Displayed Universal Algebra in UniMath: Basic Definitions and Results

Gianluca Amato[1], Matteo Calosci[2], Marco Maggesi[2] and Cosimo Perini Brogi[3]

[1]*Laboratorio di Logica Computazionale e Intelligenza Artificiale, Università di Chieti-Pescara*
[2]*Università di Firenze*
[3]*Scuola IMT Alti Studi Lucca*

Abstract

Univalent mathematics and homotopy type theory provide a structural approach to formalizing mathematical concepts. Inspired by the role of displayed categories in the univalent treatment of category theory, we develop an analogous notion of displayed algebras for universal algebra. This modular and layered approach allows us to construct and reason about algebraic structures over a fixed base. Classical constructions such as cartesian products, pullbacks, semidirect products, and subalgebras naturally arise as total algebras of suitable displayed algebras. The main results are fully formalized in the UniMath library.

Keywords

Homotopy type theory, Universal algebra, Displayed constructions, Computerized mathematics, UniMath

1. Introduction

1.1. Homotopy Type Theory and Univalent Foundations

Homotopy Type Theory (HoTT) is a modern approach to the foundations of mathematics that integrates ideas from homotopy theory, type theory, and category theory [1]. At its core, HoTT proposes a new way of thinking about mathematical objects where the notion of equality between terms can carry additional structure.

In this framework, a type can be understood as a space, and its terms as points in that space. Two terms are considered equal not just in the traditional sense, but by the existence of a path connecting them - capturing the idea that identity can be continuous and structured. In this topologically inspired point of view, two terms can be connected by multiple paths, leading us to consider paths between paths, representing equalities between equalities. Since equality in this setting brings information on how terms get identified, higher paths are not necessarily trivial (in the same way, topological spaces are not necessarily simply connected). Moreover, this can be iterated, giving rise to a rich hierarchy of higher-dimensional structures, where logical propositions and mathematical constructions can be interpreted geometrically.

Type isomorphisms correspond to homotopy equivalence; by the univalence axiom of Vladimir Voevodsky [2, 3], they can be computed as propositional equality on the Universe. This means that equivalent types can be treated as any other two equal terms in the system; moreover, theorems about one can be transported to the other. This aligns with the informal intuition of reasoning "up to isomorphism" and allows bringing this intuition into formal, machine-verifiable mathematics.

The UniMath project builds on univalent foundations to provide a large, formally verified library of mathematics [4] along with other systems such as Coq-HoTT [5] and Cubical Agda [6]. UniMath is particularly well-suited to our purposes thanks to its extensive library, which offers the necessary tools

ICTCS'25: Italian Conference on Theoretical Computer Science, September 10–12, 2025, Pescara, Italy
gianluca.amato@unich.it (G. Amato); matteo.calosci@unifi.it (M. Calosci); marco.maggesi@unifi.it (M. Maggesi); cosimo.perinibrogi@imtlucca.it (C. Perini Brogi)
https://www.sci.unich.it/~amato (G. Amato); https://sites.google.com/unifi.it/maggesi/ (M. Maggesi); https://sysma.imtlucca.it/people/cosimo-perini-brogi (C. Perini Brogi)
0000-0002-6214-5198 (G. Amato); 0009-0001-5237-7980 (M. Calosci); 0000-0003-4380-7691 (M. Maggesi); 0000-0001-7883-5727 (C. Perini Brogi)

and infrastructure for developing universal algebra in a modular and structured way. Moreover, the library also offers many implementations of algebraic structures (groups, ordered sets, …) which may be presented as instances of our constructions.

In this context, our goal is to develop a formalization of Universal Algebra that is naturally aligned with the principles and foundational philosophy of univalent mathematics in UniMath.

1.2. General framework

Both universal algebra and category theory provide unifying frameworks for studying algebraic structures in abstract terms [7, 8, 9].

In universal algebra, a structure – be it a group, a ring, or a lattice – is described as a theory, i.e. in terms of abstract operations over a carrier and a set of equations these operations are required to satisfy. This way, once a theory is specified, specific instances of algebraic structures are defined as models of that theory, and an algebra homomorphism is a structure-preserving map between models [10].

