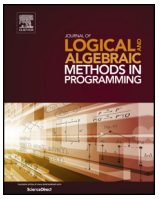
Journal of Logical and Algebraic Methods in Programming 130 (2023) 100827

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp



A rewriting logic approach to specification, proof-search, and meta-proofs in sequent systems

Carlos Olarte [a,*], Elaine Pimentel [b,*], Camilo Rocha [c,*]

[a] LIPN, CNRS UMR 7030, Université Sorbonne Paris Nord, Villetaneuse, France
[b] University College London, London, UK
[c] Pontificia Universidad Javeriana, Cali, Colombia

ARTICLE INFO

Article history:
Received 7 September 2021
Received in revised form 27 September 2022
Accepted 7 October 2022
Available online 18 October 2022

Keywords:
Rewriting logic
Proof theory
Sequent systems

ABSTRACT

This paper develops an algorithmic-based approach for proving inductive properties of propositional sequent systems such as admissibility, invertibility, cut-admissibility, and identity expansion. Although undecidable in general, these structural properties are crucial in proof theory because they can reduce the proof-search effort and further be used as scaffolding for obtaining other meta-results such as consistency. The algorithms –which take advantage of the rewriting logic meta-logical framework– are explained in detail and illustrated with examples throughout the paper. They have been fully mechanized in the `L-Framework`, thus offering both a formal specification language and off-the-shelf mechanization of the proof-search algorithms coming together with semi-decision procedures for proving theorems and meta-theorems of the object system. As illustrated with case studies in the paper, the `L-Framework` achieves a great degree of automation when used on several propositional sequent systems, including single conclusion and multi-conclusion intuitionistic logic, classical logic, classical linear logic and its dyadic system, intuitionistic linear logic, and normal modal logics.

Contents

* Corresponding authors.
E-mail addresses: olarte@lipn.univ-paris13.fr (C. Olarte), e.pimentel@ucl.ac.uk (E. Pimentel), camilo.rocha@javerianacali.edu.co (C. Rocha).

https://doi.org/10.1016/j.jlamp.2022.100827

1. Introduction

Gentzen's sequent systems [1] have proved to be a flexible and robust logical formalism for defining proof systems. With the advent of new needs in the form of techniques and technologies (*e.g.*, to reason in temporal, epistemic, and modal domains), the past decades have witnessed a vivid scene where novel sequent-based calculi have been proposed (see, *e.g.*, [2,3]). It is fair to say that the most important aims of these new systems are the ultimate goal of having both a formal specification language and a mechanization of proof-search algorithms coming together with decision procedures –whenever possible and feasible.

Cut-free sequent systems are key for this latter purpose. Intuitively, the cut-rule expresses the mathematical use of lemmas in proofs: if A follows from B and C follows from A, then C follows from B. That is, one can cut the intermediate lemma A. Cut-admissibility theorems then state that any judgment that possesses a proof making use of the cut-rule also possesses a proof that does not make use of it. This is of practical convenience for proof-search purposes because it is often much easier to find proofs when the need to mechanically come up with auxiliary lemmas can be avoided. Moreover, there are important applications of cut-admissibility, such as the reduction of consistency proofs and "subformula properties" to mere structural syntactic checks. However, structural properties of a logical system such as cut-freeness are actually inductive meta-properties and thus undecidable to check in general. In practice, proof-theoretic approaches to establish them usually require combinatorial search heuristics based on the structure of formulas of each particular system. These heuristics, in turn, may depend on the ability to prove other inductive properties. For instance, Gentzen-style proofs of cut-admissibility heavily depend on checking admissibility of other structural rules (*e.g.*, weakening and contraction) and showing that some of the sequent rules defining the system are invertible.

The main goal of this article is to present the `L-Framework` [4], a tool for algebraically specifying and analyzing propositional sequent systems. It provides both a specific scaffolding of constructs usually found in a sequent system, and algorithms for proving inductive properties such as admissibility, invertibility, cut-admissibility, and identity expansion. The proposed framework is *generic* in the sense that only mild restrictions are imposed on the formulas of the sequent system, and *modular* since admissibility of structural rules, invertibility, cut-admissibility and identity expansion can be proved incrementally.

The algorithms and their implementation in the `L-Framework` are based on the expressive power of rewriting logic [5], both as a logical and a reflective meta-logical framework [6,7]. A sequent system is specified as a rewriting logic theory $\mathscr{S}$ and a sequent as a formula φ in the syntax of $\mathscr{S}$. In this setting, the process of building a proof of φ in $\mathscr{S}$ is cast as a reachability goal of the form $\mathscr{S} \vdash \varphi \xrightarrow{*} \top$, where $\top$ denotes the empty sequent. Therefore, as a logical framework, rewriting logic directly serves the purpose of providing a proof-search algorithm at the *object* level of $\mathscr{S}$. Since rewriting logic is reflective, a universal rewrite theory $\mathscr{U}$ is further used to prove meta-logical properties (*e.g.*, cut-freeness) of the system $\mathscr{S}$. More precisely, such proofs are specified as reachability goals of the form $\mathscr{U} \vdash \left(\overline{\mathscr{S}}; \overline{\Gamma}\right) \xrightarrow{*} \left(\overline{\mathscr{S}}; \overline{\top}\right)$, where Γ is a set of sequents representing the proof-obligations to be discharged, and $\overline{\mathscr{S}}$ and $\overline{\Gamma}$ are the meta-representation of $\mathscr{S}$ and Γ in the syntax of the universal theory $\mathscr{U}$, respectively. In this way, starting with the specification of the propositional sequent system $\mathscr{S}$ in rewriting logic, the framework presented in this paper enables both the proof of theorems and inductive meta-theorems of $\mathscr{S}$.

The framework comprises a *generic* rewrite theory offering sorts, operators, and standard procedures for proof-search at the object- and meta-level. The sorts are used to represent different elements of the syntactic structure of a sequent system such as formulas, sets of formulas, sequents, and collections of sequents. The inference rules of a sequent system are specified as reversed rewrite rules, so that an inference step of the mechanized system is carried out by matching a sequent to the conclusion of an inference rule, and replacing the latter by the corresponding substitution instance of the

rule's premises. Conceptually, a proof at the object level is found when all proof-obligations are conclusions of inference rules without premises (*i.e.*, axioms).

Once a sequent system $\mathscr{S}$ is specified, proofs of its cut-freeness, admissibility, invertibility, and identity expansion properties can be searched for. The process of obtaining proofs for each one of these properties follows a similar pattern. Namely, the rules of $\mathscr{S}$ are used by pair-wise comparison to generate the inductive hypothesis $\mathscr{H}$ and the proof obligations Γ. Therefore, if for each pair $(\mathscr{H}, \Gamma)$ the (meta-)reachability goal $\mathscr{U} \vdash \left(\overline{\mathscr{S} \cup \mathscr{H}}; \overline{\Gamma}\right) \xrightarrow{*} \left(\overline{\mathscr{S} \cup \mathscr{H}}; \overline{\top}\right)$ can be answered positively, then the corresponding property of the sequent system $\mathscr{S}$ holds. The universal theory $\mathscr{U}$ contains the rewrite-based heuristics that operate over $\overline{\mathscr{S} \cup \mathscr{H}}$ and $\overline{\Gamma}$. Much of the effort in obtaining the proposed framework was devoted to designing such meta-level heuristics and fully implementing them in Maude [8], a high-performance reflective language and system supporting rewriting logic.

The case studies presented in this paper comprise several propositional sequent systems, encompassing different proof-theoretical aspects. The chosen systems include: single conclusion (G3ip) and multi-conclusion intuitionistic logic (mLJ), classical logic (G3cp), classical linear logic (LL) and its dyadic system (DyLL), intuitionistic linear logic (ILL), and normal modal logics (K and S4). Beyond advocating for the use of rewriting logic as a meta-logical framework, the novel algorithms presented here are able to automatically discharge many proof obligations and ultimately obtain the expected results. In fact, as detailed in Section 8, the only expected intervention from the user consists in manually adding the already proved theorems that can be used to complete the proof of another result. This reflects the dependencies of the different analyses, a step that can be further automated. For instance, admissibility of structural rules are usually needed to show invertibility results that, in turn, are extensively used in the proof of cut-elimination.

