Towards Computational UIP in Cubical Agda

Yee-Jian Tan, Andreas Nuyts, Dominique Devriese
KU Leuven, Belgium
{yee-jian.tan, andreas.nuyts, dominique.devriese}@kuleuven.be

Motivation Cubical Agda implements Cubical Type Theory, featuring quotient inductive types (QITs) and functional extensionality (funext) [Coh+17], but also an infinite hierarchy of equalities that is necessary to support univalence but may become unwieldy in formalisations. Fortunately, QITs and functional extensionality are both preserved even if the equality levels of Cubical Type Theory are truncated to only homotopical Sets (h-Sets) [Uni13]. Such a "h-Set Cubical Type Theory" would remove univalence (by removing the *Glue* type) but add an axiom called Uniqueness of Identity Proofs (UIP). Researchers and Cubical Agda users have expressed interest in such a theory [Dan19, Shu17]. Lacking a mature implementation, they resort to one of the following:

- Cubical Agda, explicitly passing around h-Set proofs (e.g. category theory in the Cubical Library [Agd25], dependent Lambek calculus [Sch+25], algebraic theories in Cubical Agda [Nuy24])
- Cubical Agda with postulated UIP (e.g. in Chen, Nordvall Forsberg and Tsai's TT-as-QIIRT [CNT26], more generally when formalising the syntax and models of type theory)
- Regular Agda with setoids, giving up general function extensionality, sometimes avoiding UIP (Agda Categories [HC21], (Bi)Sikkel [CND22, CND25])

In Cubical Agda, there are currently only two unsatisfying ways to achieve h-Set Cubical Type Theory: either give up on canonicity and postulate UIP, or manually prove UIP for *every* defined (small) type using the standard result that "type formers preserve h-levels" [Uni13]. The latter is, however, laborious work best suited for an automatic implementation by the proof assistant.

Contribution In this project, we implement two variants of Cubical Agda: Cubical-without-Glue (`--cubical=no-glue`) [Tan25], and our work-in-progress variant: Cubical-UIP (`--cubical=uip`). Cubical-without-Glue is a strict subvariant of both Cubical and Cubical-UIP. Cubical-UIP adds computational UIP, i.e. with computational behavior, similar to what is done for univalence in Cubical Agda. Technically, we use a more general but equivalent formulation called the square-filling property (SqFill, dependent form SqPFill), introduced later. In this paper, we report on computation rules of both SqFill and SqPFill for Π, Σ, Coproduct, and Path (equality) types. Our Cubical-UIP implementation currently implements SqFill for the abovementioned types as well as some Agda builtin types such as Bool, Nat, and List.

Related Work XTT [SAG22] is a Cubical Type Theory (without *Glue* types) with definitional UIP: two paths are *definitionally* equal if they have the same endpoints, while our computation rules provide only *propositional* equality. XTT seemed unattractive for implementation in Cubical Agda, as the type-checking algorithm would involve non-standard reduction steps relying on injectivity of type constructors w.r.t. paths, which is incompatible with at least excluded middle [CD18, Hur10]. Observational Type Theory [AMS07, PT22] and setoid models of Type Theory [Alt99, Alt+19, Hof95] add functional extensionality, propositional extensionality, and quotient types to intensional type theory, and achieve UIP definitionally via proof-irrelevant propositions. In OTT, equality *types* compute by recursion on type structure: for instance, equality of pairs reduces *definitionally* to pointwise equality of components; whereas in Cubical-UIP, the path type of pairs are only *isomorphic* to pairs of path types. Instead of types, equality *proofs* in Cubical-UIP are computed by recursion on type structure via SqFill, thus it is a modest modification of Cubical Agda which can be straightforwardly implemented by adding a single computing primitive to Cubical-without-Glue. Pujet et al. implemented CIC^{obs} in an extension of Rocq [PLT25].

