Deep Inference for Proof Search

Ozan Kahramanoğulları

University of Trento, Trento, Italy
ozan-k.com

Abstract

Deep inference can benefit proof search if non-determinism in proof search is controlled, and this way the breadth of the search space remains manageable. We support this argument by demonstrating three features of deep inference on examples, survey results, and list conjectures that generalise these features. That is, non-determinism in context management in deep inference can be controlled, deeper rule instances can be prioritised, and the inherent true concurrency can be selectively exploited for proof search.

The set-up

As the showcase, we use the deep inference presentation of multiplicative linear logic, which is the simplest deep inference system that is computationally meaningful [10]. We then generalise to other deep inference systems.

Formulae (or structures) are defined in the usual way. For example, the multiplicative linear logic *formulae* P, Q, R, ... are generated by

$$R ::= a \mid \bar{a} \mid 1 \mid \bot \mid [R \parr R] \mid (R \otimes R) \quad ,$$

where a stands for any atom; negation is defined on the atoms as a (non-identical) involution $\bar{\cdot}$, thus dual atom occurences, as a and $\bar{a}$, can appear in the formulae. 1 and $\bot$ are the units *one* and *bottom*, which are special atoms. Different kinds of brackets are used to distinguish logical operators. Negation on formulae is defined in the usual way by using (De Morgan) equations.

Formulae are considered to be equivalent modulo a congruence relation. We use the smallest congruence relation induced by the equational system consisting of the equations for *associativity* and *commutativity*. Other relations can be used for different systems, see, e.g., [4].

Inference rules are rewriting rules. We define the system MLS for multiplicative linear logic as the term rewiting system below, where r, t, u are generic terms that can match any formula, and x is a special term that can only match atoms.

$$\mathsf{s}: \quad [(r \otimes t) \parr u] \to ([r \parr u] \otimes t) \qquad\qquad \mathsf{u}_1{\downarrow}: \quad [\bot \parr r] \to r$$

$$\mathsf{ai}{\downarrow}: \quad [x \parr \bar{x}] \to 1 \qquad\qquad \mathsf{u}_2{\downarrow}: \quad (1 \otimes r) \to r$$

A *derivation* Δ is a formula or a finite chain of instances of rule instances. The left-most formula in a derivation is the *conclusion*, and the right-most formula is the *premise*. A derivation Δ whose premise is T, conclusion is R, and inference rules are in $\mathscr{S}$ is written as $R \xrightarrow{\Delta} T$. A MLS *proof* Π is a derivation whose premise is the unit 1.

Proof systems for other logics are defined similarly by including other inference rules or introducing modifications to the inference rules involving the units. For example, system KSg is obtained by including the inference rules for contraction and weakening, whereas system BV includes the rule $\mathsf{q}{\downarrow}$ for the management of the non-commutative contexts that are particular to logic BV. Both of these logics treat the units differently, see, e.g., [1, 4].

An interactive theorem prover as well as Maude modules that implement some of the ideas below can be found at our website. [1]

[1] http://ozan-k.com/software.html

Non-determinism in context management can be controlled

A significant part of the non-determinism in proof search with deep inference systems is due to the switch rule, i.e., the rule s, which is the rule responsible for context management.

Example 1. *Consider the formula* $[(\bar{a} \otimes [b \parr \bar{b}]) \parr a]$. *The switch rule can be applied to this formula in two different ways and one of them results in a proof.*

Example 2. *Consider the formula* $[([(\bar{c} \otimes \bar{d}) \parr c \parr d] \otimes \bar{a} \otimes \bar{b}) \parr a \parr b]$. *The switch rule can be applied to this formula in 24 different ways, however only 4 of these instances provide proofs. The two of these instances at the deeper positions provide shorter proofs than others.*

Definition 3. *Given a formula* r, $\mathsf{at}\, r$ *is the set of the atoms in* r. *For any two formulae* r *and* u, *we say that* r *and* u *can interact if* $\mathsf{at}\, \overline{r} \cap \mathsf{at}\, u \neq \emptyset$.

Definition 4. *An instance of the switch rule* s *is an instance of* deep lazy interaction switch (dlis) *if* r *and* u *can interact,* u *is not a disjunction, and* r *is not a conjunction.*

Example 5. *For the formula in Example 2, an instance of* dlis *is given with* $r = \bar{c}$ *and* $u = c$.