This paper is an extended version of the work [9], bringing not only new results and analyses, but also new uses of the `L-Framework` tool. In particular:

- *Properties and the spectrum of logics:* the procedures have been updated and extended to automatically check invertibility of rules in a larger class of logical systems, including modal logics and variants of linear logic. Moreover, the current analyses prove stronger properties: invertibility lemmas and admissibility of structural rules are shown to be *height-preserving* (a fact needed in the proof of cut-admissibility). For that, the definitions and necessary conditions in Section 5 were refined.
- *New reasoning techniques:* cut-admissibility is probably the most important property of sequent systems. This paper shows how to use the proposed framework to automatically discharge most of the cases in proofs of cut-admissibility. In some cases, the proposed algorithms are able to complete entire cut-admissibility proofs. The cases of failure are also interesting, since they tell much about the reasons for such failure. Some are fixable by proving and adding invertibility and admissibility lemmas. Others can be used in order to shed light on the reason for the failure. Finally, this work also addresses the dual property of identity expansion. Together with cut-admissibility, this property is key for designing harmonical systems [10].
- *Pretty printing and output:* the previous implementation has been updated to generate proof terms that can be used, *e.g.*, to produce LaTeX files with the proof trees in the meta-proofs. This allows for generating documents with complete and detailed proofs of several results in the literature, as well as identifying dependencies among the different theorems.

Outline. The rest of the paper is organized as follows. Section 2 introduces the structural properties of sequent systems that are considered in this work and Section 3 presents order-sorted rewriting logic and its main features as a logical framework. Section 4 presents the machinery used for specifying sequent systems as rewrite theories. Then, Section 5 establishes how to prove the structural properties based on a rewriting approach. Section 6 addresses cut-admissibility and identity expansion. The design principles behind the `L-Framework` are described in Section 7. Section 8 presents different sequent systems and properties that can be proved with the approach. Finally, Section 9 concludes the paper and presents some future research directions.

2. Structural properties of sequent-based proof systems

This section presents and illustrates with examples the properties of *admissibility* and *invertibility* of rules in sequent systems [11,12]. Additional notation and standard definitions are established to make the text self-contained.

Definition 1 *(Sequent).* Let $\mathscr{L}$ be a formal language consisting of well-formed formulas. A *sequent* is an expression of the form $\Gamma \vdash \Delta$ where Γ (the *antecedent*) and Δ (the *succedent*) are finite multisets of formulas in $\mathscr{L}$, and $\vdash$ is the *consequence* symbol. If the succedent of a sequent contains at most one formula, it is called *single-conclusion*; otherwise, it is called *multiple-conclusion*.

Definition 2 *(Sequent system).* A *sequent system* $\mathscr{S}$ is a finite set of inference rules of the form:

$$\frac{S_1 \quad \cdots \quad S_n}{S}\ r$$

$$\frac{}{\Gamma, p \vdash p}\ I \qquad \frac{}{\Gamma \vdash \top}\ \top_R \qquad \frac{\Gamma \vdash C}{\Gamma, \top \vdash C}\ \top_L \qquad \frac{}{\Gamma, \bot \vdash C}\ \bot_L$$

$$\frac{\Gamma, F \vdash C \quad \Gamma, G \vdash C}{\Gamma, F \vee G \vdash C}\ \vee_L \qquad \frac{\Gamma \vdash F_i}{\Gamma \vdash F_1 \vee F_2}\ \vee_{R_i} \qquad \frac{\Gamma, F, G \vdash C}{\Gamma, F \wedge G \vdash C}\ \wedge_L \qquad \frac{\Gamma \vdash F \quad \Gamma \vdash G}{\Gamma \vdash F \wedge G}\ \wedge_R$$

$$\frac{\Gamma, F \supset G \vdash F \quad \Gamma, G \vdash C}{\Gamma, F \supset G \vdash C}\ \supset_L \qquad \frac{\Gamma, F \vdash G}{\Gamma \vdash F \supset G}\ \supset_R$$

Fig. 1. System G3ip for propositional intuitionistic logic. In the I axiom, p is atomic.

where the sequent S is the *conclusion* inferred from the *premise* sequents $S_1, \ldots, S_n$ in the rule r. If the set of premises is empty, then r is an *axiom*. In a rule introducing a connective, the formula with that connective in the conclusion sequent is the *principal formula*. When sequents are restricted to having empty antecedents (hence being of the shape $\vdash \Delta$), the system is called *one-sided*; otherwise it is called *two-sided*.

As an example, Fig. 1 presents the two-sided single-conclusion propositional intuitionistic sequent system G3ip [11], with formulas built from the grammar:

$$F, G ::= p \mid \top \mid \bot \mid F \vee G \mid F \wedge G \mid F \supset G$$

where p ranges over atomic propositions. In this system, for instance, the formula $F \vee G$ in the conclusion sequent of the inference rule $\vee_L$ is the principal formula.

Definition 3 *(Derivation).* A *derivation* in a sequent system $\mathscr{S}$ (called $\mathscr{S}$-derivation) is a finite rooted tree with nodes labeled by sequents, axioms at the leaf nodes, and where each node is connected with the (immediate) successor nodes (if any) according to its inference rules. A sequent S is *derivable* in the sequent system $\mathscr{S}$, notation $\mathscr{S} \blacktriangleright S$, iff there is a derivation of S in $\mathscr{S}$. The system $\mathscr{S}$ is usually omitted when it can be inferred from the context.

It is important to distinguish the two different notions associated to the symbols $\vdash$ and $\blacktriangleright$, namely: the former is used to build sequents and the latter denotes derivability in a sequent system.

Definition 4 *(Height of derivation).* The *height* of a derivation is the greatest number of successive applications of rules in it, where an axiom has height 1. The expression

$$\mathscr{S} \blacktriangleright_n \Gamma \vdash \Delta$$

denotes that the sequent $\Gamma \vdash \Delta$ is derivable in $\mathscr{S}$ with a height of derivation *at most* n. A property is said *height-preserving* if such n is an invariant. The annotated sequent

$$\Gamma \vdash_n \Delta$$

is a shorthand for $\mathscr{S} \blacktriangleright_n \Gamma \vdash \Delta$ when $\mathscr{S}$ is clear from the context. Moreover, when S is a sequent, this notation is further simplified to $n : S$.

In what follows, annotated sequents are freely used in inference rules. For instance, if $suc(\cdot)$ represents the successor function on natural numbers, then

$$\frac{n : S_1 \quad \ldots \quad n : S_k}{suc(n) : S}\ r$$

represents the annotated rule r stating that

$$\text{if } \mathscr{S} \blacktriangleright_n S_i \text{ for each } i = 1, \ldots, k, \text{ then } \mathscr{S} \blacktriangleright_{suc(n)} S$$

Some other rules can be added to the system without changing the set of theorems (provable formulas) in it. These added rules may ease the reasoning when proving sequents and they are said to be *admissible* in the system. *Invertibility*, on the other hand, is the admissibility of "upside-down" rules, where the premises of a rule are derived from the conclusion. Invertibility is one of the most important properties in proof theory, since it is the core of proofs of cut-admissibility [1], as well as the basis for tailoring focused proof systems [13,14].

Definition 5 *(Admissibility and invertibility).* Let $\mathscr{S}$ be a sequent system. An inference rule $\dfrac{S_1 \quad \cdots \quad S_k}{S}$ is called:

i. *admissible in* $\mathscr{S}$ if S is derivable in $\mathscr{S}$ whenever $S_1, \ldots, S_k$ are derivable in $\mathscr{S}$;
ii. *invertible* in $\mathscr{S}$ if the rules $\frac{S}{S_1}, \ldots, \frac{S}{S_k}$ are admissible in $\mathscr{S}$.

The admissibility of structural rules is often required in the proof of other properties. A *structural rule* does not introduce logical connectives, but instead changes the structure of the sequent. Since sequents are built from multisets, such changes are related to the multiplicity of a formula or its presence/absence in a context.

In the intuitionistic setting, the structural rules for *weakening* and *contraction*

$$\frac{\Gamma \vdash C}{\Gamma, \Delta \vdash C}\ \mathsf{W} \qquad \text{and} \qquad \frac{\Gamma, \Delta, \Delta \vdash C}{\Gamma, \Delta \vdash C}\ \mathsf{C} \tag{1}$$

are height-preserving admissible in G3ip. A proof of the admissibility of weakening can be obtained by induction on the height of derivations (considering all possible rule applications) and it is often independent of any other results. For example, suppose that G3ip $\blacktriangleright_n \Gamma \vdash C$ and consider, *e.g.*, the case where $C = A \supset B$ and that the last rule applied in the proof of $\Gamma \vdash A \supset B$ is $\supset_R$

$$\frac{\Gamma, A \vdash_{\mathsf{i}} B}{\Gamma \vdash_{suc(i)} A \supset B}\ \supset_R$$

By inductive hypothesis, the sequent $\Gamma, \Delta, A \vdash_{\mathsf{i}} B$ is derivable and then,

$$\frac{\Gamma, \Delta, A \vdash_{\mathsf{i}} B}{\Gamma, \Delta \vdash_{suc(i)} A \supset B}\ \supset_R$$

The same exercise can be done for the other rules of the system, thus showing that W is height-preserving admissible:

$$\text{G3ip} \blacktriangleright_n \Gamma \vdash C \text{ implies G3ip} \blacktriangleright_n \Gamma, \Delta \vdash C$$