Consistency The consistency of h-Set Cubical Type Theory can be justified by a set model or translation to XTT. We sketch such a set model here: types as sets, the interval type as the singleton set, and path types as subsingleton sets which are non-empty only when the endpoints are equal.

Cubical Agda without *Glue* To modularly implement Cubical-UIP, we first implement its strict subset Cubical-without-Glue (`--cubical=no-glue`) [Tan25]. When activated, Cubical-without-Glue prohibits *Glue* types in the current module and anything it imports, thus disabling univalence and allowing users to consistently postulate axioms such as UIP and K, forming the basis for implementing Cubical-UIP (`--cubical=uip`).

UIP via the SqFill property The UIP statement states that every type has the `isSet` property, i.e. for $x\ y : A$, any two proofs $p\ q : x \equiv y$ of their equality are equal: $p \equiv q$ (Figure 1). Pictorially, we can think of the points $x\ y : A$ as reflexive sides and extend the picture into a square. Then, the original UIP statement is

definitionally equivalent to requiring that any such hollow square (with reflexive sides) has a filling (Figure 2). Perhaps surprisingly, relaxing the requirement of reflexive sides still results in an equivalent property found in the Cubical Agda library [Agd25], which we call the "square-filling" property SqFill: that any square in A (four corners lu ld ru rd : A with sides $l : \mathrm{lu} \equiv \mathrm{ld}, r : \mathrm{ru} \equiv \mathrm{rd}, u : \mathrm{lu} \equiv \mathrm{ru}, d : \mathrm{ld} \equiv \mathrm{rd}$) has a filling (Figure 3). UIP follows from SqFill as a special case, but the proof for the converse is non-trivial [Agd25].

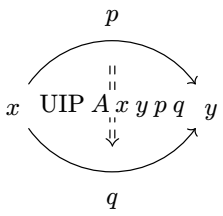

Figure 1: UIP

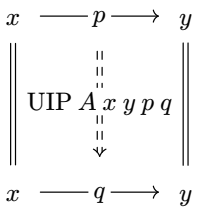

Figure 2: Reflexive sides

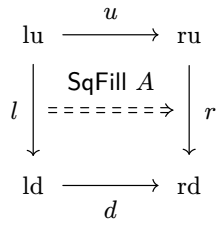

Figure 3: Square-filling

Computational UIP in Cubical Agda We implement UIP in terms of SqFill as an Agda primitive sqFill : $(A : \mathrm{Type}) \to \mathsf{SqFill}\ A$, which computes by recursing on the type A. The reduct for each type is given as a `BUILTIN` in an Agda file, and leverages the standard result that type formers (such as Π, Σ, Inductive types) preserve h-levels, reformulated w.r.t. SqFill rather than the more usual isSet property. For example, the reducts for Π and $\top$

$$\mathsf{sqFill}_\Pi : \forall\{A : \mathrm{Type}\}\ \{B : A \to \mathrm{Type}\} \to ((x : A) \to \mathsf{SqFill}\ (B\ x)) \to \mathsf{SqFill}\ (\Pi_{x:A}(B\ x))$$
$$\mathsf{sqFill}_\top : \mathsf{SqFill}\ \top$$

allow the following computation by recursing on the type $(\top \to \top)$:

$$\mathsf{sqFill}\ (\top \to \top) \Rightarrow_\beta \mathsf{sqFill}_\Pi\{\top\}\ \{\top\}\ (\lambda\ \_\ .\ \mathsf{sqFill}\ \top) \Rightarrow_\beta \mathsf{sqFill}_\Pi\{\top\}\ \{\top\}\ (\lambda\ \_\ .\ \mathsf{sqFill}_\top)$$

