From parametricity to identity types

Towards Higher Observational Type Theory

Thorsten Altenkirch[1], Ambrus Kaposi[2]*, Michael Shulman[3]†, and Elif Üsküplü[4]

[1] University of Nottingham, Nottingham, United Kingdom, `txa@cs.nott.ac.uk`
[2] Eötvös Loránd University, Budapest, Hungary, `akaposi@inf.elte.hu`
[3] University of San Diego, San Diego, USA, `shulman@sandiego.edu`
[4] Indiana University, Bloomington, IN, USA, `euskuplu@iu.edu`

We have introduced Parametric Observational Type Theory, which is a theory with internal parametricity which does not use an interval. It satisfies canonicity [2]; see [4] for an indexed version. This theory is the basis for the Narya system that we have implemented.[1] We are planning to use this type theory as a stepping stone to Higher Observational Type Theory (HOTT), a computational version of homotopy type theory that likewise does not use an interval. POTT introduces a relation for every type (called the *bridge type* [6]). The bridge type (which we will write as `Br` in Narya) can be seen as a baby version of equality: it is reflexive and a congruence, but it is not symmetric or transitive. Bridge can be heterogeneous, so given `A : B → Type`, we have `Br A : (b0 b1 : B) → Br B b0 b1 → A b0 → A b1 → Type`. Bridge of Π types says that bridge-identical inputs are mapped to bridge-identical outputs. The bridge of an inductive type is an inductive family with the same number of constructors, as usual in parametricity translations [5], and symmetrically for coinductive types. Bridge of the universe corresponds to relation space; in particular, if `A2 : Br Type A0 A1`, and `a0 : A0` and `a1 : A1`, then `A2 a0 a1 : Type`. Narya also supports Martin-Löf's inductively defined equality type family (which does not satisfy function extensionality or univalence).

Following the ideas of [8, 3] we want to construct a universe of fibrant types inside POTT, such that bridges of fibrant types correspond to equivalences (i.e. we have univalence), and using the observation that singletons are contractible we can derive the usual J-elimination rule (with weak β-equality). This is similar to the approach to cubical type theory in [7].

Our goal is to define a family `isFib : Type → Type` in Narya; then the universe of fibrant types will be `Ufib = Σ Type isFib`. HOTT will live in `Ufib`, and Narya can be seen as an outer layer for a two-level type theory [1]. We define `isFib` as a 'higher coinductive type'. Higher coinductive types are the dual of higher inductive types, and come with destructors on equalities rather than equality-constructors. In POTT and Narya, these are actually bridge-destructors. The Narya-definition of `isFib` is the following; note the 'self variable' `x` through which previous fields are accessed.

```
def isFib (A : Type) : Type := codata [
  | x .trr.p   : A.0 → A.1
  | x .trl.p   : A.1 → A.0
  | x .liftr.p : (a0 : A.0) → A.2 a0 (x.2 .trr a0)
  | x .liftl.p : (a1 : A.1) → A.2 (x.2 .trl a1) a1
  | x .id.p    : (a0 : A.0) (a1 : A.1) → isFib (A.2 a0 a1) ]
```

Destructors in Narya start with a `.`, and higher destructors end with `.p` (our chosen name for the parametricity direction) when being defined, but not when they are used. For example, given

*Ambrus Kaposi was supported by the the HOTT ERC Grant 101170308.

†Michael Shulman was supported by the United States Air Force Office of Scientific Research under award number FA9550-21-1-0009.

[1] See `https://github.com/gwaithimirdain/narya/tree/2cca3d7` and more specific links in the abstract.

```
A0 : Type,                x0 : isFib A0,
A1 : Type,                x1 : isFib A1,
A2 : Br Type A0 A1,       x2 : Br isFib A0 A1 A2 x0 x1,
```

we have `x2 .trr : A0 → A1`. Also, since a higher destructor must be applied to a *bridge in* the type being defined, the 'self variable' `x` and the parameter `A` must be replaced by such bridges when declaring the its type; the suffixes `.0`, `.1`, `.2` on the variables `A` and `x` serve to identify the three components of these arguments of the heterogeneous bridge types.

