Computational complexity of knotted substructural logics

Nick Galatos[1], Vitor Greati[2], Revantha Ramanayake[3], and Gavin St. John[4]

[1] University of Denver, ngalatos@du.edu
[2] University of Groningen, v.rodrigues.greati@rug.nl
[3] University of Groningen, d.r.s.ramanayake@rug.nl
[4] University of Cagliari, gavinstjohn@gmail.com

Abstract

We establish the exact computational complexity of knotted substructural logics. We prove that it is exactly at the Ackermannian level of the fast-growing computational hierarchy; in particular, it is decidable but not primitive recursive. To establish the upper bound we undertake a complicated proof-theoretic analysis, based on analytic sequent calculi. For the the lower bound, we encode suitable versions of and-branching counter machines.

1 Introduction

Substructural logics are generalizations of classical logic. They are usually formulated as extensions of the sequent calculus **FL**, which is similar to Gentzen's calculus **LK** for classical logic and **LJ** for intuitionistic logic, but the three basic structural rules of exchange, contraction and weakening are omitted; substructural logics owe their name to this the fact, while one or more of these basic structural rules (or other structural rules) may be added back to **FL**, resulting in various substructural logics. That these logics may lack one or more of the three basic rules, allows for reasoning that is sensitive to the multiplicity and order of propositions. As a result, they are useful for settings where propositions correspond to resources or to pieces of information.

Notably, substructural logics, apart from classical logic (obtained by reitroducing the three basic structural rules and double-negation), include intuitionistic logic (constructive mathematics), relevance logic (philosophy), many-valued logic (engineering), linear logic (functional programming), and separation logic (dynamic memory allocation and pointer management in computer science), among others.

Moreover, derivation systems coming from mathematical linguistics (relating to context-free grammars and automata theory) are obtained as variants of the system **FL**, so methods and tools of structural proof theory can be applied to these contexts, as well. The system **FL** (as well as many of its extensions) enjoys cut elimination and an accompanying extension of λ-calculus enjoys strong normalization, thus also assigning computational meaning to the calculus.

Substractural logics are algebraizable, and their algebraic semantics are known as *residuated lattices*. The latter have an independent history, first starting in classical algebra, as lattices of ideals of rings (under usual multiplication and division of ideals). They also include other diverse structures such as lattice-ordered groups (where group division realizes the implication connective) and algebras of binary relations (more generally, Tarski's relation algebras). Being semantics of substructural logics, they also include Boolean and Heyting algebras, MV-algebras, Sugihara monoids, and bunched-implication algebras, among others. We refer to [5] for the fundamentals of the theory of substructural logics and their algebraic semantics.

The three structural rules of exchange, contraction and weakening are special cases of *knotted* inference rules and they have a rich history of proof-theoretic analysis. Adding various such knotted rules, allows for fine tuning the different levels of control of resources in the calculus. In this work we study such knotted extensions of **FL**; the results we report are part of [3].

2 Knotted rules

The connections and applications of substructural logics to computer science, to classical and order algebra, and to mathematical linguistics, mentioned above, pertain even to the propositional versions of these logics. In particular, similar to modal logic, propositional substructural logics can be thought of as well-behaved fragments of first-order classical logic. They are decidable at times, but also undecidable at others. For example, provability/theoremhood of **FL** is decidable, but deducibility is undecidable. This corresponds to the fact that the variety of residuated lattices has a decidable equational theory, but an undecidable quasi-equational theory (even undecidable word problem). We prove that extensions with knotted rules have complexity exactly at the Ackermannian level of the fast-growing computational hierarchy.

In **LK** and **LJ** the metalogical comma separator on the left-hand side of sequents $\alpha_1, \alpha_2, \ldots, \alpha_n \Rightarrow \alpha$ corresponds to logical conjunction; in the absence of the three structural rules, this is not true any more. A new strong conjunction (also known as fusion or multiplication) can be conservatively added and it can be shown that it corresponds to the comma. In interpretations where formulas represent resources or pieces of information, $\phi \cdot \psi$ is not equivalent to $\phi \wedge \psi$, and even $\phi \cdot \phi$ has more information or resource content than ϕ; we use ϕ^n for repeated applications of multiplication, as usual. Also, due to the lack of exchange, commutativity of comma (and multiplication) fails in general, thus implication splits into two (left and right) implication connectives. In linguistic contents, where ϕ and ψ represent phrases in natural language, $\phi \cdot \psi$ has a different meaning than $\psi \cdot \phi$, either due to grammatical rules or due to implied temporal precedence. In these contexts, phrases with linguistic type $\phi \to \psi$ expect a phrase of type ϕ on the left to yield a phrase of type ψ, in symbols $\phi \cdot (\phi \to \psi) \Rightarrow \psi$ (which is an internationalization of Modus Ponens and is a derivable sequent in **FL**) but $(\phi \to \psi) \cdot \phi \Rightarrow \psi$ is rejected in languages where word order is important. For this reason we use two implication/-division connectives to get $\phi \cdot (\phi \backslash \psi) \Rightarrow \psi$ and $(\psi / \phi) \cdot \phi \Rightarrow \psi$. In the context of lattice-ordered groups (where again idempotency and commutativity are absent) these implications are definable as usual by $\phi \backslash \psi = \phi^{-1} \cdot \psi$ and $\psi / \phi = \psi \cdot \phi^{-1}$. If formulas are interpreted as ideals in noncommutative rings, then the usual left and right division of ideals correspond to these two connectives.

