Data-Parallel Proof Checking for Monoidal Theories (TA1.2)

1 Project & Technical Information

Symmetric monoidal theories (SMTs) are one of the chief methodologies employed by the TA1.1 creators of ARIA's Safeguarded AI Programme (eg. [4, 13, 17, 5, 14]). Generally speaking, categorical modelling for AI-relevant systems is often developed in symmetric monoidal categories (SMCs), because their structure allows for a fine-grained account of resource exchange and consumption, which can be pictorially represented as string diagrams. SMTs serve as a specification for algebraic structures arising in SMCs, as they provide a term syntax for string diagrams and a set of axioms supporting proof-theoretic reasoning — see eg. [16] for an overview.

Our proposal will develop a proof checker for SMTs, based on a data-parallel proof representation. We will provide: (1) a concrete, serialisable proof representation for SMTs, based on our work [21]; (2) data-parallel algorithms for proof checking; (3) integration with a generic proof search backend. In doing so, we contribute to the TA1.2 programme, which has **type checking**, **proof checking**, and **GPU-optimized implementations** among its primary goals.

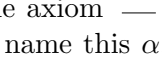
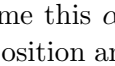
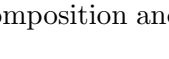
The proof representation is meant to eventually be used by TA2-track teams in proof search applications, particularly those based on reinforcement learning[1]. In particular, its role is analogous to the 'proof trees' in Monte-Carlo Tree Search (see e.g. [22], and the 'hypertree proof search' of [12] for a conceptually similar approach). However, our proof representation enjoys advantages over trees, which are detailed in the last part of Section 1.1.

Although any symmetric monoidal theory will be supported, our initial implementation will be guided by the symmetric monoidal theory of *diagrammatic first order logic* [4]. In this way, we confirm the expressivity and power of our approach, since first order logic is sufficient to encode many other formal systems. However, we aim to produce software suitable for use by any other TA1-track team working with a symmetric monoidal theory.

1.1 Approach

We take inspiration from the Metamath Zero [7] project and propose a proof representation which is **serializable**, **version-controllable**, and **reproducible**. The latter is a common frustration with existing proof assistants, where changing a tactic implementation can 'break' proofs. These desiderata are achieved by representing proofs and terms as morphisms in two distinct but related symmetric monoidal categories which we now describe.

Proof Representation The usual way of doing formal reasoning in a symmetric monoidal category (SMC) $\mathscr{C}$ is by considering *rewrites* between morphisms, as dictated by a set $\mathcal{R}$ of *rewrite rules* (eg. obtained by orienting the equations of an SMT). Indeed, via their representation as string diagrams, morphisms of SMCs can be interpreted as certain kinds of hypergraphs, and rewriting modulo the laws of SMCs as a form of double-pushout hypergraph rewriting, see [1, 2]. One can take a more abstract 2-categorical perspective on rewriting, by letting 0-cells and 1-cells be the objects and morphisms of $\mathscr{C}$, and regarding rewrites as the 2-cells [8, 6]. We can 'extract' an SMC from these data, called the *category of rewrites* $\mathcal{R}(\mathscr{C})$. We let the objects of $\mathcal{R}(\mathscr{C})$ be the morphisms of $\mathscr{C}$, and the morphisms of $\mathcal{R}(\mathscr{C})$ be the rewrites of $\mathscr{C}$-morphisms according to $\mathcal{R}$. The category $\mathcal{R}(\mathscr{C})$ will be our main object of study, as our approach takes proofs (sequences of rewrites) as first-class citizens.

We demonstrate how composition $\fatsemi$ and tensor $\otimes$ work in $\mathcal{R}(\mathscr{C})$ with a concrete example. The theory of cartesian bicategories, see eg.[4, Figure 2], is based on a set of inequalities between string diagrams, which we may view as rewrite rules. For instance, we have the axiom  . As a rewrite, this is a morphism whose source is  and whose target is  . We name this α, and depict its type as $\alpha :$. Suppose also have a rewrite $\beta :$  . The composition and tensor of α and β are depicted below left and right, respectively.

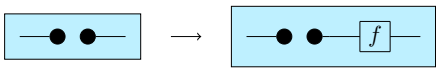
$\alpha \fatsemi \beta :$  $\qquad\qquad \alpha \otimes \beta :$

Notice that composition requires the target of α and source of β to be equal, and that the tensor $\alpha \otimes \beta$ is simply the 'pointwise' tensor of sources and targets, respectively.

