Implementation and Evaluation of Contextual Natural Deduction for Minimal Logic

Bruno Woltzenlogel Paleo

Theory and Logic Group
Vienna University of Technology
Vienna, Austria
bruno@logic.at

Abstract. The *contextual* natural deduction calculus ($\mathbf{ND^c}$) extends the usual natural deduction calculus (**ND**) by allowing the implication introduction and elimination rules to operate on formulas that occur inside contexts. It has been shown that, asymptotically in the best case, $\mathbf{ND^c}$-proofs can be quadratically smaller than the smallest **ND**-proofs of the same theorems. In this paper we describe the first implementation of a theorem prover for minimal logic based on $\mathbf{ND^c}$. Furthermore, we empirically compare it to an equally simple **ND** theorem prover on thousands of randomly generated conjectures.

1 Introduction

Natural deduction was introduced by Gentzen in [7] and one of its distinguishing features is that the meaning of a logical connective is determined by elimination and introduction rules, and not by axioms. As a result, formal natural deduction proofs are considered to be similar in structure to their informal counterparts and hence more *natural*. This subjective claim is corroborated by the observation that widely used proof assistants[1] follow a natural deduction style.

However, as exemplified in [11], the inference rules of natural deduction style calculi can be inconvenient, lengthy and ultimately unnatural for formalizing reasoning steps that modify a deeply located subformula of a formula, such as: skolemization, double negation elimination, quantifier shifting, prenexification... Because these deep reasoning steps are commonly used by automated deduction tools during preprocessing of the theorem to be proved, the resulting proofs may contain deep inferences [6]. Therefore, automatically replaying (i.e. reproving) these proofs in proof assistants (e.g. when an automated deduction tool is integrated within a proof assistant [1]) can be inefficient in terms of proving time and size of the generated shallow proof.

These challenges motivated the invention (in [11]) of the *contextual natural deduction calculus* ($\mathbf{ND^c}$), which is a simple extension of the usual natural deduction calculus (here called **ND**) allowing introduction and elimination rules

[1] e.g. Isabelle (`www.cl.cam.ac.uk/research/hvg/Isabelle/`) and Coq (`http://coq.inria.fr`).

to operate on formulas occurring inside contexts. The goals in [11] were purely theoretical. It was shown that $\mathbf{ND^c}$ is sound and complete, that proofs can be normalized, and that some proofs can be quadratically smaller than the smallest proofs of the same theorem in the usual natural deduction calculus. In contrast, the main goal of the work reported in the present paper is to evaluate $\mathbf{ND^c}$ empirically. This is important, because asymptotic proof-complexity results can be misleading when the assymptotic behaviour they describe for particular worst cases or best cases is not observed in cases that occur most often in practice.

$\mathbf{ND^c}$ can be regarded not only from a theorem proving perspective but also from a *proof compression* point of view: a given **ND**-proof ψ could be compressed by transforming it to a smaller $\mathbf{ND^c}$-proof. Since every **ND**-proof is also an $\mathbf{ND^c}$-proof, a straightforward proof compression algorithm could simply try to reprove ψ's theorem using an $\mathbf{ND^c}$ theorem prover.

The implementation of prototypical theorem provers based on **ND** and $\mathbf{ND^c}$ within the `Skeptik` framework (`github.com/Paradoxika/Skeptik`) is discussed in Section 3. These provers are restricted to minimal logic (intuitionistic logic having only the implication connective). Although it would be straightforward to extend the contextual techniques to inference rules for other connectives as well, the restriction to minimal logic implies less implementation effort and is sufficient to estimate how promising the idea of contextual natural deduction might be in practice. An experimental infra-structure, including a random formula generator, also had to be implemented, as briefly described in Section 4. The experimental results are shown and analyzed in Section 5.