On the other hand, proving invertibility of a rule may require the admissibility of weakening. For example, for proving that $\supset_R$ is height-preserving invertible in G3ip, one has to show that G3ip $\blacktriangleright_n \Gamma, F \vdash G$ whenever G3ip $\blacktriangleright_n \Gamma \vdash F \supset G$. The proof follows by induction on the height of the derivation of $\Gamma \vdash F \supset G$. Consider, *e.g.*, the case when $\Gamma = \Gamma', A \supset B$ and the last rule applied is $\supset_L$

$$\frac{\Gamma', A \supset B \vdash_{\mathsf{i}} A \quad \Gamma', B \vdash_{\mathsf{i}} F \supset G}{\Gamma', A \supset B \vdash_{suc(i)} F \supset G}\ \supset_L$$

By inductive hypothesis on the right premise, $\Gamma', B, F \vdash_{\mathsf{i}} G$ is derivable. Considering the left premise, since $\Gamma \vdash_{\mathsf{i}} A$ is derivable, height-preserving admissibility of weakening implies that $\Gamma, F \vdash_{\mathsf{i}} A$ is derivable and the result follows:

$$\frac{\Gamma, F \vdash_{\mathsf{i}} A \quad \Gamma', B, F \vdash_{\mathsf{i}} G}{\Gamma, F \vdash_{suc(i)} G}\ \supset_L$$

Note that not all introduction rules in G3ip are invertible: if p_1, p_2 are different atomic propositions, then $p_i \vdash p_1 \vee p_2$ is derivable for $i = 1, 2$, but $p_i \vdash p_j$ is not for $i \neq j$. Indeed, $\vee_{R_i}$ and $\supset_L$ are the only non-invertible rules in G3ip.

Finally, admissibility of contraction (C) often depends on invertibility results. As an example, consider that G3ip $\blacktriangleright_n \Gamma, F \vee G, F \vee G \vdash C$ with last rule applied $\vee_L$

$$\frac{\Gamma, F \vee G, F \vdash_{\mathsf{i}} C \quad \Gamma, F \vee G, G \vdash_{\mathsf{i}} C}{\Gamma, F \vee G, F \vee G \vdash_{suc(i)} C}\ \vee_L$$

The inductive hypothesis cannot be applied since the premises do not have duplicated copies of formulas. Since $\vee_L$ is height-preserving invertible, the derivability of $\Gamma, F \vee G, F \vdash_{\mathsf{i}} C$ and $\Gamma, F \vee G, G \vdash_{\mathsf{i}} C$ implies, respectively, the derivability of $\Gamma, F, F \vdash_{\mathsf{i}} C$ and $\Gamma, G, G \vdash_{\mathsf{i}} C$. By induction, $\Gamma, F \vdash_{\mathsf{i}} C$ and $\Gamma, G \vdash_{\mathsf{i}} C$ are derivable and the result follows:

$$\frac{\Gamma, F \vdash_{\mathsf{i}} C \quad \Gamma, G \vdash_{\mathsf{i}} C}{\Gamma, F \vee G \vdash_{suc(i)} C}\ \vee_L$$

Invertibility and admissibility results will be largely used for showing cut-admissibility in Section 6.

3. Rewriting logic preliminaries

This section briefly explains order-sorted rewriting logic [5] and its main features as a logical framework. Maude [8] is a language and tool supporting the formal specification and analysis of rewrite theories, which are the specification units of rewriting logic.

An *order-sorted signature* Σ is a tuple $\Sigma{=}(S, \leq, F)$ with a finite poset of sorts $(S, \leq)$ and a set of function symbols F typed with sorts in S, which can be subsort-overloaded. For $X = \{X_s\}_{s \in S}$ an S-indexed family of disjoint variable sets with each X_s countably infinite, the *set of terms of sort s* and the *set of ground terms of sort s* are denoted, respectively, by $T_\Sigma(X)_s$ and $T_{\Sigma,s}$; similarly, $T_\Sigma(X)$ and T_Σ denote the set of terms and the set of ground terms. A *substitution* is an S-indexed mapping

$\theta : X \longrightarrow T_\Sigma(X)$ that is different from the identity only for a finite subset of X and such that $\theta(x) \in T_\Sigma(X)_s$ if $x \in X_s$, for any $x \in X$ and $s \in S$. A substitution θ is called *ground* iff $\theta(x) \in T_\Sigma$ or $\theta(x) = x$ for any $x \in X$. The application of a substitution θ to a term t is denoted by $t\theta$. Acquaintance with the usual notions of *position* p in a term t, *subterm* $t|_p$ at position p, and *term replacement* $t[u]_p$ of t's subterm at position p with term u is assumed (see, *e.g.*, [15]). The expression $u \preceq t$ (resp., $u \prec t$) denotes that term u is a subterm (resp., proper subterm) of term t. Given a term $t \in T_\Sigma(X)$, $\overline{t} \in T_{(S,\leq,F\cup C_{\overline{t}})}(X)$ is the ground term obtained from t by turning each variable $x \in vars(t)$ of sort $s \in S$ into the fresh constant $\overline{x}$ of sort s and where $C_{\overline{t}} = \{\overline{x} \mid x \in vars(t)\}$.

A *rewrite theory* is a tuple $\mathscr{R} = (\Sigma, E \uplus B, R)$ with: (i) $(\Sigma, E \uplus B)$ an order-sorted equational theory with signature Σ, $E \uplus B$ the disjoint union of E, a set of (possibly conditional) equations over $T_\Sigma(X)$ and B, a set of structural axioms over $T_\Sigma(X)$ for which there is a finitary matching algorithm (*e.g.*, associativity, commutativity, and identity, or combinations of them); and (ii) R a finite set of rewrite rules over $T_\Sigma(X)$ (possibly with equational conditions). A rewrite theory $\mathscr{R}$ induces a rewrite relation $\rightarrow_{\mathscr{R}}$ on $T_\Sigma(X)$ defined for every $t, u \in T_\Sigma(X)$ by $t \rightarrow_{\mathscr{R}} u$ if and only if there is a *one-step* rewrite proof of $(\forall X) t \rightarrow u$ in $\mathscr{R}$. More precisely, $t \rightarrow_{\mathscr{R}} u$ iff there is a rule $(l \rightarrow r \textbf{ if } \gamma) \in R$, a term t', a position p in t', and a substitution $\theta : X \longrightarrow T_\Sigma(X)$ satisfying $t =_{E\uplus B} t'=t'[l\theta]_p$, $u =_{E\uplus B} t'[r\theta]_p$, and $\gamma\theta$ can be inferred from $\mathscr{R}$. When the condition γ is *true*, then the (conditional) rule $l \rightarrow r \textbf{ if } \gamma$ is simply written as $l \rightarrow r$.

The tuple $\mathscr{T}_{\mathscr{R}} = (\mathscr{T}_{\Sigma/E}, \overset{*}{\rightarrow}_{\mathscr{R}})$, where $\overset{*}{\rightarrow}_{\mathscr{R}}$ is the reflexive-transitive closure of $\rightarrow_{\mathscr{R}}$, is called the *initial reachability model of* $\mathscr{R}$ [16]. An inductive property of a rewrite theory $\mathscr{R}$ does not need to hold for any model of $\mathscr{R}$, but just for $\mathscr{T}_{\mathscr{R}}$. Using a suitable inductive inference system, for example, one based on a convenient notion of constructors as proposed in [17], the semantic entailment $\models$ in $\mathscr{T}_{\mathscr{R}}$ can be under-approximated by an inductive inference relation $\Vdash$ over $\mathscr{R}$, which is shown to be sound with respect to $\models$ (*i.e.*, for any property φ, if $\mathscr{R} \Vdash \varphi$, then $\mathscr{T}_{\mathscr{R}} \models \varphi$). A Σ-sentence $(\forall X)\, \varphi$ is called an *inductive consequence* of $\mathscr{R}$ iff $\mathscr{R} \Vdash (\forall X)\, \varphi$, and this implies that $\mathscr{T}_{\mathscr{R}} \models \varphi$.

Appropriate requirements are needed to make an equational theory $\mathscr{R}$ *executable* in Maude. It is assumed that the equations E can be oriented into a set of (possibly conditional) sort-decreasing, operationally terminating, and confluent rewrite rules $\overrightarrow{E}$ modulo B [8]. For a rewrite theory $\mathscr{R}$, the rewrite relation $\rightarrow_{\mathscr{R}}$ is undecidable in general, even if its underlying equational theory is executable, unless conditions such as coherence [18] are given (*i.e.*, rewriting with $\rightarrow_{\mathscr{R}}$ can be decomposed into rewriting with $\rightarrow_{E/B}$ and $\rightarrow_{R/B}$). The executability of a rewrite theory $\mathscr{R}$ ultimately means that its mathematical and execution semantics coincide.

