Deriving Theorems in Implicational Linear Logic, Declaratively

Paul Tarau and Valeria de Paiva

September, 2020

University of North Texas / Topos Institute

ICLP'2020

Overview

- GOALS:
  - derive a theorem prover for a *decidable* fragment of linear logic, *implicational propositional intuitionistic linear logic* (**IPILL**)
  - enable training neural networks to find the proofs terms
- THE PROBLEM:
  - how to generate all theorems of a given size in **IPILL**?
- OUR SOLUTION:
  - derive step-by-step an efficient algorithm requiring a low polynomial effort per generated theorem
  - rely on the Curry-Howard isomorphism, focus on generating simply typed lambda terms in normal form
- THE OUTCOMES:
  - an implicational intuitionistic logic prover specialized to **IPILL** formulas
  - a dataset for training neural networks
  - preliminary results: very high success rate with seq2seq encoded LSTM neural networks (not in the paper, but quite cool)

The Implicational Fragment of Propositional Intuitionistic Linear Logic (IPILL)

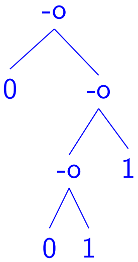
- *Linear Logic* provides the ability to constrain/control the use of formulas available as premises in a proof
- while propositional intuitionistic linear logic is already Turing complete, its *implicational fragment* is decidable
- $\Rightarrow$ polynomial algorithms for generating its theorems are useful:
  - when turned into *test sets*, combining tautologies and their proof terms can be useful for testing correctness and scalability of linear logic theorem provers
  - when turned into *datasets*, they can be used for training deep learning networks focusing on *neuro-symbolic* computations, among which theorem proving is a prototypical example

The Curry Howard Isomorphism

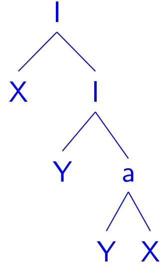
- of particular interest in the correspondence between computations and proofs is the *Curry-Howard isomorphism*
- in its simplest form, it connects the *implicational fragment of propositional intuitionistic logic* with types in the *simply typed lambda calculus*
- a low polynomial type inference algorithm associates a type (when it exists) to a lambda term
- harder, (PSPACE-complete) 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 in implicational propositional intuitionistic logic
- $\Rightarrow$ can we use combinatorial generation of lambda terms + type inference (easy) to "solve" some type inhabitation problems (hard)?

Formulas depicted as trees, together with their proof terms

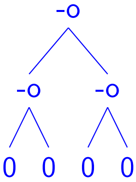
formula:



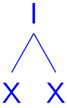
$\lambda X.\lambda Y.(Y\ X)$



formula:



$\lambda X.X$



The formula generators (details in the paper)

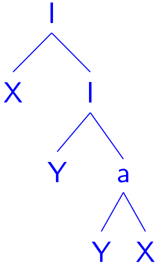
1. **IPILL** formulas (fairly simple Prolog code), built as:
   - binary trees of size N, counted by Catalan numbers $Catalan(N)$
   - labeled with variables derived from set partitions counted by $Bell(N+1)$ (see **A289679** in OEIS)
2. linear lambda terms (proof terms for the **IPILL** formulas)
   - linear skeleton Motzkin trees (binary-unary trees with constraints enforcing one-to-one mapping from variables to their lambda binders)
3. *closed* linear lambda terms
4. closed linear lambda terms in normal form
5. after a chain of refinements, we derive a compact and efficient generator for *pairs of Linear Lambda Terms in Normal Form* and their types (which always exist as they are all typable!)
6. it generates in a few hours **7,566,084,686** terms together with their corresponding types, seen as theorems in **IPILL** via the Curry-Howard isomorphism (**A062980** sequence in OEIS)

The Linear Lambda Term in Typed Normal Form Generator

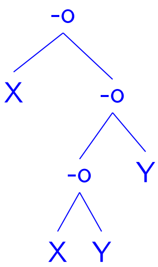
```
linear_typed_normal_form(N,E,T):-succ(N,N1),
  linear_typed_normal_form(E,T,N,0,N1,0,[]).

linear_typed_normal_form(l(X,E),(S '-o' T),A1,A2,L1,L3,Vs):-
  pred(L1,L2), % defined as L1>0,L2 is L1-1
  linear_typed_normal_form(E,T,A1,A2,L2,L3,[V:S|Vs]),
  check_binding(V,X).
linear_typed_normal_form(E,T,A1,A2,L1,L3,Vs):-
  linear_neutral_term(E,T,A1,A2,L1,L3,Vs).

linear_neutral_term(X,T,A,A,L,L,Vs):-
  member(V:TT,Vs),bind_once(V,X),T=TT.
linear_neutral_term(a(E,F),T,A1,A4,L1,L3,Vs):-pred(A1,A2),
  linear_neutral_term(E,(S '-o' T),A2,A3,L1,L2,Vs),
  linear_typed_normal_form(F,S,A3,A4,L2,L3,Vs).

bind_once(V,X):-var(V),V=v(X).
check_binding(V,X):-nonvar(V),V=v(X).
```

