Training Neural Networks as Theorem Provers via the Curry-Howard Isomorphism

Paul Tarau
Dept. of Computer Science and Engineering
University of North Texas
paul.tarau@unt.edu

Valeria de Paiva
Topos Institute, CA, USA
valeria.depaiva@gmail.com

Overview of the proposed 30 minutes CLA'20 talk

We describe a simple but effective method to train neural networks as theorem provers for an interesting logic language fragment, via combinatorial generation algorithms and the Curry-Howard isomorphism between lambda terms and formulas corresponding to their types.

Linear Logic [2] when used as a resource-control mechanism constrains the use of formulas available as premises in a proof. Its powerful proof mechanism is known to be Turing-complete even in the propositional case. Our talk will focus on generating all theorems of a given size for a restricted set of formulas, the Implicational fragment of Propositional Intuitionistic Linear Logic (**IPILL**), corresponding to principal types of lambda terms in normal form. In its simplest form, the Curry-Howard isomorphism [3] connects the implicational fragment of propositional intuitionistic logic with types in the *simply typed lambda calculus*. A low polynomial type inference algorithm associates a type (when it exists) to a lambda term. Harder (PSPACE-complete, see [6]) algorithms associate inhabitants to a given type expression with the resulting lambda term (typically in normal form) serving as a witness for the existence of a proof for the corresponding tautology.

To train neural networks as theorem provers via the Curry-Howard isomorphism we need to efficiently generate all theorems of a given size in the implicational fragment of propositional intuitionistic linear logic, with help from their lambda term couterparts.

We start by filtering for linearity the proof terms associated by a Prolog-based theorem prover for Implicational Intuitionistic Logic. This works, but using for each formula a PSPACE-complete algorithm limits it to very small formulas. We take a few back and forth walks over the bridge between proof terms and theorems, provided by the Curry-Howard isomorphism, and derive step-by-step an efficient algorithm requiring a low polynomial effort per generated theorem. The resulting Prolog program runs in $O(N)$ space for terms of size N and generates in a few hours **7,566,084,686** theorems in **IPILL**, together with their proof terms.

We will explain our results in terms of a *size-preserving bijection between linear lambda terms in normal form and their principal types*. The bijection, first proven in [4] and given a geometric interpretation in [7], has been instrumental in deriving the optimal final form of our term/formula pair generator. It is proven in [7] by exhibiting a reversible transformation of oriented edges in the tree describing the linear lambda term in normal form, into corresponding oriented edges in the tree describing the linear implicational formula, acting as its principal type.

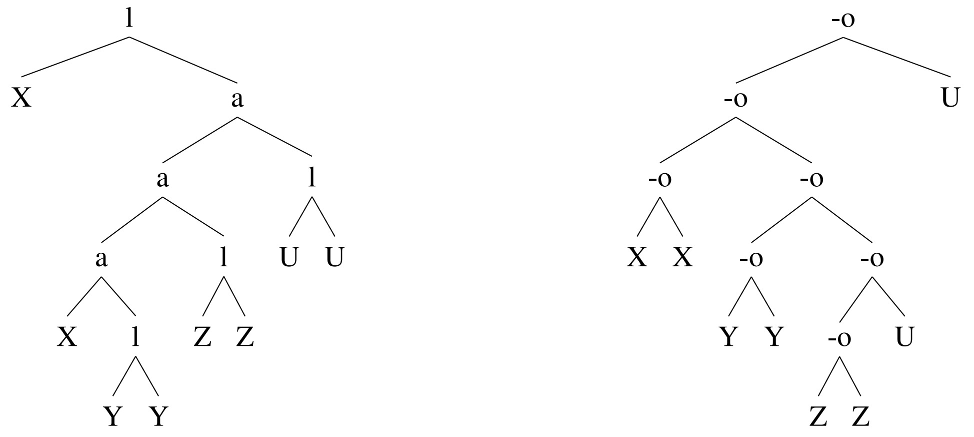
Thus, by generating all typable linear lambda terms of in normal form size N, we obtain, for free, a generator for all corresponding **IPILL** theorems of size N.

The figure below shows a lambda term normal form and its corresponding linear type (where we label lambda nodes with **l**, application nodes with **a**, linear implication nodes with **-o** and lambda and

Submitted to:
CLA'2020

type variables with uppercase letters).

$\lambda X.(((X\ \lambda Y.Y)\ \lambda Z.Z)\ \lambda U.U)$



The dataset, containing generated theorems of several sizes and their proof-terms in postfix form, is available at `http://www.cse.unt.edu/~tarau/datasets/lltaut/` and can be used for correctness, performance and scalability testing for linear logic theorem provers.

More importantly, we show that the formula/proof-term pairs in the dataset are usable to test if deep-learning systems can perform a fairly interesting theorem proving task, a topic of growing interest in deep learning [5, 1]. When trained via a `seq2seq` algorithm with an LSTM recurrent neural network on encodings of theorems and their proof-terms, the resulting model performs unusually well when tested on unseen formula/proof-term pairs.

Our experiments with training Recurrent Neural Networks using our implicational linear logic theorem dataset are available at: `https://github.com/ptarau/neuralgs` .

References

[1] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li & Denny Zhou (2019): *Neural Logic Machines*. Available at `https://openreview.net/pdf?id=B1xY-hRctX`.

[2] Jean-Yves Girard (1987): *Linear logic*. *Theoretical computer science* 50(1), pp. 1–101.

[3] W.A. Howard (1980): *The Formulae-as-types Notion of Construction*. In J.P. Seldin & J.R. Hindley, editors: *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, London, pp. 479–490.

[4] Grigori E. Mints (1992): *Closed categories and the theory of proofs*. In: *Selected Papers in Proof Theory*, Bibliopolis.

[5] Tim Rocktäschel & Sebastian Riedel (2017): *End-to-end Differentiable Proving*. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, editors: *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pp. 3788–3800. Available at `http://papers.nips.cc/paper/6969-end-to-end-differentiable-proving.pdf`.

[6] Richard Statman (1979): *Intuitionistic Propositional Logic is Polynomial-Space Complete*. *Theor. Comput. Sci.* 9, pp. 67–72, doi:10.1016/0304-3975(79)90006-9.

[7] Noam Zeilberger (2015): *Balanced polymorphism and linear lambda calculus, talk at TYPES'15*. `http://noamz.org/papers/linprin.pdf`.