Globular: An Online Proof Assistant for Higher-Dimensional Rewriting*

Krzysztof Bar[1], Aleks Kissinger[2], and Jamie Vicary[3]

1 **Department of Computer Science, University of Oxford, Oxford, UK**
`krzysztof.bar@cs.ox.ac.uk`
2 **iCIS, Radboud University Nijmegen, Nijmegen, The Netherlands**
`aleks@cs.ru.nl`
3 **Department of Computer Science, University of Oxford, Oxford, UK**
`jamie.vicary@cs.ox.ac.uk`

Abstract

This article introduces Globular, an online proof assistant for the formalization and verification of proofs in higher-dimensional category theory. The tool produces graphical visualizations of higher-dimensional proofs, assists in their construction with a point-and-click interface, and performs type checking to prevent incorrect rewrites. Hosted on the web, it has a low barrier to use, and allows hyperlinking of formalized proofs directly from research papers. It allows the formalization of proofs from logic, topology and algebra which are not formalizable by other methods, and we give several examples.

1998 ACM Subject Classification I.2.3 Deduction and Theorem Proving

Keywords and phrases higher category theory, higher-dimensional rewriting, interactive theorem proving

Digital Object Identifier 10.4230/LIPIcs.FSCD.2016.34

1 Introduction

This paper is a system description for Globular [20], an online tool for formalizing and verifying proofs in semistrict globular higher category theory. It operates from the perspective of *higher-dimensional rewriting*, with terms represented as graphical structures, and proofs constructed and visualized as sequences of rewrites on these structures. At the time of writing, the tool operates up to the level of 3-categories and is being actively developed (in parallel with the corresponding theory) to support for 4-categories and higher.

Globular is the first proof assistant of its kind, and it allows many proofs from higher category theory to be formalized, verified and visualized in a way that would not be practical in any other tool. The closest comparable tools are Quantomatic [7], which does diagrammatic rewriting for monoidal categories, and the formalisations of homotopy type theory in the proof assistants Coq and Agda [5]. The latter can indeed be used to perform logical and homotopy-theoretical proofs from a higher-categorical perspective. However, this approach diverges from ours in that it is based on the syntax of Martin-Löf type theory rather than diagrams, and identity types naturally lead one to treat higher-dimensional *invertible* structures (e.g. ∞-groupoids) as first-class citizens, rather than the more general structures we'll consider. Another comparable tool is *Orchard* [3], which allows the formalization of proofs in opetopic

* This work was supported by the European Research Council grant 320571.

(as opposed to globular) higher categories; this tool can handle ∞-categories, and has many attractive properties, although the opetopic approach to higher categories is far less dominant than the globular approach, and its associated graphical calculus is less attractive for many purposes. The higher dimensional rewriting implemented by Globular draws inspiration from the polygraphic approach to rewriting [10, 14], but extends it to allow for non-strict higher-categorical structures.

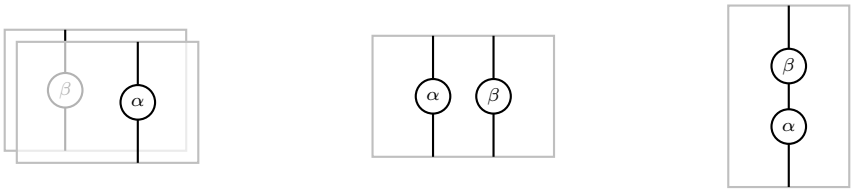
Globular was designed to make it as quick and easy as possible for users to go from zero to proving theorems and sharing proofs. Hence, it is entirely web-based, with all logic taking place client-side in the user's web browser. The most commonly used procedures run in linear time with little overhead, so this is practical on modest hardware even for large diagrams. Proofs can be stored on the remote server for later reference, or downloaded for storage locally. Permanent hyperlinks to formalized proofs can be generated and embedded as links in research papers, allowing readers instant access to the formalization without the usual barriers-to-use of downloading, installing and maintaining an executable. The tool launched in December 2015, and has been well-received by the community, with 4126 sessions across 884 unique users in the first 2 months since deployment[1].

In Section 2, we give a brief overview of the mathematical foundations of Globular, namely higher-dimensional category theory and rewriting. In Section 3 we exhibit all of the core functionality of the tool via a simple example. In Section 4 we discuss the implementation, including the architecture and relevant data structures and procedures for rewriting. We conclude by surveying a variety of interesting proofs in Globular by the authors and others, with direct links for viewing online.

