

Tools for the Investigation of Substructural and Paraconsistent Logics*

Agata Ciabattoni and Lara Spendier

Vienna University of Technology, Austria

Abstract. We present an overview of the methods in [10,7,13] and their implementation in the system **TINC**. This system introduces analytic calculi for large classes of substructural and paraconsistent logics, which it then uses to prove various results about the formalized logics.

1 Introduction

Logic is concerned with the study of reasoning and is the basis of applications in various fields. Classical logic is not adequate for all of them; for instance, it is ill-equipped to reason in presence of inconsistencies, inherently vague information, or about resources. Driven in part by the rising demand of practitioners, the last decades have witnessed an explosion of research on logics different from classical logic, and the definition of many new logics. These are often described in a declarative way within the framework due to Hilbert and Frege, which is however extremely cumbersome when it comes to finding or analyzing proofs. Moreover, a Hilbert-Frege system does not help answering useful questions about the formalized logic and its corresponding algebraic structure, such as ‘Is the logic decidable?’ or ‘Is the logic standard¹ complete?’. Therefore providing an algorithmic presentation of logics, in particular in the form of *analytic calculi*, is essential both for understanding their mathematical properties and for developing potential applications. Analyticity is crucial as it means that proofs in these calculi proceed by a step-wise decomposition of the formulas to be proved.

Since the introduction of Gentzen’s calculi *LK* and *LJ* for classical and intuitionistic logic, the *sequent calculus* has been one of the most popular frameworks for defining analytic calculi. Sequent calculi have been successfully used for studying important properties of their formalized logics such as decidability, complexity and interpolation; they have also proved useful for giving syntactic proofs of algebraic properties for which, in particular cases, semantic methods were not known, see e.g. [18]. These results all follow from the fundamental theorem of *cut-elimination*, which implies the redundancy of the cut rule and makes the calculi analytic. Despite the successful formalization of important logics, many natural and useful logics do not fit comfortably into the sequent framework. A huge range of extensions of the sequent calculus have been introduced

* Work supported by the FWF project START Y544-N23.

¹ That is complete with respect to algebras based on truth values in $[0, 1]$.

in the last few decades to define analytic calculi for logics apparently lacking a (cut-free) sequent formalization.

In this paper we describe our tools (theory and implementation) for introducing analytic calculi for large classes of substructural and of paraconsistent logics and using them to prove various results about these logics.

The idea to use computer supported tools for the investigation of logics has already been around for more than two decades, see, e.g., [24]. In recent years, several tools following this spirit of “logic engineering” have been introduced. These aim at making theoretical results in logic more accessible to researchers and practitioners who might not have deep knowledge about the logical theory, e.g. [5,25,23]. An example of a “logic engineering” tool addressing the issue of finding analytic calculi is the system *MUltlog* [5] which introduces such calculi for the class of finite-valued logics.

Our system **TINC** (Tools for the Investigation of Non-Classical logics) is created along the lines of *MUltlog* to cover a wider range of logics. It introduces sequent-style calculi for large classes of propositional substructural, intermediate and paraconsistent logics, which it then uses: (i) to check whether a substructural logic is standard complete (and hence it is a fuzzy logic in the sense of [20,17]) and (ii) to extract non-deterministic finite-valued semantics for paraconsistent and related logics and provide a uniform decidability proof for them. **TINC** implements the theoretical results in [10,7,15,13], for which this paper provides an overview and a non-technical description.

2 The System TINC

The system **TINC**, available at <http://www.logic.at/tinc>, takes as input a logic specified via suitable Kripke models or Hilbert systems, returns (a paper written in L^AT_EX containing) an analytic calculus and states certain properties of the logic. Currently **TINC** includes the following tools which handle large classes of substructural, paraconsistent and intermediate logics:

AxiomCalc transforms any suitable axiomatic extension of Full Lambek calculus with exchange and weakening **FLew** (i.e., intuitionistic linear logic with weakening) into a cut-free sequent or hypersequent calculus. Moreover, the tool exploits the generated calculus by checking a sufficient condition for the standard completeness of the input logic.

Paralyzer (PARAconsistent logics anaLYZER) transforms large classes of Hilbert axioms defining paraconsistent (and related) logics into sequent calculus rules. Moreover, it extracts non-deterministic, finite-valued semantics from the obtained calculi which show the decidability of the logics and reveal whether the calculi are analytic. *Paralyzer* also provides an encoding of the introduced calculi for the proof-assistant *Isabelle* [27] that can be used for semi-automated proof search within the considered logics.

Framinator (FRAMe condItioNs Automatically TO Rules) transforms frame conditions expressed as classical first-order formulas within the class Π_2 of the arithmetical hierarchy (i.e. formulas of the form $\forall\bar{x}\exists\bar{y}P$, for P quantifier free) into cut-free labelled sequent calculi.

AxiomCalc implements in Prolog² the results in [10,7], *Paralyzer* in [13] and *Framinator* in [15]. The whole system consists of 34 files and around 5400 lines of code (including documentation). The general structure of the implementation is depicted in Figure 1 and is instantiated with specific methods for every tool.

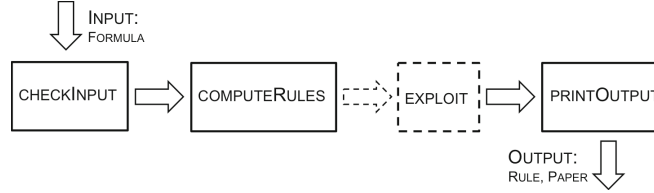


Fig. 1. Design of TINC

CHECKINPUT checks whether the syntactic form of the input formulas is correct. The core component COMPUTERULES implements the algorithm(s) to extract Gentzen-style rules out of the input formulas. The component EXPLOIT, which is not present in *Framinator*, implements methods that utilize the introduced calculus to reason about the logic and establish properties of it. PRINTOUTPUT contains everything that is related to presenting the results to the user.

The general idea behind the algorithm(s) implemented in COMPUTERULES is to start with a suitable analytic Gentzen-style calculus for a base logic and transform the Hilbert axioms or semantic conditions characterizing the logic at hand (i.e. the input formulas) into suitable rules.

Notation: henceforth we will use $\varphi, \psi, \alpha, \beta$ for (metavariables for) formulas. $\Gamma, \Delta, \Sigma, \Lambda$ will denote (metavariables for) multisets of formulas whereas Π will always stand for either a formula or the empty set.

