EPiC Series in Computing

Volume 100, 2024, Pages 370–389

Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning

Deep Inference in Proof Search: The Need for Shallow Inference

Ozan Kahramanoğulları

Free University of Bozen-Bolzano, Bolzano, Italy
ozan.kah@gmail.com

Abstract

Deep inference is a proof theoretical formalism that generalises the "shallow inference" of sequent calculus by permitting the application of inference rules on subformulae like term rewriting rules. Deep inference makes it possible to build shorter proofs than sequent calculus proofs. However, deep inference in proof search introduces higher nondeterminism, an obstacle in front of applications. Deep inference is essential for designing system BV, an extension of multiplicative linear logic (MLL) with a self-dual non-commutative operator. MLL has shallow inference systems, whereas BV is impossible with a shallow-only system. As Tiu showed, any restriction on rule depth makes a system incomplete for BV. This paper shows that any restriction that rules out shallow rules makes the system incomplete, too. Our results indicate that for system BV, shallow and deep rules must coexist for completeness. We provide extensive empirical evidence that deep inference can still be faster than shallow inference when used strategically with a proof theoretical technique for reducing nondeterminism. We show that prioritising deeper rule instances, in general, reduces the cost of proof search by reducing the size of the managed contexts, consequently providing more immediate access to shorter proofs. Moreover, we identify a class of MLL formulae with deep inference proof search times that grow linearly in the number of atoms in contrast to an exponential growth pattern with shallow inference. We introduce a large and exhaustive benchmark for MLL, with and without mix, and a proof search framework to apply various search strategies, which should be of independent interest.

1 Introduction

Deep inference [8] is a proof theoretical methodology that allows for applying the inference rules at any position inside the formulae. Deep inference generalises the shallow inference of the sequent calculus, which, in contrast, permits the rule instances only at the top-level formulae. Due to the combinatoric wealth in the possible rule instances, deep inference exposes a rich proof theoretical analysis that provides new perspectives for various logics [2, 25, 26, 8, 9, 28]. Some of these results are only possible with deep inference [30].

Deep inference proposes solutions to the logical characterisation of problems in computing [8, 2, 9, 25, 3, 29, 8, 10]. A result of particular importance from the point of view of applications is that deep inference accommodates proofs for propositional logic that are exponentially shorter than those with the sequent calculus, as was shown for Statman tautologies in [4]. Sequent

calculus proofs, on the other hand, can be straightforwardly mapped to deep inference proofs, as formally shown via polynomial simulation results [4, 6]. Given that in proof search applications, e.g., logic programming [21], even an improvement of a few steps can make a difference, the capability to obtain shorter proofs should provide fertile ground for applications. However, applications based on deep inference face the obstacle of non-determinism in proof search. Because there are many more rule instances with deep inference compared to shallow inference, the breadth of the search space increases rapidly in proof search applications, leading to a combinatoric explosion after only a few steps.

System BV [8] is a logic for which the non-determinism in deep inference proof search has particular implications. System BV is a conservative extension of multiplicative linear logic (MLL) with the rules mix and nullary mix (mix0) and a self-dual non-commutative operator. The non-commutative operator of system BV is well-suited for modelling sequential data. Like MLL [18, 19, 20], system BV is NP-complete [11]. BV is similar to Pomset logic [23], it is contained in the latter [22], and extending BV with the exponentials of linear logic [10, 28] results in a Turing-complete logic [27]. However, as Tiu demonstrated with a counter-example, system BV is impossible to conceive as a sequent calculus system or any system with bounded depth inference rules without losing expressivity [30]. Consequently, any proof search application that benefits from the full expressivity of BV needs to employ deep inference machinery.

In previous work [11, 13, 15], we introduced a technique for controlling the non-determinism in deep inference proof search, and demonstrated its applicability to classical logic, multiplicative exponential linear logic and logic BV. This technique is based on making use of the interactions between complementary formulae in proof construction and its correctness relies on the similarity between cut elimination and completeness. We have used this technique also as a proof theoretical tool to show the NP-completeness of BV [11].

Although this technique indicates a reduction in non-determinism, its performance in proof search has not been hitherto quantified with extensive empirical evidence. In particular, the availability of the interaction technique does not address the problem of how the search space should be explored: the computational bottleneck due to the larger breadth of the search space also persists when this technique is used naively to explore the complete search space. When the proved formula presents sufficient depth to apply the inference rules deeply, there are still many instances that cause the search space to expand more quickly than with shallow inference. However, the large breadth of the search space also exposes shorter proofs that are otherwise inaccessible with shallow inference.

To see these notions on an example, let us consider the formula $[([a \parr \bar{a}] \otimes \bar{b}) \parr b]$. With shallow inference, the immediate breadth of the search space has the left and middle choices below, whereas deep inference adds the third on the right, and others, depending on the logic.

$$\mathsf{s}\,\frac{([a \parr \bar{a}] \otimes [b \parr \bar{b}])}{[([a \parr \bar{a}] \otimes \bar{b}) \parr b]} \qquad \mathsf{s}\,\frac{([a \parr \bar{a} \parr b] \otimes \bar{b})}{[([a \parr \bar{a}] \otimes \bar{b}) \parr b]} \qquad \mathsf{ai}{\downarrow}\,\frac{[(1 \otimes \bar{b}) \parr b]}{[([a \parr \bar{a}] \otimes \bar{b}) \parr b]}$$

The situation exposed by this simple example becomes more significant as the number of the subformulae grows. For example, for the formula $[(\bar{a} \otimes \bar{b}) \parr a \parr b]$ the immediate breadth of the search space has 6 rule instances, and only 2 of them result in proofs. With 6 atoms, we get 42 immediate rule instances, and only 3 of them deliver proofs. However, some of the deep rule instances provide shorter proofs. For example, the third instance above results in a shorter proof. Then the question is: "How can these shorter proofs be accessed without being exposed to the larger breadth of the search space?"

A potential answer to this question has been further narrowing the proof search space by allowing only deeper rule instances. We had previously conjectured the completeness of a

restriction that rules out shallow instances in deep inference systems [13]. If shallow instances can be discarded without losing completeness, the remaining deeper instances will provide access to shorter proofs quickly with a much narrower search space breadth.

Given that all proof attempts for this conjecture failed, we searched for a counter-example. In the following, we show such a counter-example. To find this example, we have generated all the provable formulae with up to 14 atoms that are provable in MLL or BV, resulting in more than 20 million provable and non-provable MLL formulae and many more for BV. [1] We have discovered that all the formulae up to 10 atoms are provable with restrictions on context management rules that push the rule instances deeper into the formula. However, two different 12-atom formulae are counter-examples as they are otherwise provable. These are the only two among 174079 provable formulae with 12 atoms. These examples demonstrate that shallow inference is essential for the completeness of MLL and BV.

With this observation, we explored the performance of the alternative of pushing the rule instances deeper via a strategy while keeping them in the search space. We performed an exhaustive empirical analysis using our benchmark of *all the formulae* consisting of up to 14 atoms. Our results indicate that a naive approach for exploring the search space does not deliver any competitive advantage to deep inference in comparison with shallow inference. As expected, without any restrictions on the search strategy, the larger search space breadth results in a quicker expansion and creates a bottleneck. However, we show on all the formulae with up to 14 atoms that a search strategy that prioritises deeper rule instances over shallower ones provides a reduction in the average cost of proof search significantly. The difference in performance grows rapidly as the formulae grow larger.

Moreover, we introduce a normal form for MLL and include in our benchmark all the formulae in this normal form with up to 34 atoms (607.492 formulae in normal form). With this deep strategy, we show that the formulae in normal form require linear time in the number of atoms, in contrast to an exponential growth pattern with shallow inference.

In the following, we first provide the background on deep inference and the previous work on which the present study is based. We then illustrate our counter-example and introduce the benchmark, providing a link to an online repository with our modules and scripts. We incrementally report the experiments that substantiate our claims and conclude with a discussion of our results and outlook.

2 Deep Inference and System BV

Deep inference operates on equivalence classes of formulae modulo algebraic equalities for logical operators such as associativity, commutativity, and equalities for units and negation. The notion of an equivalence class of formulae, originally called *structure*, stems from a graph representation of formulae, which gives meaning to the inference rules [8]. The inference rules of BV are also transformations on these graph representations.