2 Mathematical foundations

Higher category theory is the study of *n-categories*. As well as objects and morphisms familiar from traditional category theory, which are we call 0-cells and 1-cells, an n-category also has morphisms between morphisms (2-cells), morphisms between those (3-cells), and so on, up to level n. An n-category has a n distinct composition operations, which allow cells to be combined to produce new cells.

A convenient notation for working with n-categories is the *graphical calculus*, in which a k-cell is represented as an $(n-k)$-dimensional geometrical structure[2]. Composition then corresponds to 'gluing' of these structures along the different axes of n-dimensional space. For example, in a 3-category, we represent 3-cells as points, 2-cells as lines, 1-cells as regions, and 0-cells as 'volumes'. Given 3-cells α and β, we could form the following composite 3-cells by composing along three different axes:



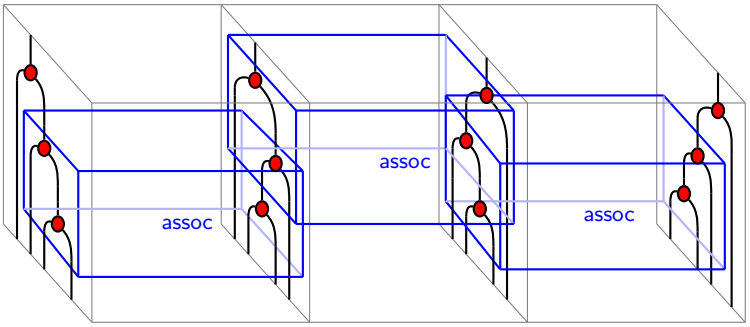
(1)

In this way we can draw diagrams to represent arbitrary composites, in principle in any dimension (although for $n > 3$, visualizing the resulting geometrical structure of course becomes nontrivial).

[1] Usage statistics from Google Web Analytics.
[2] This is rigorously developed only for $n \leq 3$ [2, 6].

We take a *rewriting* perspective on higher category theory. Suppose a $(k+1)$-cell X has source and target k-cells α and α' respectively. Then we interpret X as a way to rewrite α into α'. Since composition in higher category theory is local, this also works for composite cells: for example, we can apply X to any of the composites in (1) to obtain a new composite with α replaced by α'.

The attractive feature of this perspective is that there is no fundamental difference between the notions of *composition* and *proof*. A proof that some diagram D of k-cells can rewritten into some other diagram D' amounts to building a composite $(k+1)$-cell with source D and target D', using just the 'axiom' cells of a given theory. For instance, if we have a 3-cell called 'assoc' which captures an associativity rule of 2-cells, we can prove a theorem about associativity as a composition of 3-cells:



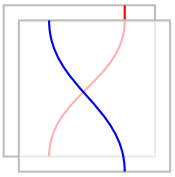
That is, we can *define* a composite $(k+1)$-cell as a rewrite sequence on composite k-cells. This gives a recursive definition of composition, which terminates with a family of 'basic' rewrite operations, which the user must specify. This is the essence of Globular's approach to higher category theory.

In higher category theory, we have some freedom to decide what it means for two things to be 'the same'. At one extreme are 'fully weak' n-categories, where all of the axioms governing the composition of cells (such as associativity and unit axioms) hold only up to higher-dimensional cells. For example, for 1-cells f, g, h, rather than requiring associativity: $f \circ (g \circ h) = (f \circ g) \circ h$, we merely assert the existence of a (weakly) invertible family of 'associator' 2-cells $(f \circ g) \circ h \to f \circ (g \circ h)$. These in turn must satisfy various coherence properties, which we again interpret only up to higher-dimensional cells (which *themselves* must satisfy coherence properties, and so on). While these structures arise naturally in many contexts, the amount of bureaucracy that arises from this structure makes it hard to work with weak n-categories as purely syntactic objects.