[1] In some TA2 settings, a 'proof' might better be thought of as a 'verifiable plan' created by an autonomous agent.

The 2-categorical structure of rewrites also yields a *third* operation: the 'pointwise' (horizontal) composition operation $\fatsemi$. This ensures any possible rewrite is representable. Continuing the example from above, the 'pointwise' composite $\alpha \fatsemi \beta$ is depicted as follows.

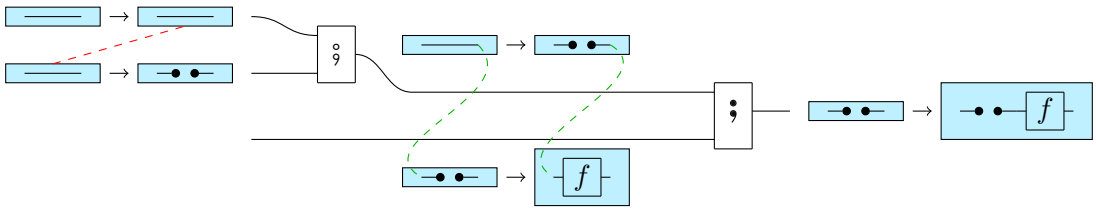
$\alpha \fatsemi \beta$:



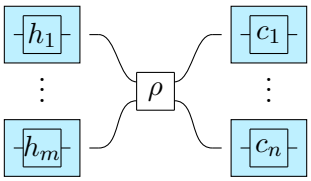
Intuitively, one may think of the tensor as rewriting two *parallel* contexts, while the pointwise composition rewrites two *sequential* contexts. A **proof** in $\mathcal{R}$ is just an arrow of $\mathcal{R}(\mathscr{C})$. That is, the *set* of proofs is the closure of $\mathcal{R}$ under the operations $\fatsemi$, $\otimes$, and $\fatsemi$.

Proof Checking The main contribution of this proposal is implementing a data-parallel algorithm for verifying that an externally supplied proof (provided eg. by proof-search) is valid. Inputs to our tool will be any candidate proof constructed with the three operations $\otimes$, $\fatsemi$, and $\fatsemi$ discussed above. Note that these are partial operations, meaning that inputs may not type-check correctly (both at the level of string diagrams and at the level of rewrites). A *valid* proof is simply a well-defined morphism of $\mathcal{R}(\mathscr{C})$. In a nutshell, our tool will inspect the algebraic structure of candidate morphisms, as given by the three operations $\otimes$, $\fatsemi$, and $\fatsemi$, and check that typing is correct.

There is a natural approach to this question: first, for each $\rho \fatsemi \sigma$ with $\rho : f_0 \to f_1$ and $\sigma : g_0 \to g_1$ verify that $f_1 = g_0$. Second, for each $\rho \fatsemi \sigma$, ensure that $\mathsf{tgt}(f_0) = \mathsf{src}(g_0)$ and $\mathsf{tgt}(f_1) = \mathsf{src}(g_1)$. This process is depicted in the example below, which represents the expression $(\mathsf{id} \fatsemi \alpha) \fatsemi \beta$. The red dashed line denotes that the two identity 1-cells must be checked for equality for the $\fatsemi$ operation to be valid, while the green dashed line shows which wires are to be identified to compute the resulting 1-cell. The key ingredient to our implementation is that this approach can be made efficient by building on the data-parallel 'fast writing' algorithm of [21, Section 8]. The essential idea is to represent all 1-cells (blue boxes) within the same datastructure, computing intermediates using this 'fast wiring' algorithm. Equality checking of source/target maps (red dashed lines) can then be regarded as an instance of (monogamous acyclic) hypergraph isomorphism on a subset of the datastructure, [2]



Integration with Proof Search We identify three main benefits of our approach to proof search. First, the representation of a single proof also works as the state of a proof *search*. Consider for example a proof ρ (pictured right) with m hypotheses $h_1 \dots h_m$ and n conclusions $c_1 \dots c_n$.Each conclusion c_i can be thought of as a potential point on a 'path' to some goal term g; the search state being the set of points on the 'frontier'. A proof search state can therefore also use the same data-parallel representation as a proof.



Secondly, by augmenting the category of proofs $\mathcal{R}(\mathscr{C})$ with *cartesian* structure, sub-proofs can share intermediate terms. Continuing the example above, we might continue the search by using conclusions c_i and c_j with some axiom to get a new conclusion d_i. Using cartesian structure, we can *share* the intermediate conclusions c_i and c_j with some other new conclusion d_j.

