Learning Structure-Aware Representations of Dependent Types – Extended Abstract

Konstantinos Kogkalidis[1], Orestis Melkonian[2], and Jean-Philippe Bernardy[3]

[1] Aalto University, Finland
[2] Input Output, Global
[3] University of Gothenburg and Chalmers University of Technology, Sweden

1 Introduction

In this paper, we describe a structure-driven *neural premise selection* setup for a dependent type theory, and present an implementation for Agda. In a premise selection setup, the statistical learner is presented with a goal statement and a collection of theorems or lemmas, some of which are potentially relevant for the proof process, whereas others are not. A statistical model is then tasked with selecting which of those lemmas are relevant.

Modeling strategies can be broadly grouped into two general categories. Historically, the dominant approach has been sequence-based encoder and/or decoder architectures, operating on the user-facing representation of formal objects (theorems, formulas, lemmas, *etc.*) – more recently, this approach has been further empowered by its direct compatibility with large language models. The alternative approach has sought to explicate the relations that exist within and between these objects, *e.g.,* with tree and graph architectures.

In this paper, we seek to address two key limitations in the existing literature.

- **Machine Learning for Agda** While a plurality of datasets, models and tools are available for a handful of languages, chiefly HOL [1, 6], Coq [4] and Lean [16], none exist for Agda. We believe that this state of affairs is an effect of historical momentum and methodological precedence, rather than inherent merit. Agda spearheads developments in constructive type theory with an array of innovative features, gaining adoption and a rapidly maturing ecosystem. Aiming to aid Agda with meeting and expanding her potential, we implemented an algorithm, AGDA2TRAIN, that extracts intermediate compilation steps, and produces them in the form of human-readable partial proof states. Applying AGDA2TRAIN on established high-quality libraries, we obtain and release an elaborate and extensive dataset of Agda program-proofs, at a scale and form that can support various machine learning applications – the first of its kind.
- **Modeling Type Structure** We also note a significant gap between the rigor of the structures modeled and the lenience of the architectures employed in prior work [3, 5, 10, 12–14, 17]. While there are efforts to reflect certain aspects of type structure in models, [2, 9, 11, 15] our aim is to develop a learning scheme which *faithfully represents of expressions involving dependent types*. We apply our methodology on the extracted dataset to produce QUILL: a proof-of-concept neural guidance tool for Agda. Beyond its current use case, the system is *universal*, in the sense of being applicable to any language that uses type theory as its foundational core.

2 Data

Data is extracted from source Agda programs in a way that is mindful of type structure. We allow Agda to infer and type-check program-proofs, and then consult her for their internal representations, which we store mostly unchanged. From then, we consider every possible subterm, and extract pairs of: (1) the typing context of the subterm (hereafter a *hole*) and (2) lemmas used by the user to construct the subterm. The model can then be trained to predict (2) from (1). We obtain competitive results for this task.

3 Model

Our key motivation is generalizability; *i.e.,* direct applicability to new data with minimal performance degradation. One can view generalizability as the aggregate of *completeness* and appropriate *inductive biases*. Completeness entails being able to model any new data point in the domain, regardless of possible distributional shifts, whereas the incorporation of suitable inductive biases is crucial for recognizing and effectively generalizing from the structural patterns in the training data.

The commonly employed sequential approaches attain (some form of) completeness by tokenizing expressions in the domain according to the co-occurrence frequencies of their constituent substrings. In doing so, however, they provide the model with the *wrong* inductive biases. The grammar underlying formal objects is not *just any* string grammar, but a precise inductive system. String-formatted representations obfuscate this structure.

Furthermore, Agda types are *dependent*, which means that their abstract syntax trees may often contain references to other types. These references cannot be considered as strings either. Indeed, shifts in naming conventions, use of libraries that are stylistically distant or targeted at novel domains, *etc.* might necessitate the evocation of symbols not present in a trainable embedding table, collapsing representations into meaningless generics (*e.g.,* multiple different entries might be made the same due to an excess of `[UNK]` tokens). A non-nominal resolution of these complex referencing patterns necessitates an extension of the traditional AST-encoding approaches. Crucially, we need to able to structurally refer to both locally defined variables (intra-AST) as well as other scope entries in their entirety (inter-AST).