In this paper, the rewriting logic specification of a sequent system $\mathscr{S}$ is a rewrite theory $\mathscr{R}_{\mathscr{S}} = (\Sigma_{\mathscr{S}}, E_{\mathscr{S}} \uplus B_{\mathscr{S}}, R_{\mathscr{S}})$ where: $\Sigma_{\mathscr{S}}$ is an order-sorted signature describing the syntax of the logic $\mathscr{S}$; $E_{\mathscr{S}}$ is a set of executable equations modulo the structural axioms $B_{\mathscr{S}}$; and $R_{\mathscr{S}}$ is a set of executable rewrite rules modulo $B_{\mathscr{S}}$ capturing those non-deterministic aspects of logical inference in $\mathscr{S}$ that require proof search. The point is that although both the computation rules $E_{\mathscr{S}}$ and the deduction rules $R_{\mathscr{S}}$ are executed by rewriting modulo $B_{\mathscr{S}}$, by the executability assumptions on $\mathscr{R}_{\mathscr{S}}$, the rewrite relation $\rightarrow_{E_{\mathscr{S}}/B_{\mathscr{S}}}$ has a single outcome in the form of a canonical form and thus can be executed blindly with a "don't care" non-deterministic strategy. Furthermore, $B_{\mathscr{S}}$ provides yet one more level of computational automation in the form of $B_{\mathscr{S}}$-matching and $B_{\mathscr{S}}$-unification algorithms. This interplay between axioms, equations, and rewrite rules can ultimately make the executable specification $\mathscr{R}_{\mathscr{S}}$ very efficient with modest memory requirements.

The expression $CSU_B(t, u)$ denotes the *complete set of unifiers* of terms t and u modulo the structural axioms B. Recall that for each substitution $\sigma : X \longrightarrow T_\Sigma(X)$, such that $t\sigma =_B u\sigma$, there are substitutions $\theta \in CSU_B(t, u)$ and $\gamma : X \longrightarrow T_\Sigma(X)$ satisfying $\sigma =_B \theta\gamma$. For a combination of free and associative and/or commutative and/or identity axioms B, except for symbols that are associative but not commutative, such a finitary unification algorithm exists [19].

The unification problems solved here comprise terms of sort `Sequent` (defined in the next section), where the set B corresponds to identity, commutativity, and associativity (ACU). Those are the usual structural axioms for multisets of formulas needed to represent the left and right contexts in sequents. Hence, the finitary algorithm implemented in Maude is used [20]. Unification is used to mechanize some of the inference rules in the proposed framework by exploiting the structure of terms, and thus implementing reasoning with the inductive relation $\Vdash$. Moreover, matching modulo ACU is needed because sequents are included as part of the simplification tasks internally performed by the proposed heuristics to discharge proof obligations. In Maude, both matching and unification modulo ACU are built-in procedures, and used here as blackboxes. A recent account of such features can be found in [21].

4. A rewriting view of sequent systems

This section presents the machinery used for specifying a propositional sequent system $\mathscr{S}$ as a rewrite theory $\mathscr{R}_{\mathscr{S}}$. The framework is equipped with sorts that represent formulas, multisets of formulas, sequents, and lists of sequents. The user of the framework is expected to inhabit the sort for formulas in $\mathscr{R}_{\mathscr{S}}$ with the concrete syntax of the system $\mathscr{S}$. This has the immediate effect of fully inhabiting the remaining sorts of $\mathscr{R}_{\mathscr{S}}$. As shown below, rewrite rules in $\mathscr{R}_{\mathscr{S}}$ correspond to backwards inference rules of $\mathscr{S}$, so that proof-search in the former is successful whenever all leaves in a proof-tree are instances of axioms. The system G3ip (Fig. 1) is used throughout the section for illustrating the proposed approach.

The notation of Maude [8] is adopted as an alternative representation to the rewriting logic one introduced in Section 3. This decision has the immediate effect of producing an executable specification of sequent systems, while providing a precise mathematical semantics of the given definitions. Additional details about the implementation in Maude are given in Section 7.

As a reference for the development of this section, Maude declarations are summarized next.

```
mod M is ... endm                --- Rewrite theory M
sort S .                         --- Declaration of sort S
subsort S1 < S2 .                --- Subsort relation
op f : S1 ... Sn -> T .          --- Funct. sym. of sort T with
                                 ---    params in S1 x ... x Sn
op c : -> T .                    --- Constant c of sort T
rl  [rule-name] : l => r .       --- Rewrite rule with name 'rule-name'
crl [rule-name] : l => r if c .  --- Conditional rewrite rule
```

An order-sorted signature $\Sigma_{\mathscr{S}}$ defining the sorts for building formulas (and multisets of formulas and sequents, which are introduced later), is assumed:

```
sort Prop .               --- Atomic propositions
sort Formula .            --- Formulas
subsort Prop < Formula .  --- Atomic propositions are formulas
```

The object logic to be specified must provide suitable constructors for these two sorts. For instance:

```
mod G3i is
 ...
 op p : Nat -> Prop .                        --- Atomic  Propositions
 op _/\_ : Formula Formula -> Formula .      --- Conjunction
 op False : -> Formula .                     --- False/bottom
 ...
endm
```

Atomic propositions take the form $p(n)$, where the natural number n is the identifier of the proposition (*e.g.*, the proposition p_3 is written as `p(3)`). Constructors for `Prop` are not allowed to have arguments of type `Formula`.

Multisets of formulas are built in the usual way:

```
sort MSFormula .              --- Multiset of Formulas
subsort Formula < MSFormula . --- A formula is also a singleton
op * : -> MSFormula .         --- Empty multiset
--- Multiset Union
op _;_ : MSFormula MSFormula -> MSFormula [assoc comm id: * ] .
```

Given two multisets M and N, the term `M;N` denotes the multiset union of M and N. Note that the mixfix operator `op _;_` (in Maude, _ denotes the position of the parameters) is declared with three structural axioms: associativity, commutativity, and the empty multiset as its identity element. Due to the subsort relation `Formula < MSFormula`, a formula F is also a singleton containing F.

Sequents, as expected, are built as pairs of multisets of formulas. A *goal* is a list of sequents to be proved:

```
sorts Sequent  .
op _|--_ : MSFormula MSFormula -> Sequent .  --- Sequents
--- Goals: List of sequents to be proved
sorts Goal .
subsort Sequent < Goal .
op proved : -> Goal .                        --- The empty list
op _|_ : Goal Goal -> Goal [frozen(2) id: proved] .
```

The attribute `frozen` used in the declaration of the operator `op _|_` (goals' concatenation) means that inference steps can only be performed on the head of a non-empty list of goals (*i.e.*, the usual 'head-tails' recursive structure). More precisely, the attribute `frozen(2)` defines a rewriting strategy where the second parameter cannot be subject of rewriting. In this way, only the first sequent `S` in the list `S | G` can be reduced until it becomes `proved` (when possible), as will be explained shortly.

Inference rules in the theory $\mathscr{R}_{\mathscr{S}}$ are specified as rules that rewrite list of sequents (*i.e.*, goals). There are two options for expressing an inference rule as a rewrite rule. Namely, they can be axiomatized as *backwards* inference (*i.e.*, from conclusion to premises) or as *forward* inference (*i.e.*, from premises to conclusion). In this paper, as explained in Section 3, the backwards inference approach is adopted, so that a proof-search process advances by rewriting the target goal of an inference rule to its premises, thus implementing a goal directed proof-search procedure. For instance, the initial rule, and the left and right introduction rules for conjunction in G3ip are specified as follows:

```
var P : Prop .
var Gamma : MSFormula .
vars A B C : Formula .
--- Rules
```

```
rl [I] : P ; Gamma  |-- P => proved .
rl [AndL] : A /\ B ; Gamma |-- C => A ; B ; Gamma |-- C .
rl [AndR] : Gamma |-- A /\ B => (Gamma |-- A) | (Gamma |-- B) .
```

Variables in rules are implicitly universally quantified. The type of variables is specified with the syntax `var x : T .` Hence, the initial rule must be read as

$$(\forall P : \texttt{Prop}, \Gamma : \texttt{MSFormula})((P, \Gamma \vdash P) \rightarrow \texttt{proved})$$

The constant `proved` denotes the empty list of goals or, equivalently, the empty collection of (pending) proof obligations. Rules with more than one premise, such as `AndR`, are specified using `op _|_`. Modulo the axioms $B_{\mathscr{S}}$, a term `proved | G` is structurally equal to `G`, thus making the goal `G` automatically active for proof-search under the rewrite strategy declared for goals.