A Normal Form and its Corresponding Linear Type (I).

term: $\lambda X.\lambda Y.(Y\ X)$



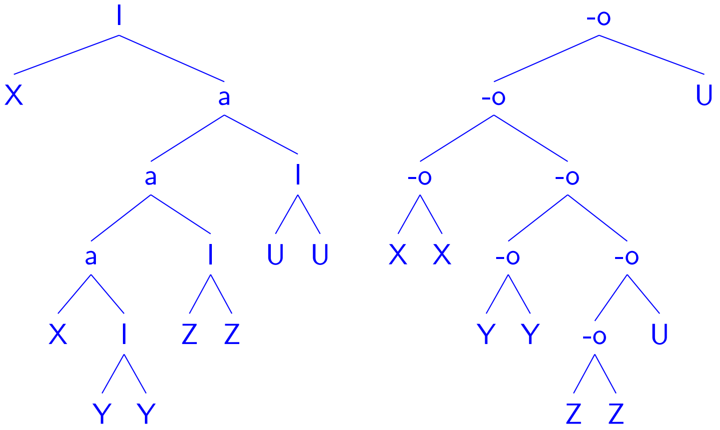
its linear type:



Note that all linear lambda terms are typable!

A Normal Form and its Corresponding Linear Type (II).

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



Note the symmetries between linear terms and their types!

The Eureka Moment

- it looks like we see some interesting symmetries in the pictures!
  - there are exactly two occurrences of each variable both in the theorems and their proof terms of which they are the principal types
  - theorems and their proof terms *have the same size*, counted as number of internal nodes
- *thus, we can solve the problem of generating all* **IPILL** *tautologies size N*

  IF

  the predicate `linear_typed_normal_form` *implements a generator of their proof-terms of size N*

Theorems for Free: the Size Preserving Bijection

- the GOOD NEWS: there's a *size-preserving bijection between linear lambda terms in normal form and their principal types*!
- a proof follows immediately from a paper by Noam Zeilberger who attributes this observation to Grigori Mints (refs. in the paper)
- the bijection is proven 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
- $\Rightarrow$ we have obtained a generator for all theorems of implicational linear intuitionistic propositional logic of a given size, as measured by the number of lollipops, without having to prove theorems!
- this is a "Goldilocks" situation that points out the very special case that implicational formulas have in linear logic and equivalently, linear types have in type theory!

Discussion

- our initial interest was to generate a benchmark of intuitionistic linear logic provers
- a motivation for this research was using linear lambda-calculus to investigate proofs in **ILL** and translations between it and intuitionistic logic proofs
- a long term goal of ours is to improve on the already known translations of intuitionistic logic into intuitionistic linear logic
- for that, we need to know more about the universe of existing linear proofs, like how many there are, their shapes, invariant properties
- this work is a step forward in that direction, covering the simpler case of the implicational fragment of propositional linear logic

Applications

- the dataset containing generated theorems and their proof-terms in postfix form (as well as their LaTeX tree representations marked as Prolog "%" comments) is available at `http://www.cse.unt.edu/~tarau/datasets/lltaut/`
- it can be used for correctness, performance and scalability testing of linear logic theorem provers
- the `<formula, proof-term>` pairs in the dataset are usable to test deep-learning systems on theorem proving tasks
- in fact, our `seq2seq` LSTM recurrent neural network trained on encodings of theorems and their proof-terms performs unusually well
- the experiments with training the neural networks using the IPILL theorem dataset are available at: `https://github.com/ptarau/neuralgs`

Accuracy of the LSTM seq2seq neural network on our formula/proof term dataset

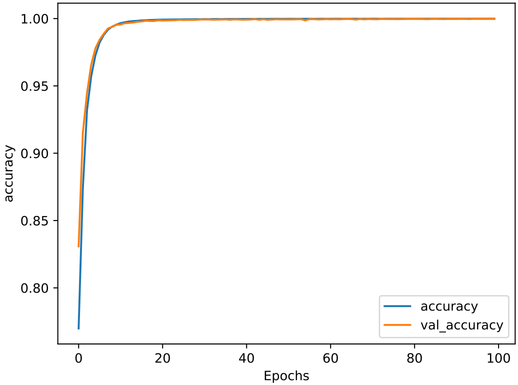


Figure: Accuracy curve for 100 epochs

Conclusions

- we have obtained a generator for all **IPILL** theorems of a given size, without needing a theorem prover by combining a generator for their proof terms and a type inference algorithm
- we sketched their use as a dataset for training neural networks, turning them into reliable theorem provers, for the harder inverse problem: given a formula in **IPILL**, find a proof term for it
- the dataset is at `http://www.cse.unt.edu/~tarau/datasets/`
- it now contains also a training set for implicational propositional intuitionistic logic
- the Appendix of the paper also contains an intuitionistic logic theorem prover, constrained to work on **IPILL** formulas, covering the case of formulas that are not in normal form