In category theory, Lawvere theories provide a presentation-invariant version of algebraic theories, where algebraic operations are given by morphisms in the Lawvere theory. Specific instances of algebraic structures are defined as product-preserving functors from a Lawvere theory to categories with finite products. This reveals universal algebra as the theory of all structures that can be defined in categories with finite products [11].

The work started with [12] showed that many categorical constructions can be developed modularly both at the objects and morphisms levels (as well as at higher categorical levels [13]), by adding progressively layers of further structure. Similarly, in universal algebra, it is often possible to modularly construct an algebraic structure. In this work, we show how to rephrase the methods of displayed categories in order to deal with classical concepts of universal algebra, making the analogy we have just sketched more precise. We thus widen the investigations on the notion of *displayed algebras* [14, 15, 16], with a focus on developing universal algebra with the displayed-category style.

Concretely, given a base algebra, we *display* the additional algebraic structure – elements and operations – in a structured layer above such base. This approach not only streamlines proofs of properties and constructions but also facilitates the reuse of generic lemmas across different algebraic contexts. This way we can reap the organizational benefits of displayed formalisms, both for the mathematical developments of universal algebra and their computer implementations.

1.3. Source code

Our code is freely available from our GitHub repository[1] and it is already integrated in the official UniMath library. When discussing concepts which are available in our source code, we will provide specific links below, denoted by the icon ↗.

2. Universal Algebra in Univalent Foundations

2.1. Development in UniMath

This work extends our previous development [17, 18], which introduced the first (precategorical) implementation of universal algebra in the UniMath system. There, we formalized multi-sorted signatures, algebras, homomorphisms, and categories of algebras, and addressed the absence of general inductive types in UniMath by introducing a stack-based encoding of terms. This encoding enabled the construction of term algebras with desired computational properties and of homotopy W-types with finite branching. The present work builds upon and generalizes that foundation by developing a theory of displayed algebras, which allows for modular, layered constructions over a given base algebra.

[1] https://github.com/UniMathUA/UniMath/tree/ICTCS-2025

We report here some of the key constructions of our development. We refer to our earlier work for further details. We start with the basic definition of signature and algebra. We require that the type of sorts has a decidable equality and that the type of operations is an hSet (a set in the homotopical sense).

```
Definition signature : UU := ∑ (S: decSet) (O: hSet), O → list S × S.
```

```
Definition algebra (σ: signature): UU
  := ∑ A: sUU (sorts σ), ∏ nm: names σ, A⋆ (arity nm) → A (sort nm).
```

The construction of the algebra of terms is based on the idea of a stack machine that we roughly sketch below:

1. A sequence of function symbols is thought of as a series of commands to be executed by a *stack machine* whose stack is made of sorts, and which we define by means of a maybe monad.

   ```
   Definition oplist (σ: signature):= list (names σ).
   ```

   ```
   Definition stack (σ: signature): UU := maybe (list (sorts σ)).
   ```

2. When an operation symbol is executed, its arity is popped out from the stack and replaced by its range. When a stack underflow occurs, or when the sorts present in the stack are not the ones expected by the operator, the stack goes into an error condition which is propagated by successive operations. We implement this process by means of two functions:

   ```
   Definition opexec (nm: names σ): stack σ → stack σ.
   ```

   ```
   Definition oplistexec (l: oplist σ): stack σ.
   ```

   The former is the stack transformation corresponding to the execution of the operation symbol `nm`. The latter returns the stack corresponding to the execution of the entire `oplist` argument starting from the empty stack. The list is executed from the last to the first operation symbol.

   Several additional lemmas are required in order to make us able to handle stacks – by concatenating, splitting, etc. – without incurring failures breaking down the whole process.[2]

3. Finally, we define a term to be just a list of operation symbols that, after being executed by `oplistexec`, returns a list of length one with appropriate sort:[3]

   ```
   Definition isaterm (s: sorts σ) (l: oplist σ): UU
     := oplistexec l = just [s].
   ```

   ```
   Definition term (σ: signature) (s: sorts σ): UU
     := ∑ t: oplist σ, isaterm s t.
   ```

2.2. Displayed algebras and current development

The natural starting point of our work is the formal definition of displayed algebras.