Dependent or non-dependent? We can generalise SqFill to the dependent case of a square of types $I \to I \to A$, where the corners have types $\mathrm{lu} : A\ 0\ 0, \mathrm{ld} : A\ 0\ 1, \mathrm{ru} : A\ 1\ 0, \mathrm{rd} : A\ 1\ 1$ respectively, and the sides are dependent paths such as $(l : \mathrm{PathP}\ (\lambda\, i.A\ 0\ i)\ \mathrm{lu}\ \mathrm{ld})$. We call this the dependent square-filling property SqPFill, and it is equivalent to SqFill [Agd25]. We gave preservation proofs of both SqFill and SqPFill for Π, Σ, Coproducts, and Path types in [TND25], formalised in Agda Cubical-without-Glue. Instead of relying on existing proofs in Cubical Agda [Agd25], which specialise more general theorems, we prove the preservation theorems for SqFill and SqPFill more directly. In the process, we noticed a discrepancy in the complexity of the proofs: for example, the proof for Π-types was easier for SqFill than SqPFill, but the reverse is true for Σ types:

	SqFill (non-dependent Square-Filling)	SqPFill (dependent Square-Filling)
Pi	Trivial (no Kan operations)	**Complicated**: transport-fill-align
Sigma	**Complicated**: transport-fill-align	Trivial (no Kan operations)
Coproducts	Encode-decode (J, hcomp) [Uni13]	Encode-decode (J, comp) [Uni13]
Path Types	Simple (a single hcomp)	**Complicated**: transport-fill-align

Table 1: Complexity of SqFill and SqPFill preservation proofs

Since neither version has overall simpler computation rules than the other, we arbitrarily chose to implement the non-dependent SqFill for Cubical-UIP.

Conclusion We implemented SqFill on a small set of type formers such as Π, Σ, booleans, natural numbers, the coproduct, and path types, and will evaluate its usefulness on concrete use cases. On the other hand, we are keeping the treatment of the universe [Nuy23], general (co-)inductive (indexed, higher) types, and record types as future work. The development fork can be found on https://github.com/yeejian-tan/agda/tree/cubical-uip.

Acknowledgements

Andreas Nuyts holds a Postdoctoral Fellowship from the Research Foundation - Flanders (FWO; 12AB225N). This research is partially funded by the Research Fund KU Leuven and the Internal Funds KU Leuven.

Bibliography

[Agd25] The Agda Community. 2025. Cubical Agda Library. Retrieved from https://github.com/agda/cubical

[Alt+19] Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi, and Nicolas Tabareau. 2019. Setoid Type Theory - A Syntactic Translation. In *Mathematics of Program Construction - 13th International Conference, MPC 2019, Porto, Portugal, October 7-9, 2019, Proceedings (Lecture Notes in Computer Science)*, 2019. Springer, 155–196. https://doi.org/10.1007/978-3-030-33636-3_7

[AMS07] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. 2007. Observational equality, now!. In *Proceedings of the ACM Workshop Programming Languages meets Program Verification, PLPV 2007, Freiburg, Germany, October 5, 2007*, 2007. ACM, 57–68. https://doi.org/10.1145/1292597.1292608

[Alt99] Thorsten Altenkirch. 1999. Extensional Equality in Intensional Type Theory. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, 1999. IEEE Computer Society, 412–420. https://doi.org/10.1109/LICS.1999.782636

[CND22] Joris Ceulemans, Andreas Nuyts, and Dominique Devriese. 2022. Sikkel: Multimode Simple Type Theory as an Agda Library. In *Proceedings Ninth Workshop on Mathematically Structured Functional Programming, MSFP@ETAPS 2022, Munich, Germany, 2nd April 2022 (EPTCS)*, 2022. 93–112. https://doi.org/10.4204/EPTCS.360.5

[CND25] Joris Ceulemans, Andreas Nuyts, and Dominique Devriese. 2025. BiSikkel: A Multimode Logical Framework in Agda. *Proc. ACM Program. Lang.* 9, POPL (2025), 210–240. https://doi.org/10.1145/3704844