In the following, we use the term formula as a synonym for structure: a formula is a representation of all the formulae in its equivalence class. For example, the two formulae $[a \parr b]$ and $[b \parr a]$ are in the same equivalence class, and we can use any of them to denote their equivalence class. Let us now formally introduce BV formulae.

Definition 1. *There are infinitely many* positive atoms *and* negative atoms. *Atoms, positive or negative, are denoted by* $a, b, \ldots$ *Formulae are denoted by* $S, P, Q, R, T, U, \ldots$ BV *formulae*

[1] We provide the scripts for generating these formulae in a GitHub repository. The formulae are also available for download at https://www.inf.unibz.it/~okahramanogullari/Formulae/.

are generated by the following grammar.

$$R ::= a \mid \circ \mid \bar{R} \mid [R \parr R] \mid (R \otimes R) \mid \langle R \lhd R \rangle$$

Here, $\circ$ is the unit, $\bar{R}$ is the negation *of the formula R, $[R \parr R]$ is a* par *formula, $(R \otimes R)$ is a* copar *(times) formula and $\langle R \lhd R \rangle$ is a* seq *formula. A* formula context*, $S\{\ \}$, is a formula with a hole that is not in the scope of a negation. The formula R is a* subformula *of $S\{R\}$, and $S\{\ \}$ is its* context*. We denote the empty context $\{\ \}$ by $\circ$. Context braces are omitted if no ambiguity is possible, e.g., if $S\{\ \} = [\{\ \} \parr a]$, and $R = \bar{a}$, $S\{R\} = [\bar{a} \parr a]$.*

BV extends MLL with the rules mix, mix0, and an associative non-commutative operator. Due to the rules mix and mix0, the units $\bot$ and $\mathsf{1}$ collapse into a single unit [8].

Definition 2. BV *formulae are equivalent modulo the relation $\approx$, which is the smallest congruence relation induced by the equalities for associativity and commutativity of par and copar, associativity of seq, the equations*

$$[R \parr \circ] \approx R, \qquad (R \otimes \circ) \approx R, \qquad \langle R \lhd \circ \rangle \approx R, \qquad \langle \circ \lhd R \rangle \approx R$$

for unit and the equations

$$\overline{[R \parr T]} \approx (\overline{R} \otimes \overline{T}), \quad \overline{(R \otimes T)} \approx [\overline{R} \parr \overline{T}], \quad \overline{\langle R \lhd T \rangle} \approx \langle \overline{R} \lhd \overline{T} \rangle, \quad \overline{\overline{R}} \approx R, \quad \overline{\circ} \approx \circ$$

for negation*. We denote the formulae in the same equivalence class by picking a formula from the equivalence class. If there is no ambiguity, we drop the superfluous brackets. For simplicity, we assume that formulae are in negation normal form, which is obtained by pushing the negation to the atoms in the usual way.*

The set of equalities above is specific to system BV, which differs from those for deep inference systems for other logics [8, 25, 26, 2, 29, 3, 10, 28].

Example 3. *The two formulae below are equivalent.*

$$[[a \parr b] \parr [c \parr (\langle \bar{a} \lhd \langle (\bar{b} \otimes \bar{c}) \lhd [d \parr \bar{d}] \rangle \rangle \otimes \circ)]] \approx [[[c \parr \circ] \parr [b \parr a]] \parr (\langle \langle \bar{a} \lhd (\bar{b} \otimes \bar{c}) \rangle \lhd [d \parr \bar{d}] \rangle \otimes \circ)]$$

We can denote both of these formulae with $[a \parr b \parr c \parr \langle \bar{a} \lhd (\bar{b} \otimes \bar{c}) \lhd [d \parr \bar{d}] \rangle]$.

Definition 4. *A deep inference rule is a scheme of the kind*

$$\rho \, \frac{S\{T\}}{S\{R\}} \,,$$

where ρ is the name *of the rule, $S\{T\}$ is its* premise*, and $S\{R\}$ is its* conclusion*. For system* BV*, such a rule determines the linear implication $T \multimap R$, or equivalently $[\bar{T} \parr R]$, inside a generic context $S\{\ \}$. In an instance of ρ, we say that R is the* redex *and T is the* contractum*. A system $\mathscr{S}$ is a set of inference rules.*

An inference rule can be seen as a rewriting rule modulo an equational theory. Such a consideration becomes useful, e.g., in proof search [17], as is the case for the present work.

Definition 5. *A* formal system*, denoted by S, is a set of rules. A* derivation *in a system S is a finite chain of instances of rules of S and is denoted by Δ. A derivation can consist of just one formula. The topmost formula in a derivation is called* premise*, and the bottommost formula in a derivation is called* conclusion*. A* proof *is a derivation with the premise $\circ$.*

Definition 6. *System* BV *is shown in Figure 1. The rules are called* atomic interaction *(*ai↓*),* switch *(*s*), and* seq *(*q↓*). System* FBV *is the system consisting of only the rules* ai↓ *and* s*.*

$$\mathsf{ai}{\downarrow}\,\frac{S\{\circ\}}{S[a \parr \bar{a}]} \qquad \mathsf{s}\,\frac{S([R \parr U] \otimes T)}{S[(R \otimes T) \parr U]} \qquad \mathsf{q}{\downarrow}\,\frac{S\langle [P \parr Q] \lhd [R \parr T]\rangle}{S[\langle P \lhd R\rangle \parr \langle Q \lhd T\rangle]}$$

Figure 1: System BV

3 The Need for Shallow Inference

Tiu showed that the applicability of inference rules in deeper contexts is essential for system BV [30]. Any restriction on the depth of the inference rules of BV delivers a strictly less expressive logical system. The counter-example in [30] is based on a formula, the provability of which necessitates deep inference to access the deepest formula. This formula can be nested to generate formulae requiring increasingly deeper applicability of the inference rules. In its simplest form, the following example illustrates the counter-example.

Example 7. *The only way to prove the formula* $[\langle [a \parr b] \lhd c\rangle \parr \langle \bar{a} \lhd [\bar{b} \parr \bar{c}]\rangle]$ *requires an instance of the rule* $\mathsf{q}{\downarrow}$ *that sequentializes at least one of the par formulae* $[a \parr b]$ *and* $[\bar{b} \parr \bar{c}]$*. Two such proofs are depicted below, where the redex in each rule instance is colour-marked.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\circ}{[c \parr \bar{c}]}\mathsf{ai}{\downarrow}}{\langle [b \lhd \bar{b}] \lhd [c \parr \bar{c}]\rangle}\mathsf{ai}{\downarrow}}{\langle [a \parr \bar{a}] \lhd [b \lhd \bar{b}] \lhd [c \parr \bar{c}]\rangle}\mathsf{ai}{\downarrow}}{\langle [a \parr \bar{a}] \lhd [\langle b \lhd c\rangle \parr \langle \bar{b} \lhd \bar{c}\rangle]\rangle}\mathsf{q}{\downarrow}}{[\langle a \lhd b \lhd c\rangle \parr \langle \bar{a} \lhd \bar{b} \lhd \bar{c}\rangle]}\mathsf{q}{\downarrow}}{[\langle a \lhd b \lhd c\rangle \parr \langle \bar{a} \lhd [\bar{b} \parr \bar{c}]\rangle]}\mathsf{q}{\downarrow}}{[\langle [a \parr b] \lhd c\rangle \parr \langle \bar{a} \lhd [\bar{b} \parr \bar{c}]\rangle]}\mathsf{q}{\downarrow} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\circ}{[c \parr \bar{c}]}\mathsf{ai}{\downarrow}}{\langle [a \lhd \bar{a}] \lhd [c \parr \bar{c}]\rangle}\mathsf{ai}{\downarrow}}{\langle [a \parr \bar{a}] \lhd [\langle [b \parr \bar{b}] \lhd c\rangle \parr \bar{c}]\rangle}\mathsf{ai}{\downarrow}}{\langle [a \parr \bar{a}] \lhd [\langle b \lhd c\rangle \parr \bar{b} \parr \bar{c}]\rangle}\mathsf{q}{\downarrow}}{[\langle a \lhd b \lhd c\rangle \parr \langle \bar{a} \lhd [\bar{b} \parr \bar{c}]\rangle]}\mathsf{q}{\downarrow}}{[\langle [a \parr b] \lhd c\rangle \parr \langle \bar{a} \lhd [\bar{b} \parr \bar{c}]\rangle]}\mathsf{q}{\downarrow}$$

The one on the left sequentializes both seq formulae to make the dual atoms interact. The one on the right sequentializes only one of the two par formulae, which is also sufficient.