Related work: due to the increasing maturity of automated deduction tools, there has been a lot of recent work on the development of algorithms for simplifying the generated proofs in a post-processing phase. These methods have focused mostly on propositional resolution proofs output by SAT- and SMT-solvers so far, but generalizations to first-order resolution have been proposed as well. There are also algorithms aimed at compressing and structuring sequent calculus proofs by eliminating or introducing cuts [13, 10] or by extracting Herbrand sequents from proofs.[11] is probably the first work considering, from a theoretical perspective, the compressibility of natural deduction proofs, and the present paper reports the first realization of this idea in practice. Contextual inferences have a lot in common with the related idea of *deep inference*, which has been intensively investigated in the last decade, especially for classical logic (e.g. [2, 5, 4, 9]) but also for intuitionistic logic [12, 3, 8]. Despite the technical differences, deep inference calculi were an inspiration for the development of contextual natural deduction.

2 Contextual Natural Deduction

In this paper a *derivation* is a tree of inferences (instances of the inference rules), operating on sequents of the form $\Gamma \vdash t : T$, where Γ is a (possibly empty) set of named *hypotheses* $h_1 : H_1, \ldots, h_n : H_n$, t is a (contextual) lambda term (whose free variables are among the names in Γ and whose bound variables are assumed to have unique names) and T is a minimal logic formula (or equivalently, by the Curry-Howard isomorphism, the type of t). A derivation ψ is a *proof* of a theorem T if and only if its leaves are axiom inferences and it ends in $\vdash t : T$, for some term t. Figure 1 shows the inference rules of the *contextual natural deduction calculus* $\mathbf{ND^c}$ along with a corresponding extension of the lambda calculus. $\mathbf{ND^c}$ extends $\mathbf{ND}$ by allowing the inference rules to operate on subformulas located deeply inside the premises. The notation $\mathcal{C}_\pi[F]$ indicates a formula that has the subformula F in position π. $\mathcal{C}_\pi[\_]$ is called the *context* of F in the formula $\mathcal{C}_\pi[F]$. A *position* π is encoded as a binary string indicating the path from the root of $\mathcal{C}_\pi[F]$ to F in the tree structure of $\mathcal{C}_\pi[F]$; thus, a subformula at position π of a formula P, denoted $\mathrm{At}_\pi(P)$, can be retrieved by traversing the formula according to the following inductive definition:

$$\mathrm{At}_\epsilon(A) = A \qquad \mathrm{At}_{0\pi}(A \to B) = \mathrm{At}_\pi(B) \qquad \mathrm{At}_{1\pi}(A \to B) = \mathrm{At}_\pi(A)$$

A position is said to be *positive* (*negative*) if and only if it contains an even (odd) number of digits 1. In other words, in the tree structure of a formula, a node and its left (right) child always occupy positions with opposite (same) polarities, and the root position is positive. Moreover, a position is *strongly positive* if and only if it does not contain any digit 1. $\mathbf{ND^c}$ has two implication elimination rules because the contexts can be combined in two different ways. The usual natural deduction calculus $\mathbf{ND}$ can be considered a restriction of $\mathbf{ND^c}$ enforcing empty contexts. For convenience, the rules of $\mathbf{ND}$ are shown in the appendix.

Note: π, π_1 and π_2 must be positive positions.

$$\frac{}{\Gamma, a : A \vdash a : A}$$

$$\frac{\Gamma, a : A \vdash b : \mathcal{C}_\pi[B]}{\Gamma \vdash \lambda_\pi a^A.b : \mathcal{C}_\pi[A \to B]} \to_I (\pi)$$

Contextual Soundness Condition:
a is allowed to occur in b only if π is strongly positive.

$$\frac{\Gamma \vdash f : \mathcal{C}^1_{\pi_1}[A \to B] \qquad \Gamma \vdash a : \mathcal{C}^2_{\pi_2}[A]}{\Gamma \vdash (f\ a)^{\rightharpoonup}_{(\pi_1;\pi_2)} : \mathcal{C}^1_{\pi_1}[\mathcal{C}^2_{\pi_2}[B]]} \to^{\rightharpoonup}_E (\pi_1;\pi_2)$$

$$\frac{\Gamma \vdash f : \mathcal{C}^1_{\pi_1}[A \to B] \qquad \Gamma \vdash a : \mathcal{C}^2_{\pi_2}[A]}{\Gamma \vdash (f\ a)^{\leftharpoonup}_{(\pi_1;\pi_2)} : \mathcal{C}^2_{\pi_2}[\mathcal{C}^1_{\pi_1}[B]]} \to^{\leftharpoonup}_E (\pi_1;\pi_2)$$