Thirdly, the 'categorical viewpoint' of proofs as rewrites in an SMC means that proofs can easily be 'ported' between different theories enjoying a functorial relationship. This reduces the search space by restricting proofs to a 'domain specific' theory which can then be used within a more general theory.

1.2 Motivation and Differentiation from Related Work

While a number of theorem provers exist in various states of development, to our knowledge none satisfy all three requirements highlighted at the beginning of Section 1. Moreover, none at all have a proof representation designed specifically for data-parallel computing. We now provide a more detailed comparison.

[2] While graph isomorphism is in complexity class GI, the condition of monogamous acyclicity [1] means that that our hypergraph representation has bounded valence, and the isomorphism problem is therefore likely in the class P. See e.g. [15] for a result on graphs with bounded valence.

Mainstream proof assistants such as LEAN [9] and Agda [19]. These are not typically designed for use as kernels (i.e. components of a larger system) but instead as a user-facing tool in their own right. Moreover, their proof backends are are not designed for data-parallelism, and they do not focus on symmetric monoidal theories specifically. Instead, these typically choose a particular logic with which to work. (As an aside, note that our proposal has similarities with the method of semantic tableaux— the PAPERPROOF project [10] provides a similar conceptual interface on top of LEAN. However, such project does not address the desiderata above.)

Reproducible proof kernels such as Metamath Zero (mm0) [7] provide a reproducible/serializable proof representation, and focus on performance. mm0 in particular shows a 40x speedup over metamath on a library of 30,000 proofs. However, mm0 is not data-parallel, and does not focus on symmetric monoidal theories. Despite this, our approach draws much inspiration from the mm0 project. In particular, the aspects of a minimal foundations based on substitution as well as reproducible and serializable proofs.

Category-theoretic approaches based on hypergraph rewriting [1, 2, 3] like Chyp [11] are perhaps closest to this proposal. However, proofs in Chyp are not directly represented; a 'proof' is actually the *computation* of running a sequence of tactics. This means that proofs are not reproducible: when tactic implementations change, a proof may no longer be valid. This latter point is exemplified by the following quote from the Metamath Zero project: "We embrace the difference between fully explicit proofs that are verified by a trusted verifier, and proof scripts that are used by front-end tools (..) to generate proofs" [7].

1.3 Why ARIA

Both the PIs have participated regularly in the workshops of ARIA's Safeguarded AI programme, with Zanasi also being one of the TA1.1 creators. Our proposal is informed by the feedback gathered at these events, and our vision closely aligns with the aims of the programme. No other funding agency comes close to this level of affinity. In particular, this call represents a unique opportunity: academic research in theoretical computer science, overly focused on publications, hardly provides the resources to deploy a fully-fledged implementation. Conversely, software development often privileges short-term goals above principled design. This project would finally provide us sufficient resources to develop a production-grade implementations, while at the same time maintaining strong ties with the TA1 programme.

1.4 Interaction with Other Creators

Zanasi is a TA1.1 creator, and both PIs have regular collaborations with other TA1.1 creators, as witnessed by coauthored publications, joint projects, and mutual research visits. Particularly relevant to our proposal are the TA1.1 approaches currently being developed at University College London, University of Pisa, Tallinn University of Technology, and University of Oxford. These provide the first motivating examples to be integrated in our software, and serve as acceptance criteria for our implementation: theories and proofs within them must be representable and checkable. During the project we will ensure regular interaction via attendance of ARIA's workshops, participation in international conferences and workshops in relevant research areas, and research visits.

2 Team and Project Management

2.1 Project Team

Our proposal requires a mix of very different expertises, ranging from categorical algebra to GPU optimisation. We carefully assembled a team able to fulfil any gaps in relevant competency. We now introduce the team members.

- **Prof. Fabio Zanasi** (co-PI, 20% FTE) is a Professor of Computer Science at University College London, who has made several contributions to categorical semantics, including the framework for string diagram rewriting [1, 2, 3] at the core of this proposal. The theories designed in the context of his TA1.1 project provide a key test case for the proposed implementation. He has extensive experience in leadership, project management, and supervision. In addition, he has an established academic network of collaborators, including other TA1 creators. He will be responsible for overall project management and long-term vision, as

well as training and development of team members for what concerns the mathematical background of the project.

