Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types

Andrea Vezzosi[1], Anders Mörtberg[23], and Andreas Abel[4]

[1] IT University Copenhagen, Copenhagen, Denmark
[2] Stockholm University, Stockholm, Sweden
[3] Carnegie Mellon University, Pittsburgh, USA
[4] Chalmers and Gothenburg University, Gothenburg, Sweden

A core idea in programming and mathematics is *abstraction*: the exact details of how an object is represented should not affect its abstract properties. The principle of univalence captures this by extending the equality on the universe of types to incorporate equivalent types. This provides a form of abstraction, or invariance up to equivalence, in the sense that equivalent types will share the same structures and properties. The fact that equality is *proof relevant* in dependent type theory is the key to enabling this. The data of an equality proof can store the equivalence and transporting along this equality should then apply the function underlying the equivalence. In particular, this allows programs and properties to be transported between equivalent types, hereby increasing modularity and decreasing code duplication. A concrete example are the equivalent representations of natural numbers in unary and binary format. In a univalent system it is possible develop theory about natural numbers using the unary representation, but compute using the binary representation, and as the two representations are equivalent they share the same properties.

The principle of univalence is the major new addition in Homotopy Type Theory and Univalent Foundations (HoTT/UF) [The Univalent Foundations Program, 2013]. However, these new type theoretic foundations add univalence as an *axiom* which disrupts the good constructive properties of type theory. In particular, if we transport addition on binary numbers to the unary representation we will not be able to compute with it as the system would not know how to reduce the univalence axiom. Cubical Type Theory (CTT) [Cohen et al., 2015a] addresses this problem by introducing a novel representation of equality proofs and thereby providing computational content to univalence. This makes it possible to constructively transport programs and proofs between equivalent types. This representation of equality proofs has many other useful consequences, in particular functional and propositional extensionality and the equivalence between bisimilarity and equality for coinductive types [Vezzosi, 2017].

Dependently typed functional languages such as `Agda`, `Coq`, `Idris`, and `Lean`, provide rich and expressive environments supporting both programming and proving within the same language. However, the extensionality principles mentioned above are not available out of the box and need to be assumed as axioms just as in HoTT/UF. Unsurprisingly, this suffers from the same drawbacks as it compromises the computational behavior of programs that use these axioms. It even makes subsequent proofs more complicated as equational properties do not hold by computation.

So far, CTT has been developed with the help of a prototype `Haskell` implementation called `cubicaltt` [Cohen et al., 2015b], but it has not been integrated into one of the main dependently typed functional languages. Recently, an effort was made, using `Coq`, to obtain effective transport for restricted uses of the univalence axiom [Tabareau et al., 2018], because, as the authors mention, *"it is not yet clear how to extend [proof assistants] to handle univalence internally"*.

We achieve this, and more, by making `Agda` into a cubical programming language with native support for univalence and higher inductives types (HITs). We call this extension `Cubical Agda` [2019] as it incorporates and extends CTT. In addition to providing a fully constructive univalence theorem, `Cubical Agda` extends the theory by allowing proofs of equality by copatterns, HITs as in Coquand et al. [2018] with nested pattern matching, and interval and partial pre-types. The extension of dependent (co)pattern matching [Cockx and Abel, 2018] to the equality type allows for convenient programming with HITs and univalence. We demonstrate this by the proof that the torus is equal to two circles in `Cubical Agda`.

```
data S¹ : Set where
  base : S¹
  loop : base ≡ base

data Torus : Set where
  point  : Torus
  line1  : point ≡ point
  line2  : point ≡ point
  square : PathP (λ i → line1 i ≡ line1 i) line2 line2

t2c : Torus → S¹ × S¹
t2c point        = (base   , base)
t2c (line1 i)    = (loop i , base)
t2c (line2 j)    = (base   , loop j)
t2c (square i j) = (loop i , loop j)

c2t : S¹ × S¹ → Torus
c2t (base , base)   = point
c2t (loop i , base) = line1 i
c2t (base , loop j) = line2 j
c2t (loop i , loop j) = square i j
```

The proof that t2c and c2t are inverses is just reflexivity in each of the four cases. We can then the isomorphism into an equality, using the implementation of univalence.

While this is a rather elementary result in topology it had a surprisingly non-trivial proof in HoTT because of the lack of definitional computation for higher constructors [Sojakova, 2016, Licata and Brunerie, 2015]. With the additional definitional computation rules and pattern-matching of `Cubical Agda` this proof is now almost entirely trivial.

References

Agda developers. *Agda 2.6.0 documentation*, 2019. http://agda.readthedocs.io/en/v2.6.0/.

J. Cockx and A. Abel. Elaborating dependent (co)pattern matching. *Proc. ACM Program. Lang.*, 2 (ICFP), 2018. http://doi.acm.org/10.1145/3236770.

C. Cohen, T. Coquand, S. Huber, and A. Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In *TYPES'15*, vol. 69 of *LIPIcs*. Dagstuhl, 2015a. https://doi.org/10.4230/LIPIcs.TYPES.2015.5.

C. Cohen, T. Coquand, S. Huber, and A. Mörtberg. Cubicaltt. https://github.com/mortberg/cubicaltt, 2015b.

T. Coquand, S. Huber, and A. Mörtberg. On higher inductive types in cubical type theory. In *LICS'18*. ACM, 2018. https://doi.org/10.1145/3209108.3209197.

D. R. Licata and G. Brunerie. A cubical approach to synthetic homotopy theory. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'15*. ACM, 2015.

K. Sojakova. The Equivalence of the Torus and the Product of Two Circles in Homotopy Type Theory. *ACM Transactions on Computational Logic*, 17(4), 2016.

N. Tabareau, E. Tanter, and M. Sozeau. Equivalences for free: Univalent parametricity for effective transport. *Proc. ACM Program. Lang.*, 2(ICFP), 2018. http://doi.acm.org/10.1145/3236787.

The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

A. Vezzosi. Streams for cubical type theory. http://saizan.github.io/streams-ctt.pdf, 2017.