As these two proofs and the example in the introduction demonstrate, deeper rule instances can provide shorter proofs when they are available. This is because deeper instances often reduce the number of nested logical operators due to the annihilation of subformulae. This reduction, in return, reduces non-determinism as fewer nested logical operators imply fewer rule instances in the subsequent search tree branches. Although this mechanism is useful in providing quicker access to shorter proofs, we show below that it cannot be imposed to reduce non-determinism without losing completeness. Shallow rule instances are essential for completeness.

Definition 8. *A formula context* $S\{\ \}$ *is* shallow *if there are no formulae* $S_1 \neq \circ$*,* $S_2 \neq \circ$*, and* S_3 *such that* $S = [S_1 \parr (S_2 \otimes S_3\{\ \})]$ *or* $S = [S_1 \parr \langle S_2 \lhd S_3\{\ \}\rangle]$ *or* $S = [S_1 \parr \langle S_3\{\ \} \lhd S_2\rangle]$*.*

The definition above is the most general form of shallowness that illustrates our case. However, more restricted definitions can also be considered.

Example 9. *Both* $([a \parr \bar{a}] \otimes \{\ \})$ *and* $[(a \otimes [a \parr \bar{a}]) \parr \{\ \}]$ *as well as* $\{\ \}$ *and* $([a \parr \bar{a}] \otimes [a \parr \{\ \}])$ *are shallow contexts, whereas the contexts* $[a \parr (\bar{a} \otimes \{\ \})]$ *and* $[a \parr \langle \bar{a} \lhd \{\ \}\rangle]$ *are not.*

$$\mathsf{ai}{\downarrow}\frac{\circ}{[a \parr \bar{a}]}$$
$$\mathsf{ai}{\downarrow}\frac{}{[\bar{a} \parr \langle a \lhd [a \parr \bar{a}]\rangle]}$$
$$\mathsf{q}{\downarrow}\frac{}{[\bar{a} \parr \bar{a} \parr \langle a \lhd a\rangle]}$$
$$\mathsf{ai}{\downarrow}\frac{}{[\langle [a \parr \bar{a}] \lhd [\bar{a} \parr \bar{a}]\rangle \parr \langle a \lhd a\rangle]}$$
$$\mathsf{q}{\downarrow}\frac{}{[\langle a \lhd [\bar{a} \parr \bar{a}]\rangle \parr \bar{a} \parr \langle a \lhd a\rangle]}$$
$$\mathsf{ai}{\downarrow}\frac{}{[\langle a \lhd [\bar{a} \parr \bar{a}]\rangle \parr \langle [\bar{a} \parr \langle a \lhd a\rangle] \lhd [\bar{a} \parr a]\rangle]}$$
$$\mathsf{ai}{\downarrow}\frac{}{[\langle a \lhd [\bar{a} \parr \bar{a}]\rangle \parr \langle [\bar{a} \parr \langle a \lhd a\rangle] \lhd [\bar{a} \parr \langle a \lhd [a \parr \bar{a}]\rangle]\rangle]}$$
$$\mathsf{q}{\downarrow}\frac{}{[\langle a \lhd [\bar{a} \parr \bar{a}]\rangle \parr \langle [\bar{a} \parr \langle a \lhd a\rangle] \lhd [\bar{a} \parr \bar{a} \parr \langle a \lhd a\rangle]\rangle]}$$
$$\mathsf{ai}{\downarrow}\frac{}{[\langle a \lhd [\bar{a} \parr \bar{a}]\rangle \parr \langle [\bar{a} \parr \langle a \lhd a\rangle] \lhd [\langle a \lhd a\rangle \parr \langle [\bar{a} \parr \bar{a}] \lhd [a \parr \bar{a}]\rangle]\rangle]}$$
$$\mathsf{q}{\downarrow}\frac{}{[\langle a \lhd [\bar{a} \parr \bar{a}]\rangle \parr \langle [\bar{a} \parr \langle a \lhd a\rangle] \lhd [\langle a \lhd a\rangle \parr \bar{a} \parr \langle a \lhd [\bar{a} \parr \bar{a}]\rangle]\rangle]}$$
$$\mathsf{q}{\downarrow}\frac{}{[\langle [\bar{a} \parr \langle a \lhd a\rangle] \lhd [\bar{a} \parr \langle a \lhd a\rangle]\rangle \parr \langle a \lhd [\bar{a} \parr \bar{a}]\rangle \parr \langle a \lhd [\bar{a} \lhd \bar{a}]\rangle]}$$

Figure 2: A proof of the formula in Example 11.

Definition 10. *A rule instance is shallow if it is in a shallow context. A rule is deep-only if its shallow instances are excluded when a deep (non-shallow) instance is available.*

Example 11. *The formula below can be proven as depicted in Figure 2.*

$$[\langle [\bar{a} \parr \langle a \lhd a\rangle] \lhd [\bar{a} \parr \langle a \lhd a\rangle]\rangle \parr \langle a \lhd [\bar{a} \parr \bar{a}]\rangle \parr \langle a \lhd [\bar{a} \parr \bar{a}]\rangle]$$

There are 28 rule instances of the rule $\mathsf{q}{\downarrow}$ *with this formula in the conclusion. The instances that are not shallow do not lead to a proof.*

Due to the units, the rule $\mathsf{q}{\downarrow}$ can be applied to this formula in 28 different ways [16], making the case analysis difficult. We can produce a similar example involving copar formulae instead of seq formulae. This formula is simpler as it involves fewer rule instances.

Example 12. *The formula below can be proven as depicted in Figure 3.*

$$[([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a}])]$$

Remark 13. *The formula in Example 11 is not provable in* BV *with deep-only seq rule. The formula in Example 12 is not provable in* BV *with deep-only switch rule.*

The two example formulae are very similar as the rules s and $\mathsf{q}{\downarrow}$ are derived from the same rule in [8]. We discuss the formula in Example 12, which is simpler with fewer cases, and suffices to show that shallow inference is essential for provability for BV and MLL. This formula does not include any seq operators. Thus, the rule $\mathsf{q}{\downarrow}$ instances can be discarded from the case analysis. There are 15 rule instances of s with this formula in the conclusion. The Maude module `counter_example.maude` in our repository [2] shows that only the instance in Figure 3 results in a proof [17]. However, there are many other proofs, where this instance is the bottom-most.

[2] https://github.com/ozan-k/deep/blob/main/counter_example.maude

$$
\begin{array}{rc}
 & \circ \\ \hline
\mathsf{ai}\!\downarrow & [a \parr \bar{a}] \\ \hline
\mathsf{ai}\!\downarrow & [\bar{a} \parr (a \otimes [a \parr \bar{a}])] \\ \hline
\mathsf{s} & [\bar{a} \parr \bar{a} \parr (a \otimes a)] \\ \hline
\mathsf{ai}\!\downarrow & [([a \parr \bar{a}] \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes a)] \\ \hline
\mathsf{s} & [(a \otimes [\bar{a} \parr \bar{a}]) \parr \bar{a} \parr (a \otimes a)] \\ \hline
\mathsf{ai}\!\downarrow & [(a \otimes [\bar{a} \parr \bar{a}]) \parr ([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr a])] \\ \hline
\mathsf{ai}\!\downarrow & [(a \otimes [\bar{a} \parr \bar{a}]) \parr ([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes [a \parr \bar{a}])])] \\ \hline
\mathsf{s} & [(a \otimes [\bar{a} \parr \bar{a}]) \parr ([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr \bar{a} \parr (a \otimes a)])] \\ \hline
\mathsf{ai}\!\downarrow & [(a \otimes [\bar{a} \parr \bar{a}]) \parr ([\bar{a} \parr (a \otimes a)] \otimes [(a \otimes a) \parr ([\bar{a} \parr \bar{a}] \otimes [a \parr \bar{a}])])] \\ \hline
\mathsf{s} & [(a \otimes [\bar{a} \parr \bar{a}]) \parr ([\bar{a} \parr (a \otimes a)] \otimes [(a \otimes a) \parr \bar{a} \parr (a \otimes [\bar{a} \parr \bar{a}])])] \\ \hline
\mathsf{s} & [([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a}])]
\end{array}
$$

Figure 3: A proof of the formula in Example 12.