Knotted axioms have the form $\phi^n \Rightarrow \phi^m$ for different values of n, m; in particular, $\phi^1 \Rightarrow \phi^2$ is equivalent to the structural rule of contraction, while $\phi^1 \Rightarrow \phi^0$ is equivalent to the structural rule of weakening. The algebraic rendering of a knotted rule is the inequality $x^n \leq x^m$ and can be thought of as a controlled and predescribed version/generalization of contraction and weakening that is permitted in the system. Now, it turns out that adding $\phi^n \Rightarrow \phi^m$ to **FL** breaks cut elimination, but [1] describes a multi-step process for converting it to a structural rule that yields an analytic sequent calculus, thus making such knotted extensions $\mathbf{FL}^{n,m}$ amenable to the tools of proof theory.

3 Computational complexity: upper bounds

The further extension $\mathbf{FL}_{\mathbf{e}}^{n,m}$ with exchange is also analytic (has cut elimination and the subformula property). To establish an upper complexity bound we undertake a complicated proof-theoretic analysis. In particular, we first show that despite the fact that a naive inverse proof search in the (cut free) calculus $\mathbf{FL}_{\mathbf{e}}^{n,m}$ starting with a given end-sequent does not terminate, there is a way to restrict to an equivalent subclass of well-behaved proofs where termination is guaranteed. This is done by defining a relation on the (infinite) set of all sequents using formulas from the end sequent and showing that this yields a (not necessarily linear) well quasi-ordered

set (wqo). By proving an analogue of Curry's lemma, this helps with showing decidability, but not with controlling the complexity.

Toward bounding the computational complexity, we define a norm on the sequents of the wqo and we show that even though the norm does not decrease as we read branches of the proof upward, still the increase in the norm can be bounded by a suitable function in the following sense. There is no such function that gives a universal bound (i.e., even though all bad sequences in the wqo are finite, there is no bound on the length of a bad sequence in the wqo), but if we introduce one more dependence on a parameter that depends on certain features of the end sequent, then there is a function (with one more argument) that does control this growth in the norms.

On the way we develop parts of the theory or wqo sets that is new and needed for our case, we develop parts of the theory or the subrecursive, Grzegorczyk and fast-growing hierarchies, and we obtain length theorems for the wqos that we study. Having controlled the parametrized growth of the norm along bad sequences, we undertake a further proof-theoretic analysis (introducing measures such as the formula multiplicity and the active component number, and performing various proof-theoretic reductions in auxiliary proof systems) in order to put a parameterized bound on the length of bad sequences and use it to further control the parameterized size of the whole inverse-proof tree. In the end we show that this size is at most Ackermannian in sense of the fast growing hierarchy (i.e., controlled by a function that contains a single application of Ackermann's function); these results are contained in Chapters 3, 4 and 5 of [3].

4 Computational complexity: lower bounds

For the lower bounds we encode suitable variants of counter machines. The hardware of a standard counter machine consists of a finite set of internal states (one designated as final) and a finite number of registers each able to hold a non-negative integer value. The software consists of instructions of three types: if in state q, change to state q' and increment by 1 the value of register i; if in state q, change to state q' and decrement by 1 the value of register i, if possible (i.e., if non zero); if in state q, and the value of register i is zero, then change to state q'. It is well known that counter machines with at least 3 registers are universal (they can simulate Turing machines), with 1 register are decidable, and with 2 registers they are undecidable but not universal (they cannot perform exponentiation, but by attaching a simple exponentiation machine they become universal). Note that counter machines are naturally nondeterministic as multiple instructions could be applicable to a configuration, and acceptance of an initial configuration is defined existentially, by some sequence of instructions that leads to the final state with all register values equal to zero (this is the final configuration). We can view the current configuration of the machine as a commutative monoid word $qr_1^{n_1} \cdots r_k^{n_k}$, where n_i denotes the non-negative integer stored in counter i. Then applications of increment and decrement instructions are compatible with multiplication, but the zero-test instruction is not.

A better behaved and equivalent variant is that of an and-branching counter machine (ACM) that works on more expanded hardware (multiple sets of counters) and replaces zero test instructions with forking instructions: if at state q, duplicate the whole register setup and copy the register contents to both setups, assigning state q' to the first setup and q'' to the second setup. Acceptance is defined conjunctively (no presence of non-determinism here) where all setups must lead to the final configuration. After multiple applications of forking (and other) instructions, we may be in a setup with multiple configurations that have resulted from suc-

cessive forkins; these can be represented by joins $C_1 \vee \ldots \vee C_n$ of configurations, are known as instantaneous descriptions, and form the expanded hardware of and-branching machines; as a result, instantaneous descriptions live in an idempotent semiring (with multiplication and join). The three types of instructions of these machines are compatible with multiplication and join, thus the new machines behave better than regular counter machines.