[CNT26] Liang-Ting Chen, Fredrik Nordvall Forsberg, and Tzu-Chun Tsai. 2026. Can We Formalise Type Theory Intrinsically without Any Compromise? A Case Study in Cubical Agda. In *Proceedings of the 15th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '26)*, 2026. Association for Computing Machinery, Rennes, France, 201–215. https://doi.org/10.1145/3779031.3779090

[CD18] Jesper Cockx and Dominique Devriese. 2018. Proof-relevant unification: Dependent pattern matching with only the axioms of your type theory. *J. Funct. Program.* 28, (2018), e12. https://doi.org/10.1017/S095679681800014X

[Coh+17] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2017. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. *FLAP* 4, 10 (2017), 3127–3170. Retrieved from http://collegepublications.co.uk/ifcolog/?00019

[Dan19] Nils Anders Danielsson. 2019. Issue: A variant of Cubical Agda that is consistent with UIP. Retrieved from https://github.com/agda/agda/issues/3750

[Hof95] Martin Hofmann. 1995. A Simple Model for Quotient Types. In *Typed Lambda Calculi and Applications, Second International Conference on Typed Lambda Calculi and Applications, TLCA '95, Edinburgh, UK, April 10-12, 1995, Proceedings (Lecture Notes in Computer Science)*, 1995. Springer, 216–234. https://doi.org/10.1007/BFB0014055

[HC21] Jason Z. S. Hu and Jacques Carette. 2021. Formalizing category theory in Agda. In *CPP '21: 10th ACM SIGPLAN International Conference on Certified Programs and Proofs, Virtual Event, Denmark, January 17-19, 2021*, 2021. ACM, 327–342. https://doi.org/10.1145/3437992.3439922

[Hur10] Chung Kil Hur. 2010. [Coq-Club] Agda with the excluded middle is inconsistent?. Retrieved from https://sympa.inria.fr/sympa/arc/coq-club/2010-01/msg00007.html

[Nuy23] Andreas Nuyts. 2023. Issue: Computational UIP in Agda Cubical. Retrieved from https://github.com/agda/agda/issues/6696

[Nuy24] Andreas Nuyts. 2024. Algebraic theories with contexts, in Cubical Agda. Retrieved from https://github.com/anuyts/ctx-alg

[PT22] Loïc Pujet and Nicolas Tabareau. 2022. Observational equality: now for good. *Proc. ACM Program. Lang.* 6, POPL (2022), 1–27. https://doi.org/10.1145/3498693

[PLT25] Loïc Pujet, Yann Leray, and Nicolas Tabareau. 2025. Observational Equality Meets CIC. *ACM Trans. Program. Lang. Syst.* 47, 2 (2025), 6:1–6:35. https://doi.org/10.1145/3719342

[Sch+25] Steven Schaefer, Nathan Varner, Pedro Henrique Azevedo de Amorim, and Max S. New. 2025. Intrinsic Verification of Parsers and Formal Grammar Theory in Dependent Lambek Calculus. *Proc. ACM Program. Lang.* 9, PLDI (June 2025). https://doi.org/10.1145/3729281

[Shu17] Michael Shulman. 2017. Cubical type theory with UIP. Retrieved from https://groups.google.com/g/homotopytypetheory/c/UegPjMrnx6I/m/UNBLHfZNBAAJ

[SAG22] Jonathan Sterling, Carlo Angiuli, and Daniel Gratzer. 2022. A Cubical Language for Bishop Sets. *Log. Methods Comput. Sci.* 18, 1 (2022). https://doi.org/10.46298/LMCS-18(1:43)2022

[TND25] Yee-Jian Tan, Andreas Nuyts, and Dominique Devriese. 2025. Towards Computational UIP in Cubical Agda. Retrieved from https://arxiv.org/abs/2511.21209

[Tan25] Yee-Jian Tan. 2025. Cubical — Agda 2.9.0 Documentation : Cubical Agda without Glue. Retrieved from https://agda.readthedocs.io/en/latest/language/cubical.html#cubical-without-glue

[Uni13] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study. Retrieved from https://homotopytypetheory.org/book