Let us analyse these cases. 6 of the remaining instances are of the following form, where we colour-mark R and T.

$$
\frac{S([R \parr \circ] \otimes T)}{S[(R \parr T) \otimes \circ]}
$$

The following instances do not lead to a proof.

$$([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)] \otimes [(a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a}])])$$
$$([([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a}])] \otimes a \otimes [\bar{a} \parr \bar{a}])$$
$$[([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a}] \otimes a \otimes [\bar{a} \parr \bar{a}])]$$
$$[([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)] \otimes a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a}])]$$
$$[([\bar{a} \parr (a \otimes a)] \otimes \bar{a} \otimes a \otimes a) \parr (a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a}])]$$
$$[([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes \bar{a} \otimes \bar{a})]$$

There are 8 other instances of s that result in non-provable formulae. Each can have multiple ways to instantiate R, T, and U. We mark R and U for one such choice for each case.

$$([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a) \parr (a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a}])])$$
$$(a \otimes [\bar{a} \parr \bar{a} \parr ([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a}])])$$
$$([\bar{a} \parr \bar{a}] \otimes [a \parr ([\bar{a} \otimes (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a}])])$$
$$[(a \otimes [\bar{a} \parr \bar{a}]) \parr ([\bar{a} \parr \bar{a}] \otimes [a \parr ([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)])])]$$
$$[([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr ([\bar{a} \parr \bar{a}] \otimes [a \parr (a \otimes [\bar{a} \parr \bar{a}])])]$$
$$[(a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a} \parr ([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)])])]$$
$$[([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a} \parr (a \otimes [\bar{a} \parr \bar{a}])])]$$
$$[(a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes ([a \parr \bar{a}] \otimes [\bar{a} \parr (a \otimes a)]))]$$

Definition 14. *Two systems $\mathscr{S}$ and $\mathscr{S}'$ are* equivalent *if they prove the same formulae.*

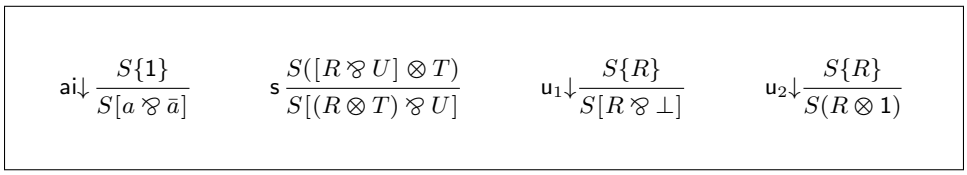

Figure 4: System MSu

Proposition 15. *No system consisting of deep-only rules can be equivalent to system* BV.

The statement follows from Examples 11, and 12 as the only rule instance that delivers a proof is in a shallow context in both cases. By using the formulae in Examples 7, 11, and 12, it is easy to obtain others that require both deep and shallow rule instances for provability.

Example 16. *The following formula requires both deep and shallow rule instances in a proof.*

$$([([\bar{a} \parr (a \otimes a)] \otimes [\bar{a} \parr (a \otimes a)]) \parr (a \otimes [\bar{a} \parr \bar{a}]) \parr (a \otimes [\bar{a} \parr \bar{a}])] \otimes [\langle [a \parr b] \lhd c \rangle \parr \langle \bar{a} \lhd [\bar{b} \parr \bar{c}] \rangle])$$

Remark 17. *In our benchmark, there are 174079 provable formulae with 12 atoms. The formula in Example 12 is one of the two formulae (and their symmetric versions with respect to atoms) that have no proof with deep-only rules. The other counter-example is the following.*

$$[(\bar{a} \otimes \bar{a}) \parr (\bar{a} \otimes [a \parr a]) \parr ([a \parr (\bar{a} \otimes \bar{a})] \otimes [a \parr \bar{a} \parr (a \otimes a)])]$$

Remark 18. *The rule* ai↓ *can be permuted up in the proof [26, 28], which implies that this rule can be restricted as deep-only.*

4 Non-determinism in Proof Search

In BV, a part of the non-determinism in the applicability of the inference rules is due to the context management rules s and q↓. Below, we review the technique in [11, 13, 15] that reduces the non-determinism in rule s. To be able to provide a comparison with a shallow system, we focus on MLL. However, we include in our benchmark also BV formulae.

Definition 19. *Given a formula R,* at R *is the set of the atoms in R.*

Example 20. *For $R = [a \parr \bar{a} \parr b \parr (\bar{a} \otimes c) \parr (a \otimes \bar{b})]$, we have* at $R = \{a, \bar{a}, b, \bar{b}, c\}$.

Definition 21. *For any two formulae R and U, we say that R and U can interact if R and U contain complementary atoms, that is,* at $\overline{R} \cap$ at $U \neq \emptyset$.

Definition 22. *Let* MSu *denote the deep inference system for* MLL *with the rules* s, ai↓, *and the rules for the multiplicative units* u1↓ *and* u2↓ *in Figure 4. An instance of the switch rule* s

1. *is an instance of* interaction switch (is) *if R and U can interact,*
2. *it is* lazy switch (ls) *if U is not a par,*
3. *it is* deep switch (ds) *if R is not a copar.*

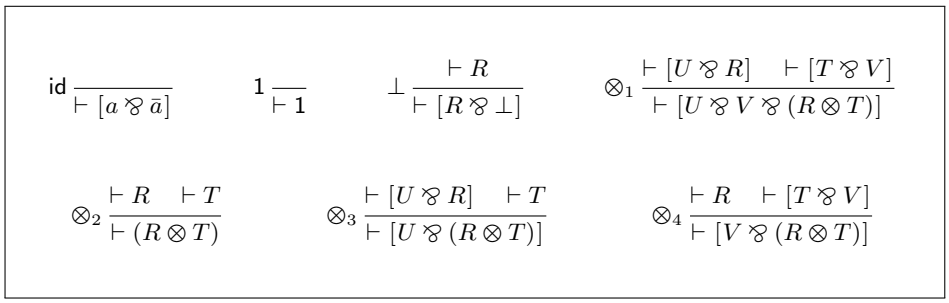

Figure 5: System MLL

By combining these restrictions, we obtain 7 new *rules. For example, by imposing all of them, we obtain* deep lazy interaction switch *(*dlis*). The system obtained by replacing in* MSu *the rule* s *with one of these* 7 *rules is denoted by adding the prefix of the switch rule. For example, the* system MSu with deep lazy interaction switch*, denoted by* MSdli*, is* $\{\mathsf{ai}{\downarrow}, \mathsf{dlis}, \mathsf{u}_1{\downarrow}, \mathsf{u}_2{\downarrow}\}$.

System MSu differs from the deep inference system for MLL in [26]: we use an explicit treatment of the units with the inference rules *unit-one* ($\mathsf{u}_1{\downarrow}$) and *unit-two* ($\mathsf{u}_2{\downarrow}$) instead of the equations for unit in [26]. Because negation appears only on atoms as before, we can discard the equations for negation and unit in the systems we use for proof search.

Theorem 23. *[11, 13, 15] Systems* MSdli *and* MLL *are equivalent.*

Example 24. *For the formula in Example 12, there are 6 applicable instances of the rule* dlis*, including the bottom-most rule instance in Figure 3 in contrast to 15 instances of the rule* s*.*

To conclude our discussion on setting the stage, let us introduce the shallow inference system MLL in Figure 5, defined as in the sequent calculus, however, with the omission of the meta-level rules, replaced by the associativity and commutativity of the par connective. In this setting, derivations are trees, as usual.

Let us now apply the notion of interaction to the shallow inference setting.

Definition 25. *The rules* $\otimes_1^{\mathsf{i}}$, $\otimes_3^{\mathsf{i}}$ *and* $\otimes_4^{\mathsf{i}}$ *are obtained from the rules* $\otimes_1$, $\otimes_3$ *and* $\otimes_4$ *in Figure 5 by imposing the condition that* R *and* U *can interact in* $\otimes_1^{\mathsf{i}}$ *and* $\otimes_3^{\mathsf{i}}$*, and that* T *and* V *can interact in* $\otimes_1^{\mathsf{i}}$ *and* $\otimes_4^{\mathsf{i}}$*. The system with these conditions in* MLL *is denoted by* MLLi*.*

Proposition 26. *Systems* MLL *and* MLLi *are equivalent.*

5 The Benchmark and Proof Search Implementation

We have created a benchmark using the equational term rewriting features of the Maude language. Let us first collect the definitions to build the benchmark.