Following [10] (and its generalization to display calculi in [16], and to labelled deductive systems in [15]) the key ingredients of the algorithm(s) are:

- (1) the *invertibility* of the logical rules of the base calculus, and
- (2) the following lemma, which allows formulas to change the side of the sequent by moving from the rule conclusion to the rule premise. For instance, its formulation for commutative (multiple-conclusion) sequent calculi is:

Lemma 1. *The sequent $\Gamma, \varphi \Rightarrow \psi, \Delta$ is interderivable with the rules (Γ' and Δ' are new metavariables):*

$$\frac{\Gamma' \Rightarrow \varphi, \Delta'}{\Gamma, \Gamma' \Rightarrow \psi, \Delta', \Delta} \quad \text{and} \quad \frac{\Gamma', \psi \Rightarrow \Delta'}{\Gamma, \Gamma', \varphi \Rightarrow \Delta', \Delta}$$

by using cut and the identity axiom $\alpha \Rightarrow \alpha$.

The transformation algorithm for substructural and for paraconsistent logics will be explained in the next sections which also contain examples of the corresponding tools *AxiomCalc* and *Paralyzer*.

² We used swi-prolog by Jan Wielemaker <http://www.swi-prolog.org>.

3 Substructural Logics

Substructural logics are obtained by dropping some of the structural rules from Gentzen's sequent calculus LJ . They encompass among many others classical, intuitionistic, intermediate, fuzzy, linear and relevant logics. Substructural logics are usually defined as axiomatic extensions of *full Lambek calculus* \mathbf{FL} , that is non-commutative intuitionistic linear logic.

In this section we give an overview of the theoretical results in [10,12] and their use in the tool *AxiomCalc* focusing on substructural logics extending \mathbf{FLew} , that is \mathbf{FL} with the rules for exchange and weakening. Connectives in these logics are \wedge (additive conjunction), \cdot (multiplicative conjunction/fusion), \vee (disjunction), \rightarrow (implication) and the constants 1 and 0.

As usual, $\neg\varphi$ is used as an abbreviation for $\varphi \rightarrow 0$.

In the following we refer to [22,11,18] for all concepts of universal algebra and to [26] for sequent calculi.

3.1 From Axioms to Structural Rules

The algorithm in [10], which is the core of the implementation of *AxiomCalc*, transforms large classes of axioms into structural sequent and hypersequent rules.

Which axioms can we handle? The axioms that belong to the classes \mathcal{N}_2 and \mathcal{P}_3 of the substructural hierarchy – a syntactic classification of axioms over \mathbf{FLew} (or, equivalently, of algebraic equations over integral and commutative residuated lattices) introduced in [10] for \mathbf{FLe} (and in [12] for the non-commutative version). The hierarchy is based on the *polarity* of the connectives of the base sequent calculus for \mathbf{FLew} ; recall that a connective has *positive* (resp. *negative*) *polarity* if its left (resp. right) logical rule is invertible, i.e., the conclusion implies the premises, see [1]. The classes \mathcal{P}_n and \mathcal{N}_n contain axioms/equations with leading positive ($1, \vee, \cdot$) and negative connectives ($0, \rightarrow, \wedge$).

Definition 1 (Substructural Hierarchy [10]). *For $n \geq 0$, the sets $\mathcal{P}_n, \mathcal{N}_n$ of formulas are defined as follows ($\mathcal{P}_0, \mathcal{N}_0$ contain all atomic formulas):*

$$\begin{aligned} \mathcal{P}_{n+1} &::= \mathcal{N}_n \mid \mathcal{P}_{n+1} \cdot \mathcal{P}_{n+1} \mid \mathcal{P}_{n+1} \vee \mathcal{P}_{n+1} \mid 1 \\ \mathcal{N}_{n+1} &::= \mathcal{P}_n \mid \mathcal{P}_{n+1} \rightarrow \mathcal{N}_{n+1} \mid \mathcal{N}_{n+1} \wedge \mathcal{N}_{n+1} \mid 0 \end{aligned}$$

Intuition: the different classes are defined by alternating connectives of different polarity. This accounts for the difficulty to deal with the corresponding axioms proof theoretically (and, as shown in [12], with the preservation under suitable order theoretic completions of the corresponding equations).

Example 1. Examples of Hilbert axioms within the classes \mathcal{N}_2 and \mathcal{P}_3 are

Class	Axiom	Name
\mathcal{N}_2	$\varphi \rightarrow \varphi \cdot \varphi$	contraction
	$\neg(\varphi \wedge \neg\varphi)$	weak contraction
	$\varphi \cdot \varphi \rightarrow \varphi \cdot \varphi \cdot \varphi$	3-contraction
\mathcal{P}_2	$\varphi \vee \neg\varphi$	excluded middle
	$(\varphi \rightarrow \psi) \vee (\psi \rightarrow \varphi)$	prelinearity
\mathcal{P}_3	$\neg\varphi \vee \neg\neg\varphi$	weak excluded middle
	$\neg(\varphi \cdot \psi) \vee (\varphi \wedge \psi \rightarrow \varphi \cdot \psi)$	weak nilpotent minimum

The procedure in [10] transforms each axiom within the class \mathcal{N}_2 into structural sequent calculus rules that preserve cut-elimination when added to the calculus $FLew$ for **FLew** (see below). These rules are however not powerful enough to capture axioms beyond the class \mathcal{N}_2 . Indeed, as shown in [10,12] they can only formalize properties that are already valid in intuitionistic logic ([10]) and among them only those whose corresponding algebraic equations are closed under the order theoretic completion known as Dedekind MacNeille³, in the context of integral residuated lattices ([12]). These results ensure, for instance, that no structural sequent rule can capture the prelinearity axiom (see Example 1), which is present in all formalizations of Fuzzy Logic [20]. A structural rule for this axiom was introduced in [2] using the hypersequent calculus – a simple and natural generalization of Gentzen sequent calculus.

As proved in [10], and recalled below, the hypersequent calculus can indeed deal with all the axioms within the class \mathcal{P}_3 .

Definition 2. A hypersequent is a multiset of sequents written as $\Gamma_1 \Rightarrow \Delta_1 \mid \dots \mid \Gamma_n \Rightarrow \Delta_n$ where each $\Gamma_i \Rightarrow \Delta_i$, $i = 1, \dots, n$, is a sequent, called component of the hypersequent. If all components of a hypersequent contain at most one formula in the succedent, the hypersequent is called single-conclusion, and multiple-conclusion otherwise.

The intuitive interpretation of the symbol “ \mid ” is disjunctive. This is reflected by the rules (*EC*) and (*EW*) (see Table 1), which are present in all hypersequent calculi. The base hypersequent calculus we use is $HFLew$ (see Table 1). Its sequent version $FLew$ is simply obtained by dropping the rules (*EC*) and (*EW*) and removing the side hypersequent G everywhere.