Fig. 1: The contextual natural deduction calculus $\mathbf{ND^c}$

3 Implementation

Implementation was done in `Scala`, leveraging and extending the `Skeptik` proof compression library. Although `Skeptik`'s original focus was on propositional resolution proofs, the addition of data structures for natural deduction was easy and required no refactoring, because `Skeptik` has always taken advantage of `Scala`'s object-orientation features to be agnostic with respect to proof systems.

In `Skeptik`, inference rules are classes. In order to increase the confidence on the correctness of inference rules, each rule class includes correctness checking code and is kept as small as possible. The 3 classes for the **ND** rules are only 13 lines long. The single class `ImpElimC` for the two contextual implication elimination rules has 17 lines and the soundness condition for the `ImpIntroC` rule is a 10-line long trait. To the extent that these few lines of code are trusted, any proof constructed using these inference rules is correct. Any code that is not essential to the rule is written not in the class but in its companion object.

The class `SimpleProver` implements a theorem prover that is generic in the sense that it takes arbitrary (companion objects of) inferences rules and then performs bottom-up proof search using the given inference rules. For each open goal, the prover tries all inference rules in a bottom-up manner in parallel, generating all possible subgoals. Then it recursively tries to prove the subgoals in parallel. When returning from the recursion, the prover chooses the smallest subproof among all alternative subproofs returned by the recursive calls. This exhaustive search strategy is appropriate in the context of proof compression, where the goal is to find small proofs not necessarily as fast as possible. The depth of the recursion is bounded by the maximum proof height specified as a parameter of `SimpleProver`.

The companion objects of the inference rules implement a standard interface, which provides methods that generate subgoals as required by the prover and reconstruct the proof when the subgoals are proved. In the case of **ND**'s implication elimination rule, when generating subgoals for a goal of the form $\Gamma \vdash B$, it is necessary to guess a formula A in order to generate the subgoals $\Gamma_1 \vdash A \rightarrow B$ and $\Gamma_2 \vdash A$. Exhaustively guessing all possible formulas would be inefficient. Instead, the rule searches for hypotheses of the form $D_1 \rightarrow (\ldots \rightarrow (D_k \rightarrow B)\ldots)$ in Γ and then generates the subgoals $\Gamma \vdash (D_k \rightarrow B)$ and $\Gamma \vdash D_k$. Although proof search is still complete under this restriction, the proofs it finds are always normalized. Consequently, a proof found by this procedure is not necessarily the smallest possible proof, because sometimes non-normal proofs can be smaller. In the case of $\mathbf{ND^c}$ rules, the generation of subgoals is complicated further by the need to take positions into account. For a goal of the form $\Gamma \vdash F$, the contextual implication elimination rule first searches for all positive positions π_1 and π_2 such that $F = \mathcal{C}_{\pi_1}[\mathcal{C}_{\pi_2}[B]]$ for some B. Then it searches for a hypothesis of the form $\mathcal{C}_{\pi_1}[D_1 \rightarrow (\ldots \rightarrow (D_k \rightarrow B)\ldots)]$ (or $\mathcal{C}_{\pi_2}[D_1 \rightarrow (\ldots \rightarrow (D_k \rightarrow B)\ldots)]$) and, if it succeeds, it generates the subgoals $\Gamma \vdash \mathcal{C}_{\pi_1}[(D_k \rightarrow B)]$ and $\Gamma \vdash \mathcal{C}_{\pi_2}[D_k]$ (or, respectively, $\Gamma \vdash \mathcal{C}_{\pi_2}[(D_k \rightarrow B)]$ and $\Gamma \vdash \mathcal{C}_{\pi_1}[D_k]$).