Definition 27. *The* seed formulae *of* MLL *are given by the grammar:*

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

A seed with $2n$ *atoms is called an* n-seed*, where* n *is the* seed size*. A* seed generated formula *is obtained by iteratively applying the switch rule top-down.*

All formulae

n	2	3	4	5	6	7
# provable	6	53	665	10042	174079	3329979
# non-provable	9	124	2047	38518	795621	17655292

Table 1: The numbers of the n-seed generated MLL formulae in our benchmark.

Example 28. *The formula* $([a \parr \bar{a}] \otimes [a \parr \bar{a}] \otimes [a \parr \bar{a}])$ *is a 3-seed whereas the formulae* $[(\bar{a} \otimes \bar{a} \otimes \bar{a}) \parr a \parr a \parr a]$ *and* $([a \parr \bar{a}] \otimes [a \parr a \parr (\bar{a} \otimes \bar{a})])$ *are 3-seed generated.*

Proposition 29. *Any seed-generated formula has a proof in* MLL.

For the BV formulae, we included in the seed grammar in Definition 27 also the seq formulae. We implemented the definitions above in Maude as equational term rewriting systems [17] and interfaced Python scripts to control these modules. To obtain the formulae, we created n-seeds and generated the formulae using Maude modules that implement the top-down application of the context management rules, switch and seq. For the provable formulae, we restricted the formulae to those consisting of multiple instances of the same atom and its negation as this is the most general and harder case for proof search [7], providing the combinatoric wealth in all possible pairings of dual atoms, required for NP-hardness [18, 20, 11]. For the non-provable formulae, we replaced in each seed one of the atoms with a different atom. This way, we obtained all the n-seed generated formulae for $n \in \{2, \ldots, 7\}$. The numbers of these formulae are shown in Table 1. The Maude modules and the Python scripts are available for download at our GitHub repository[3]. An example of a module that generates formulae is the following.

```
mod S is
  sorts Atom formula . subsort Atom < formula .
  op -_ : Atom -> Atom [ prec 50 ] .
  op [_,_] : formula formula -> formula [assoc comm] .
  op {_,_} : formula formula -> formula [assoc comm] .
  ops a : -> Atom .  var R T U : formula .

  rl [switch] :  { [ R , U ] , T } => [ { R , T } , U ] .
endm
```

6 Comparing Proof Search Performance

We compared the performance of the four systems by using the same implementation. These systems are MLL (shallow), MLLi (shallow with interaction), MSu (deep), and MSdli (deep with interaction). In our comparison, we first set a baseline using Maude's built-in breadth-first search, which is highly optimised. This way, we verified how the search space expands for provable and non-provable formulae. Then, to experiment with different search strategies, we implemented the search stack using the Maude Python library [24]. This way, we could introduce programmatic control over the search space stack. Despite the considerable overhead it introduces, we settled for this option to be able to compare different systems and strategies using the same implementation. Consequently, we measured with different search strategies (*i*) the number of search steps, (*ii*) the search time, (*iii*) the length of the proof, defined as the

[3] https://github.com/ozan-k/deep

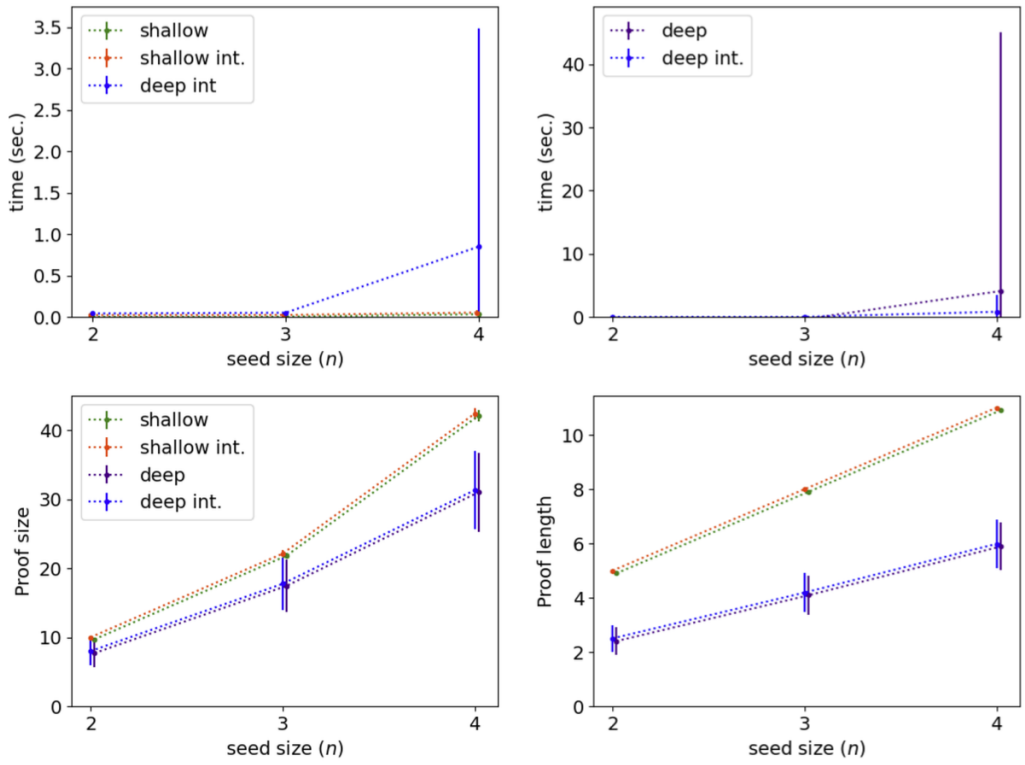

Figure 6: Mean search time (top), proof size (bottom-left), and proof length (bottom-right) with breadth-first search for all the n-seed formulae for $n < 5$. The error bars indicate the standard deviation. The search timed out for $n \geq 5$ for all the systems.

number of rule instances in the proof, and (*iv*) the size of the proof, defined as the number of atoms in the proof. With depth-first search, by applying different functions for sorting the immediate breadth of the search space at every proof step, we could implement and test different strategies for accessing the depth of the search space. The data plotted in all of the figures below are listed in the appendix.

6.1 Search space is smaller with shallow inference

We first compared the search space size with deep and shallow inference by performing breadth-first search. For these experiments we set 30000 steps as the cut-off for time out. The search timed-out for 5-seed formulae and larger; the results that are shown in Fig. 6 are for $n < 5$.

We found that with uncontrolled search, deep inference requires more search steps to find a proof due to a larger search space with proof times proportional to the search space size. However, the deep inference proofs are shorter and smaller than the shallow inference proofs. Despite the larger search space, with the 5-seed formulae, the deep inference system (without the interaction condition) timed out after proving 552 formulae, whereas the shallow system timed out after proving 15 formulae. This difference is due to the shorter deep inference proofs

that are accessed more quickly with breadth-first search. However, the larger search space creates a bottleneck as the lengths of the proofs increase.

We then compared the performance of the deep and shallow interaction systems. As expected, with breadth-first search, the deep interaction system performed better than the deep system. However, the shallow system with interaction was faster than all the others due to the narrower breadth of the search space. Because the interaction systems found the same proofs as the non-interaction systems, however faster, the proof lengths and sizes remained unaffected with the interaction systems. More interestingly, with 5-seed formulae, the deep inference interaction system timed out after proving 2224 formulae, an improvement in comparison to 552 formulae of non-interaction deep inference system and the 15 formulae of the shallow system also with the interaction. This confirms that the interaction condition provides a reduction in the search space size for the deep system, which is not the case for the shallow system, with or without the interaction condition.

6.2 Prioritising deeper rule instances delivers shorter proofs quickly

Breadth-first search explores the complete search space to find a proof of length k before attempting to find a proof of length $k+1$. As the length of the proofs increases, finding proofs become harder due to explosion in search space size. In depth-first search, on the other hand, the ordering of the breadth of the search space determines which part of the search space is explored first, in-depth, instead of another. In such a setting, applying, for example, the invertible rules sooner reduces the number of search steps of a successful search as, this way, these instances are prevented from propagating to the search space.