Definition 3. Let r and r' be two (hyper)sequent calculus rules. We say that r and r' are equivalent in a (hyper)sequent calculus S if the set of sequents provable from the same (hyper)sequent assumptions in $S \cup \{r\}$ and in $S \cup \{r'\}$ coincide. The definition naturally extends to sets of rules.

The algorithm in [10]: Following the idea sketched in Section 2, to transform axioms in the class \mathcal{P}_3 into equivalent structural (hyper)sequent rules we use

- (1) the invertible logical rules of the base calculus $HFLew$, that are $(1, l)$, (\vee, l) , (\cdot, l) , $(0, r)$, (\rightarrow, r) and (\wedge, r) , and
- (2) the following version of Lemma 1:

³ It is a generalization of Dedekind completion to ordered algebras, see e.g. [18].

$\frac{}{G \mid \varphi \Rightarrow \varphi}$ (<i>init</i>)	$\frac{G \mid \Gamma \Rightarrow \Pi}{G \mid 1, \Gamma \Rightarrow \Pi}$ (<i>1, l</i>)	$\frac{G \mid \varphi, \psi, \Gamma \Rightarrow \Pi}{G \mid \varphi \cdot \psi, \Gamma \Rightarrow \Pi}$ (<i>·, l</i>)	$\frac{G \mid \Gamma \Rightarrow \varphi \quad G \mid \Delta \Rightarrow \psi}{G \mid \Gamma, \Delta \Rightarrow \varphi \cdot \psi}$ (<i>·, r</i>)
$\frac{}{G \mid \Rightarrow 1}$ (<i>1, r</i>)	$\frac{G \mid \Gamma \Rightarrow \Pi}{G \mid \Gamma, \varphi \Rightarrow \Pi}$ (<i>w, l</i>)	$\frac{G \mid \varphi, \Gamma \Rightarrow \psi}{G \mid \Gamma \Rightarrow \varphi \rightarrow \psi}$ (<i>→, r</i>)	$\frac{G \mid \Gamma \Rightarrow \varphi \quad G \mid \psi, \Delta \Rightarrow \Pi}{G \mid \Gamma, \varphi \rightarrow \psi, \Delta \Rightarrow \Pi}$ (<i>→, l</i>)
$\frac{G \mid \Gamma \Rightarrow}{G \mid \Gamma \Rightarrow 0}$ (<i>0, r</i>)	$\frac{G}{G \mid \Gamma \Rightarrow \Pi}$ (<i>EW</i>)	$\frac{G \mid \Gamma \Rightarrow \varphi_i}{G \mid \Gamma \Rightarrow \varphi_1 \vee \varphi_2}$ (<i>∨, r</i>)	$\frac{G \mid \Gamma \Rightarrow \varphi \quad G \mid \varphi, \Delta \Rightarrow \Pi}{G \mid \Gamma, \Delta \Rightarrow \Pi}$ (<i>cut</i>)
$\frac{G \mid \Gamma \Rightarrow}{G \mid \Gamma \Rightarrow \varphi}$ (<i>w, r</i>)	$\frac{G \mid \Gamma \Rightarrow \Pi \mid \Gamma \Rightarrow \Pi}{G \mid \Gamma \Rightarrow \Pi}$ (<i>EC</i>)	$\frac{G \mid \varphi, \Gamma \Rightarrow \Pi \quad G \mid \psi, \Gamma \Rightarrow \Pi}{G \mid \varphi \vee \psi, \Gamma \Rightarrow \Pi}$ (<i>∨, l</i>)	

Table 1. Hypersequent calculus *HFLew*

Lemma 2. *The hypersequent $G \mid G' \mid \varphi_1, \dots, \varphi_n \Rightarrow \psi$ is equivalent to $(\Gamma_1, \dots, \Gamma_n, \Delta, \Pi$ are new metavariables)*

$$\frac{G \mid \Gamma_1 \Rightarrow \varphi_1 \cdots G \mid \Gamma_n \Rightarrow \varphi_n}{G \mid G' \mid \Gamma_1, \dots, \Gamma_n \Rightarrow \psi} \quad \text{and} \quad \frac{G \mid \psi, \Delta \Rightarrow \Pi}{G \mid G' \mid \varphi_1, \dots, \varphi_n, \Delta \Rightarrow \Pi}$$

These two key ingredients are then integrated in the transformation procedure as follows. Given any axiom $\varphi \in \mathcal{N}_2$ or $\varphi \in \mathcal{P}_3$:

(i) We start with the sequent $\Rightarrow \varphi$ if $\varphi \in \mathcal{N}_2$ or with hypersequents $G \mid \Rightarrow \varphi_1 \mid \cdots \mid \Rightarrow \varphi_n$ if $\varphi \in \mathcal{P}_3$ (and hence its normal form is a conjunction of formulas of the form $\varphi_1 \vee \cdots \vee \varphi_n$ with $\varphi_1, \dots, \varphi_n \in \mathcal{N}_2$). By utilizing the invertibility of the logical rules, we decompose φ as much as possible and obtain an equivalent set of (hyper)sequent rules R without premises. As an example, consider the axiom $\neg(\varphi \cdot \psi) \vee (\varphi \wedge \psi \rightarrow \varphi \cdot \psi) \in \mathcal{P}_3$, contained in the fuzzy logic **WNM** [17]:

$$G \mid \Rightarrow \neg(\varphi \cdot \psi) \mid \Rightarrow \varphi \wedge \psi \rightarrow \varphi \cdot \psi \quad \longrightarrow^{(i)} \quad \overline{G \mid \varphi, \psi \Rightarrow \mid \varphi \wedge \psi \Rightarrow \varphi \cdot \psi}$$

(ii) We apply Lemma 2 to each $r \in R$ to change side of the sequents of those formulas that cannot be decomposed by logical rules in their current position; continuing our example we move $\varphi \wedge \psi$ and $\varphi \cdot \psi$ and get $(\Sigma, \Lambda, \Pi$ are new metavariables)

$$\longrightarrow^{(ii)} \quad \frac{G \mid \Lambda \Rightarrow \varphi \wedge \psi \quad G \mid \Sigma, \varphi \cdot \psi \Rightarrow \Pi}{G \mid \varphi, \psi \Rightarrow \mid \Lambda, \Sigma \Rightarrow \Pi}$$

