HOLSTEP: A MACHINE LEARNING DATASET FOR HIGHER-ORDER LOGIC THEOREM PROVING

Cezary Kaliszyk
University of Innsbruck
cezary.kaliszyk@uibk.ac.at

François Chollet, Christian Szegedy
Google Research
{fchollet,szegedy}@google.com

ABSTRACT

Large computer-understandable proofs consist of millions of intermediate logical steps. The vast majority of such steps originate from manually selected and manually guided heuristics applied to intermediate goals. So far, machine learning has generally not been used to filter or generate these steps. In this paper, we introduce a new dataset based on Higher-Order Logic (HOL) proofs, for the purpose of developing new machine learning-based theorem-proving strategies. We make this dataset publicly available under the BSD license. We propose various machine learning tasks that can be performed on this dataset, and discuss their significance for theorem proving. We also benchmark a set of baseline deep learning models suited for the tasks (including convolutional neural networks and recurrent neural networks). The results of our baseline models shows the promise of applying deep learning to HOL theorem proving.

1 INTRODUCTION

As the usability of interactive theorem proving (ITP) systems (Harrison et al., 2014) grows, its use becomes a more common way of establishing the correctness of software as well as mathematical proofs. ITPs are used today for software certification projects ranging from compilers (Leroy, 2009) and operating system components (Chen et al., 2016; Klein et al., 2014), to establishing the absolute correctness of large proofs in mathematics such as the Kepler conjecture (Hales et al., 2015) and the Feit-Thomson Theorem (Gonthier et al., 2013).

For results of such significance to be possible, the theorem libraries of the ITPs must contain all necessary basic mathematical properties, accompanied with formal proofs. This means that the size of many ITP libraries can be measured in dozens of thousands of theorems (Grabowski et al., 2010; Blanchette et al., 2015) and billions of individual proof steps. While the general direction of the proofs is specified by humans (by providing the goal to prove, specifying intermediate steps, or applying certain automated tactics), the majority of such proof steps are actually found by automated reasoning-based proof search (Kaliszyk & Urban, 2015), with very little application of machine learning techniques so far.

At the same time, fast progress is achieved in machine learning applied to tasks that involve logical inference, such as natural question answering (Sukhbaatar et al., 2015), knowledge base completion (Socher et al., 2013), automated translation (Wu et al., 2016), and premise selection in the context of theorem proving (Alemi et al., 2016). Deep learning in particular has proven a powerful tool for embedding semantic meaning and logical relationships into geometric spaces, specifically via models such as convolutional neural networks, recurrent neural networks, and tree-recursive neural networks. These advances strongly suggest that deep learning may have become mature enough to yield significant advances in automated theorem proving. Remarkably, it has recently become possible to build a system, AlphaGo (Silver et al., 2016), blending classical AI techniques such as Monte-Carlo tree search and modern deep learning techniques, capable of playing the game of Go at super-human levels. We should note that theorem proving and Go playing are conceptually related, since both consist in searching for specific nodes in trees of states with extremely large arity and relatively large depth, which involves node evaluation decision (how valuable is this state?) and policy decisions (which node should be expanded next?). The success of AlphaGo can thus serve as encouragement on the road to building deep learning-augmented theorem provers that would blend classical techniques developed over the past few decades with the latest machine learning advances.

Fast progress in specific machine learning verticals has occasionally been achieved thanks to the release of specialized datasets (often with associated competitions, e.g. the ImageNet dataset for large-scale image classification (Deng et al., 2009)) serving as an experimental testbed and public benchmark of current progress, thus focusing the efforts of the research community. We hope that releasing a theorem proving dataset suited for specific machine learning tasks can serve the same purpose in the vertical of applying machine learning to theorem proving.

1.1 Contribution and Overview

First, we develop a dataset for machine learning based on the proof steps used in a large interactive proof section 2. We focus on the HOL Light (Harrison, 2009) ITP, its multivariate analysis library (Harrison, 2013), as well as the formal proof of the Kepler conjecture (Hales et al., 2010). These formalizations constitute a diverse proof dataset containing basic mathematics, analysis, trigonometry, as well as reasoning about data structures such as graphs. Furthermore these formal proof developments have been used as benchmarks for automated reasoning techniques (Kaliszyk & Urban, 2014).