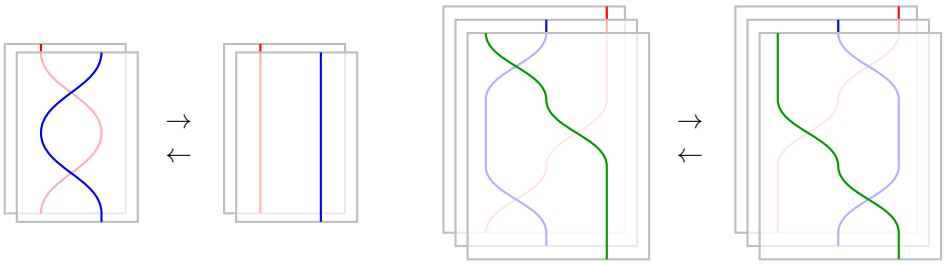
At the other extreme are the *strict n-categories* which require all the axioms involving composition of cells to hold as on-the-nose equalities. These are quite easy to define [11], and admit an evident notion of finite presentation, called a *polygraph* or *computad*, and have a reasonably well-behaved *higher-dimensional rewrite theory* [4]. However, for $n > 2$, it is *not* the case that every weak n-category is equivalent to a strict one. To see where this richness of weak categories comes from, we consider the interchange law, which in a 2-category acts as follows as a rewrite on composite 2-cells:

$$(\boldsymbol{f} \circ_1 1_B) \circ_2 (1_{A'} \circ_1 \boldsymbol{g}) \quad := \quad \xrightarrow{I} \quad =: \quad (1_A \circ_1 \boldsymbol{g}) \circ_2 (\boldsymbol{f} \circ_1 1_{B'})$$

When we stop at two dimensions, there is no problem treating this 'node-sliding' rule simply as an equation between diagrams. But seen as a 3-cell in a 3-category, the source and target of I become the bottom and top slice of a 3D picture, the nodes become wires, and the 'sliding' becomes a braiding:



By the invertibility and naturalness properties, these braidings then behave exactly how you would expect genuine topological braids to behave. For instance, the following higher rewrites exist:

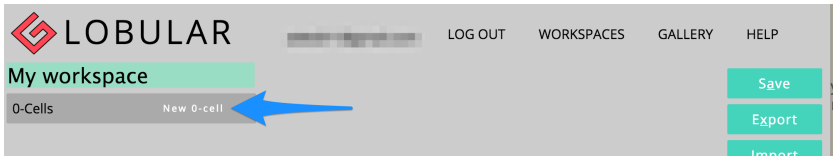


In general, overcrossings and undercrossings are distinct, so it is possible for wires to become tangled. Requiring interchangers to be identities, as in the theory of strict 3-categories, trivializes this part of the theory, and means that it is no longer fully general, in the precise sense that not every 3-category is equivalent to a strict 3-category.

It follows that the strict n-categorical setting in which the polygraph community work is not sufficiently general to reason about arbitrary n-categories. The solution is to work instead with *semistrict* n-categories, which allows a small amount of weak structure, sufficient to ensure that every weak n-category is equivalent to semistrict n-category. For $n = 3$, *Gray categories* have this property; they are defined as 3-categories in which all weak structure is the identity, *except* for interchangers[3]. The version of Globular which is operational at the time of writing implements the axioms of a Gray category.

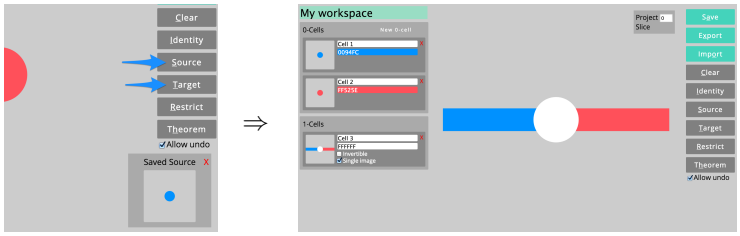
3 Using Globular

Constructing a theory and proving theorems in Globular is an inductive process, whereby lower-dimensional objects are used to construct higher-dimensional objects. This is done by building up a *signature*, i.e. a collection of generators, in parallel with increasingly higher-dimensional diagrams. From an empty signature, the only thing to do is add new 0-cells:

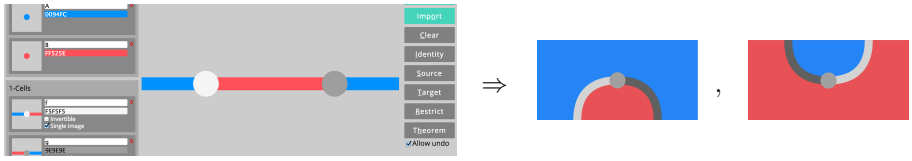


[3] A definition of semistrict n-category for $n > 3$ has not yet been generally accepted.