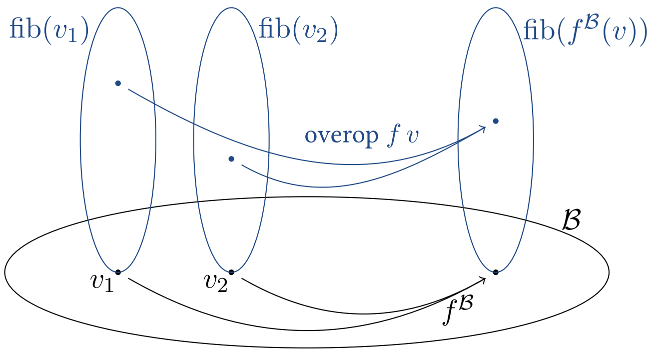
Definition. *Given an algebra $\mathcal{B}$ over a multi-sorted signature σ, a displayed algebra over $\mathcal{B}$ consists of:*

- *a family of "fiber" types indexed over terms of $\mathcal{B}$;*
- *a family of "overop" functions indexed over any operation name f of σ and any vector v of terms of $\mathcal{B}$ each having the appropriate sort specified by the arity of f. Each function has the product of the fiber of the components of v as domain and the fiber of $f^{\mathcal{B}}(v)$ as codomain.*

This is depicted in the following:

[2] In particular, since we need to decide when a stack is correctly executed and when an underflow occurs, we see the reasons for choosing sorts to constitute a decidable set.

[3] From a purely HoTT-perspective, we can easily see also that the type of stacks over σ is an hSet, so that the property of being a term is not proof-relevant (`isapropisaterm`).



Unfortunately, formalized mathematics sometimes makes obscure concepts which are otherwise quite natural. The formal definition of a displayed algebra in UniMath is the following:

```
Definition disp_alg {σ: signature} (B: algebra σ) :=
  ∑ fib : (∏ (s: sorts σ) (b: support B s), UU),
   ∏ (nm: names σ) (base_xs: hvec (vec_map B (arity nm))),
     hvec (h1map_vec (v := arity nm) fib base_xs)
     → fib (sort nm) (ops B nm (base_xs)).
```

In a straightforward and natural way, every displayed category gives rise to a total category. Analogously, from a displayed algebra we can build the associated total algebra. The carriers are dependent pairs with the carriers of $\mathcal{B}$ in the first component and the fiber types in the second one. Operations are built from the ones of $\mathcal{B}$ and the overop functions.

```
Definition total_alg {A: algebra σ} (D: disp_alg A) : algebra σ.
```

This construction is one of the primary motivations for working with displayed structures since it provides a clear, modular framework for assembling algebras from simpler components.

In the categorical setting, a displayed category encodes the same information as a functor into the base category. In our algebraic setting, we show the analogous result:

Theorem. *Given an algebra $\mathcal{B}$ over a multi-sorted signature σ with hSet carriers , the type of algebra morphisms targetting $\mathcal{B}$ is equivalent to the type of displayed algebras over $\mathcal{B}$.*

Proof sketch. From any homomorphism h targeting $\mathcal{B}$, we construct the displayed algebra of its fibers. Fiber types are given by the fiber under h of the specific index term, and overop functions are derived from the operations in the source algebra: their well-typedness depends on the fact that h is a homomorphism.

On the other hand taking the projections on the first component of the total algebra carriers leads to the definition of the forgetful homomorphism.

We finally show that taking the displayed algebra of the fiber and taking the forgetful homomorphism of a total algebra are operations inverse to each other. This also requires verifying higher coherence conditions between the type components of the whole construction.

This final step is both conceptually and technically the most delicate part of the development: it has no direct analogue in classical mathematics, as it relies crucially on the dependent and homotopical nature of types in univalent foundations. The argument is relatively long and involves managing multiple layers of dependent data and coherence, which would be invisible or collapsed in a set-theoretic setting. □

2.3. Motivating Examples.

We emphasize how various familiar algebraic constructions – in particular, those admitting a staged or layered description – can be systematically recovered by taking the total algebra of a displayed algebra. Here are some examples:

Cartesian Products: A displayed algebra can encode how algebraic operations behave componentwise on a product of carriers. Collecting these componentwise structures into a single object yields the familiar cartesian product algebra as its total algebra.

Pullbacks: By generalizing the previous example, one can display algebraic structure along pullback squares to obtain a pullback of algebras. The resulting total algebra thus inherits its operations from the displayed structure, reflecting the universal property of the pullback on the level of algebras.