4 Experimental Setup

In order to evaluate the provers, a random formula generator was implemented. It takes a desired size s and a desired number of distinct atomic formulas q as input. Then it generates a list of length s containing q distinct atomic formulas. The list is grown recursively, and at each iteration, every atomic formula is equally likely to be selected. At this stage, care is taken to avoid generating formulas that are isomorphic modulo variable renaming (e.g. $A \to B$, $B \to A$, $A \to C$, $\dots$; only $B \to A$ can be generated). Subsequently, the generator transforms this list into a minimal logic formula by recursively introducing implications at random positions in the list. The positions are equally likely to be selected (i.e. $A \to (A \to A)$ and $(A \to A) \to A$ are equally probable to be generated).

The experiments varied the value of s from 3 to 15 and the value of q from 1 to $s - 1$. For each pair of values (s, q), 1000 formulas were generated, except for small values of s and q, for which there are less than 1000 distinct formulas that could be generated. In total, 76755 formulas were generated.

For each generated formula f, the **ND** prover with a timeout of 30 seconds and a maximum proof height of 20. The $\mathbf{ND^c}$ prover, on the other hand, had a timeout of 300 seconds and a maximum proof height of $h + 1$, where h is the height of proof of f found by the **ND** prover. The larger timeout was chosen because $\mathbf{ND^c}$'s contextual rules clearly result in a larger search space, with more subgoals to try. With the larger timeout, it is possible to measure the impact of the larger search space in the proof search time.

5 Results of the Experiments

30127 formulas were proved by the **ND** prover. 46628 formulas were shown to be countersatisfiable by the **ND** prover, because it terminated before the timeout exhausting the proof search space without finding a proof. There was no case of timeout for the **ND** prover. There were 533 cases of timeout for the $\mathbf{ND^c}$ prover.

Among the 29594 formulas on which both provers were successful, 2557 (8.49%) had shorter proofs in $\mathbf{ND^c}$ than in **ND**. The total length of the $\mathbf{ND^c}$-proofs was 2.97% lesser than the total length of the proofs found by the **ND** prover on all 29594 formulas. The total length of the $\mathbf{ND^c}$-proofs was 27.8% lesser than the total length of the **ND**-proofs on the 2557 proofs that admit shorter $\mathbf{ND^c}$ proofs.

In Figure 2a, each dot represents a generated formula and its position indicates the length of the proof found by each prover. Overlapped dots are shown as darker dots. In the standard box-whiskers plot of Figure 2b, formulas have been grouped by the length of their **ND**-proofs. For each group, the chart shows the median length of $\mathbf{ND^c}$-proofs, as well as the quantiles, fences and outliers. This chart, together with the bar charts in Figure 3 indicate a dependence of the compressibility of proofs on the length. Figure 3a shows that the proportion of formulas that admit shorter $\mathbf{ND^c}$-proofs (i.e. whose **ND**-proofs could be compressed to shorter $\mathbf{ND^c}$-proofs) tends to grow with the length of the **ND**-proofs. In a larger proof, the likelihood of an opportunity for compression is greater; however, the compression might be less significant in comparison to the proof length, as indicated by the decreasing trend in Figure 3b.

The 3D-charts in Figure 4 shed further light on what influences the proportion of formulas that admit shorter $\mathbf{ND^c}$-proofs and the total compression ratios (including all 29594 formulas). The proportion of formulas admitting shorter $\mathbf{ND^c}$-proofs (and their total compression ratios) is higher the lower the number of distinct atoms they contain and the larger they are. The depths[2] of the formulas also seem to play a role, with greater compression proportions and total compression ratios for intermediary depth values.

Figure 2c (which takes into account all 76755 formulas) shows that the **ND** prover rarely took longer than 10 milliseconds on a formula. Surprisingly, the $\mathbf{ND^c}$ prover was faster than the **ND** prover in the majority of the cases (46733 formulas; 60.88%), probably due to the stricter upper-bound on proof height. However, when the $\mathbf{ND^c}$ prover was slower, it tended to be significantly slower, as shown in the Figure.