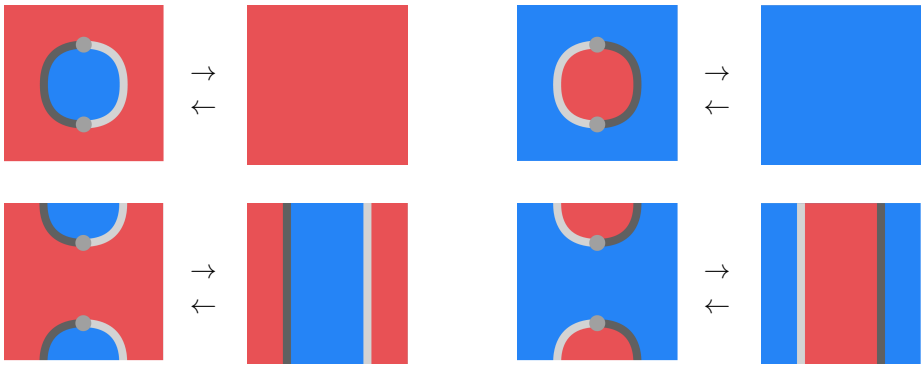
Once we have some 0-cells, these can be made the sources and targets of new 1-cells:



At this point things start to get interesting, since 1-cells can be *attached* to each other to form non-trivial diagrams. These diagrams can then form the sources and targets of new 2-cells:



In turn, these 2-cells can be composed to form larger diagrams, which and form the sources and targets of new 3-cells. We can either interpret these new 3-cells as new generators, or as *equations* between 2d diagrams. For example, we can make our 'cap' and 'cup' 2-cells invertible by adding the following 3-cells to our theory:



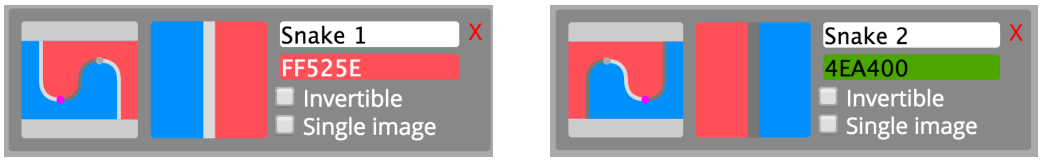
These invertible 'cup' and 'cap' 2-cells yield a familiar categorical structure.

▶ **Definition 1.** In a 2-category, an *equivalence* is a pair of objects A and B, a pair of 1-cells $A \xrightarrow{F} B$ and $B \xrightarrow{G} A$ and invertible 2-cells $F \circ G \xrightarrow{\alpha} \mathrm{id}_A$ and $\mathrm{id}_A \xrightarrow{\beta} G \circ F$, denoted as follows:

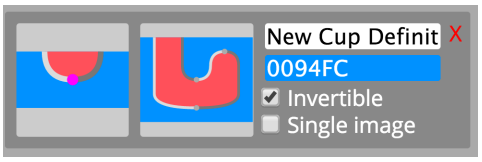
A special case is where the 2-category is **Cat**, in which case this yields the usual notion of equivalence of categories. Then the following is a well-known fact about equivalences in a 2-category:

▶ **Theorem 2.** *In a 2-category, every equivalence gives rise to a dual equivalence.*

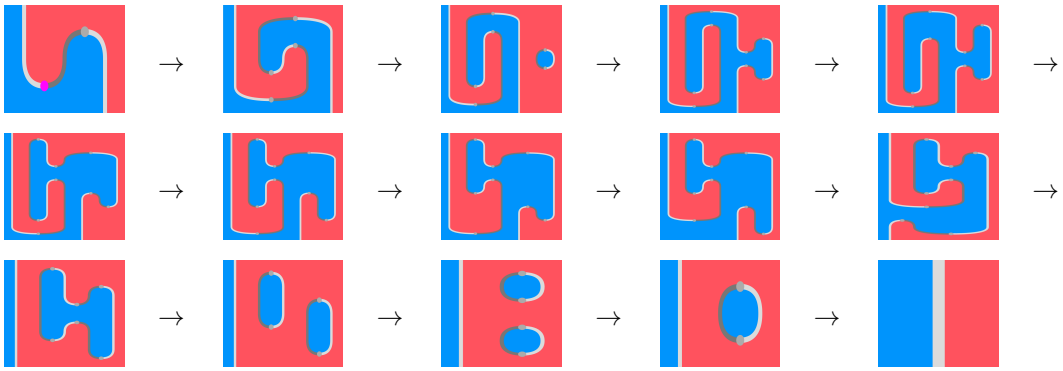
An equivalence is called a *dual equivalence* if it additionally satisfies the *snake equations*, shown here as theorems in Globular:



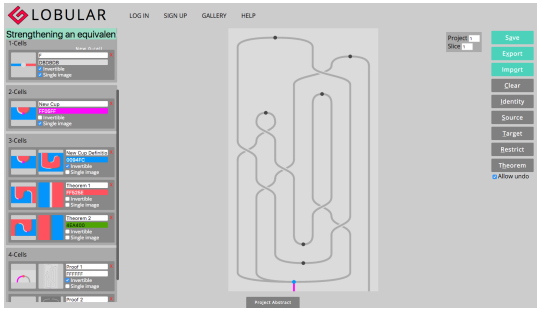
We can prove these theorems by replacing the 'cup' with a 'sock', defined in terms of the old cup and cap:



We can show that our new 'cup' satisfies the snake equation, with the original 'cap'. To prove the first snake equation, we perform the following (non-trivial!) sequence of rewrites in Globular:



This proof is itself a 3-cell. In Globular, we can either browse through it slice-by-slice, or we can see the overall structure of the proof as a single diagram, by choosing 'Project=1' in the interface:



This projects out one dimension so we call look at this entire 3-cell 'side-on'. The nodes represent applications of rewrite rules, and the wires represent 2-cells. From this view, we can refactor the proof by eliminating redundant steps (e.g. a rewrite immediately followed by its inverse) or by re-ordering rewrites that are applied to independent parts of the diagram.

Once a proof has been constructed, it can be saved privately to the server, or made public by *publishing* it. This assigns the workspace a permanent unique link, which can be shared with others or linked from a research paper. For example, the proof in this section is based on the formalization available here: globular.science/1512.007.

4 Implementation

In this section we give an overview of the implementation of Globular, and the basic structures and algorithms that underlie its operation. The tool itself can be used simply by visiting:

`http://globular.science`

The proof assistant itself runs client-side in the user's web browser, and is written in Javascript. A Node.js back-end serves the client pages, and administers a system of user accounts allowing users to register, store private proofs that are under construction, and (irrevocably) publish proofs that they would like to share publicly. The project is open-source, and the code is available at globular.science/source. Graphics are implemented in SVG.

4.1 Data types

The fundamental structures that Globular makes use of are *signatures*, which are a lists of basic generating cells that the user has specified as to define a theory, and *diagrams*, which are particular composites of generators from a given signature. These two types can be defined very compactly in a mutually-recursive fashion. For clarity, we write these both as type families in dependent type-style notion. Let 'g : Set' declare a finite set g (which we then treat as a type), let List(α) be the type of lists, and $\langle \alpha_1, \alpha_2, \ldots \rangle$ the type of tuples where types in α_j are allowed to depend on α_i for $i < j$. Let Sig(0) and Diag$(0, *)$ both be the unit type $\{*\}$. Then, for $n > 0$:

$$\mathrm{Sig}(n : \mathbb{N}) := \left\langle \begin{array}{l} g : \mathrm{Set}, \\ \sigma : \mathrm{Sig}(n-1), \\ s, t : g \to \mathrm{Diag}(n-1, \sigma) \end{array} \right\rangle \qquad \mathrm{Diag}(n : \mathbb{N}, \sigma : \mathrm{Sig}(n)) := \left\langle \begin{array}{l} s : \mathrm{Diag}(n-1, \sigma), \\ \delta : \mathrm{List}(\ \langle a : \sigma.g,\ c : \mathrm{List}(\mathbb{N}) \rangle\ ) \end{array} \right\rangle$$

An n-signature $\Sigma : \mathrm{Sig}(n)$ therefore consists of an $(n-1)$-signature $\Sigma.\sigma$, and a set of generators $\Sigma.g$, such that each $x : \Sigma.g$ has a source and target $(n-1)$-diagrams $\Sigma.s(x)$ and $\Sigma.t(x)$ respectively, which each contain cells from the $(n-1)$-signature $\Sigma.\sigma$.