Semidirect Products: The semidirect product of groups can also be viewed as the total algebra of a suitable displayed algebra. Here, one "displays" how a normal subgroup and a quotient group interact, and then reassembles this information into the total structure defining the semidirect product.[4]

Subalgebras: Consider a displayed algebra that restricts the underlying carriers of a larger algebra to subsets closed under its operations. When one collects this restricted (or "sub") structure into a single object, the total algebra precisely captures the notion of a subalgebra.

3. Related and future work

This paper builds upon our original UniMath library for universal algebra [18]. Other formalizations of universal algebra in dependent type theory are [20, 21, 22, 23, 24, 25, 26]. In the univalent setting, universal algebra is formalised in [27, 28].

Our novel implementation of displayed algebras is inspired by the work in displayed categories started with [12]; in future, we plan to explore further connections between their displaying methods for categories and our constructions at the algebraic level. At the same time, we plan to extend our original UniMath library for universal algebra by making extensive use of the techniques of displayed algebras that we communicate here. Finally, it would be relevant to bridge our formalization of universal algebra in UniMath with the existing library on Lawvere theories[5] in the same univalent system and investigate potential transfers of results between the categorical and universal languages for algebraic structures using a unified displayed formalism.

Acknowledgments

This work was partially supported by: the project SERICS 'Security and Rights in CyberSpace' – PE0000014, financed within PNRR, M4C2 I. 1. 3, funded by the European Union - NextGenerationEU; the MIUR project PRIN 2017JZ2SW5 'Real and Complex Manifolds: Topology, Geometry and holomorphic dynamics'; the PNRR project FAIR – Future AI Research (PE00000013), Spoke 9 - Green-aware AI, under the NRRP MUR program funded by the NextGenerationEU; Istituto Nazionale di Alta Matematica – INdAM groups GNAMPA, GNCS and GNSAGA.

Declaration on Generative AI

During the preparation of this work, the authors used X-GPT-4 to: Drafting content, Paraphrase and reword, Improve writing style. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

[4] While semidirect products of groups have well-known generalizations to other contexts in universal algebra [19], those generalizations are not treated here.

[5] https://github.com/UniMath/UniMath/tree/master/UniMath/AlgebraicTheories

References

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

[2] P. Aczel, On Voevodsky's univalence axiom, Mathematical Logic: Proof Theory, Constructive Mathematics, Samuel R. Buss, Ulrich Kohlenbach, and Michael Rathjen (Eds.). Mathematisches Forschungsinstitut Oberwolfach, Oberwolfach (2011) 2967.

[3] M. H. Escardó, A self-contained, brief and complete formulation of Voevodsky's Univalence Axiom, https://arxiv.org/abs/1803.02294, 2018. doi:10.48550/ARXIV.1803.02294.

[4] V. Voevodsky, B. Ahrens, D. Grayson, et al., Unimath — a computer-checked library of univalent mathematics, Available at http://unimath.org, 2025. URL: https://github.com/UniMath/UniMath. doi:10.5281/zenodo.8427604.

[5] A. Bauer, J. Gross, P. L. Lumsdaine, M. Shulman, M. Sozeau, B. Spitters, The HoTT library: a formalization of homotopy type theory in Coq, in: Y. Bertot, V. Vafeiadis (Eds.), Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16-17, 2017, ACM, 2017, pp. 164–172. URL: https://doi.org/10.1145/3018610.3018615. doi:10.1145/3018610.3018615.

[6] A. Vezzosi, A. Mörtberg, A. Abel, Cubical agda: a dependently typed programming language with univalence and higher inductive types, Proc. ACM Program. Lang. 3 (2019). URL: https://doi.org/10.1145/3341691. doi:10.1145/3341691.

[7] M. Hyland, J. Power, The category theoretic understanding of universal algebra: Lawvere theories and monads, Electronic Notes in Theoretical Computer Science 172 (2007) 437–458.

[8] G. Grätzer, Universal algebra, Springer Science & Business Media, 2008.

[9] J. Adámek, J. Rosickỳ, E. M. Vitale, Algebraic theories: a categorical introduction to general algebra, volume 184, Cambridge University Press, 2010.

[10] H. P. Sankappanavar, S. Burris, A course in universal algebra, Graduate Texts Math 78 (1981) 56.