The dataset consists of 2,013,046 training examples and 196,030 testing examples that originate from 11,400 proofs. Precisely half of the examples are statements that were useful in the currently proven conjectures and half are steps that have been derived either manually or as part of the automated proof search but were not necessary in the final proofs. The dataset contains only proofs of non-trivial theorems, that also do not focus on computation but rather on actual theorem proving. For each proof, the conjecture that is being proven as well as its dependencies (axioms) and may be exploited in machine learning tasks. Furthermore, for each statement both its human-readable (pretty-printed) statement and a tokenization designed to make machine learning tasks more manageable are included.

Next, in section 3 we discuss the proof step classification tasks that can be attempted using the dataset, and we discuss the usefulness of these tasks in interactive and automated theorem proving. These tasks includes unconditioned classification (without access to conjectures and dependencies) and conjecture-conditioned classification (with access to the conjecture) of proof steps as being useful or not in a proof. We outline the use of such classification capabilities for search space pruning and internal guidance, as well as for generation of intermediate steps or possible new lemma statements.

Finally, in section 4 we propose three baseline models for the proof step classification tasks and experimentally evaluate the models on the data in section 5. The models considered include both a relatively simple regression model, as well as models based on convolutional and recurrent neural networks.

1.2 Related Work

The use of machine learning in interactive and automated theorem proving has so far focused on three main tasks: premise selection, strategy selection, and internal guidance. We shortly explain these tasks.

Given a large library of proven facts and a user given conjecture to prove, the multi-label classification problem of selecting the facts that are most likely to lead to a successful proof of the conjecture has been usually called *relevance filtering* or *premise selection* (Alama et al., 2014). This task is crucial for the efficiency of the state-of-the-art automation techniques for ITPs (Blanchette et al., 2016). Similarly most competitive automated theorem provers today (Sutcliffe, 2016) implement the SInE classifier (Hoder & Voronkov, 2011).

A second theorem proving task where machine learning has been of importance is *strategy selection*. With the development of automated theorem provers came many parameters that control their execution. In fact, modern ATPs, such as E (Schulz, 2013) and Vampire (Kovács & Voronkov, 2013), include complete strategy description languages that allow a user to specify the orderings, weighting functions, literal selection strategies, etc. Rather than optimizing the search strategy globally, one can choose the strategy based on the currently considered problem. For this some frameworks use machine learning (Bridge et al., 2014; Kühlwein & Urban, 2015).

Finally, an automated theorem prover may use machine learning for choosing the actual inference steps. It has been shown to significantly reduce the proof search in first-order tableaux by the selection of extension steps to use (Urban et al., 2011), and has been also successfully applied in monomorphic higher-order logic proving (Färber & Brown, 2016). Data/proof mining has also been applied on the level of interactive theorem proving tactics (Duncan, 2007) to extract and reuse repeating patterns.

2 Dataset Extraction

We focus on the HOL Light theorem prover for two reasons. First, it is an LCF style interactive theorem prover. In the LCF style more complicated inferences are reduced to the most primitive ones, all of which go through the small trusted core, called the *kernel*. This means that many modifications can be restricted to this small core, and it is relatively easy to extract proof steps and an arbitrary selected level of granularity. Second, it implements higher-order logic as its foundation, which on the one hand is powerful enough to encode most of today's formal proofs, and on the other hand allows for an easy integration of many powerful automation mechanisms (Baader & Nipkow, 1998; Paulson, 1999).

When selecting the theorems to record, we choose an intermediate approach between HOL Light ProofRecording (Obua & Skalberg, 2006) and the HOL/Import one (Kaliszyk & Krauss, 2013). The top-level values that are of type theorem based on standard proof functions are patched to record their names and remaining values are extracted from the underlying OCaml toplevel. Additionally, the intermediate theorem values that are used across proof blocks are considered theorem boundaries, avoiding any reused unrelated subproofs.