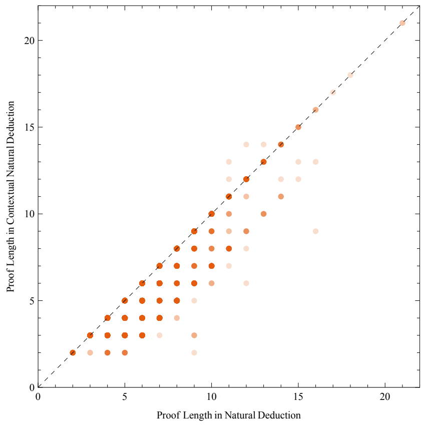
6 Conclusions

The empirical investigation reported in this paper confirms the expectation that $\mathbf{ND^c}$ can provide shorter proofs in a significant number of cases. This complements and strengthens the previous asymptotic complexity theorem proved in [11], which showed the $\mathbf{ND^c}$-proofs could be quadratically shorter than **ND**-proofs for one particular sequence of proofs. The price to pay for shorter proofs is currently a much longer proving time.

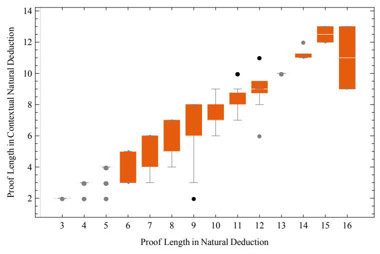
References

1. Sascha Böhme and Tobias Nipkow. Sledgehammer: Judgement day. In Jürgen Giesl and Reiner Hähnle, editors, *IJCAR*, volume 6173 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2010.
2. Kai Brünnler. Atomic cut elimination for classical logic. In Matthias Baaz and Johann A. Makowsky, editors, *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 2003.
3. Kai Brünnler and Richard McKinley. An algorithmic interpretation of a deep inference system. In *15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 5330 of *LNCS*, pages 482–496. Springer, 2008.
4. Paola Bruscoli and Alessio Guglielmi. On the proof complexity of deep inference. *ACM Transactions on Computational Logic*, 10:1–34, 2009.
5. Paola Bruscoli, Alessio Guglielmi, Tom Gundersen, and Michel Parigot. A quasipolynomial cut-elimination procedure in deep inference via atomic flows and threshold formulae. In Edmund M. Clarke and Andrei Voronkov, editors, *LPAR (Dakar)*, volume 6355 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2010.

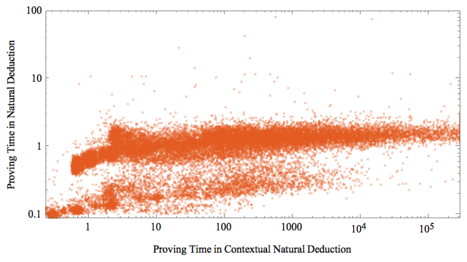
[2] $\text{depth}(A) = 1$, for an atomic A; $\text{depth}(B \to C) = \max(\text{depth}(B), \text{depth}(C)) + 1$.



(a)

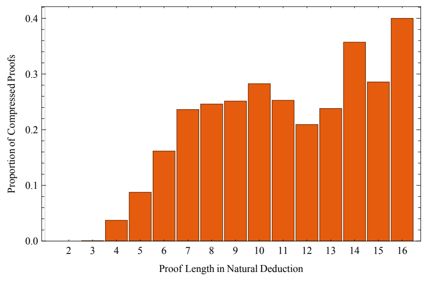


(b)

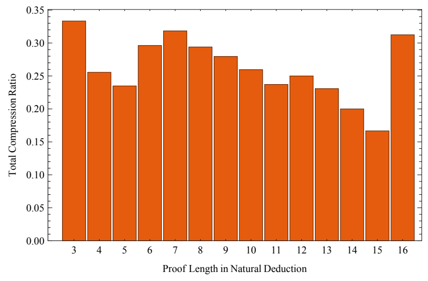


(c)

Fig. 2



(a)



(b)

Fig. 3: Bar Charts

6. David Deharbe, Pascal Fontaine, and Bruno Woltzenlogel Paleo. Quantifier inference rules in the proof format of verit. In *1st International Workshop on Proof Exchange for Theorem Proving*, 2011.