(iii) We utilize again the invertibility of the logical rules to decompose the compound formulas in the premises of each rule, resulting in a set of structural (hyper)sequent rules R_s . In our case R_s contains:

$$\longrightarrow^{(iii)} \quad \frac{G \mid \Lambda \Rightarrow \varphi \quad G \mid \Lambda \Rightarrow \psi \quad G \mid \Sigma, \varphi, \psi \Rightarrow \Pi}{G \mid \varphi, \psi \Rightarrow \mid \Lambda, \Sigma \Rightarrow \Pi}$$

(iv) The final step is a *completion procedure* that transforms each $r' \in R_s$ into an equivalent (hyper)sequent rule that preserves cut-elimination and the subformula property once it is added to the base calculus:

(iv.a) Using Lemma 2 we replace all the metavariables in the rule conclusions standing for formulas by new metavariables for multisets of formulas. Back to our example we get (Γ and Δ are new):

$$\xrightarrow{(iv.a)} \frac{G \mid \Lambda \Rightarrow \varphi \quad G \mid \Lambda \Rightarrow \psi \quad G \mid \Sigma, \varphi, \psi \Rightarrow \Pi \quad G \mid \Gamma \Rightarrow \varphi \quad G \mid \Delta \Rightarrow \psi}{G \mid \Gamma, \Delta \Rightarrow \mid \Lambda, \Sigma \Rightarrow \Pi}$$

(iv.b) We remove all the metavariables that appear in the premises and not in the conclusion. When those variables appear on the left and on the right hand side of different premises we close the obtained rules under all possible applications of (*cut*). For the rule above we therefore get:

$$\frac{G \mid \Gamma, \Lambda, \Sigma \Rightarrow \Pi \quad G \mid \Sigma, \Lambda, \Lambda \Rightarrow \Pi \quad G \mid \Sigma, \Gamma, \Delta \Rightarrow \Pi \quad G \mid \Sigma, \Lambda, \Delta \Rightarrow \Pi}{G \mid \Gamma, \Delta \Rightarrow \mid \Lambda, \Sigma \Rightarrow \Pi} \text{ (wnm)}$$

Theorem 1 ([10]). *Given any axiom $\varphi \in \mathcal{N}_2$ ($\varphi \in \mathcal{P}_3$), the rules generated by the above algorithm are equivalent to φ in $FLew$ and they preserve cut elimination when added to the sequent calculus $FLew$ (hypersequent calculus $HFLew$).*

The above algorithm is implemented in the tool *AxiomCalc* that, given an input axiom, first determines the class in the hierarchy to which the axiom belongs and, if it is within \mathcal{P}_3 , it automates the Steps (i)-(iv) above.

Example 2. Figure 2 below shows how to use *AxiomCalc* to define an analytic calculus for $FLew$ extended with the axiom $\varphi \cdot \varphi \rightarrow \varphi \cdot \varphi \cdot \varphi \in \mathcal{N}_2$.

3.2 An Application: Standard Completeness

The introduced calculi can be further utilized to check whether the corresponding logics are standard complete, i.e. complete for algebras with a real unit interval lattice reduct and hence whether they are fuzzy logics in the sense of [20,17]. The check is done using a sufficient condition for a hypersequent calculus to admit the elimination of the so-called density rule (below left is its Hilbert version and below right its hypersequent version in single-conclusion calculi):

$$\frac{(A \rightarrow p) \vee (p \rightarrow B) \vee C}{(A \rightarrow B) \vee C} \text{ (density)} \quad \frac{G \mid \Gamma \Rightarrow p \mid \Sigma, p \Rightarrow \Pi}{G \mid \Gamma, \Sigma \Rightarrow \Pi} \text{ (hdensity)}$$

where p is a propositional variable not occurring in any instance of A , B , or C (Γ , Σ and Π). Ignoring C , *density* can intuitively be read contrapositively as saying (very roughly) “if $A > B$, then $A > p$ and $p > B$ for some p ”; hence the name “density”. The connection between the elimination of the density rule and standard completeness is as follows: as shown in [22], adding *density* to any axiomatic extension \mathcal{L} of \mathbf{FLew} with prelinearity (see Example 1) makes

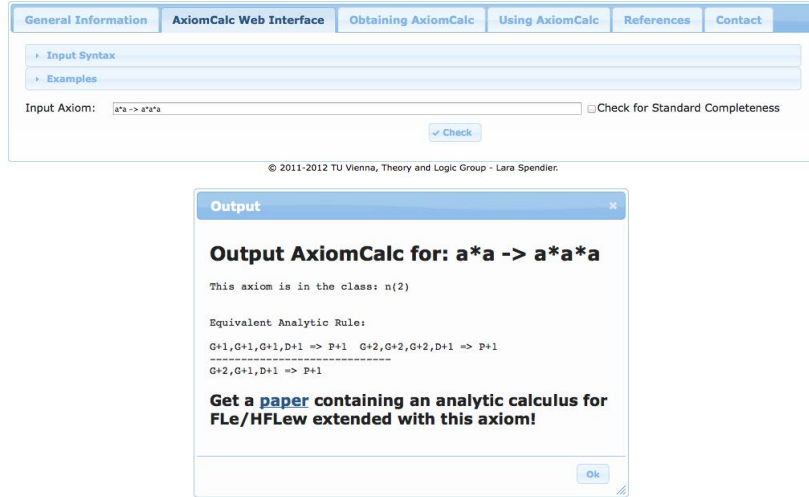
TINC - AxiomCalc

Fig. 2. Above: Main screen of *AxiomCalc* with the input axiom; below: Output

the corresponding logic *rational complete*, i.e., complete with respect to a corresponding class of (a) linearly and (b) densely ordered algebras; (a) is due to the prelinearity axiom, while (b) to the density rule. Hence by showing that the addition of *density* does not enlarge the set of provable formulas (i.e. *density* is an admissible or an eliminable rule) we get rational completeness for \mathcal{L} . Standard completeness with respect to algebras with lattice reduct $[0, 1]$ can then be obtained in many cases by means of a Dedekind MacNeille-style completion.

A syntactic condition which guarantees the elimination of the density rule from a suitable hypersequent calculus was introduced in [7]. Using this result a uniform proof of standard completeness that applies to large classes of logics is as follows: let \mathcal{L} be the logic obtained by extending **FLe** with prelinearity and with any (set of) axiom(s) within the class \mathcal{P}_3 :

- We first introduce a hypersequent calculus $H\mathcal{L}$ for \mathcal{L} ;
- If $H\mathcal{L}$ satisfies the sufficient condition in [7] then \mathcal{L} is rational complete, and by [11] it is also standard complete being all algebraic equations corresponding to axioms within the class \mathcal{P}_3 closed under Dedekind-MacNeille completion when applied to subdirectly irreducible algebras.