All kernel-level inferences are recorded together with the necessary arguments in a trace file. The trace is processed offline to extract the facts dependencies, detect used proof boundaries, mark the used and unused steps, and mark the training and testing examples. Only proofs that have sufficiently many used and unused steps are considered useful for the dataset. The proof trace with additional annotations is processed again by a HOL kernel, this time saving the actual training and testing examples. Training and testing examples are grouped by proof: for each proof the conjecture (statement that is finally proved) and the dependencies of the theorem are constant, and a list of used and not used intermediate statements is provided.

For each statement, whether it is the conjecture, a proof dependency, or an intermediate statement, both a fully parenthesised HOL Light human-like printout is provided, as well as a predefined tokenization. The standard HOL Light printer uses parentheses and operator priorities to make its notations somewhat similar to textbook-style mathematics, while at the same time preserving the complete unambiguity of the order of applications (this is particularly visible for associative operators). The tokenization that we propose attempts to reduce the number of parentheses. To do this we compute the maximum number of arguments that each symbol needs to be applied to, and only mark partial application. This means that fully applied functions (more than 90% of the applications) do not require neither application operators nor parentheses. Toplevel universal quantifications are eliminated, bound variables are represented by their de Bruijn indices and free variables are renamed canonically. Since the Hindley-Milner type inference mechanisms will be sufficient to reconstruct the most-general types of the expressions well enough for automated-reasoning techniques Kaliszyk et al. (2015) we erase all type information. Table 1 presents some dataset statistics.

The dataset together with the description of the used format is available from:

http://cl-informatik.uibk.ac.at/cek/holstep/

3 Machine Learning Tasks

3.1 Tasks description

This dataset makes possible several tasks well-suited for machine learning most of which are highly relevant for theorem proving:

	Train	Test	Positive	Negative
Examples	2013046	196030	1104538	1104538
Avg. length	503.18	440.20	535.52	459.66
Avg. tokens	87.01	80.62	95.48	77.40
Conjectures	9999	1411	-	-
Avg. dependencies	29.58	22.82	-	-

Table 1: HolStep dataset statistics

- Predicting whether a statement is useful in the proof of a given conjecture;
- Predicting the dependencies of a proof statement (premise selection);
- Predicting whether a statement is an important one (human named);
- Predicting which conjecture a particular intermediate statement originates from;
- Predicting the name given to a statement;
- Generating intermediate statements useful in the proof of a given conjecture;
- Generating the conjecture the current proof will lead to.

In what follows we focus on the first task: classifying proof step statements as being useful or not in the context of a given proof. This task may be further specialized into two different tasks:

- Unconditioned classification of proof steps: determining how likely a given proof is to be useful for the proof it occurred in, based solely on the content of statement (i.e. by only providing the model with the step statement itself, absent any context).
- Conditioned classification of proof steps: determining how likely a given proof is to be useful for the proof it occurred in, with “conditioning” on the conjecture statement that the proof was aiming to attain, i.e. by providing the model with both the step statement and the conjecture statement).

In the dataset, for every proof we provide the same number of useful and non-useful steps. As such, the proof step classification problem is a balanced two-class classification problem, where a random baseline would yield an accuracy of 0.5.