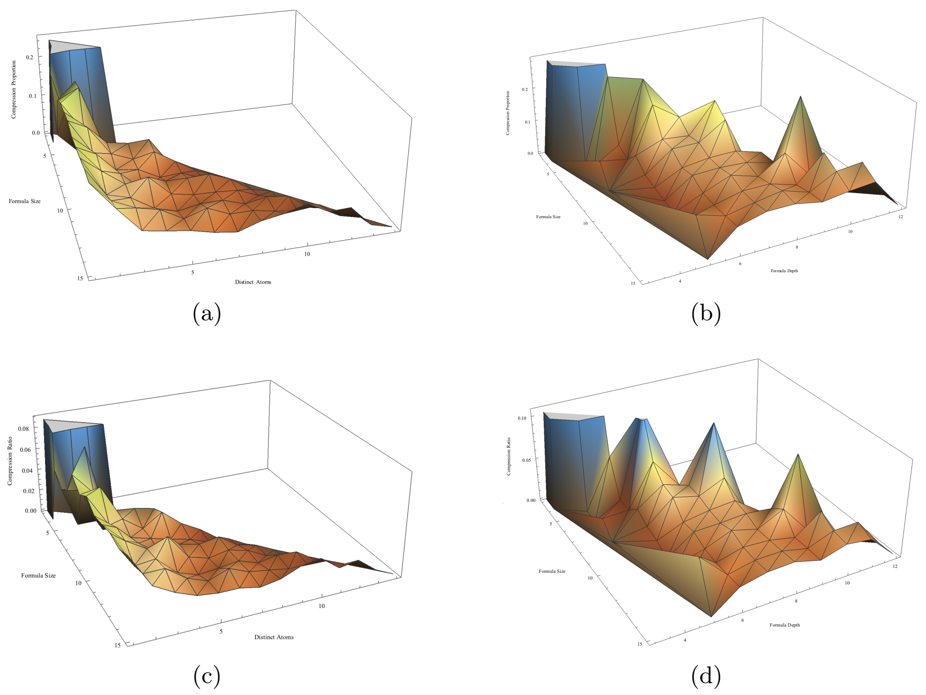


Fig. 4: proportion of compressed proofs and total compression ratio as functions of formula size, formula depth and number of distinct atoms.

7. G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1934–1935.
8. Nicolas Guenot. Nested proof search as reduction in the lambda-calculus. In Peter Schneider-Kamp and Michael Hanus, editors, *PPDP*, pages 183–194. ACM, 2011.
9. Alessio Guglielmi. A system of interaction and structure. *CoRR*, cs.LO/9910023, 1999.
10. Stefan Hetzl, Alexander Leitsch, and Daniel Weller. Towards algorithmic cut-introduction. In *LPAR*, 2012.
11. Bruno Woltzenlogel Paleo. Contextual natural deduction. In *Logical Foundations of Computer Science, International Symposium, LFCS 2013, San Diego, CA, USA, January 6-8, 2013. Proceedings*, pages 372–386, 2013.
12. Alwen Tiu. A local system for intuitionistic logic. In Miki Hermann and Andrei Voronkov, editors, *LPAR*, volume 4246 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2006.
13. Bruno Woltzenlogel Paleo. Atomic cut introduction by resolution: Proof structuring and compression. In Edmund Clarke and Andrei Voronkov, editors, *16th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, number 6355 in LNAI, pages 463 – 480. Springer, 2010.

A Appendix

$$\frac{}{\Gamma, a : A \vdash a : A}$$

$$\frac{\Gamma, a : A \vdash b : B}{\Gamma \vdash \lambda a^{A}.b : A \to B} \to_I$$

$$\frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash (f\ a) : B} \to_E$$

Fig. 5: The natural deduction calculus **ND**