This general approach contrasts with the logic-specific techniques usually employed to prove standard completeness, e.g. [19,17]. Moreover, it allows the discovery of new fuzzy logics in a completely automated way. The whole procedure is implemented in *AxiomCalc* and is started by ticking the checkbox “Check for Standard Completeness”, see Figure 2 above.

4 Paraconsistent and Related Logics

Paraconsistent logics are logics suitable for reasoning in the presence of inconsistent information. The most important family of paraconsistent logics is that of *C-systems* [8], where the notion of consistency is internalized in the object language by a unary consistency operator \circ ; $\circ\varphi$ has the intuitive meaning of “ φ is consistent”. For many of these logics, finding an analytic calculus has been an open problem.

In this section we give an overview of the theoretical results in [13] and their use in *Paralyzer*. The logics we consider are paraconsistent (and other) logics all obtained by extending the positive fragment of propositional classical logic \mathbf{CI}^+ (containing conjunction \wedge , disjunction \vee and implication \supset) with finitely many unary connectives from a set \mathcal{U} ; these logics include the most well known C-systems.

4.1 From Axioms to Logical Rules

The algorithm in [13], which is the core of the implementation of the tool *Paralyzer*, transforms axioms into logical sequent rules.

Which axioms can we handle? All axioms belonging to the set of formulas \mathbf{Ax} [13] that are (i) generated by the following grammar (where \mathcal{G} is the initial variable and $\mathcal{U} = \{\star_1, \dots, \star_n\}$):

$$\begin{aligned} \mathcal{G} &= R_1 \mid R_2 \mid R_3 & R_3 &= (R_3 \diamond P_1) \mid (P_1 \diamond R_3) \mid \star \triangleright p_1 \\ R_1 &= (R_1 \diamond P_1) \mid (P_1 \diamond R_1) \mid \star p_1 & P_1 &= (P_1 \diamond P_1) \mid \star p_1 \mid p_1 \mid p_2 \\ R_2 &= (R_2 \diamond P_2) \mid (P_2 \diamond R_2) \mid \star(p_1 \diamond p_2) & P_2 &= (P_2 \diamond P_2) \mid \star p_1 \mid p_1 \mid \star p_2 \mid p_2 \\ \diamond &= \wedge \mid \vee \mid \supset & \star, \triangleright &= \star_1 \mid \dots \mid \star_n \end{aligned}$$

and (ii) satisfy the following technical condition: some subformula $\star p_1$ of a formula generated by R_1 (the subformulas $\star \triangleright p_1$ or $\star(p_1 \diamond p_2)$ of a formula generated by R_3 or R_2 , resp.) must not be contained in

- (a) a positively⁴ occurring (sub)formula of the form $\psi_1 \wedge \psi_2$, and
- (b) a negatively occurring (sub)formula of the form $\psi_1 \vee \psi_2$ or $\psi_1 \supset \psi_2$.

Example 3. Examples of formulas in \mathbf{Ax} are ($\diamond \in \{\vee, \wedge, \supset\}$ and $\neg, \circ \in \mathcal{U}$):

$$\begin{array}{ll} \mathbf{(n_1)} & p_1 \vee \neg p_1 & \mathbf{(n_2)} & p_1 \supset (\neg p_1 \supset p_2) \\ \mathbf{(c)} & \neg \neg p_1 \supset p_1 & \mathbf{(e)} & p_1 \supset \neg \neg p_1 \\ \mathbf{(n_\wedge^1)} & \neg(p_1 \wedge p_2) \supset (\neg p_1 \vee \neg p_2) & \mathbf{(n_\wedge^r)} & (\neg p_1 \vee \neg p_2) \supset \neg(p_1 \wedge p_2) \\ \mathbf{(n_\vee^1)} & \neg(p_1 \vee p_2) \supset (\neg p_1 \wedge \neg p_2) & \mathbf{(n_\vee^r)} & (\neg p_1 \wedge \neg p_2) \supset \neg(p_1 \vee p_2) \\ \mathbf{(n_\supset^1)} & \neg(p_1 \supset p_2) \supset (p_1 \wedge \neg p_2) & \mathbf{(n_\supset^r)} & (p_1 \wedge \neg p_2) \supset \neg(p_1 \supset p_2) \\ \mathbf{(b)} & p_1 \supset (\neg p_1 \supset (\circ p_1 \supset p_2)) & \mathbf{(r_\diamond)} & \circ(p_1 \diamond p_2) \supset (\circ p_1 \vee \circ p_2) \\ \mathbf{(k)} & \circ p_1 \vee (p_1 \wedge \neg p_1) & \mathbf{(i)} & \neg \circ p_1 \supset (p_1 \wedge \neg p_1) \\ \mathbf{(o_\diamond^1)} & \circ p_1 \supset \circ(p_1 \diamond p_2) & \mathbf{(o_\diamond^2)} & \circ p_2 \supset \circ(p_1 \diamond p_2) \\ \mathbf{(a_\diamond)} & (\circ p_1 \wedge \circ p_2) \supset \circ(p_1 \diamond p_2) & \mathbf{(a_\neg)} & \circ p_1 \supset \circ \neg p_1 \end{array}$$

⁴ A subformula φ occurs *negatively* (*positively*, resp.) in ψ if there is an odd (even, resp.) number of implications \supset in ψ having φ as a subformula of its antecedent.

Most C-systems (see, e.g., [8]) are obtained by employing suitable combinations of the above axioms which express various properties of negation and of the consistency operator \circ .

The algorithm in [13]: Following the idea sketched in Section 2, to transform the axioms within \mathbf{Ax} into equivalent logical sequent rules we use:

- (1) The invertible logical rules of the base sequent calculus LK^+ for $\mathbf{C1}^+$, which is LK without negation (note that in LK^+ all rules for connectives are invertible), and
- (2) Lemma 1.

The procedure to transform any $\varphi \in \mathbf{Ax}$ into equivalent rules (cf. Definition 3) then works as follows:

(i) Starting from $\Rightarrow \varphi$, by utilizing the invertibility of the logical rules of LK^+ as much as possible, φ is decomposed into its subformulas, thus obtaining an equivalent set of rules R without premises.