3.2 Relevance to interactive and automated theorem proving

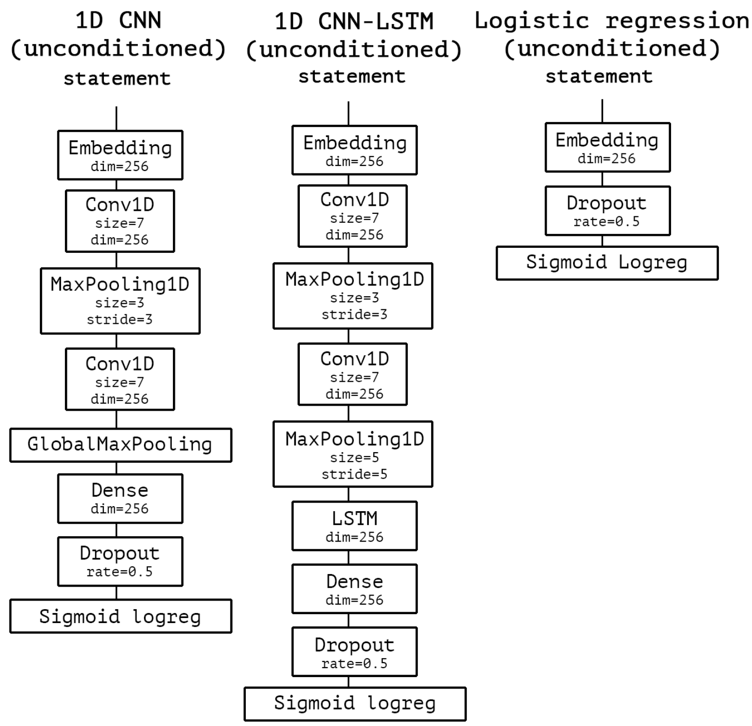
In the interaction with an interactive theorem prover, the tasks that require most human time are: the search for good intermediate steps; the search for automation techniques able to justify the individual steps, and the searching of theorem proving libraries for the necessary simpler facts. These three problems directly correspond to the machine learning tasks proposed in the previous subsection. Being able to predict the usefulness of a statement will significantly improve many automation techniques. The generation of good intermediate lemmas or intermediate steps can improve level of granularity of the proof steps. Understanding the correspondence between statements and their names can allow users to search for statements in the libraries more efficiently (Aspinall & Kaliszyk, 2016). Premise selection and filtering are already used in many theorem proving systems, and generation of succeeding steps corresponds to conjecturing and theory exploration.

4 Baseline Models

For each task (conditioned and unconditioned classification), we propose three different deep learning architectures, meant to provide a baseline for the classification performance that can be achived on this dataset. Our models cover a range of architecture features (from convolutional networks to recurrent networks), aiming at probing what characteristics of the data are the most helpful for usefulness classification.

Our models are implemented in TensorFlow (Abadi et al., 2015) using the Keras framework (Chollet, 2015). Each model was trained on a single Nvidia K80 GPU. Training only takes a few hours per

Figure 1: Unconditioned classification model architectures.



model, which makes running these experiments accessible to most people (they could even be run on a laptop CPU). We are releasing all of our benchmark code as open-source software so as to allow others to reproduce our results and improve upon our models.

4.1 UNCONDITIONED CLASSIFICATION MODELS

Our three models for this task are as follow:

- Logistic regression on top of learned token embeddings. This minimal model aims to determine to which extent simple differences between token distribution between useful and non-useful statements can be used to distinguish them. It provides an absolute floor on the performance achievable on this task.
- 2-layer 1D convolutional neural network (CNN) with global maxpooling for sequence reduction. This model aims to determine the importance of local patterns of tokens.
- 2-layer 1D CNN with LSTM (Hochreiter & Schmidhuber, 1997) sequence reduction. This model aims to determine the importance of order in the features sequences.

