) :
∀ la lb,  List_R A B R la lb → List_mymap A B
R la = lb.
```

```
Definition List_mRRmK (A B : Type) (R : Param40.Rel A B) la lb (IR : List_R A B R la lb),
List_mR A B R la lb (List_Rm A B R la lb IR) = IR.
```

Trocq parametricity translations for inductive types

Tomás Vallejos Parada
Team Gallinette, Inria, LS2N
2026-01-17

talk soon @ tvallejos.cl

Bibliography

[1] J.-P. Bernardy, P. Jansson, and R. Paterson, "Parametricity and dependent types," in *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming, ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010*, P. Hudak and S. Weirich, Eds., ACM, 2010, pp. 345–356. doi: 10.1145/1863543.1863592.

[2] N. Tabareau, É. Tanter, and M. Sozeau, "The Marriage of Univalence and Parametricity," *J. ACM*, vol. 68, no. 1, pp. 5:1–5:44, 2021, doi: 10.1145/3429979.

[3] M. Dénès, A. Mörtberg, and V. Siles, "A Refinement-Based Approach to Computational Algebra in Coq," in *Interactive Theorem Proving - Third International Conference, ITP 2012, Princeton, NJ, USA, August 13-15, 2012. Proceedings*, L. Beringer and A. P. Felty, Eds., in Lecture Notes in Computer Science, vol. 7406. Springer, 2012, pp. 83–98. doi: 10.1007/978-3-642-32347-8_7.

[4] C. Cohen, E. Crance, and A. Mahboubi, "Trocq: Proof Transfer for Free, Beyond Equivalence and Univalence," *ACM Trans. Program. Lang. Syst.*, vol. 47, no. 3, Sept. 2025, doi: 10.1145/3737283.

[5] E. Tassi, "Deriving Proved Equality Tests in Coq-Elpi: Stronger Induction Principles for Containers in Coq," in *10th International Conference on Interactive Theorem Proving, ITP 2019, Portland, OR, USA, September 9-12, 2019*, J. Harrison, J. O'Leary, and A. Tolmach, Eds., in LIPIcs, vol. 141. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 29:1–29:18. doi: 10.4230/LIPICS.ITP.2019.29.