Example 30. *Consider the three proofs of the same formula below, where the early instance of the rule* $\mathsf{u}_2{\downarrow}$ *in proof construction makes the shorter proof immediately accessible. However, another important observation is that the early instance of the rule* $\mathsf{u}_2{\downarrow}$ *is possible only because of the deep instance of the rule* $\mathsf{ai}{\downarrow}$ *below it, which is applied before in bottom-up proof construction.*

$$\dfrac{\mathsf{ai}{\downarrow}\dfrac{\mathsf{u}_2{\downarrow}\dfrac{\mathsf{ai}{\downarrow}\dfrac{1}{[a \parr \bar{a}]}}{[(1 \otimes \bar{a}) \parr a]}}{[([a \parr \bar{a}] \otimes \bar{a}) \parr a]}} \qquad \dfrac{\mathsf{ai}{\downarrow}\dfrac{\mathsf{u}_2{\downarrow}\dfrac{1}{(1 \otimes 1)}}{(1 \otimes [a \parr \bar{a}])}}{\mathsf{s}\dfrac{[(1 \otimes \bar{a}) \parr a]}{[([a \parr \bar{a}] \otimes \bar{a}) \parr a]}} \qquad \dfrac{\mathsf{ai}{\downarrow}\dfrac{\mathsf{u}_2{\downarrow}\dfrac{1}{(1 \otimes 1)}}{([a \parr \bar{a}] \otimes 1)}}{\mathsf{s}\dfrac{([a \parr \bar{a}] \otimes [a \parr \bar{a}])}{[([a \parr \bar{a}] \otimes \bar{a}) \parr a]}}$$

Proposition 31. *For any formula* R *and context* S, $S\{R\}$ *has a proof in* MSu *if and only if (i)* $S[R \parr \bot]$ *has a proof; and (ii)* $S(R \otimes 1)$ *has a proof.*

To see the effect of prioritising invertible and deeper rules, we implemented a depth-first search strategy, where the immediate breadth of the search space is sorted by two functions.

Definition 32 (deeper inference)**.** *Given a formula* R, $\sigma(R)$ *denotes the number of symbols in* R. *Given an instance of the switch rule of the form*

$$\mathsf{s}\dfrac{S([U \parr R] \otimes T)}{S[U \parr (R \otimes T)]}$$

we define the function $f = 1/(\sigma(U)+\sigma(R))$. *Then, given two instances of the switch rule* $\mathsf{s}_1, \mathsf{s}_2$, *we define the* deeper inference *relation* $\succ$ *such that* $\mathsf{s}_1 \succ \mathsf{s}_2$ *if and only if* $f(\mathsf{s}_1) > f(\mathsf{s}_2)$. *The relation* $\succ$ *is extended to the other rules of system* MSu *as follows.*

$$\mathsf{u}_1{\downarrow} \succ \mathsf{u}_2{\downarrow} \succ \mathsf{ai}{\downarrow} \succ \mathsf{s}$$

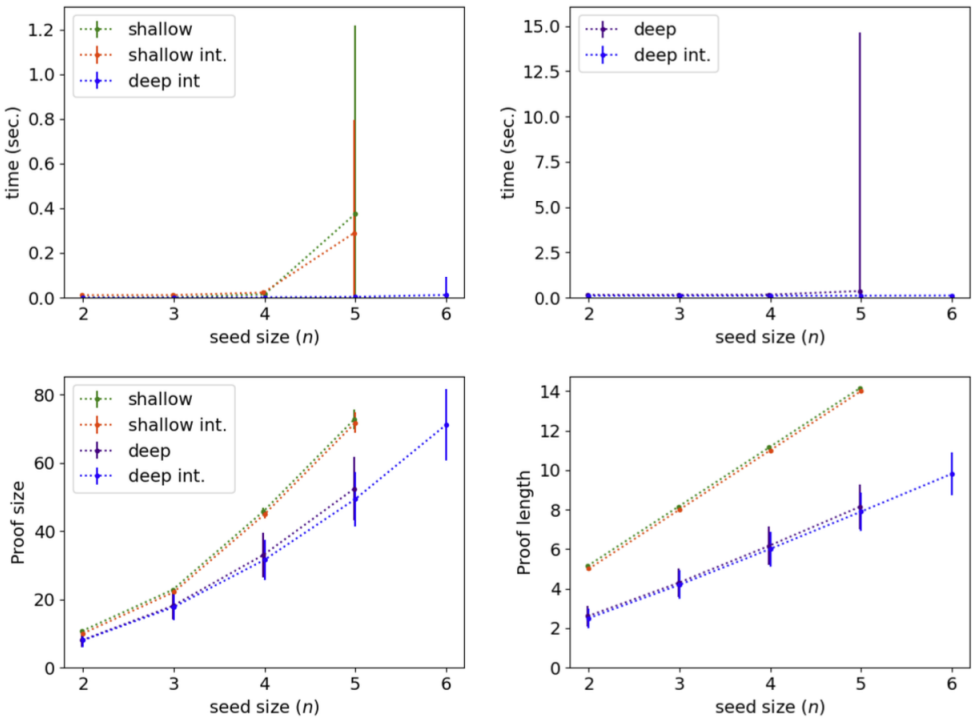

Figure 7: Mean search time (top), proof size (bottom-left) and length (bottom-right) with depth-first search for all the n-seed formulae for $n < 6$ and 6-seed formulae for deep interaction system. The search timed out for $n \geq 6$ for all the systems except the deep interaction system. For sorting the immediate breadth of the search space, the deeper inference strategy is used. The error bars indicate the standard deviation.

We implemented Definition 32 as a strategy to sort the immediate breadth of the deep systems so that the greatest rule instance with respect to the relation $\succ$ determines which part of the search space is explored in depth-first search before others. In the shallow systems, the rule instances follow a certain order whereby context management rules are applied always before other rules, hence there are no deeper rules to prioritise. Consequently, the strategy imposed by the sorting of the breadth of search space does not apply to the shallow systems. However, for a comparison, we singled out the better performing order on the rule instances of the shallow systems and used this for the experiments.

The results with depth-first search are depicted in Fig. 7. The shallow and shallow interaction systems required comparable number of search steps, whereby the interaction system performed slightly better. However, both systems were outperformed by the deep systems. Moreover, the deep interaction system provided proofs also for all the 6-seed formulae in less than 1000 steps. For all the 7-seed 3.329.979 formulae, the deep interaction system terminated with proofs. However, for 811 of these formulae, the search required more than 30000 steps. The deep interaction system outperformed all the other systems in proof times with a large margin by delivering proofs in milliseconds for all the formulae up to 6-seeds, also with smaller

Normal formulae

n	2	3	4	5	6	7	8	9
# formulae	3	8	21	52	124	284	629	1352
n	10	11	12	13	14	15	16	17
# formulae	2829	5777	11544	22620	43529	82409	153677	282634

Table 2: The numbers of the n-seed generated normal formulae in our benchmark.

proof sizes and lengths. The performance gap between deep and shallow interaction systems grew rapidly with the seed size.

To verify that the performance differences above are not only due to depth-first search but the strategy that prioritises the deeper rule instances, we ran experiments with the reverse order of the deeper inference strategy with depth-first search. As expected, the search performance was similar to the ones observed with breadth-first search (see the data in the appendix).

7 A Deeper Class of Formulae

We found that a class of formulae have exponentially faster proof search performance with the deeper search strategy.

Definition 33. *A formula is* normal *if there is no instance of the switch rule with this formula in the premise.*

Example 34. *Consider the formulae in Example 28.* $[(\bar{a} \otimes \bar{a} \otimes \bar{a}) \parr a \parr a \parr a]$ *is normal, whereas* $([a \parr \bar{a}] \otimes [a \parr a \parr (\bar{a} \otimes \bar{a})])$ *is not normal.*

We have generated exhaustively all the seed-generated normal formulae up to a seed size of 17. Table 2 gives the numbers of these formulae. We ran experiments on all these formulae with all four systems. We found that both search steps and search time, as well as the proof length, follow a linear trend with the deep interaction system for this class of formulae (Fig. 8). In contrast, with shallow inference, proof times of these formulae increase rapidly following an exponential pattern. These results indicate that the normal formulae constitute a class that drastically distinguishes deeper inference in proof search performance.