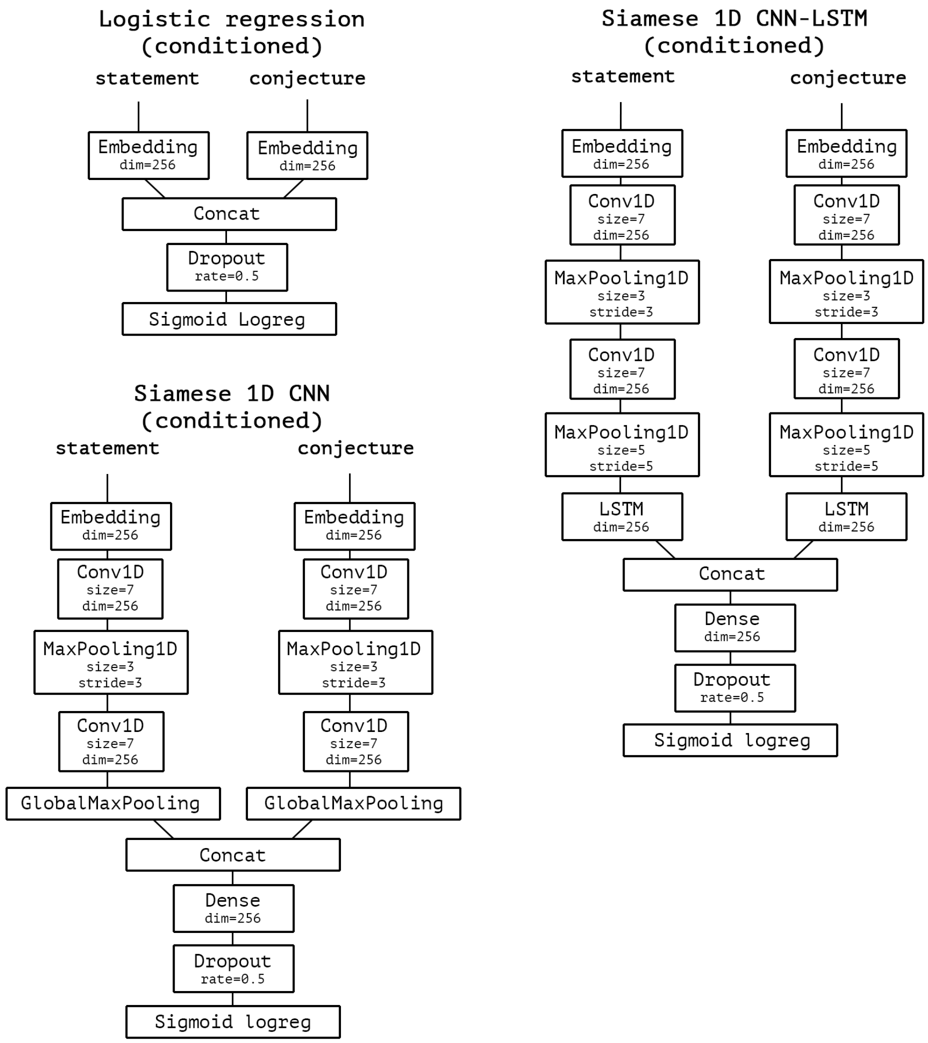
See figure 1 for a layer-by-layer description of these models.

4.2 CONDITIONED CLASSIFICATION MODELS

For this task, we use versions of the above models that have two siamese branches (identical branches with shared weights), with one branch processing the proof step statement being considered, and the other branch processing the conjecture. Each branch outputs an embedding; these two embeddings (step embedding and conjecture embedding) are then concatenated and the classified by a fully-connected network.

See figure 2 for a layer-by-layer description of these models.

Figure 2: Conditioned classification model architectures.



4.3 INPUT STATEMENTS ENCODING

It should be noted that all of our models start with an Embedding layer, mapping tokens or characters in the statements to dense vectors in a low-dimensional space. We consider two possible encodings for presenting the input statements (proof steps and conjectures) to the Embedding layers of our models:

- Character-level encoding of the human-readable versions of the statements, where each character (out of a set of 83 unique characters) in the pretty-printed statements is mapped to a 256-dimensional dense vector. This encoding yields longer statements (training statements are 347 character long on average).
- Token-level encoding of the versions of the statements rendered with our proposed high-level tokenization scheme. This encoding yields shorter statements (training statements are 51 token long on average), while considerably increasing the size of set of unique tokens (approximately 100,000 total tokens in the training set).

Table 2: HolStep proof step classification accuracy without conditioning

	Logistic regression	1D CNN	1D CNN-LSTM
Accuracy with char input	0.71	0.82	**0.83**
Accuracy with token input	0.71	0.82	0.76

Table 3: HolStep proof step classification accuracy with conditioning

	Logistic regression	Siamese 1D CNN	Siamese 1D CNN-LSTM
Accuracy with char input	0.71	0.81	**0.83**
Accuracy with token input	0.71	0.82	0.76

5 RESULTS

Experimental results are presented in tables 2 and 3 as well as figs. 3a, 3b, 4a and 4b.

5.1 INFLUENCE OF MODEL ARCHITECTURE

Our unconditioned logistic regression model yields an accuracy of 71%, both with character encoding and token encoding (tables 2 and 3). This demonstrates that differences in token or character distributions between useful and non-useful steps alone, absent any context, is sufficient for discriminating between useful and non-useful statements to a reasonable extent. This also demonstrates that the token encoding is not fundamentally more informative than raw character-level statements.