Let us recall that system BV is the deep inference system that extends multiplicative linear logic with a self-dual non-commutative logic operator, and the rules mix and mix-0 [4]. System KSg is the deep inference classical propositional logic system in [1]. System ELS is the deep inference multiplicative exponential linear logic system in [13].

Theorem 6. *A formula* R *has a proof in system* $\mathscr{S} \in \{\mathsf{MLS}, \mathsf{BV}, \mathsf{KSg}, \mathsf{ELS}\}$ *if and only if* R *has a proof in the system obtained by replacing the rule* s *with the rule* dlis *in system* $\mathscr{S}$.

For a more detailed exposure to this statement as well as the proofs see, e.g., [8, 7].

Conjecture 7. *A formula* R *has a proof in any deep inference system* $\mathscr{S}$ *if and only if* R *has a proof in the system obtained by replacing the rule* s *with the rule* dlis *in system* $\mathscr{S}$.

Definition 8. *For any deep inference rule* $\rho : \ell \to r$, *the pairs of variables in* r *that are connected by a disjunction at the top-level, but are not connected by a disjunction at the top level in* ℓ *are called* activated. *For example, in* s, *the variables* r *and* u *are activated.*

Definition 9. *For a deep inference system* $\mathscr{S}$ *we say that* $\mathscr{S}$ *has cut elimination if the following property holds: for any formula* R, R *has a proof in* $\mathscr{S}$ *if and only if* R *has a proof in the system that extends system* $\mathscr{S}$ *with the contrapositive of all the rules in* $\mathscr{S}$.

Example 10. *The contrapositive of the rules in* MLS *are listed below, which include the rule* s.

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{s}: & [(r \otimes t) \parr u] & \to ([r \parr u] \otimes t) & \mathsf{u}_1{\uparrow}: & r & \to (1 \otimes r) \\ \mathsf{ai}{\uparrow}: & \bot & \to (x \otimes \bar{x}) & \mathsf{u}_2{\uparrow}: & r & \to [\bot \parr r] \end{array}$$

Conjecture 11. *A formula* R *has a proof in any deep inference system* $\mathscr{S}$ *with cut elimination if and only if* R *has a proof in the system obtained by replacing any rule* ρ *in system* $\mathscr{S}$ *with a rule that imposes the interaction condition in Definition 3 on the activated variables in* ρ.

The promotion rule of multiplicative exponential linear logic and the k-rule of modal logic are examples to rule instances, where the activation condition in Conjecture 11 can be imposed.

Deeper rule instances can be prioritised

Because of the exponential blow up in proof search that is a consequence of hard complexity bounds [10, 7], the margin of successful applications are determined by the interplay between the breadth of the search space, nondeterminism and length of proofs. In general, deep inference provides short proofs due to a more immediate access to subformulae [2, 8]. However, the greater nondeterminism and the resulting large breadth of search space hinders broad proof search applications that benefit from these short proofs.

Consider the formulae in Examples 1 and 2. Both of these formulae have proofs that are shorter than the proofs provided by any sequent calculus system for multiplicative linear logic. These shorter proofs are precisely those that prioritise the instances of the inference rules that are applied to the sub-formulae at deeper positions in the formulae in comparison to the others. The following definition and conjecture take advantage of these observations.

Definition 12. *For any deep inference rule $\rho : \ell \to r$, an instance of the rule ρ is called* profound *if no instance of the rule ρ can be applied inside the redex of the instance of ρ.*

Conjecture 13. *A formula R has a proof in any deep inference system $\mathscr{S}$ with cut elimination if and only if R has a proof in system $\mathscr{S}$ where all the instances of the rule* s *are profound.*

The inherent true concurrency of proofs can be exploited

In [9], we have provided a true-concurrency characterisation of proofs in terms of event structures as a means to reduce proof length, which becomes effective when it is used in conjunction with the methods presented above. This is because an untamed introduction of concurrency can result in an excessive increase in the breadth of the search space.