As illustrated with the running example, the syntax of the object logic in $\mathscr{R}_{\mathscr{S}}$, as well as its inference rules, is straightforward. This is usually called in rewriting logic the *ε-representational distance* [22], where the system being specified mathematically as a rewrite theory and its codification in Maude as a system module are basically the same. This is certainly an appealing feature that can widen the adoption of the framework proposed here for implementing and analyzing sequent systems.

In face of the ε-representational distance, the adequacy of $\mathscr{R}_{\mathscr{S}}$ with respect to $\mathscr{S}$ follows from the soundness of rewriting logic itself.

Proposition 1 *(Adequacy). Let $\mathscr{S}$ be a sequent system, $\mathscr{R}_{\mathscr{S}}$ the resulting rewrite theory encoding the syntax and inference rules of $\mathscr{S}$, S a sequent in $\mathscr{S}$, and t_S its representation in $\mathscr{R}_{\mathscr{S}}$. Then,*

$$\mathscr{S} \blacktriangleright S \quad \textit{iff} \quad \mathscr{R}_{\mathscr{S}} \Vdash t_S \xrightarrow{*} \texttt{proved}.$$

Observe that the proof of a sequent S in $\mathscr{S}$ may follow different strategies depending on the order the subgoals are proved. Such strategies are irrelevant because all the branches in the derivation of a proof must be closed (*i.e.*, ending with an instance of an axiom). Hence, if S is provable, it can be proved using the strategy enforced by `op _|_` in $\mathscr{R}_{\mathscr{S}}$: always trying to first solve the left-most subgoal of the pending goals.

Let r be a sequent rule in $\mathscr{S}$ and R the corresponding rewrite rule in $\mathscr{R}_{\mathscr{S}}$. Modulo the associativity and commutativity of multisets of formulas, it is easy to show that, for any sequent S and its representation t_S, $t_S \xrightarrow{1}_{R_{\mathscr{S}}} t'_S$ iff t'_S is the representation of the premises (where `proved` means no premises) obtained when r is applied (on the same active formula) in S. Hence, rewrite steps are in one-to-one correspondence with proof search steps in $\mathscr{S}$-derivations.

When using the proposed framework, the resulting rewrite theory becomes a proof search procedure. For instance, Maude's command

```
search p(1) /\ p(2) |-- p(2) /\ p(1) =>* proved .
```

answers the question of whether the sequent $p_1 \wedge p_2 \vdash p_2 \wedge p_1$ is provable in G3ip.

For the proof analyses later developed in this paper, operations to the rewrite theory $\mathscr{R}_{\mathscr{S}}$ are added. A new constructor for annotated sequents (see Definition 4) and also a copy of the inference rules dealing with height annotations are included:

```
op _:_ : Nat Sequent -> Sequent .  --- Annotated sequents
var k : Nat .
--- Automatically generated annotated sequent rules
rl [I]    : suc(k) : P ; Gamma  |-- P => proved.
rl [AndL] : suc(k) : A /\ B ; Gamma |-- C => k : A ; B ; Gamma |-- C .
rl [AndR] : suc(k) : Gamma |-- A /\ B => ( k : Gamma |-- A) |  ( k : Gamma |-- B) .
```

The function application `suc(.)` denotes the successor function on `Nat`. Note that axioms are annotated with an arbitrary height $k \geq 1$. In rules with two premises, both of them are marked with the same label. This is without loss of generality since it is always possible to obtain larger annotations from shorter ones (see Lemma 1).

In the rest of the paper, given a sequent system $\mathscr{S}$, $\mathscr{R}_{\mathscr{S}}$ will denote the resulting rewrite theory that encodes the syntax, inference, and annotated inference rules of $\mathscr{S}$. By abusing the notation, when a sequent $S \in \mathscr{S}$ is used in the context of the rewrite theory $\mathscr{R}_{\mathscr{S}}$ (*e.g.*, in $S \rightarrow_{\mathscr{R}_{\mathscr{S}}} S'$), such S must be understood as the corresponding term $t_S \in \Sigma_{\mathscr{R}_{\mathscr{S}}}(X)$ representing S in $\mathscr{R}_{\mathscr{S}}$. Similarly, "a sequent rule r" in the context of the theory $\mathscr{R}_{\mathscr{S}}$ must be understood as "the representation of r in $\mathscr{R}_{\mathscr{S}}$". The expression $\mathscr{R}_1 \cup \mathscr{R}_2$ denotes the union of the theories $\mathscr{R}_1$ and $\mathscr{R}_2$. If S is a sequent, the expression $\mathscr{R} \cup \{S\}$ denotes the extension resulting from $\mathscr{R}$ by adding the sequent S as an axiom, understood as a zero-premise rule (*i.e.*, $\mathscr{R}$ is extended with the rule `rl [ax] S => proved .`). Moreover, given a rewrite theory $\mathscr{R}$ and a sequent S, the notation $\mathscr{R} \Vdash S$ means $\mathscr{R} \Vdash S \xrightarrow{*} \texttt{proved}$, *i.e.*, there is a derivation of S in the system specified by $\mathscr{R}$. Similarly, for annotated sequents, $\mathscr{R} \Vdash n : S$ means $\mathscr{R} \Vdash (n : S) \xrightarrow{*} \texttt{proved}$.

Lemma 1. *Let $\mathscr{S}$ be a sequent system, S a sequent, and $k \geq 1$. Then,*

$$\mathscr{R}_{\mathscr{S}} \Vdash k : S \quad \textit{implies} \quad \mathscr{R}_{\mathscr{S}} \Vdash suc(k) : S.$$

Proof. By induction on k. If $k = 1$, then S is necessarily an instance of an axiom and $suc(1) : S \rightarrow_{\mathscr{R}_{\mathscr{S}}} \texttt{proved}$ using the same axiom. The case $k > 1$ is immediate by applying induction on the premises (with shorter derivations). An instance of this theorem will be (automatically) proved for each one of the systems studied in Section 8. □

5. Proving admissibility and invertibility

This section presents rewrite-based techniques for proving admissibility and invertibility of inference rules of a sequent system specified as a rewrite theory $\mathscr{R}_{\mathscr{S}}$ (see Section 4). They are presented as meta-theorems about sequent systems with the help of rewrite-related scaffolding, such as terms and substitutions, and they provide sufficient conditions for obtaining the desired properties. The system G3ip is, as in previous sections, used as running example (Section 8 presents the complete set of case studies).

The procedures proposed here heavily depend on *unification* and a brief discussion on the subject is presented next. Since no additional axioms are added to the symbols in the syntax of the object logic $\mathscr{S}$, the existence of a unification algorithm for terms in the sort `Sequent` (whose only symbol defined with axioms is `_;_` for building multiset of formulas) is guaranteed. Furthermore, recall from Section 4 that rewrite rules in $\mathscr{R}_{\mathscr{S}}$ correspond to backwards inference rules of $\mathscr{S}$, so that proof-search in the former is successful whenever all leaves in a proof-tree are instances of axioms.

As an example, consider the unification of the conclusions of the right and left rules for conjunction in G3ip. Unification problems take the form `t1 =? t1' /\ ... /\ tn =? tn'` and Maude's command `irredundant unify` computes the set of unifiers modulo the declared axioms in the theory[1]:

```
irredundant unify suc(n) : A /\ B ; Gamma  |-- C          =?
                  suc(n') :         Gamma' |-- A' /\ B'   .

Unifier #1
n --> %3:FNat                       n' --> %3:FNat
A --> %1:Formula                    A' --> %4:Formula
B --> %2:Formula                    B' --> %5:Formula
C --> %4:Formula /\ %5:Formula
Gamma --> %6:MSFormula
Gamma' --> %6:MSFormula ; %1:Formula /\ %2:Formula

No more unifiers.
```

Observe that the variables of the second rule are renamed to avoid clash of names. The term `%1:Formula` denotes a fresh variable of sort `Formula`. Let t and t' be the two terms in the above unification problem and θ the (unique) substitution computed. Consider the least signature $\Sigma'_{\mathscr{R}_{\mathscr{S}}}$ that contains $\Sigma_{\mathscr{R}_{\mathscr{S}}}$ as well as a fresh constant `%i.Type` for each variable `%i:Type` in $t\theta$. Note the ":" in the variable `%i:Type` and the "." in the constant `%i.Type`. To avoid confusion, when the sort can be inferred from the context, the constant `%i.Type` is written as `%i`, but variables always carry their typing information. In this example, $\Sigma'_{\mathscr{R}_{\mathscr{S}}}$ extends $\Sigma_{\mathscr{R}_{\mathscr{S}}}$ with constants

`op %1 : -> Formula .`, `op %6 : -> MSFormula .`, etc.

Recall from Section 3 that given a term t, $\overline{t}$ is the ground term obtained from t by replacing its variables with fresh constants. In the current example, the ground term $\overline{t\theta}$ in the signature $\Sigma'_{\mathscr{R}_{\mathscr{S}}}$ is `suc(%4) : %1 /\ %2 ; %6 |-- %4 /\ %5`. As expected, on this (ground) sequent it is possible to apply both `AndL` and `AndR`.