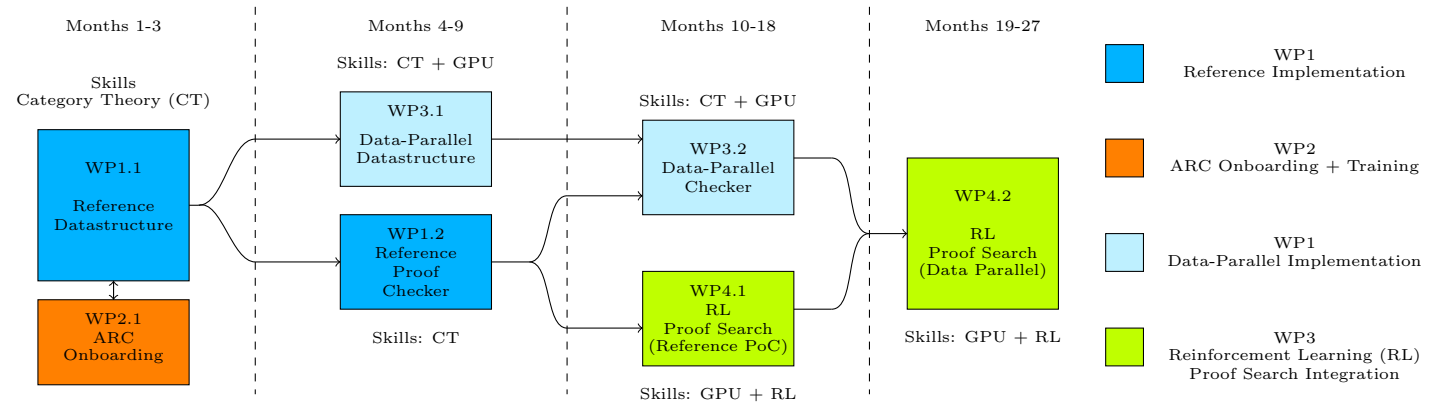
- **Dr. Paul Wilson** (co-PI, 40% FTE) has expertise both on the theoretical and applied aspects of the proposal. Together with Zanasi, he coauthored the work [21] providing the data parallel data structures and algorithms at the core of this project. Also, they previously collaborated on CARTOGRAPHER [18], a prototypical proof assistant for monoidal theories, precursor of Chyp [11]. Wilson also has experience in high performance computing, having developed the deep learning compiler CATGRAD [20]. He will manage the day-to-day coordination of the software development team and contribute himself to the prototype implementation.
- **Research Software Engineers** (150% FTE) The *Centre for Advanced Research Computing* (ARC) at University College London has granted availability of two to three Research Software Engineers (RSEs) over the course of the project (each working between 50% and 100% FTE), covering three key expertises: GPU implementation, proof assistants, and reinforcement learning. The RSEs will form the core team responsible of designing and implementing the software of the project. They will also contribute to 'quality of life' features such as documentation, supporting tools, and performance testing and improvements. Collaborating with ARC also gives us access to a large amount of GPU compute resources to test the project software. As ARC is part of UCL, staff and resources will be available from the get-go, ensuring a smooth start of the project.
- **Dr. Mario Carneiro** (Chalmers University of Technology), agreed to act on an advisory role. His expertise is highly relevant on the project. He is an expert in theorem provers who designed and built the Metamath Zero [7] system, and a major contributor to the LEAN [9] project.

2.2 Brief Profile of the PIs

Fabio Zanasi is Professor of Computer Science at UCL. His research develops compositional methods for programming languages, machine learning, and graphical models of computations (eg. circuits, networks, flow graphs), using category theory as a unifying mathematical perspective. He has over 80 peer-reviewed publications, including two monographs (Cambridge University Press), and papers appearing in the *Journal of the ACM*, *ACM TOCL*, POPL (×2), LICS (×5), CONCUR (×4), and ETAPS conferences (×9). He received a EATCS Best Paper Award, a CALCO Best Paper Award, and a CALCO Best Presentation Award. He has been multiple times invited speaker at international conferences and workshops, and served as programme committee chair, workshop organiser, steering committee member, Executive Journal editor, and Special Issue editor. He has led as PI in 4 different research projects in the last 8 years, attracting over £1m in funding. He has supervised 5 postdocs and 9 PhD students.