In [6], ACMs are encoded in the deducibility of $\mathbf{FL_e}$ (and most of its extensions, including knotted ones) thus establishing undecidability. As mentioned in the preceding section, $\mathbf{FL_e}^{n,m}$ is decidable, so it is impossible to encode in it arbitrary ACMs. However, we show that we can encode ACMs such that the values of the registers, during the whole computation, do not exceed the value of the Ackermann function on (length of) the initial configuration. This is done by adding to such an ACM a special counter, and appending an auxiliary machine (before the run of the main machine) that initializes the counter to the Ackermann value of the input, and a second auxiliary machine (after the run of the main machine) that zeros the counter. If the values of the registers in the main ACM exceeded the Ackermann bound, then this would be detected by one of the two auxiliary machines.

Further, we show that allowing applications of the knotted rule (which can be viewed as glitches that can spontaneously alter the contents of the counter between regular instructions) do not increase the accepted configurations; this is done by first investigating the wqo on the branches of the (forking) computation of the overall machine and distinguish two cases about the type of the knotted rule considered. To complete the argument that we do not over compute, we argue semantically. In detail, we employ the method of residuated frames developed in [4], which constitute relational semantics for substructural logics and have proved to be a versatile tool for establishing both proof-theoretic and algebraic results. These are two-sorted relational structures, where the two sorts correspond to the two sides of sequents in a calculus; in algebraic applications they correspond to join and meet irreducibles in a lattice. In our case, the first sort corresponds to configurations of the machine and the second sort to sections over them. All these results are contained in Chapters 8, 9 and 10 of [3]

5 Extensions

Having established exact/tight complexity bounds for all knotted extensions of $\mathbf{FL_e}$, we extend these results in multiple directions.

5.1 To the non-commutative setting

The removal of the exchange rule from the logics $\mathbf{FL_e}^{n,m}$ mentioned above results in logics of the form $\mathbf{FL}^{n,m}$; these are known to have undecidable deducibility relation. However, we identify more general versions of the exchange rule and establish our complexity results in this setting also. These generalizations of exchange correspond to monoid equations that allow conditional movement of variables and the study of their dynamics results in some finitistic aspects, which we employ. The complexity remains Ackermaniann in these more general settings; these results are contained in Chapter 6 of [3]. If the knotted rule is integrality, then without any form of exchange we can obtain upper complexity bounds (hyper-Ackermaniann); this result is contained in Chapter 7 of [3].

5.2 To hypersequents

In [1] the *substructural hierarchy* is defined in analogy to the arithmetical hierarchy, but by tracking alternations of positive and negative connectives instead of quantifiers. Sequent structural rules correspond to the level $\mathcal{N}_2$ of the hierarchy; to handle extensions by axioms in the $\mathcal{P}_3$ level, hypersequent calculi need to be employed. The resulting theory of such calculi and their relationship to semantics is established in [2]. Our work extends to the hypersequent setting in the sense that we can consider further extensions of $\mathbf{FL}_{\mathbf{e}}^{n,m}$ by arbitrary sets of structural hypersequent rules. This results in a jump of the complexity to the hyper-Ackermaniann level of the fast-growing hierarchy.

5.3 To increasing/decreasing rules

The lower complexity bounds we establish actually apply to a more general class than knotted rules, which come in two forms: joinand-increasing rules and joinand-decreasing rules (Chapters 9 and 10 of [3]).

5.4 From deducibilty to provability

For knotted rules where $n < m$, we actually establish a deduction theorem (as opposed to a local one) and this allows us to translate deductions into formulas. As a result, the complexity bounds for deducibility transfer to theoremhood (i.e., to the equational theory of the corresponding algebraic semantics); this result is contained in Section 9.1 of [3].

References

[1] A. Ciabattoni, N. Galatos and K. Terui, *Algebraic proof theory for substructural logics: cut-elimination and completions*, ***Annals of Pure and Applied Logic***, vol. 163 (2012), no. 3, pp. 266–290.

[2] A. Ciabattoni, N. Galatos and K. Terui, *Algebraic proof theory for substructural logics: hypersequents*, ***Annals of Pure and Applied Logic***, vol. 168 (2017), no. 3, pp. 693–737.

[3] N. Galatos, V. Greati, R. Ramanayake, G. St. John, *Complexities of Well-Quasi-Ordered Substructural Logics*, `https://doi.org/10.48550/arXiv.2504.21674`, pp. 166+.

[4] N. Galatos and P. Jipsen, *Residuated frames with applications to decidability*, ***Transactions of the AMS***, vol. 365 (2013), no. 3, pp. 1219–1249.

[5] N. Galatos, P. Jipsen, T. Kowalski and H. Ono, *Residuated Lattices: an algebraic glimpse at substructural logics*, ***Studies in Logics and the Foundations of Mathematics***, Elsevier, pp. 509+ (2007).

[6] N. Galatos and G. St.John, *Most simple extensions of FLe are undecidabe*, ***Journal of Symbolic Logic***, vol. 87 (2022), no. 3, pp. 1156–1200.