5.1. Admissibility of rules

This section introduces sufficient conditions to prove theorems of the form

$$\text{if } \mathscr{S} \blacktriangleright_n \Gamma' \vdash \Delta' \text{, then } \mathscr{S} \blacktriangleright_n \Gamma \vdash \Delta$$

i.e., height preserving admissibility of the rule[2]

[1] The "irredundant" version of the command `unify` in Maude 3.2.1 allows for filtering the computed unifiers against each other, thus obtaining a minimal set of most general unifiers.

[2] The admissibility problems considered in this section correspond to one-premise rules only. Observe that a general approach for proving admissibility of rules with more than one premise does not exist. Indeed, cut-admissibility itself is an admissibility problem, which will be addressed in Section 6.

$$\frac{\Gamma' \vdash \Delta'}{\Gamma \vdash \Delta}\ r$$

Admissibility is often proved by induction on the height of the derivation π of the sequent $\Gamma' \vdash \Delta'$ in combination with case analysis on the last rule applied in π. It turns out that this analysis may depend on other results. For example, as illustrated in Section 2, proving admissibility of contraction often depends on invertibility results. Hence, any general definition of admissibility of rules in the rewriting setting has to internalize such reasoning. This will be formalized by *closing* the leaves in π w.r.t. theorems of the form "if the sequent S_1 is provable, so is the sequent S_2". Such theorems will be encoded as rewrite rules of the form $r : S_1 \to S_2$. More precisely, let t_s be a ground term denoting a sequent and $r : t \to t'$ be a rule. The set

$$[\![t_s]\!]_r = \{t_s\} \cup \{t'\theta \mid \theta \in CSU(t, t_s)\}$$

is the least set containing t_s and the resulting premises after the instantiation of r in t_s. Let T_s be a set of ground terms and $R = \{r_1, ...r_m\}$ be a set of theorems of the form $r_k : t^k \to t_1^k$. Then,

$$[\![T_s]\!]_R = T_s \cup \bigcup_{k \in 1..m} \{t_1^k\theta \mid t \in T_s, \theta \in CSU(t^k, t)\}.$$

Proofs of admissibility need to consider all the cases of the last rule applied in a proof π. Definition 6 identifies rules that are height-preserving admissible *relative* to one of the sequent rules of the system.

Definition 6 *(Local admissibility).* Let $\mathscr{S}$ be a sequent system, $\mathscr{I}$ be a (possibly empty) set of rules, and $r_t, r_s \in \mathscr{S}$ be rules given by

$$\frac{k : T_1 \ \cdots \ k : T_n}{suc(k) : T}\ r_t \qquad \frac{S_1}{S}\ r_s$$

The rule r_s *is height-preserving admissible relative to* r_t *in* $\mathscr{S}$ under the assumptions $\mathscr{I}$ iff assuming that $i : S_1$ is provable then, for each $\theta \in CSU(i : S_1, suc(k) : T)$,

$$\mathscr{R}_{\mathscr{S}} \cup [\![\{\overline{(k : T_j)\theta} \mid j \in 1..n\}]\!]_{\mathscr{I}} \cup \{(\forall \vec{x})(\overline{k\theta} : S_1 \to \overline{k\theta} : S)\} \Vdash \overline{(i : S)\theta},$$

where $vars(i : S_1) \cap vars(suc(k) : T) = \emptyset$ and $\vec{x} = vars(S) \cup vars(S_1)$.

For proving admissibility of the rule r_s, the goal is to prove that, if S_1 is derivable with height i, then S is also derivable with at most the same height. The proof follows by induction on i (the height of a derivation π of S_1). Suppose that the last rule applied in π is r_t. This is only possible if S_1 and T "are the same", up to substitutions. Hence, the idea is that each unifier θ of $i : S_1$ and $suc(k) : T$ covers the cases where the rule r_t can be applied on the sequent S_1. For computing this unifier, it is assumed that the rules do not share variables (the implementation takes care of renaming the variables if needed). A proof obligation is generated for each unifier. Consider, for instance, the proof obligation of the ground sequent $\overline{(i : S)\theta}$ for a given $\theta \in CSU(i : S_1, suc(k) : T)$. This means that, as hypothesis, the derivation below is valid

$$\frac{(k : T_1)\theta \quad \cdots \quad (k : T_n)\theta}{(i : S_1)\theta}\ r_t \tag{2}$$

Hence, all the premises in this derivation are assumed derivable. This is the purpose of extending the theory with the following set of ground sequents (which are interpreted as rules of the form $t \to \texttt{proved}$):

$$\left\{\overline{(k : T_j)\theta} \mid j \in 1..n\right\} \tag{3}$$

The proof of admissibility theorems may require auxiliary lemmas. Assuming the theorems $\mathscr{I}$, the sequents resulting after an application of $r \in \mathscr{I}$ to the sequents in Equation (3) can also be assumed to be provable (notation $[\![\cdot]\!]_{\mathscr{I}}$). The typical instantiation of $\mathscr{I}$ in admissibility analysis will be the already proved invertibility lemmas.

If $\theta \in CSU(i : S_1, suc(k) : T)$, then θ should map k to a fresh variable, say `%1:Nat` and i to `suc(%1:Nat)`. Hence, the (ground) goal $\overline{(i : S)\theta}$ to be proved takes the form `suc(%1)` $: \overline{S\theta}$ where `%1`$= \overline{k\theta}$ is the freshly generated constant in the extended signature. By induction, it can be assumed that the theorem (*i.e.*, S_1 implies S) is valid for shorter derivations, *i.e.*, derivations of height at most `%1`. This is the purpose of the added rule

$$\{(\forall \vec{x})(\overline{k\theta} : S_1 \to \overline{k\theta} : S)\} \tag{4}$$

where the height of the derivation is "frozen" to be the constant $\overline{k\theta}$. This allows for applying r_s only on sequents of height $\overline{k\theta}$. In particular, induction can be used on all the premises of the rule r_t in Equation (2).

If it is possible to show that the ground sequent $\overline{(i : S)\theta}$ is derivable from the extended rewrite theory, then the admissibility result will work for the particular case when r_t is the last applied rule in the derivation π of S_1. Since a complete set of unifiers is finite for terms of the sort `Sequent`, then there are finitely many proof obligations to discharge in order to check if a rule is admissible relative to a rule in a sequent system. Observe that the set $CSU(i : S, suc(k) : T)$ may be empty. In this case, the set of proof obligations is empty and the property vacuously holds.

The base cases in this proof correspond to axiom rules. Consider for instance the case where r_t is the initial rule in the system G3ip. Since there are no premises, the set in Equation (3) is empty and hence there are no ground sequents of the form `%1:`$\overline{T_j\theta}$ as hypothesis. Consider the hypothetical case that `suc(%1)`$:\overline{S\theta}$ is rewritten to a term of the form `%1: S'` by using another rule of the system (different from the initial rule r_t). Note that it is not possible to use the rule in Equation (4) to reduce the goal to `proved`: an application of this rule produces inevitably yet another sequent of the form `%1: S''`. Thus the only hope to finishing the proof is to apply the initial rule on the sequent `suc(%1)`$:\overline{S\theta}$ and the rule in Equation (4) is never used in such a proof.

The notion of admissibility relative to a rule is the key step in the inductive argument, when establishing sufficient conditions for the admissibility of a rule in a sequent system.

Theorem 1. *Let $\mathscr{S}$ be a sequent system, $\mathscr{I}$ be a (possibly empty) set of rules, and* $\dfrac{\Gamma' \vdash \Delta'}{\Gamma \vdash \Delta}\ r_s$ *be an inference rule. If r_s is admissible relative to each r_t in $\mathscr{S}$ under the assumption $\mathscr{I}$, then r_s is height-preserving admissible in $\mathscr{S}$ under the assumption $\mathscr{I}$, i.e., $\mathscr{S} \blacktriangleright_n \Gamma' \vdash \Delta'$ implies $\mathscr{S} \blacktriangleright_n \Gamma \vdash \Delta$ assuming the theorems $\mathscr{I}$.*

Proof. Assume that $\Gamma' \vdash_n \Delta'$ is derivable in the system $\mathscr{S}$. The proof proceeds by induction on n with case analysis on the last rule applied. Assume that the last applied rule is r_t. By hypothesis (using Definition 6), it can be concluded that $\Gamma \vdash_n \Delta$ is derivable and the result follows. □

It is important to highlight the rationale behind Definition 6, which is similar to the results in the forthcoming sections. The proof search procedures try to show that $\overline{(i : S)\theta}$ is provable by reducing it to `provable` (Proposition 1). Since rules in $\mathscr{R}_{\mathscr{S}}$ are encoded in a backward fashion (rewriting the conclusion into the premises), the procedure attempts to build a goal directed derivation of that sequent. The set of assumptions (Equation (3)) is added as axioms and it is closed under the rules in $\mathscr{I}$. The closure reflects a forward reasoning: if the (ground) sequent S is provable and theorem $r \in \mathscr{I}$ applies on it, then the right-hand side of r can also be assumed to be provable. During the search procedure, a goal is immediately discharged if it belongs to that set of assumptions. Finally, the induction principle is encoded by specializing the theorem to be proved, and allowing applications of it to ground sequents with shorter height annotations (Equation (4)).