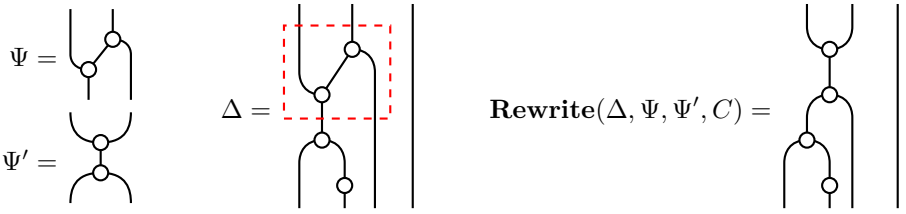
Given a signature $\sigma : \mathrm{Sig}(n)$, a diagram $\Delta : \mathrm{Diag}(n, \sigma)$ consists of a source $(n-1)$-diagram $\Delta.s$, and a list of n-cells that act sequentially on that source. The kth n-cell is given by a pair $\Delta.\delta[k]$, whose first element $\Delta.\delta[k].a$ is a generating cell drawn from the signature σ, and whose second element $\Delta.\delta[k].c$ is a list of numbers which specify the coordinates at which the chosen generating rewrite acts. For example, a 2-diagram consists of a list of 2-cells which are stacked vertically, and this coordinate consists of a single number giving the horizontal position of each 2-cell. We leave the target $(n-1)$-cell implicit, as it can be recovered from the other data (e.g. via the **Slice** procedure below).

4.2 Procedures

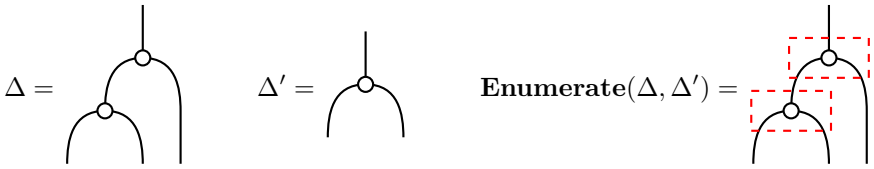
Here we give the type specifications of the basic procedures that manipulate our diagram structures, along with brief descriptions of their functionality.

- **Match**$\big(\Delta : \mathrm{Diag}(n, \sigma), \Delta' : \mathrm{Diag}(n, \sigma)\big) : \mathrm{Bool}$
  Determines whether two diagrams are equal. For Δ, Δ' we first recursively compare whether $\Delta.s$ and $\Delta'.s$ match. If not, return **false**. Otherwise, we compare corresponding elements of $\Delta.\delta$ and $\Delta'.\delta$, if there is a pair δ_k, δ'_k such that either types $\delta_k.a, \delta'_k.a$ or coordinates do not match, then return **false**, otherwise return **true**.

- **Identity**$\big(\Delta : \mathrm{Diag}(n, \sigma)\big) : \mathrm{Diag}(n+1, \sigma)$
  Given an n-diagram, this operation transforms it into an identity $(n+1)$-diagram $\Delta'(n+1, \sigma)$. The set of generators $\Delta'.\delta$ is empty, while both $\Delta'.s$ and $\Delta'.t$ are set to Δ.
- **Rewrite**$(\Delta : \mathrm{Diag}(n, \sigma), \Psi : \mathrm{Diag}(n, \sigma), \Psi' : \mathrm{Diag}(n, \sigma), C : \mathrm{List}(\mathbb{N})) : \mathrm{Diag}(n, \sigma)$
  Here Δ is the diagram that is being rewritten, Ψ is the source of the rewrite, Ψ' is the target of the rewrite, and C is the list of coordinates of where the rewrite is to be applied. $|\Psi.\delta|$ consecutive rewrites in $\Delta.\delta$ starting from position C_{n-1} are removed, with the rewrites in $\Psi'.\delta$ inserted, with their coordinates offset by C. We illustrate this with a simple example, where C is denoted by the dashed rectangle:

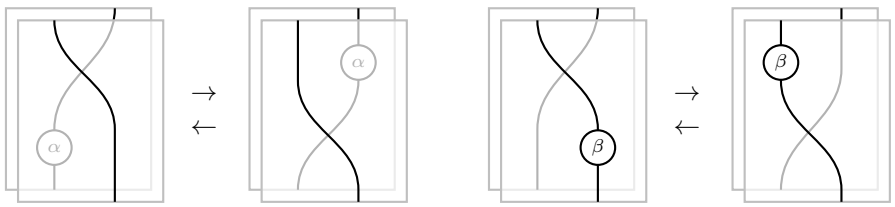


- **Attach**$(\Delta : \mathrm{Diag}(n, \sigma), \Delta' : \mathrm{Diag}(k, \sigma), P : \{s, t\}, C : \mathrm{List}(\mathbb{N})) : \mathrm{Diag}(n, \sigma)$
  Attaches the diagram Δ' to the diagram Δ, where P is a boolean distinguishing between attachment to the source or target boundary of Δ', and C are the coordinates within this boundary of where Δ' is to be attached.
- **Slice**$(\Delta : \mathrm{Diag}(n, \sigma), k : \mathbb{N}) : \mathrm{Diag}(n-1, \sigma.\sigma)$
  Rewrite the source boundary $\Delta.s$ using the leading k n-cells in the ordered set $\Delta.\delta$, to obtain an intermediate diagram in the rewrite sequence.
- **Enumerate**$\big(\Delta : \mathrm{Diag}(n, \sigma), \Delta' : \mathrm{Diag}(n, \sigma)\big) : \mathrm{List}(\mathrm{List}(\mathbb{N}))$
  Enumerates the locations at which Δ' occurs as a subdiagram of Δ. For example, for Δ and Δ' 2-dimensional diagrams as given, the procedure returns a list of length 2:



  The implementation is as follows. First we loop through the elements of the rewrite list $\Delta.\delta$. For the rewrite at depth k, we recursively call **Enumerate**(**Slice**$(\Delta, k-1), \Delta'.s)$. If the result is the empty list, we increment k and retry. If the result is nonempty, at most 1 can be consistent with the structure of Δ, and we then compare types and coordinates of the corresponding generators in $\Delta.\delta$ and $\Delta'.\delta$. If they match, we append k to the coordinate list returned by the recursive call, and we add the coordinate list as a witness to the instance of Δ' being a sub-diagram of Δ.

Globular also has procedures which generate cell coming from the semistrict n-category structure on demand. These consist of *interchangers*, which we've already seen, and *pull-throughs*, which capture the naturality of interchangers:



These cells are generated as they are required because they in fact form an infinite family of cells. This is because f, g and the wires depicted above might be basic generators, but could also be diagrams of generators.

5 Examples

Here we give examples of formalized proofs from algebra and topology. In each case we briefly describe the mathematical context of the proof, and give some details of its formalization. Direct hyperlinks are provided to the formalized proofs on the Globular website; to navigate these proofs, use the *Project* and *Slice* controls at the top-right, and move your mouse cursor over the different parts of the main diagram to understand its components. Documentation on how to use Globular is available [20]. To our knowledge, none of these results have previously been formalized by any existing tool.

▶ **Example 3** (Frobenius implies associative, globular.science/1512.004, length 12)**.** *In a monoidal category, if multiplication and comultiplication morphisms are unital, counital and Frobenius, then they are associative and coassociative.* We formalize this in Globular using a 2-category with a single 0-cell, since this is algebraically equivalent to a monoidal category. Such a proof would be traditionally written out as a series of pictures; for example, see the textbook [8]. Globular produces these pictures automatically.

▶ **Example 4** (Strengthening an equivalence, globular.science/1512.007, length 14)**.** *In a 2-category, an equivalence gives rise to an adjoint equivalence.* This is a classic result from the category theory community [1, 18]; it can be considered one of the first nontrivial theorems of 2-category theory. We investigate it in further detail in Section 3.

▶ **Example 5** (Swallowtail comes for free, globular.science/1512.006, length 12)**.** *In a monoidal 2-category, a weakly-dual pair of objects gives rise to a strongly-dual pair, satisfying the swallowtail equations.* This theorem plays an important role in the singularity theory of 3-manifolds [17]. For the formalization, we model a monoidal 2-category as a 3-category with one 0-cell.

▶ **Example 6** (Pentagon and triangle implies $\rho_I = \lambda_I$, globular.science/1512.002, length 62)**.** *In a monoidal 2-category, a pseudomonoid object satisfies $\rho_I = \lambda_I$.* A *pseudomonoid* is a higher algebraic structure categorifying the concept of monoid; it has the property that a pseudomonoid in **Cat** is the same as a monoidal category. Such a structure is known to be coherent [9], in the sense that all equations commute, and here we give an explicit proof of the equation $\rho_I = \lambda_I$, which played an important role in the early study of coherence for monoidal categories.

▶ **Example 7** (The antipode is an algebra homomorphism, globular.science/1512.011, length 68)**.** *For a Hopf algebra structure in a braided monoidal category, the antipode is an algebra homomorphism.* Hopf algebras are algebraic structures which play an important role in representation theory and physics [12, 19]. Proofs involving these structures are usually presented in *Sweedler notation*, a linear syntax which represents coalgebraic structures using strings of formal variables with subscripts; we do not know of any existing approaches to formal verification for Sweedler proofs. This formalization in Globular is translated from a Sweedler proof given in [15]. For the formalization, we model a braided monoidal category as a 3-category with one 0-cell and one 1-cell.