As an example, let $\varphi := p_1 \supset (\neg p_1 \supset (\circ p_1 \supset p_2)) \in \mathbf{Ax}$:

$$\begin{array}{ccc} \Rightarrow p_1 \supset (\neg p_1 \supset (\circ p_1 \supset p_2)) & \longrightarrow^{(i)} & \overline{p_1 \Rightarrow \neg p_1 \supset (\circ p_1 \supset p_2)} \\ \longrightarrow^{(i)} \overline{p_1, \neg p_1 \Rightarrow \circ p_1 \supset p_2} & & \longrightarrow^{(i)} \overline{p_1, \neg p_1, \circ p_1 \Rightarrow p_2} \end{array}$$

(ii) We remove all rules $r \in R$ containing $p_i \Rightarrow p_i$ for $i \in \{1, 2\}$ in their conclusion. Moreover, if a rule does not contain $\star(p_1 \diamond p_2)$ with $\star \in \mathcal{U}$ and $\diamond \in \{\vee, \wedge, \supset\}$, we can safely remove all variables p_2 . In our example it gives:

$$\longrightarrow^{(ii)} \overline{p_1, \neg p_1, \circ p_1 \Rightarrow}$$

(iii) By using Lemma 1 all remaining formulas but one are moved to the premises of each rule, changing the side of the sequent. The formula that remains in the conclusion will be the one introduced by the rule and will be either of the form $\star p_1$ (when φ was generated by R_1), $\star \triangleright p_1$ or $\star(p_1 \diamond p_2)$, resp. (when φ was generated by R_3 or R_2 , resp.) for any $\star, \triangleright \in \mathcal{U}$ and $\diamond \in \{\wedge, \vee, \supset\}$.

Continuing our example (using weakening and contraction of LK^+), we obtain

$$\longrightarrow^{(iii)} \frac{\Gamma \Rightarrow \Delta, p_1 \quad \Gamma \Rightarrow \Delta, \neg p_1}{\circ p_1, \Gamma \Rightarrow \Delta} \quad \text{or} \quad \longrightarrow^{(iii)} \frac{\Gamma \Rightarrow \Delta, p_1 \quad \Gamma \Rightarrow \Delta, \circ p_1}{\neg p_1, \Gamma \Rightarrow \Delta}$$

Theorem 2 ([13]). *Any axiom $\varphi \in \mathbf{Ax}$ can be transformed into sequent rules equivalent in LK^+ having the following form ($\star, \triangleright, * \in \mathcal{U}$ and $\diamond \in \{\wedge, \vee, \supset\}$):*

unary-one rules	binary rules	unary-two rules
$\frac{\mathcal{S}_1}{\Gamma, \star \varphi \Rightarrow \Delta}$	$\frac{\mathcal{S}_2}{\Gamma, \star(\varphi_1 \diamond \varphi_2) \Rightarrow \Delta}$	$\frac{\mathcal{S}_1}{\Gamma, \star \triangleright \varphi \Rightarrow \Delta}$
$\frac{\mathcal{S}_1}{\Gamma \Rightarrow \Delta, \star \varphi}$	$\frac{\mathcal{S}_2}{\Gamma \Rightarrow \Delta, \star(\varphi_1 \diamond \varphi_2)}$	$\frac{\mathcal{S}_1}{\Gamma \Rightarrow \Delta, \star \triangleright \varphi}$

where \mathcal{S}_1 may contain premises of the form $\Gamma, \varphi \Rightarrow \Delta$; $\Gamma, * \varphi \Rightarrow \Delta$; $\Gamma \Rightarrow \Delta, \varphi$; and $\Gamma \Rightarrow \Delta, * \varphi$, while \mathcal{S}_2 of the form $\Gamma, \varphi_i \Rightarrow \Delta$; $\Gamma, * \varphi_i \Rightarrow \Delta$; $\Gamma \Rightarrow \Delta, \varphi_i$, and $\Gamma \Rightarrow \Delta, * \varphi_i$ where $i \in \{1, 2\}$.

4.2 An Application: Non-deterministic Semantics

The introduced calculi, obtained by extending LK^+ with the special rules described in Theorem 2, are used to extract new semantics for the corresponding logics using *partial non-deterministic matrices (PNmatrices)*. These are a natural generalization of the standard multi-valued matrices, which allow the truth-value assigned to a complex formula to be chosen *non-deterministically* out of a given (possibly⁵ empty) set of options.

Our semantics guarantee the decidability of the considered logics (Corollary 1) and are used to check whether the defined calculi satisfy a generalized notion of the subformula property (Theorem 4). Regarding the latter, note that the addition of the new logical sequent rules to LK^+ does not necessarily result in a cut-free system (or in a system satisfying some form of subformula property). In fact, checking whether this is the case requires a “global view” of the resulting calculus, which takes into account the way in which all the rules of the calculus mentioning the same connectives interact. This view is provided by our semantics and, as shown in [6], it amounts to checking only whether the resulting PNmatrix contains an empty set in the truth tables of the connectives.

Definition 4 ([6]). A partial non-deterministic matrix (PNmatrix) \mathcal{M} for a propositional language \mathcal{L} consists of:

- (1) A set $\mathcal{V}_{\mathcal{M}}$ of truth values.
- (2) A subset $\mathcal{D}_{\mathcal{M}} \subseteq \mathcal{V}_{\mathcal{M}}$ of designated truth values.
- (3) A truth table $\diamond_{\mathcal{M}} : \mathcal{V}_{\mathcal{M}}^n \rightarrow P(\mathcal{V}_{\mathcal{M}})$ for every n -ary connective \diamond of \mathcal{L} .

From sequent calculi to PNmatrices: Let G be sequent calculus obtained by adding to LK^+ any set R of unary-one, binary and unary-two rules with set \mathcal{U} of unary connectives. The truth values $\mathcal{V}_{\mathcal{M}}$ of the PNmatrix for G are tuples over $\{0, 1\}$ of size “# of unary connectives in \mathcal{U} ”+1; the tuples store the information about the value of a formula φ and also of $\star\varphi$ for each $\star \in \mathcal{U}$. The matrix is then constructed using the rules in R , which play different roles according to their type. More precisely,

- *unary-one* rules reduce the set $\mathcal{V}_{\mathcal{M}}$ of truth values,
- *unary-two* rules determine the truth tables of the unary connectives, and
- *binary* rules determine the truth tables of the binary connectives.

We show below how to construct a PNmatrix out of a concrete sequent calculus (see [6] for the general procedure and all technical details).

⁵ The possibility of having empty spots in the matrices make PNmatrices a generalization of non-deterministic matrices Nmatrices [4].