[11] F. W. Lawvere, Functorial semantics of algebraic theories, Proceedings of the National Academy of Sciences 50 (1963) 869–872.

[12] B. Ahrens, P. L. Lumsdaine, Displayed Categories, Logical Methods in Computer Science Volume 15, Issue 1 (2019). URL: https://lmcs.episciences.org/5252. doi:10.23638/LMCS-15(1:20)2019.

[13] B. Ahrens, D. Frumin, M. Maggesi, N. Veltri, N. van der Weide, Bicategories in univalent foundations, Mathematical Structures in Computer Science 31 (2021) 1232–1269. doi:10.1017/S0960129522000032.

[14] A. Kaposi, A. Kovàcs, T. Altenkirch, Constructing quotient inductive-inductive types, Proceedings of the ACM on Programming Languages 3 (2019) 1–24. URL: https://doi.org/10.1145/3290315. doi:10.1145/3290315.

[15] A. Kovács, Type-Theoretic Signatures for Algebraic Theories and Inductive Types, Ph.D. thesis, Eötvös Loránd University Doctoral School of Informatics, 2022. URL: https://arxiv.org/abs/2302.08837.

[16] K. Sojakova, Higher inductive types as homotopy-initial algebras, in: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '15, ACM, New York, NY, USA, 2015, p. 31–42. doi:10.1145/2676726.2676983.

[17] G. Amato, M. Maggesi, M. Parton, C. Perini Brogi, Universal algebra in UniMath, in: Workshop on Homotopy Type Theory/Univalent Foundations – HoTT/UF2020, 2020, pp. 1–4. URL: https://hott-uf.github.io/2020/.

[18] G. Amato, M. Calosci, M. Maggesi, C. Perini Brogi, Universal algebra in UniMath, Mathematical Structures in Computer Science 34 (2024) 869 – 891. doi:10.1017/S0960129524000367.

[19] A. Facchini, D. Stanovský, Semidirect products in universal algebra, 2023. URL: https://arxiv.org/abs/2311.04321. arXiv:2311.04321.

[20] V. Capretta, Universal algebra in type theory, in: Y. Bertot, G. Dowek, A. Hirschowits, C. Paulin, L. Théry (Eds.), Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs '99, volume 1690 of *LNCS*, Springer, 1999, pp. 131–148. URL: http://www-sop.inria.fr/lemme/

Venanzio.Capretta/universal_algebra.html.
[21] E. Gunther, A. Gadea, M. Pagano, Formalization of universal algebra in Agda, Electronic Notes in Theoretical Computer Science 338 (2018) 147–166. URL: https://www.sciencedirect.com/science/article/pii/S1571066118300768.
[22] A. Abel, Birkhoff's Completeness Theorem for Multi-Sorted Algebras Formalized in Agda, CoRR abs/2111.07936 (2021). URL: https://arxiv.org/abs/2111.07936. arXiv:2111.07936.
[23] W. DeMeo, J. Carette, A Machine-Checked Proof of Birkhoff's Variety Theorem in Martin-Löf Type Theory, in: H. Basold, J. Cockx, S. Ghilezan (Eds.), 27th International Conference on Types for Proofs and Programs (TYPES 2021), volume 239 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022, pp. 4:1–4:21. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TYPES.2021.4. doi:10.4230/LIPIcs.TYPES.2021.4.
[24] W. DeMeo, The Agda Universal Algebra Library, Part 1: Foundation, arXiv preprint arXiv:2103.05581 (2021).
[25] W. DeMeo, The Agda Universal Algebra Library, Part 2: Structure, arXiv preprint arXiv:2103.09092 (2021).
[26] W. DeMeo, J. Carette, The Agda Universal Algebra Library (agda-algebras), GitHub.com, 2021. doi:10.5281/zenodo.5765793, documentation available at https://ualib.org.
[27] A. Lynge, Universal algebra in HoTT, 2017. URL: https://github.com/andreaslyn/Work/blob/master/Math-Bachelor.pdf, bachelor's thesis, Department of Mathematics, Aarhus University.
[28] A. Lynge, B. Spitters, Universal algebra in HoTT, in: TYPES 2019, 25th International Conference on Types for Proofs and Programs, 2019, pp. 1–2. URL: http://www.ii.uib.no/~bezem/abstracts/TYPES_2019_paper_7.