Additionally, our unconditioned 1D CNN model yields an accuracy of 83% to 84%, both with character encoding and token encoding (tables 2 and 3). This demonstrates that patterns of characters or patterns of tokens are considerably more informative than single tokens for the purpose of usefulness classification.

Finally, our unconditioned convolutional-recurrent model does not improve upon the results of the 1D CNN, which indicates that our models are not able to meaningfully leverage order in the feature sequences into which the statements are encoded.

5.2 INFLUENCE OF INPUT ENCODING

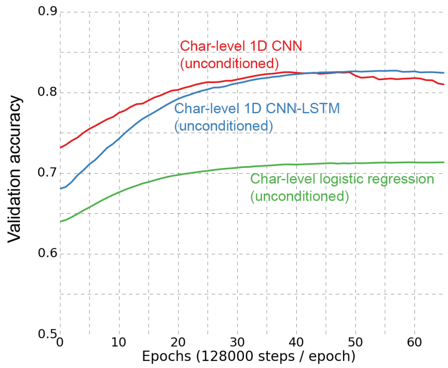
For the logistic regression model and the 2-layer 1D CNN model, the choice of input encoding seems to have little impact. For the convolutional-recurrent model, the use of the high-level tokenization seems to cause a large decrease in model performance (figs. 3b and 4b). This may be due to the fact that token encoding yields shorter sequences, making the use of a LSTM less relevant.

5.3 INFLUENCE OF CONDITIONING ON THE CONJECTURE

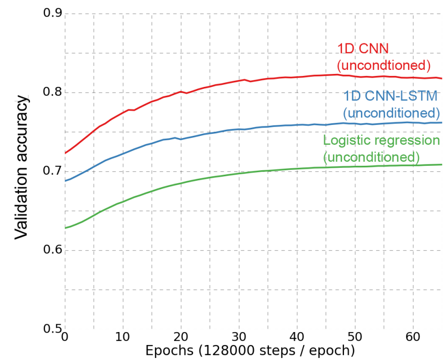
None of our conditioned models appear to be able to improve upon the unconditioned models, which indicates that our architectures are not able to leverage the information provided by the conjecture. The presence of the conditioning does however impact the training profile of our models, in particular by making the 1D CNN model converge faster and overfit significantly quicker (figs. 4a and 4b).

6 CONCLUSIONS

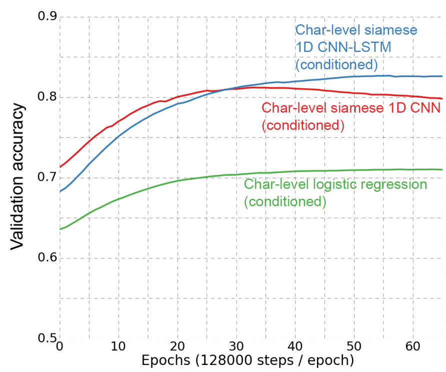
Our baseline deep learning models, albeit fairly weak, are still able to predict statement usefulness with a remarkably high accuracy, making them valuable for a number of practical theorem proving applications. This includes making tableaux-based (Paulson, 1999) and superposition-based (Hurd, 2003) internal ITP proof search significantly more efficient, which in turn would make formalization easier.



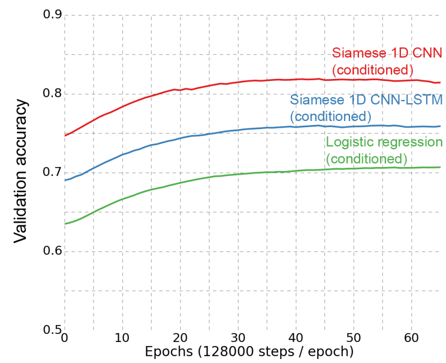
(a) Training profile of the three unconditioned baseline models with character input.



(b) Training profile of the three unconditioned baseline models with token input.



(a) Training profile of the three conditioned baseline models with character input.



(b) Training profile of the three conditioned baseline models with token input.