Let $\mathcal{U} = \{\star\}$, and consider the sequent calculus LK^+ extended with the two rules (equivalent to the axioms $p_1 \supset (\star p_1 \supset p_2)$ and $\star \star p_1 \supset p_1$, respectively):

$$\frac{\Gamma \Rightarrow \Delta, \varphi}{\Gamma, \star \varphi \Rightarrow \Delta} (r_{uo}) \quad \text{and} \quad \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma, \star \star \varphi \Rightarrow \Delta} (r_{ut})$$

We first construct the set of truth values $\mathcal{V}_{\mathcal{M}}$, starting with the set of all possible truth values of size 2 (i.e. $|\{\star\}| + 1$):

$$\{\langle 0, 0 \rangle, \langle 0, 1 \rangle, \langle 1, 0 \rangle, \langle 1, 1 \rangle\}$$

Note that a tuple $\langle x, y \rangle$ stands for $\langle \varphi : x, \star \varphi : y \rangle$, i.e., it says that x (resp. y) is the value of φ (resp. of $\star \varphi$).

As usual, we interpret formulas occurring on the right (left) hand side of a sequent as taking the value 1 (0). Hence the rule (r_{uo}) says that all tuples in which φ takes the value 1 (in the rule premise φ occurs on the right hand side) must also have that $\star \varphi$ takes the value 0. As $\langle 1, 1 \rangle$ does not satisfy this condition, the set of truth values is reduced to

$$\mathcal{V}_{\mathcal{M}} = \{\langle 0, 0 \rangle, \langle 0, 1 \rangle, \langle 1, 0 \rangle\}$$

The set of designated truth values $\mathcal{D}_{\mathcal{M}}$ contains all elements of $\mathcal{V}_{\mathcal{M}}$ where the first element is 1; in our case $\mathcal{D}_{\mathcal{M}} = \{\langle 1, 0 \rangle\}$.

The truth table of the unary connective \star is determined in two steps:

First, we set up the basic truth table for \star where we assign to every tuple $u \in \mathcal{V}_{\mathcal{M}}$ all tuples where φ coincides with $\star \varphi$ of u (see left table below), e.g. $u = \langle 0, 0 \rangle$ has $\langle 0, 0 \rangle, \langle 0, 1 \rangle$ as possible values.

Then, we have to consider the rule (r_{ut}) which says that for every tuple u in which φ takes the value 0 (in the rule premise, φ occurs on the left side) we delete the assigned tuples not having 0 for $\star \star \varphi$. E.g., for $u = \langle 0, 0 \rangle$ we delete the tuple $\langle 0, 1 \rangle$ as $\star \varphi$ takes 1 (we are in the truth table for the unary connective \star , and hence $\star \varphi$ corresponds to $\star \star \varphi$).

$$\begin{array}{c|c} \star & \\ \hline \langle 0, 0 \rangle & \{\langle 0, 0 \rangle, \langle 0, 1 \rangle\} \\ \langle 0, 1 \rangle & \{\langle 1, 0 \rangle\} \\ \langle 1, 0 \rangle & \{\langle 0, 0 \rangle, \langle 0, 1 \rangle\} \end{array} \xrightarrow{(r_{ut})} \begin{array}{c|c} \star & \\ \hline \langle 0, 0 \rangle & \{\langle 0, 0 \rangle\} \\ \langle 0, 1 \rangle & \{\langle 1, 0 \rangle\} \\ \langle 1, 0 \rangle & \{\langle 0, 0 \rangle, \langle 0, 1 \rangle\} \end{array}$$

Note that the PNmatrix for a calculus G can be automatically computed by the tool *Paralyzer* (see Example 4).

Theorem 3 ([13]). *Let G be the sequent calculus for the logic defined by extending \mathbf{CI}^+ with any $\varphi \in \mathbf{Ax}$, and \mathcal{M} its associated PNmatrix, both obtained by the procedures sketched above. A sequent is provable in G iff it is valid⁶ in \mathcal{M} .*

⁶ That is it takes a designated truth value under all interpretations in \mathcal{M} .

As each logic characterized by a finite PNmatrix is decidable [6] we immediately have:

Corollary 1. *All logics extending \mathbf{CI}^+ with axioms in \mathbf{Ax} are decidable.*

The PNmatrix \mathcal{M} is also used to check the analyticity of the corresponding calculus G :

Theorem 4 ([13]). *\mathcal{M} does not contain empty sets in its truth tables iff whenever a sequent s is provable in G , s can be proved by using only its subformulas and their extensions with unary connectives from \mathcal{U} .*

Example 4. Consider the axioms $p_1 \vee \neg p_1$ and $\circ p_1 \supset \circ \neg p_1$, which are given as input (in a slightly adapted syntax) for *Paralyzer*, see Figure 3.

TINC - Paralyzer

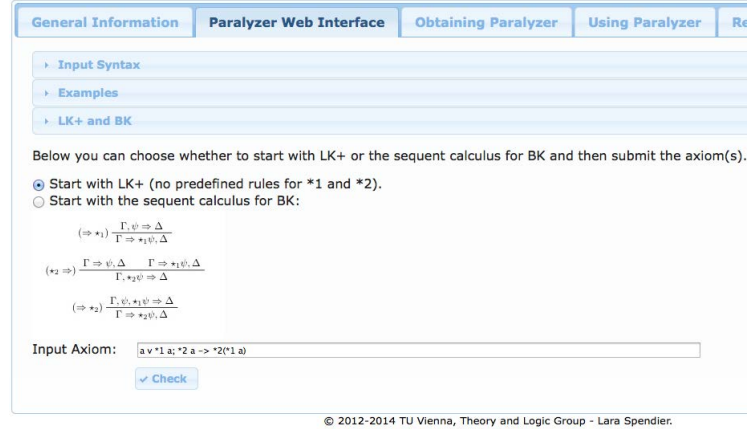


Fig. 3. Main screen of *Paralyzer* with the input axioms. Note that the user can choose the base calculus between LK^+ (default option) and BK [3].

The computed rules and the associated PNmatrix are in Figure 4.

5 Future Research

The tools described in this paper provide automated support for the introduction of analytic calculi and the investigation of interesting properties (standard completeness, non-deterministic semantics and decidability) for many substructural and for many paraconsistent and related logics.

Many practical and theoretical issues are still to be addressed; among them extending our results to new logics including first-order logics. In the substructural case the main challenge is to find the right formalism (and method) to capture axioms beyond the level \mathcal{P}_3 of the substructural hierarchy.

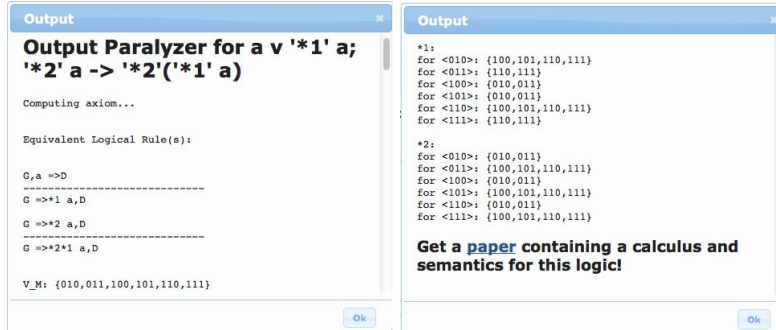


Fig. 4. Left: output containing the equivalent sequent rules and \mathcal{V}_M of the PNmatrix. Right: truth tables for the unary connectives.