With the above in mind, we design and implement a system utilizing:

- **Structured Attention** We employ a tailor-made attention scheme that alters attention coefficients by adjusting symbol representations (content) according to their relative positions (structure), where positions are defined in the context of an underlying AST [7]. This allows us to make use of fully-attentive models of sequential computation without foregoing structural discipline or bottlenecking vectorial computation.
- **Nameless Representations of References and Variables** A standard name-agnostic representation for variables are de Bruijn indices Nonetheless, these are not effective for learning purposes. Indeed, a single index carries no "meaning" of its own, other than as an address-referencing instruction. To facilitate learning, we take the extra step of actually resolving the indexing instruction by using a pointer to the node introducing the referenced variable, represented as a positional reference. Furthermore, iteratively representing definitions following along their topological sort, we get access to a dynamic collection of in-context representations without having to use or peak at their names.

The resulting system is truly structure-faithful and name-free. Initial experiments conducted on the major Agda libraries achieve competitive results with a fraction of the parameter count and training cost of modern approaches. For further details, we refer the interested reader to our full paper [8].

References

[1] Kshitij Bansal, Sarah Loos, Markus Rabe, Christian Szegedy, and Stewart Wilcox. HOList: An environment for machine learning of higher order logic theorem proving. In *International Conference on Machine Learning*, pages 454–463. PMLR, 2019.

[2] Lasse Blaauwbroek, Miroslav Olšák, Jason Rute, Fidel Ivan Schaposnik Massolo, Jelle Piepenbrock, and Vasily Pestun. Graph2tac: Online representation learning of formal math concepts. 2024. URL https://arxiv.org/abs/2401.02949.

[3] Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward Ayers, and Stanislas Polu. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations*, 2021.

[4] Daniel Huang, Prafulla Dhariwal, Dawn Song, and Ilya Sutskever. GamePad: A learning environment for theorem proving. In *International Conference on Learning Representations*, 2018.

[5] Albert Qiaochu Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. Lisa: Language models of Isabelle proofs. In *6th Conference on Artificial Intelligence and Theorem Proving*, pages 378–392, 2021.

[6] Cezary Kaliszyk, François Chollet, and Christian Szegedy. HolStep: A machine learning dataset for higher-order logic theorem proving. In *International Conference on Learning Representations*, 2016.

[7] Konstantinos Kogkalidis, Jean-Philippe Bernardy, and Vikas Garg. Algebraic positional encodings. *Advances in Neural Information Processing Systems*, 37, 2024.

[8] Konstantinos Kogkalidis, Orestis Melkonian, and Jean-Philippe Bernardy. Learning structure-aware representations of dependent types. *Advances in Neural Information Processing Systems*, 37, 2024.

[9] Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence C Paulson. IsarStep: a benchmark for high-level mathematical reasoning. In *International Conference on Learning Representations*, 2020.

[10] Maciej Mikuła, Szymon Antoniak, Szymon Tworkowski, Albert Qiaochu Jiang, Jin Peng Zhou, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. Magnushammer: A transformer-based approach to premise selection. *arXiv preprint arXiv:2303.04488*, 2023.

[11] Aditya Paliwal, Sarah Loos, Markus Rabe, Kshitij Bansal, and Christian Szegedy. Graph representations for higher-order logic and theorem proving. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2967–2974, Apr. 2020. doi: 10.1609/aaai.v34i03.5689. URL https://ojs.aaai.org/index.php/AAAI/article/view/5689.

[12] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

[13] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-P7G-8dmSh4.

[14] Josef Urban and Jan Jakubův. First neural conjecturing datasets and experiments. In *Intelligent Computer Mathematics: 13th International Conference, CICM 2020, Bertinoro, Italy, July 26–31, 2020, Proceedings 13*, pages 315–323. Springer, 2020.

[15] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. *Advances in neural information processing systems*, 30, 2017.

[16] Kaiyu Yang and Jia Deng. Learning to prove theorems via interacting with proof assistants. In *International Conference on Machine Learning*, pages 6984–6994. PMLR, 2019.

[17] Kaiyu Yang, Aidan M Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.