However, our models do not appear to be able to leverage order in the input sequences, nor do they appear to be able to leverage conditioning on the conjectures. This is due to the fact that these models are not doing any form of logical reasoning on their input statements; rather they are doing simple pattern matching at the level of n-grams of characters or tokens.

This shows the need to focus future efforts on new forms deep learning models that can do *reasoning*, or alternatively, on systems that blend explicit reasoning with deep learning-based feature learning.

6.1 Future work

The dataset focuses on one interactive theorem prover. It would be interesting if the proposed techniques generalize, primarily across ITPs that use the same foundational logic, for example using the (Hurd, 2011) standard, and secondarily across fundamentally different ITPs or even ATPs.

Finally, two of the proposed task for the dataset have been premise selection and intermediate sentence generation. It would be interesting to define more ATP-based ways to evaluate the selected premises, as well as to evaluate generated sentences (Kaliszyk et al., 2015). The set is a relatively large one when it comes to proof step classification, however the number of available premises makes the set a medium-sized set for premise selection in comparison with those of the Mizar Mathematical Library or the seL4 development.

Acknowledgements

Supported by the Austrian Science Fund (FWF) grant P26201.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise selection for mathematics by corpus analysis and kernel methods. *J. Autom. Reasoning*, 52(2): 191–213, 2014. doi: 10.1007/s10817-013-9286-5.

Alex A. Alemi, François Chollet, Geoffrey Irving, Christian Szegedy, and Josef Urban. DeepMath – Deep sequence models for premise selection. 2016. URL https://arxiv.org/abs/1606.04442.

David Aspinall and Cezary Kaliszyk. What's in a theorem name? In Jasmin Christian Blanchette and Stephan Merz (eds.), *Interactive Theorem Proving (ITP 2016)*, volume 9807 of *LNCS*, pp. 459–465. Springer, 2016. doi: 10.1007/978-3-319-43144-4.

Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998. ISBN 978-0-521-45520-6.

Jasmin C. Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016. ISSN 1972-5787. doi: 10.6092/issn.1972-5787/4593.

Jasmin Christian Blanchette, Maximilian P. L. Haslbeck, Daniel Matichuk, and Tobias Nipkow. Mining the Archive of Formal Proofs. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge (eds.), *Intelligent Computer Mathematics (CICM 2015)*, volume 9150 of *LNCS*, pp. 3–17. Springer, 2015.

James P. Bridge, Sean B. Holden, and Lawrence C. Paulson. Machine learning for first-order theorem proving - learning to select a good heuristic. *J. Autom. Reasoning*, 53(2):141–172, 2014. doi: 10.1007/s10817-014-9301-5.

Haogang Chen, Daniel Ziegler, Tej Chajed, Adam Chlipala, M. Frans Kaashoek, and Nickolai Zeldovich. Using crash Hoare logic for certifying the FSCQ file system. In Ajay Gulati and Hakim Weatherspoon (eds.), *USENIX 2016*. USENIX Association, 2016.

François Chollet. Keras. https://github.com/fchollet/keras, 2015.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Hazel Duncan. *The Use of Data-Mining for the Automatic Formation of Tactics*. PhD thesis, University of Edinburgh, 2007.

Michael Färber and Chad E. Brown. Internal guidance for Satallax. In Nicola Olivetti and Ashish Tiwari (eds.), *International Joint Conference on Automated Reasoning (IJCAR 2016)*, volume 9706 of *LNCS*, pp. 349–361. Springer, 2016. doi: 10.1007/978-3-319-40229-1.

Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A machine-checked proof of the odd order theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie (eds.), *Interactive Theorem Proving (ITP 2013)*, volume 7998 of *LNCS*, pp. 163–179. Springer, 2013.

Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz. Mizar in a nutshell. *J. Formalized Reasoning*, 3(2):153–245, 2010. doi: 10.6092/issn.1972-5787/1980.

Thomas Hales, John Harrison, Sean McLaughlin, Tobias Nipkow, Steven Obua, and Roland Zumkeller. A revision of the proof of the Kepler Conjecture. *Discrete & Computational Geometry*, 44(1):1–34, 2010.

Thomas C. Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *CoRR*, abs/1501.02155, 2015.