For paraconsistent and related logics the introduction of analytic calculi could be easily adapted to capture e.g. paraconsistent logics extending intuitionistic logic, substructural paraconsistent logics or first-order logics; however the construction of the corresponding PNmatrices would require a deeper investigation. For the time being there is indeed no theory of PNmatrices for first-order logics, intuitionistic logics or substructural logics (that in fact lack even a theory of Nmatrices [4]). A step forward in this direction has been done in the recent work [14] that generalizes the results described in Section 4 to axioms in **Ax** with a possible nesting of unary connectives of any fixed depth. On a more practical level, the encoding into the proof-assistant *Isabelle* [27] of the calculi computed by *Paralyzer* allows us to find proofs of theorems in the considered logics in a semi-automated way. The definition of automated deduction procedures is currently under investigation; following [21], a possible approach is to search for suitable encodings of our calculi into SAT.

Finally, we plan to extend the system **TINC** with new tools that cover further classes of interesting logics, e.g. modal logics defined by Hilbert axioms.

References

1. Andreoli, J.-M.: Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation* 2(3), 297–347 (1992)
2. Avron, A.: Hypersequents, logical consequence and intermediate logics for concurrency. *Annals of Mathematics and Artificial Intelligence* 4, 225–248 (1991)
3. Avron, A., Konikowska, B., Zamansky, A.: Cut-free sequent calculi for C-systems with generalized finite-valued semantics. *Journal of Logic and Computation* 21(3), 517–540 (2013)
4. Avron, A., Lev, I.: Non-deterministic multiple-valued structures. *Journal of Logic and Computation* 15(3), 241–261 (2005)
5. Baaz, M., Fermüller, C.G., Salzer, G., Zach, R.: MUltlog 1.0: Towards an expert system for many-valued logics. In: McRobbie, M.A., Slaney, J.K. (eds.) *CADE 1996*. LNCS, vol. 1104, pp. 226–230. Springer, Heidelberg (1996)
6. Baaz, M., Lahav, O., Zamansky, A.: Finite-valued semantics for canonical labelled calculi. *Journal of Automated Reasoning* 51(4), 401–430 (2013)

7. Baldi, P., Ciabattoni, A., Spendier, L.: Standard completeness for extensions of MTL: An automated approach. In: Ong, L., de Queiroz, R. (eds.) WoLLIC 2012. LNCS, vol. 7456, pp. 154–167. Springer, Heidelberg (2012)
8. Carnielli, W.A., Marcos, J.: A taxonomy of C-systems. In: Carnielli, W.A., Coniglio, M.E., Ottaviano, I.D. (eds.) Paraconsistency: The Logical Way to the Inconsistent, pp. 1–94 (2002)
9. Chagrov, A., Zakharyashev, M.: Modal Logic. Clarendon Press, Oxford (1997)
10. Ciabattoni, A., Galatos, N., Terui, K.: From axioms to analytic rules in nonclassical logics. In: Proceedings of LICS 2008, pp. 229–240 (2008)
11. Ciabattoni, A., Galatos, N., Terui, K.: MacNeille Completions of FL-algebras. Algebra Universalis 66(4), 405–420 (2011)
12. Ciabattoni, A., Galatos, N., Terui, K.: Algebraic proof theory for substructural logics: cut-elimination and completions. Annals of Pure and Applied Logic 163(3), 266–290 (2012)
13. Ciabattoni, A., Lahav, O., Spendier, L., Zamansky, A.: Automated support for the investigation of paraconsistent and other logics. In: Artemov, S., Nerode, A. (eds.) LFCS 2013. LNCS, vol. 7734, pp. 119–133. Springer, Heidelberg (2013)
14. Ciabattoni, A., Lahav, O., Spendier, L., Zamansky, A.: Taming paraconsistent (and other) logics: An algorithmic approach (submitted 2014)
15. Ciabattoni, A., Maffezioli, P., Spendier, L.: Hypersequent and labelled calculi for intermediate logics. In: Galmiche, D., Larchey-Wendling, D. (eds.) TABLEAUX 2013. LNCS, vol. 8123, pp. 81–96. Springer, Heidelberg (2013)
16. Ciabattoni, A., Ramanayake, R.: Structural extensions of display calculi: A general recipe. In: Libkin, L., Kohlenbach, U., de Queiroz, R. (eds.) WoLLIC 2013. LNCS, vol. 8071, pp. 81–95. Springer, Heidelberg (2013)
17. Cintula, P., Hájek, P., Noguera, C. (eds.): Handbook of Mathematical Fuzzy Logic, vol. 1. Studies in Logic, Mathematical Logic and Foundations, vol. 37. College Publications (2011)
18. Galatos, N., Jipsen, P., Kowalski, T., Ono, H.: Residuated Lattices: An algebraic glimpse at substructural logics. Studies in Logics and the Foundations of Mathematics. Elsevier (2007)
19. Jenei, S., Montagna, F.: A proof of standard completeness for Esteva and Godo’s MTL logic. Studia Logica 70(2), 183–192 (2002)
20. Hájek, P.: Metamathematics of Fuzzy Logic. Springer (1998)
21. Lahav, O., Zohar, Y.: SAT-based decision procedure for analytic pure sequent calculi. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS (LNAI), vol. 8562, pp. 76–90. Springer, Heidelberg (2014)
22. Metcalfe, G., Montagna, F.: Substructural fuzzy logics. Journal of Symbolic Logic 7(3), 834–864 (2007)
23. Nigam, V., Pimentel, E., Reis, G.: An extended framework for specifying and reasoning about proof systems. Journal of Logic and Computation (accepted)
24. Ohlbach, H.J.: Computer support for the development and investigation of logics. Logic Journal of the IGPL 4(1), 109–127 (1996)
25. Tishkovsky, D., Schmidt, R.A., Khodadadi, M.: The tableau prover generator MetTeL2. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS, vol. 7519, pp. 492–495. Springer, Heidelberg (2012)
26. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory. Cambridge University Press (2000)
27. Wenzel, M., Paulson, L.C., Nipkow, T.: The Isabelle Framework. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 33–38. Springer, Heidelberg (2008)