▶ **Example 8** (The Perko knots are isotopic, globular.science/1512.012, length 251)**.** *The Perko knots are isotopic.* The Perko knots are a pair of 10-crossing knots stated by Little in 1899

to be distinct, but proven by Perko in 1974 to be isotopic [16]. Here we give the isotopy proof, adapted from [13]. A nice feature is that the second and third Reidemeister moves do not have to be entered, since they are already implied by the 3-category axioms. The proof consists of a series of 251 atomic deformations, which rewrite the first Perko knot into the second. By stepping through the proof one rewrite at a time, the isotopy itself can be visualized as a movie.

Acknowledgements. We would like to thank John Baez, Manuel Bärenz, Bruce Bartlett, Eugenia Cheng, Chris Douglas, Eric Finster, Nick Gurski, André Henriques, Samuel Mimram and Dominic Verdon for useful discussions.

References

1 John C. Baez and Aaron D. Lauda. Higher-dimensional algebra V: 2-groups. *Theory and Applications of Categories*, 12:423–491, 2004. URL: `http://www.tac.mta.ca/tac/volumes/12/14/12-14abs.html`.

2 John W. Barrett, Catherine Meusburger, and Gregor Schaumann. Gray categories with duals and their diagrams. *J. Diff. Geom., to appear*. URL: `http://arxiv.org/abs/1211.0529`.

3 Eric Finster. The *Orchard* proof assistant. URL: `https://github.com/ericfinster/orchard`.

4 Yves Guiraud. Polygraphs for termination of left-linear term rewriting systems. 2007. URL: `http://arxiv.org/abs/cs/0702040`.

5 HoTT Formalisations in Coq and Agda. URL: `http://homotopytypetheory.org/coq`.

6 André Joyal and Ross Street. The geometry of tensor calculus, I. *Adv. Math.*, 88(1):55–112, 1991. `doi:10.1016/0001-8708(91)90003-p`.

7 Aleks Kissinger and Vladimir Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In *CADE-25 – 25th International Conference on Automated Deduction*, volume 9195 of *LNCS*. Springer, 2015. `doi:10.1007/978-3-319-21401-6\_22`.

8 Joachim Kock. *Frobenius Algebras and 2D Topological Quantum Field Theories*. Cambridge University Press (CUP), 2003. `doi:10.1017/cbo9780511615443`.

9 Stephen Lack. A coherent approach to pseudomonads. *Adv. Math.*, 152(2):179–202, 2000. `doi:10.1006/aima.1999.1881`.

10 Yves Lafont. Algebra and geometry of rewriting. *Applied Categorical Structures*, 15(4):415–437, 2007. `doi:10.1007/s10485-007-9083-6`.

11 Tom Leinster. A survey of definitions of n-category. *Theory and Applications of Categories*, 10(1):1–70, 2002. `http://arxiv.org/abs/math/0107188`. URL: `http://tac.mta.ca/tac/volumes/10/1/10-01abs.html`.

12 Shahn Majid. *A Quantum Groups Primer*. Cambridge University Press (CUP), 2002. `doi:10.1017/cbo9780511549892`.

13 MathForum. Perko pair knots. URL: `http://mathforum.org/mathimages/index.php/Perko_pair_knots`.

14 Samuel Mimram. Towards 3-dimensional rewriting theory. *Logical Methods in Computer Science*, 10(2), 2014. `doi:10.2168/LMCS-10(2:1)2014`.

15 Bodo Pareigis. In Stefaan Caenepeel and Fred Van Oystaeyen, editors, *Hopf Algebras in Noncommutative Geometry and Physics*, chapter On Symbolic Computations in Braided Monoidal Categories, pages 269–280. CRC Press, 2004.

16 Kenneth A. Perko. On the classification of knots. *Proceedings of the AMS*, 45(2):262–262, 1974. `doi:10.1090/s0002-9939-1974-0353294-x`.

17 Piotr Pstragowski. On dualizable objects in monoidal bicategories. Master's thesis, Bonn University, 2014. URL: `http://arxiv.org/abs/1411.6691`.

18 Saavedra Rivano. *Catégories Tannakiennes*, volume 265 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, 1972. `doi:10.1007/bfb0059108`.

19 Ross Street. *Quantum Groups*. Cambridge University Press (CUP), 2007. `doi:10.1017/cbo9780511618505`.

20 The Globular Team. Globular documentation. URL: `https://ncatlab.org/nlab/show/Globular`.