John Harrison. HOL Light: An overview. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel (eds.), *Theorem Proving in Higher Order Logics (TPHOLs 2009)*, volume 5674 of *LNCS*, pp. 60–66. Springer, 2009.

John Harrison. The HOL Light theory of Euclidean space. *J. Autom. Reasoning*, 50(2):173–190, 2013. doi: 10.1007/s10817-012-9250-9.

John Harrison, Josef Urban, and Freek Wiedijk. History of interactive theorem proving. In Jörg Siekmann (ed.), *Handbook of the History of Logic vol. 9 (Computational Logic)*, pp. 135–214. Elsevier, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Kryštof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans (eds.), *CADE-23*, volume 6803 of *LNAI*, pp. 299–314. Springer, 2011.

Joe Hurd. First-order proof tactics in higher-order logic theorem provers. In Myla Archer, Ben Di Vito, and César Muñoz (eds.), *Design and Application of Strategies/Tactics in Higher Order Logics (STRATA 2003)*, number NASA/CP-2003-212448 in NASA Technical Reports, pp. 56–68, September 2003. URL http://www.gilith.com/research/papers.

Joe Hurd. The OpenTheory standard theory library. In Mihaela Gheorghiu Bobaru, Klaus Havelund, Gerard J. Holzmann, and Rajeev Joshi (eds.), *NASA Formal Methods (NFM 2011)*, volume 6617 of *LNCS*, pp. 177–191. Springer, 2011.

Cezary Kaliszyk and Alexander Krauss. Scalable LCF-style proof translation. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie (eds.), *Interactive Theorem Proving (ITP 2013)*, volume 7998 of *LNCS*, pp. 51–66. Springer, 2013.

Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *J. Autom. Reasoning*, 53(2):173–213, 2014. doi: 10.1007/s10817-014-9303-3.

Cezary Kaliszyk and Josef Urban. Learning-assisted theorem proving with millions of lemmas. *J. Symbolic Computation*, 69:109–128, 2015. doi: 10.1016/j.jsc.2014.09.032.

Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. Learning to parse on aligned corpora. In Christian Urban and Xingyuan Zhang (eds.), *Proc. 6h Conference on Interactive Theorem Proving (ITP'15)*, volume 9236 of *LNCS*, pp. 227–233. Springer-Verlag, 2015. doi: 10.1007/978-3-319-22102-1_15.

Gerwin Klein, June Andronick, Kevin Elphinstone, Toby C. Murray, Thomas Sewell, Rafal Kolanski, and Gernot Heiser. Comprehensive formal verification of an OS microkernel. *ACM Trans. Comput. Syst.*, 32(1):2, 2014.

Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith (eds.), *Computer-Aided Verification (CAV 2013)*, volume 8044 of *LNCS*, pp. 1–35. Springer, 2013.

Daniel Kühlwein and Josef Urban. MaLeS: A framework for automatic tuning of automated theorem provers. *J. Autom. Reasoning*, 55(2):91–116, 2015. doi: 10.1007/s10817-015-9329-1.

Xavier Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, 2009.

Steven Obua and Sebastian Skalberg. Importing HOL into Isabelle/HOL. In Ulrich Furbach and Natarajan Shankar (eds.), *International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNCS*, pp. 298–302. Springer, 2006.

Lawrence C. Paulson. A generic tableau prover and its integration with Isabelle. *J. Universal Computer Science*, 5(3):73–87, 1999.

Stephan Schulz. System description: E 1.8. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov (eds.), *Logic for Programming, Artificial Intelligence (LPAR 2013)*, volume 8312 of *LNCS*, pp. 735–743. Springer, 2013.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings.*, pp. 926–934, 2013. URL http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pp. 2431–2439, 2015.

Geoff Sutcliffe. The CADE ATP system competition - CASC. *AI Magazine*, 37(2):99–101, 2016.

Josef Urban, Jiří Vyskočil, and Petr Štěpánek. MaLeCoP: Machine learning connection prover. In Kai Brünnler and George Metcalfe (eds.), *TABLEAUX 2011*, volume 6793 of *LNCS*. Springer, 2011.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.