`isFib` expresses that given two bridge-identical types, we can transport between them in both directions (via `.trr` and `.trl`), these transports preserve the bridge-relation `A.2` (via `.liftr` and `.liftl`), and finally, the bridge-relation itself is fibrant (via `.id`; this last destructor makes `isFib` into a coinductive type rather than just a record).

Higher coinductive types are terminal coalgebras; in Narya we can construct their elements via co-pattern matching. Similarly, the bridge of a higher coinductive type is another one. This allows us to prove that a fibrant one-to-one correspondence between fibrant types, i.e. an equivalence, gives rise to a bridge of those types; thus we have univalence. Univalence is *definitional*, which means that if we turn a function which is an equivalence into an element of the `Br` type, and transport along it, we get back the function definitionally (unlike in cubical type theories such as Cubical Agda).

We can also prove using co-pattern matching and corecursion that basic type formers preserve fibrancy. Fibrancy for the empty type is trivial using an empty match. For the unit type, we show that bridge of unit is equivalent to unit, with the notion of equivalence using the Martin-Löf equality type. Knowing that fibrancy is preserved by such equivalences, we prove fibrancy of unit via corecursion. For natural numbers, we first show that bridge is equivalent to the recursively defined equality (which only uses empty and unit, already shown to be fibrant). Then fibrancy of `Nat` is a simple corecursion. Fibrancy of Σ-types, Π-types, W-types, and M-types can be shown with analogous methods, although we need to assume as axioms that the Martin-Löf equality type satisfies function extensionality (for W-types) and coinductive extensionality (for M-types). The more serious challenge of showing that `Ufib` itself is fibrant is work in progress.

We shouldn't forget the elephant in the room, which is a semantic justification of higher coinductive types in POTT and an extension of the proof of canonicity to their presence. We also aim to extend the proof of canonicity to a proof of normalization, based on the algorithm implemented by Narya. Work is also in progress to directly implement a mode for HOTT in Narya, in which all types will automatically be fibrant.

The above results were formalised in Narya by the third author. Note that Narya currently does not have termination or universe-level checking; thus while our corecursive proofs of fibrancy appear semantically productive, they have not been syntactically checked to be so.

References

[1] Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. Extending homotopy type theory with strict equality. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. URL: https://doi.org/10.4230/LIPIcs.CSL.2016.21, doi:10.4230/LIPICS.CSL.2016.21.

[2] Thorsten Altenkirch, Yorgo Chamoun, Ambrus Kaposi, and Michael Shulman. Internal parametricity, without an interval. *Proc. ACM Program. Lang.*, 8(POPL):2340–2369, 2024. doi:10.1145/3632920.

[3] Thorsten Altenkirch and Ambrus Kaposi. Towards a cubical type theory without an interval. In *21st International Conference on Types for Proofs and Programs (TYPES 2015)(2018)*, pages 3–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[4] Thorsten Altenkirch, Ambrus Kaposi, Michael Shulman, and Elif Usküplü. Internal relational parametricity, without an interval. In *30th International Conference on Types for Proofs and Programs TYPES 2024–Abstracts*, page 56, 2024.

[5] Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. Proofs for free — parametricity for dependent types. *J. Funct. Program.*, 22(2):107–152, 2012. doi:10.1017/S0956796812000056.

[6] Evan Cavallo. *Higher Inductive Types and Internal Parametricity for Cubical Type Theory.* PhD thesis, Carnegie Mellon University, USA, 2021.

[7] Ian Orton and Andrew M. Pitts. Axioms for modelling cubical type theory in a topos. *Logical Methods in Computer Science*, 14, December 2018. arXiv:1712.04864. doi:10.23638/LMCS-14(4:23)2018.

[8] Michael Shulman. Towards third-generation HoTT, part 1, 2021. YouTube lecture. URL: https://www.youtube.com/watch?v=FrxkVzItMzA.