8 Discussion

The interaction condition on context management provides some control on nondeterminism in deep inference proof search [11, 13, 15]. However, our results indicate that a naive use of this technique does not provide a computational advantage for proof search applications. The main advantage of the interaction technique manifests itself when it is coupled with the strength of deep inference in providing access to the shorter proofs that are otherwise, with shallow inference, impossible. Our results, however, indicate that shallow rule instances are needed to preserve completeness.

In most cases in our experiments, deep inference systems outperform shallow systems when the interactions between dual atoms are prioritised in deeper contexts. Because shorter proofs are constructed by annihilating the atoms as early as possible during the proof construction process, the size and length of the proofs obtained this way are also smaller. Constructing the proofs by starting from subformulae removes the redundant information that would otherwise

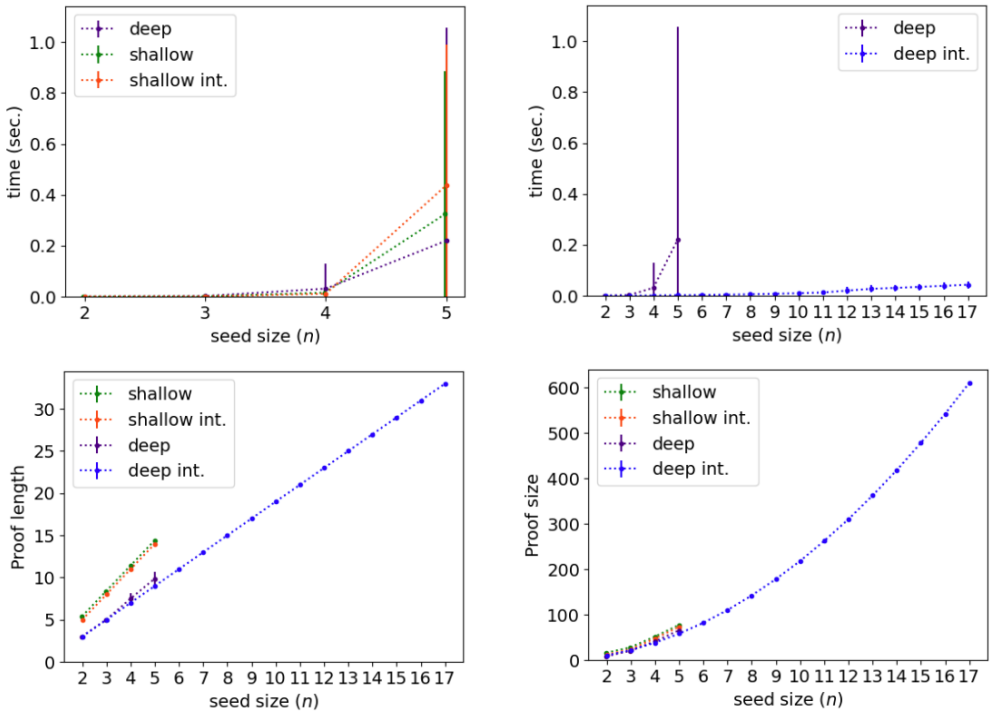

Figure 8: Mean search time, proof length, and proof size with depth-first search for all the n-seed normal formulae for $n \leq 17$. The search timed out for $n \geq 6$ for the deep system without interaction. For sorting the immediate breadth of the search space, the deeper inference strategy is used. The error bars indicate the standard deviation.

be carried along with the proof. In addition, checking the interaction condition for the redexes in deeper contexts is generally cheaper as the size of the subformulae gets smaller in deeper contexts. The empirical evidence in our results indicates that the deeper inference search strategy implements a mechanism that brings together these concepts and, in this way, delays exploring the complete search space, delivering shorter proofs quickly.

Although the deeper inference strategy delays the shallow rule instances, these instances still need to exist in the search space to conserve completeness. This has implications on performance, especially when non-provable formulae are considered. In applications, non-provable formulae can be accommodated with a time-out mechanism as in competitive first-order logic theorem provers. However, the theoretical baseline in deep inference proof search can be pushed further using a mechanism that couples splitting theorems [8, 10, 13, 15] and focusing [1, 5], the latter being orthogonal to the approach here that prioritises deeper rule instances. From the point of view of applications, exploiting the concurrency in proofs is a related topic that should provide performance improvements [12, 14], for example, in logic programming applications [21] and interactive and automated provers for different logics. Progress in these lines of research should also pave the way for applications of deep inference, especially for BV, which needs deep and shallow context management.

References

[1] Jean-Marc Andreoli. Focussing and proof construction. *Ann. Pure Appl. Logic*, 107(1-3):131–163, 2001.

[2] Kai Brünnler. Atomic cut elimination for classical logic. In M. Baaz and J. A. Makowsky, editors, *CSL 2003*, volume 2803 of *LNCS*, pages 86–97. Springer, 2003.

[3] Kai Brünnler. Locality for classical logic. *Notre Dame Journal of Formal Logic*, 47(4):557–580, 2006.

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

[5] 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.

[6] Anupam Das. On the relative proof complexity of deep inference via atomic flows. *Logical Methods in Computer Science*, 11(1:4):1–27, 2015.

[7] Stefano Guerrini. Correctness of multiplicative proof nets is linear. In *In Fourteenth Annual IEEE Symposium on Logic in Computer Science*, pages 454–463, 1999.

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

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

[10] Alessio Guglielmi and Lutz Strassburger. A system of interaction and structure V: The exponentials and splitting. *Mathematical Structures in Computer Science*, 21(3):563–584, 2010.

[11] Ozan Kahramanoğulları. System BV is NP-complete. *Annals of Pure and Applied Logic*, 152(1–3):107–121, 2008. Preliminary version appeared in the Proceedings of WoLLIC 2005.

[12] Ozan Kahramanoğulları. On linear logic planning and concurrency. *Information and Computation*, 207(11):1229–1258, 2009.

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

[14] Ozan Kahramanoğulları. True concurrency of deep inference proofs. In Jouko Vaananen, Asa Hirvonen, and Ruy de Queiroz, editors, *Logic, Language, Information, and Computation 23rd International Workshop, WoLLIC 2016, Puebla, Mexico, August 16-19th, 2016. Proceedings*, volume 9803 of *LNCS*. Springer, 2016.

[15] Ozan Kahramanoğulları. Deep proof search in MELL. In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12*, volume 46 of *EPiC Series in Computing*, pages 106–124. EasyChair, 2017.

[16] Ozan Kahramanoğulları. System BV without the equalities for unit. In *Proceedings of the 19th International Symposium on Computer and Information Sciences, ISCIS'04*, volume 3280 of *LNCS*, pages 986–995, Antalya, Turkey, 2004. Springer.

[17] Ozan Kahramanoğulları. Maude as a platform for designing and implementing deep inference systems. In *Proc. of the Eighth International Workshop on Rule-Based Programming, RULE'07*, volume 219 of *ENTCS*, pages 35–50. Elsevier, 2008.

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

[19] Max Kanovich. Horn programming in linear logic is NP-complete. In *Proccedings of the 7th Annual IEEE Symposium on Logic in Computer Science, Santa Cruz*, pages 200–210, Los Alamitos, California, 1992. IEEE Computer Society Press.

[20] Patrick Lincoln and Timothy C. Winkler. Constant-only multiplicative linear logic is NP-complete.

Theoretical Computer Science, 135(1):155–169, 1994.

[21] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51(1–2):125–157, 1991.

[22] Lê Thành Dũng (Tito) Nguyên and Lutz Strassburger. A system of interaction and structure III: the complexity of system BV and pomset logic. *Logical Methods in Computer Science*, 19(4):1–25, 2023.

[23] Christian Retoré. Pomset logic: A non-commutative extension of classical linear logic. In Ph. de Groote and J. R. Hindley, editors, *Typed Lambda Calculus and Applications, TLCA'97*, volume 1210 of *LNCS*, pages 300–318. Springer, 1997.

[24] Rubén Rubio. Maude as a library: An efficient all-purpose programming interface. In Kyungmin Bae, editor, *Rewriting Logic and Its Applications - 14th International Workshop, WRLA@ETAPS 2022, Munich, Germany, April 2-3, 2022, Revised Selected Papers*, volume 13252 of *Lecture Notes in Computer Science*, pages 274–294. Springer, 2022.