The proof of admissibility of weakening for G3ip and some other systems studied in Section 8 does not rely on any result (and hence, $\mathscr{I}$ is empty). Also, the proof of Lemma 1, mechanized by stating it as the admissibility of the annotated rule

$$\frac{n : S}{suc(n) : S}\ \mathsf{H} \tag{5}$$

is also proved with $\mathscr{I} = \emptyset$ in all the systems in Section 8. The proof of contraction requires some invertibility lemmas and they must be added to $\mathscr{I}$. The use of $\mathscr{I}$ thus presents a convenient and modular approach because properties can be proved incrementally.

Section 8.3 presents some further examples where the above definitions can be applied for proving that other rules with one premise satisfy the admissibility condition, *i.e.*, the provability of the premise implies the provability of the conclusion.

5.2. Invertibility of rules

This section gives a general definition for proving height-preserving invertibility of rules. Observe that such analysis is done premise-wise. The case of rules with several premises is performed for each one of them separately. For instance, consider the rule $\supset_L$ of G3ip, and let $\supset_{L1}$ and $\supset_{L2}$ be the rules

$$\frac{\Gamma, F \supset G \vdash F \quad \Gamma, G \vdash C}{\Gamma, F \supset G \vdash C}\ \supset_L \qquad \frac{\Gamma, F \supset G \vdash C}{\Gamma, F \supset G \vdash F}\ \supset_{L1} \qquad \frac{\Gamma, F \supset G \vdash C}{\Gamma, G \vdash C}\ \supset_{L2}$$

It can be shown that $\supset_{L1}$ is not admissible while $\supset_{L2}$ is. Hence, $\supset_L$ is invertible only in its right premise (see Definition 5).

Some invertibility results depend on, *e.g.*, the admissibility of weakening. Hence, for modularity, the invertibility analysis is parametric under a set of rules $\mathscr{H}$ of admissible rules of the system (that can be used during the proof search procedure).

Definition 7 *(Local invertibility).* Let $\mathscr{S}$ be a sequent system and $\mathscr{H}$ be a (possibly empty) set of rules. Consider the following annotated inference rules:

$$\frac{k : S_1 \ \cdots\ k : S_p}{suc(k) : S}\ r_s \qquad \frac{m : T_1 \ \cdots\ m : T_n}{suc(m) : T}\ r_t$$

Under the assumption $\mathscr{H}$, the premise $l \in 1..p$ of the rule r_s *is height-preserving invertible relative to the rule* r_t iff for each $\theta \in CSU(suc(k) : S, suc(m) : T)$:

$$\mathscr{H} \cup \mathscr{R}_{\mathscr{S}} \cup \left\{\overline{(m : T_j)\theta} \mid j \in 1..n\right\} \cup \bigcup_{j \in 1..n} \{\overline{(m : S_l)\gamma} \mid \gamma \in CSU(k : S, \overline{(m : T_j)\theta})\} \Vdash \overline{(k : S_l)\theta}$$

where the variables in S and T are assumed disjoint. Under the assumption $\mathscr{H}$, the rule r_s is height-preserving invertible relative to r_t if all its premises are.

In order to show the invertibility of a rule r_s, the goal is to check that derivability is not lost when moving from the conclusion S to the premises $S_1, \cdots, S_p$. Each premise S_l entails a different proof obligation. Let $\dfrac{S_l}{S}\ r_{sl}$ be the "sliced" version of rule r_s when the case of the premise $l \in 1..p$ is considered. The proof is by induction on the derivation π of S. Suppose that the last rule applied in π is r_t. Observe that this is only possible if S and T unify. Let $\theta \in CSU(suc(k) : S, suc(m) : T)$ and assume that $\theta(k) = \texttt{\%1:Nat}$ (and hence $\theta(m) = \texttt{\%1:Nat}$). The resulting derivation π takes the form

$$\frac{\%1 : \overline{T_1\theta} \ldots \%1 : \overline{T_n\theta}}{suc(\%1) : \overline{S\theta}}\ r_t \tag{6}$$

The premises of this derivation are assumed to be provable and the theory is extended with the axioms $\overline{(m : T_j)\theta}$. Since those premises have shorter derivations, induction applies on them. More precisely, given that the ground sequent $\overline{(m : T_j)\theta}$ is provable and the rule r_{sl} can be applied on it, the resulting sequent after the application of r_{sl} ($T_j S_l$ below) is also provable with the same height of derivation $\overline{m\theta}$:

$$\text{if } \frac{T_j S_l}{\overline{T_j\theta}}\ r_{sl} \text{then the sequent } T_j S_l \text{ is provable with height } \%1 \tag{7}$$

This inductive reasoning is captured by the expression

$$\bigcup_{j \in 1..n} \{\overline{(m : S_l)\gamma} \mid \gamma \in CSU(k : S, \overline{(m : T_j)\theta})\}$$

The unifier γ checks whether it is possible to apply r_{sl} on the premise T_j. If this is the case, the resulting premise $S_l\gamma$ is added as an axiom. If, from the extended theory, it is possible to prove derivable the premise S_l with height $\overline{k\theta}$, then the invertibility result will work for the particular case when r_t was the last applied rule in the derivation π of S. If the set $CSU(suc(k)S, suc(m) : T)$ is empty, it means that the rules r_t and r_s cannot be applied on the same sequent and the property vacuously holds for that particular case of r_t. For instance, in G3ip, the proof of invertibility of $\wedge_R$ does not need to consider the case of invertibility relative to $\vee_R$ since it is not possible to have, at the same time, a conjunction and a disjunction on the succedent of the sequent. In multiple-conclusion systems as e.g, G3cp (see Section 8.3), this proof obligation is not vacuously discharged.

The next theorem presents sufficient conditions for checking the invertibility of a rule r_s in a sequent system. The proof is similar to the one given for Theorem 1.

Theorem 2. *Let $\mathscr{S}$ be a sequent system, $\mathscr{H}$ be a set of rules, and r_s be an inference rule with $p \geq 1$ premises in $\mathscr{S}$. Let $l \in 1..p$. If the premise l of r_s is height-preserving invertible relative to each r_t in $\mathscr{S}$ under the assumption $\mathscr{H}$, then the premise l in r_s is height-preserving invertible in $\mathscr{S}$ under the assumption $\mathscr{H}$.*

Note that if r_s is a rule without premises (*i.e.*, an axiom), invertibility vacuously holds according to Definition 5. Hence, only the case for $p \geq 1$ premises is considered in the theorem above.

Most of the proofs of invertibility require Lemma 1 and, in some cases, admissibility of weakening. Hence, the assumption $\mathscr{H}$ must contain those theorems for each case. The dependencies among the different proofs will be stated in Section 8 for each of the systems under analysis.

6. Proving cut-admissibility and identity expansion

One of the most fundamental properties of a proof system is *analyticity*. Analytic calculi are calculi in which proof search proceeds by a stepwise decomposition of the formula to be proved, normally implying that it has the sub-formula property [23]. In sequent calculus, the sub-formula property is often proved by showing that the *cut-rule* is admissible. Roughly, the cut-rule introduces an auxiliary lemma A in the reasoning "if A follows from C and B follows from A, then B follows from C". The admissibility of the cut rule states that adding intermediary lemmas does not change the set of theorems of the logical system. That is, the lemma A is not needed in the proofs of the system. This implies that, if B is provable from the hypothesis C, then there exists a direct (cut-free) proof of this fact.

The cut-rule may take different forms depending on the logical system at hand. For concreteness, consider the following cut-rule for the system G3ip:

$$\frac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash B}\ Cut \tag{8}$$

This rule has an *additive* flavor: the context Γ is shared by the premises. Later, different cut-rules will be considered, including multiplicative-like cut-rules (splitting the context Γ in the premises) and also cut-rules for one-sided sequent systems.

Gentzen's style proof of admissibility of the cut-rule [1] generally proceeds by proving that top-most applications of cut can be eliminated. This is usually done via a nested induction on the *complexity* of the cut-lemma A and sub-induction on the *cut-height*, *i.e.*, the sum of the heights of the derivations of the premises. In the following, the rationale behind this cut-elimination procedure will be formalized by means of rewrite-based conditions. The next section discusses how these conditions become a semi-automatic procedure for checking the cut-admissibility property for different logical systems.