Paul Wilson holds a Ph.D in computer science from the University of Southampton. He is co-founder of Hellas.AI, a venture-backed startup using categorical machine learning to build a two-sided marketplace for AI compute. Previously, he worked as an independent researcher funded by the Ethereum Foundation for categorical approaches to Zero-Knowledge ML, and has more than 10 years experience in software engineering, data science, and machine learning research.

2.3 Timeline



References

[1] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski, and Fabio Zanasi. String diagram rewrite theory i: Rewriting with frobenius structure, 2020.

[2] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski, and Fabio Zanasi. String diagram rewrite theory ii: Rewriting with symmetric monoidal structure, 2021.

[3] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. String diagram rewrite theory iii: Confluence with and without frobenius, 2021.

[4] Filippo Bonchi, Alessandro Di Giorgio, Nathan Haydon, and Pawel Sobocinski. Diagrammatic algebra of first order logic. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706608. doi: 10.1145/3661814.3662078. URL https://doi.org/10.1145/3661814.3662078.

[5] Filippo Bonchi, Cipriano Junior Cioffo, Alessandro Di Giorgio, and Elena Di Lavore. Tape diagrams for monoidal monads, 2025. URL https://arxiv.org/abs/2503.22819.

[6] Albert Burroni. Higher-dimensional word problems with applications to equational logic. *Theoretical Computer Science*, 115(1):43–62, 1993. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(93)90054-W. URL https://www.sciencedirect.com/science/article/pii/030439759390054W.

[7] Mario Carneiro. Metamath zero. URL https://github.com/digama0/mm0.

[8] Andrea Corradini and Fabio Gadducci. Categorical rewriting of term-like structures1 1research partially supported by the ec tmr network getgrats (general theory of graph transformation systems), by the esprit working group appligraph (applications of graph transformation), and by the italian murst project tosca (teoria della concorrenza, linguaggi di ordine superiore e strutture di tipi). *Electronic Notes in Theoretical Computer Science*, 51:108–121, 2002. ISSN 1571-0661. doi: https://doi.org/10.1016/S1571-0661(04)80195-6. URL https://www.sciencedirect.com/science/article/pii/S1571066104801956. GETGRATS Closing Workshop.

[9] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25*, pages 378–388, Cham, 2015. Springer International Publishing. ISBN 978-3-319-21401-6.

[10] Evgenia Karunus et al. Paperproof. URL https://github.com/Paper-Proof/paperproof.

[11] Aleks Kissinger. Chyp: Composing hypergraphs, proving theorems. URL https://github.com/akissinger/chyp.

[12] Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. Hypertree proof search for neural theorem proving, 2022. URL https://arxiv.org/abs/2205.11491.

[13] Elena Di Lavore, Mario Roman, and Pawel Sobocinski. Partial markov categories, 2025. URL https://arxiv.org/abs/2502.03477.

[14] Gabriele Lobbia, Wojciech Rozowski, Ralph Sarkis, and Fabio Zanasi. Quantitative monoidal algebra: Axiomatising distance with string diagrams. *CoRR*, abs/2410.09229, 2024. doi: 10.48550/ARXIV.2410.09229. URL https://doi.org/10.48550/arXiv.2410.09229.

[15] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 42–49, 1980. doi: 10.1109/SFCS.1980.24.

[16] Robin Piedeleu and Fabio Zanasi. An introduction to string diagrams for computer scientists, 2023. URL https://arxiv.org/abs/2305.08768.

[17] Robin Piedeleu, Mateo Torres Ruiz, Alexandra Silva, and Fabio Zanasi. A complete axiomatisation of equivalence for discrete probabilistic programming. In *ESOP*, To appear, 2025.

[18] Paweł Sobociński, Paul W. Wilson, and Fabio Zanasi. CARTOGRAPHER: A Tool for String Diagrammatic Reasoning. In Markus Roggenbach and Ana Sokolova, editors, *8th Conference on Algebra and Coalgebra in Computer Science (CALCO 2019)*, volume 139 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:7, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-120-7. doi: 10.4230/LIPIcs.CALCO.2019.20. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CALCO.2019.20.

[19] Agda Development Team. Agda, 2007–2024. https://wiki.portal.chalmers.se/agda/.

[20] Paul Wilson. Catgrad. URL https://catgrad.com.

[21] Paul Wilson and Fabio Zanasi. Data-parallel algorithms for string diagrams, 2023. URL https://arxiv.org/abs/2305.01041.

[22] Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search, 2024. URL https://arxiv.org/abs/2408.08152.