[25] Lutz Straßburger. A local system for linear logic. In M. Baaz and A. Voronkov, editors, *LPAR 2002*, volume 2514 of *LNAI*, pages 388–402. Springer, 2002.

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

[27] Lutz Straßburger. System NEL is undecidable. In Ruy De Queiroz, Elaine Pimentel, and Lucília Figueiredo, editors, *10th Workshop on Logic, Language, Information and Computation (WoLLIC)*, volume 84 of *Electronic Notes in Theoretical Computer Science*, 2003.

[28] Lutz Strassburger and Alessio Guglielmi. A system of interaction and structure IV: The exponentials and decomposition. *ACM Trans. on Comp. Logic*, 12(4), 2011.

[29] Alwen Fernanto Tiu. A local system for intuitionistic logic. In Miki Hermann and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, Proceedings of the 13th International Conference, LPAR 2006, Phnom Penh, Cambodia*, volume 4246 of *LNCS*, pages 242–256. Springer, 2006.

[30] Alwen Fernanto Tiu. A system of interaction and structure II: the need for deep inference. *Logical Methods in Computer Science*, 2 (2:4):1–24, April 2006.

9 Appendix: Data Used in the Figures

shallow, all formulae, breadth-first, Fig. 6

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	6.0	0.577	0.0003	0.0002	10.0	0.0	5.0	0.0
3	53	59.245	22.333	0.0027	0.0013	22.113	0.462	8.0	0.0
4	665	710.759	527.719	0.0314	0.0262	42.378	0.783	11.0	0.0

shallow int., all formulae, breadth-first, Fig. 6

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	6.0	0.577	0.0004	0.0002	10.0	0.0	5.0	0.0
3	53	59.245	22.333	0.0027	0.0013	22.113	0.462	8.0	0.0
4	665	710.759	527.719	0.0315	0.0262	42.379	0.784	11.0	0.0

deep, all formulae, breadth-first, Fig. 6

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	2.3333	2.13437	0.0011	0.0007	8.0	2.0	2.5	0.5
3	53	52.811	66.8964	0.0154	0.0222	17.7358	3.7475	4.1887	0.7283
4	665	3307.85	7253.19	4.389	41.0617	31.2992	5.6936	5.9835	0.8816

deep int., all formulae, breadth-first, Fig. 6

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	1.6666	0.7453	0.0007	0.0002	8.0	2.0	2.5	0.5
3	53	29.113	28.017	0.0098	0.0113	17.7358	3.74745	4.1887	0.7283
4	665	1321.51	2377.51	0.8079	2.6450	31.2992	5.6936	5.9835	0.8816

shallow, all formulae, depth-first, deeper inference, Fig. 7

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	4.0	0.0	0.0007	0.0002	10.0	0.0	5.0	0.0
3	53	8.774	2.447	0.0012	0.0004	22.1132	0.4622	8.0	0.0
4	665	156.914	137.225	0.0146	0.0135	45.1098	1.0809	11.0	0.0
5	10042	3105.81	4156.83	0.3746	0.8446	72.0295	2.8973	14.0	0.0

shallow int., all formulae, depth-first, deeper inference, Fig. 7

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	4.0	0.0	0.0005	0.0003	10.0	0.0	5.0	0.0
3	53	8.283	2.0223	0.0010	0.0003	22.113	0.4621	8.0	0.0
4	665	120.206	95.6769	0.0122	0.0101	44.902	1.082	11.0	0.0
5	10042	1907.58	2427.16	0.2795	0.5054	71.724	2.999	14.0	0.0

deep, all formulae, depth-first, deeper inference, Fig. 7

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	1.8333	1.067	0.0003	0.0001	8.0	2.0	2.5	0.5
3	53	7.9811	11.64	0.0012	0.0017	18.075	3.8453	4.1886	0.7283
4	665	43.426	136.74	0.0083	0.029	33.028	6.5855	6.0661	0.9679
5	10042	229.39	783.72	0.2102	14.290	52.414	9.3063	8.0438	1.1468

deep int., all formulae, depth-first, deeper inference, Fig. 7

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	1.5	0.5	0.0002	6.755	8.0	2.0	2.5	0.5
3	53	3.3019	1.020	0.0005	0.0002	17.7358	3.7474	4.1886	0.7283
4	665	6.0751	5.376	0.0013	0.001	31.5579	5.8216	6.0090	0.8791
5	10042	12.2749	32.019	0.0033	0.0073	49.350	8.0574	7.8912	0.9850
6	174079	31.6363	197.639	0.01166	0.0818	71.119	10.5108	9.8061	1.0680

deep, depth-first, all formulae, deeper inference reverse

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	4.1666	2.7938	0.0004	0.0002	11.3333	2.9814	3.3333	0.745
3	53	85.566	118.016	0.0161	0.024	30.2264	6.7924	6.0378	1.132
4	665	1340.65	2192.82	0.3644	0.9551	53.2583	9.9515	8.192	1.217

deep int., all formulae, depth-first, deeper inference reverse

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	6	3.5	1.5	0.0003	7.885	11.3333	2.981	3.333	0.7453
3	53	54.1886	60.0654	0.0056	0.0067	29.622	6.961	5.9622	1.1320
4	665	1603.16	3058.83	0.2817	0.817	56.544	11.70251	8.6256	1.4151

shallow, normal formulae, depth-first, deeper inference, Fig. 8

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	3	4.0	0.0	0.00034	0.0002	10.0	0.0	5.0	0.0
3	8	10.0	2.4495	0.00075	0.0002	22.0	0.0	8.0	0.0
4	21	350.714	161.033	0.0153	0.0124	45.428	0.904	11.0	0.0
5	52	8699.56	4726.35	0.3241	0.561	71.0	2.0	14.0	0.0

shallow int., normal formulae, depth-first, deeper inference, Fig. 8

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	3	4.0	0.0	0.0005	0.00034	10.0	0.0	5.0	0.0
3	8	9.5	2.784	0.0007	0.00019	22.0	0.0	8.0	0.0
4	21	260.91	107.777	0.01123	0.0084	45.333	0.9428	11.0	0.0
5	52	7825.92	4898.05	0.4371	0.55412	71.115	2.1983	14.0	0.0

deep, normal formulae, depth-first, deeper inference, Fig. 8

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	3	2.666	0.9428	0.0005	0.0001	10.0	0.0	3.0	0.0
3	8	16.25	21.6376	0.0024	0.0030	22.25	0.662	5.0	0.0
4	21	207.952	566.76	0.0313	0.0988	42.7619	5.290	7.476	0.663
5	52	832.730	2304.62	0.2196	0.8384	67.3076	7.685	9.866	0.760

deep int., normal formulae, depth-first, deeper inference, Fig. 8

seed size	# form.	Search steps	steps SD	time (sec.)	time SD	Proof size	size SD	Proof length	length SD
2	3	2.0	0.0	0.0003	0.0002	10.0	0.0	3.0	0.0
3	8	4.0	0.0	0.0007	0.0001	22.0	0.0	5.0	0.0
4	21	6.0	0.0	0.0011	0.0004	38.0	0.0	7.0	0.0
5	52	8.0	0.0	0.0019	0.0008	58.0	0.0	9.0	0.0
6	124	10.0	0.0	0.0030	0.0015	82.0	0.0	11.0	0.0
7	284	12.0	0.0	0.0041	0.0017	110.0	0.0	13.0	0.0
8	629	14.0	0.0	0.0056	0.0022	142.0	0.0	15.0	0.0
9	1352	16.0	0.0	0.0075	0.0029	178.0	0.0	17.0	0.0
10	2829	18.0	0.0	0.0098	0.0037	218.0	0.0	19.0	0.0
11	5777	20.0	0.0	0.0125	0.0047	262.0	0.0	21.0	0.0
12	11544	22.0	0.0	0.0201	0.0138	310.0	0.0	23.0	0.0
13	22620	24.0	0.0	0.0270	0.0144	362.0	0.0	25.0	0.0
14	43529	26.0	0.0	0.0307	0.0127	418.0	0.0	27.0	0.0
15	82409	28.0	0.0	0.0347	0.0123	478.0	0.0	29.0	0.0
16	153677	30.0	0.0	0.0387	0.0127	542.0	0.0	31.0	0.0
17	282634	32.0	0.0	0.0432	0.0136	610.0	0.0	33.0	0.0