Example 14. *In system* MLS*, the shortest proof of $[a \parr (\bar{a} \otimes [a \parr \bar{a}])]$ has length three. However, by using the event structures characterisation of derivations, we can obtain proofs of length two by composing rules in parallel. Below we first apply the composition of the causally independent rules* s *and* $\mathsf{ai}{\downarrow}$*, and then apply the composition of the rules* $\mathsf{u}_2{\downarrow}$ *and* $\mathsf{ai}{\downarrow}$*.*

$$[a \parr (\bar{a} \otimes [a \parr \bar{a}])] \overset{\mathsf{s}|\mathsf{ai}{\downarrow}}{\to} ([a \parr \bar{a}] \otimes 1) \overset{\mathsf{u}_2{\downarrow}|\mathsf{ai}{\downarrow}}{\to} 1$$

The event structures representation of derivations in [9] provides a canonical representation of the derivations that differ only with respect to permutation of inference rules, but that are in essence identical. Such a characterisation provides a qualification of identity of proofs with respect to rule permutations, while distinguishing proofs that differ in the inference steps that they take. A similar characterisation is provided, for example, by proof nets for the case of multiplicative linear logic. However, proof nets lack any information about the deductive steps performed in the proof. In contrast, in our approach, proofs that have identical proof nets can be distinguished with respect to their different configurations that result from distinct inference strategies. Such considerations are also addressed by atomic flows for classical logic [5, 6].

The true-concurrency characterisation of the derivations provides a point of view of the rule instances that exposes their independence in proof search. This in return provides a means for concurrent application of inference rules in a way that reduces the length of the proofs. Although uncontrolled use of concurrency can result in an excessive increase in the breadth of search space, introducing control by means of other methods as described above should result in improvement in proof search applications. Similar methods have been readily applied to planning problems with significant performance improvements [11, 12].

Discussion

We have surveyed three directions of research that can independently contribute to the problem of taming the non-determinism in deep inference proof search. Because deep inference provides shorter proofs than the sequent calculus, the proposed methods can provide improvements for proof search applications. A promising technique with a similar aim and overlapping notions, however with a more syntactic execution, is focused proofs [3].

We believe that the three methods we have listed above are orthogonal. To examine this as well as the validity of our conjectures above, we have so far resorted to methods that have previously proven successful such as splitting, permutation of inference rules and decompositions of proofs, see, e.g., [7, 8]. However, at the time of writing, we are not able to provide evidence for any of the claims. The recent developments on subatomic proof systems by Tubella and Guglielmi [14] have a potential to shed light to these claims, which is a topic of ongoing work.

References

[1] Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. PhD thesis, Technische Universität Dresden, 2003.

[2] Paola Bruscoli and Alessio Guglielmi. On the proof complexity of deep inference. *ACM Transactions on Computational Logic*, 2(14):1–34, 2009.

[3] Kaustuv Chaudhuri, Nicolas Guenot, and Lutz Straßburger. The focused calculus of structures. In Marc Bezem, editor, *Computer Science Logic (CSL'11) - 25th International Workshop/20th Annual Conference of the EACSL*, volume 12, pages 159–173. LIPICS, 2011.

[4] Alessio Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1):1–64, 2007.

[5] Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1:9):1–36, 2008.

[6] Alessio Guglielmi, Tom Gundersen, and Michel Parigot. A proof calculus which reduces syntactic bureaucracy. In *Proceedings of the International Conference on Rewriting Techniques and Applications 2010 (Edinburgh)*, pages 135–150. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik 2010 LIPIcs, 2010.

[7] Ozan Kahramanoğulları. System BV is NP-complete. *Annals of Pure and Applied Logic*, 152(1–3):107–121, 2008.

[8] Ozan Kahramanoğulları. Interaction and depth against nondeterminism in proof search. *Logical Methods in Computer Science*, 10 (2:5):1–49, 2014.

[9] Ozan Kahramanoğulları. True concurrency of deep inference proofs. In *Logic, Language, Information, and Computation 23rd International Workshop, WoLLIC 2016, Puebla, Mexico, August 16-19th, 2016. Proceedings*, volume 9803 of *LNCS*. Springer, 2016.

[10] Max Kanovich. The multiplicative fragment of linear logic is NP-complete. Technical Report X-91-13, Institute for Language, Logic, and Information, 1991.

[11] Sıtar Kortik and Uluç Saranlı. Lingraph: a graph-based automated planner for concurrent task planning based on linear logic. *Applied Intelligence*, 47(3):914–934, 2017.

[12] Sıtar Kortik and Uluç Saranlı. Robotic task planning using a backchaining theorem prover for multiplicative exponential first-order linear logic. *Journal of Intelligent & Robotic Systems*, pages 1–13, 2019.

[13] Lutz Straßburger. MELL in the calculus of structures. *Theoretical Computer Science*, 309:213–285, 2003.

[14] Andrea Aler Tubella and Alessio Guglielmi. Subatomic proof systems: Splittable systems. *ACM Transactions on Computational Logic (TOCL)*, 19(5), 2018.