The analyses in Section 5 showed how to inductively reason on the height of derivations inside the rewriting logic framework. But what about the induction on the complexity of formulas? In general, it is possible to inductively reason about terms built from algebraic specifications. The unification of a sequent against the conclusion of an inference rule $r_s \in \mathscr{S}$ uniquely determines the active formula F introduced by r_s. Since terms in the sort `Formula` are inductively generated from the constructors of that sort, special attention can be given to the sub-terms (if any) of F since those are the *only* candidates that are useful to build the needed inductive hypothesis in the cut-admissibility proof. Such sub-terms are called *auxiliary formulas*.

Gentzen's procedure consists of reducing topmost cut-applications to the atomic case, then showing that these cuts can be eliminated. The reduction step to be performed depends on the status of the cut-formula in the premises: *principal* on both or *non-principal* on at least one premise. This procedure is formalized next.

Principal cases. If the cut-formula is principal (see Definition 2) in both premises of the cut-rule, then induction on the *complexity* of the cut-formula is applied. For instance, consider the case when the cut-formula is $A \wedge B$:

$$\dfrac{\dfrac{\Gamma \vdash_{\overline{n}} A \quad \Gamma \vdash_{\overline{n}} B}{\Gamma \vdash_{\overline{suc(n)}} A \wedge B}\ \wedge_R \quad \dfrac{\Gamma, A, B \vdash_{\overline{m}} C}{\Gamma, A \wedge B \vdash_{\overline{suc(m)}} C}\ \wedge_L}{\Gamma \vdash C}\ Cut \quad \leadsto \quad \dfrac{\Gamma \vdash A \quad \dfrac{\dfrac{\Gamma \vdash B}{\Gamma, A \vdash B}\ W \quad \Gamma, A, B \vdash C}{\Gamma, A \vdash C}\ Cut}{\Gamma \vdash C}\ Cut$$

Both applications of the cut-rule in the right-hand derivation are on smaller formulas and then induction applies. Note that this kind of reduction does not necessarily preserve the height of the derivation. Hence, no height-annotation can be used on the resulting premises. Note also that weakening is needed in order to match the leaves of the left derivation.

If the cut-formula $A \wedge B$ is "frozen" (making A, B constants of type `Formula`), then the inductive reasoning is formalized by generating the following rules:

```
rl [cutF] : Gamma |-- C => (Gamma |-- cA) | (Gamma, cA |-- C)
rl [cutF] : Gamma |-- C => (Gamma |-- cB) | (Gamma, cB |-- C)
```

Here cA, cB are constants representing the sub-terms of the ground term $\overline{A \wedge B}$. The other terms are variables. More generally, if the cut formula is a term of the form $f(t_1, ..., t_n)$, each t_i of sort `Formula` gives rises to a different rule. In the case of constants (*e.g.*, `False`) and atomic propositions (no sub-terms of sort `Formula`), the set of generated rules is empty.

Non-principal cases. The non-principal cases require permuting up the application of the cut-rule w.r.t. the application of an inference rule, thus reducing the *cut-height*. As an example, assume that the left premise of the cut-rule is the conclusion of an application of the rule $\wedge_L$. Hence, it must be the case that the antecedent of the sequent contains a conjunct $F \wedge G$. The reduction is as follows:

$$\dfrac{\dfrac{\Gamma, F, G \vdash_{\overline{n}} A}{\Gamma, F \wedge G \vdash_{\overline{suc(n)}} A}\ \wedge_L \quad \Gamma, F \wedge G, A \vdash_{\overline{suc(m)}} B}{\Gamma, F \wedge G \vdash B}\ Cut \quad \leadsto \quad \dfrac{\Gamma, F, G \vdash_{\overline{n}} A \quad \Gamma, F, G, A \vdash_{\overline{suc(m)}} B}{\dfrac{\Gamma, F, G \vdash B}{\Gamma, F \wedge G \vdash B}\ \wedge_L}\ Cut$$

Permuting up cuts results in an application of the cut-rule on shorter derivations. The top-rightmost sequent in the right-hand side derivation is deduced via the height-preserving invertibility of the $\wedge_L$ rule and the fact that $\Gamma, F \wedge G, A \vdash_{\overline{suc(m)}} B$ is provable. Similar reductions are possible when the cut formula A is not principal in the right premise of the cut-rule.

If the height of the premises in the above derivation is "frozen", it is possible to extend the theory $\mathscr{R}_{\mathscr{S}}$ with two new rules that exactly mimic the behavior of replacing a non-principal cut with a smaller one. Those rules are:

```
rl[CutH]:Gamma |-- G  => (cn : Gamma |-- cA) | (suc(cm) : Gamma, cA |-- G)
rl[CutH]:Gamma |-- G  => (suc(cn) : Gamma |-- cA) | (cm : Gamma, cA |-- G)
```

where cn (resp., cm) is the "frozen" constant resulting from n (resp., m) and cA the ground term representing the cut-formula. The first rule reflects the principle behind the above reduction where the height of the left premise of the cut-rule is reduced. The second rule reflects the case where the cut-formula is not principal in the right premise. Note that the first rule cannot be applied directly on the sequent $\Gamma, F \wedge G \vdash B$: the left premise $\Gamma, F \wedge G \vdash A$ is provable with height $suc(n)$ but, not necessarily, with height n. Similarly for the second rule and the right premise of the cut-rule.

6.1. Cut admissibility

The scenario is ready for establishing the necessary conditions for the admissibility of cut. The specification of the additive cut-rule for the system G3ip can be written as

```
rl [cut] : Gamma |-- C => (Gamma |-- A) | (Gamma, A |-- C) [nonexec] .
```

This rewrite rule has an extra variable (A) in the right-hand side and cannot be used for execution (unless a strategy is provided). Hence, the attribute `nonexec` identifies this rule as non-executable. In the following, `lcut` (resp., `rcut`) is used to denote the term `Gamma |-- A` (resp., `Gamma, A |-- C`) whose set of variables is {`Gamma, A`} (resp., {`Gamma, A, C`}). Moreover, `hcut` denotes the head/conclusion of the cut-rule, *i.e.*, the term `Gamma |-- C`.

As already illustrated, admissibility results and invertibility lemmas are usually needed in order to complete the proof of cut-admissibility. Such auxiliary results are specified in the analysis, respectively, as $\mathscr{H}$ and $\mathscr{I}$. As explained in Sections 5.1 and 5.2, a rule in $\mathscr{H}$ encodes an admissible rule of the system that can be used, in a backward fashion, during the proof-search procedure. Moreover, a rule in $\mathscr{I}$ is used to close the assumptions under the invertibility results.

Given a logical system $\mathscr{S}$, every possible derivation of the premises of the cut-rule should be considered. This means that there is a proof obligation for each $r_s, r_t \in \mathscr{S}$ s.t. r_s (resp., r_t) is applied on the left (resp., right) premise (*i.e.*, when r_s matches `lcut` and r_t marches `rcut`). More precisely:

Definition 8 *(Local cut-admissibility).* Let $\mathscr{S}$ be a sequent system, and $\mathscr{H}, \mathscr{I}$ be set of rules. Let

$$\frac{n:S_1 \;\cdots\; n:S_m}{suc(n):S}\; r_s \qquad \frac{k:T_1 \;\cdots\; k:T_n}{suc(k):T}\; r_t$$

be inference rules in $\mathscr{S}$. Under the assumptions $\mathscr{H}$ and $\mathscr{I}$, the cut rule is admissible relative to r_s and r_t iff for each $\theta \in CSU(S, \texttt{lcut})$ and $\gamma \in CSU(T, \texttt{rcut}\theta)$:

$$[\![\{\overline{(n:S_j)\gamma}, \overline{S_j\gamma} \mid j \in 1..m\} \cup \{\overline{(n:T_j)\gamma}, \overline{T_j\gamma} \mid j \in 1..n\}]\!]_{\mathscr{H} \cup \mathscr{R}_{\mathscr{S}} \cup \textit{ind-F} \cup \textit{ind-H} \cup \mathscr{I}} \Vdash \overline{\texttt{hcut}\gamma}$$

where the variables in S and T are assumed disjoint and

$$\begin{aligned} \textit{ind-F} &= \left\{(\texttt{hcut} \to \texttt{lcut} \mid \texttt{rcut})[t/A] \mid t \prec \overline{A\gamma} \text{ and } t \text{ has sort } \texttt{Formula}\right\} \\ \textit{ind-H} &= \{\texttt{hcut} \to (\overline{n\gamma} : \texttt{lcut} \mid \overline{suc(n)\gamma} : \texttt{rcut})[\overline{A\gamma}/A]\} \cup \\ &\quad\; \{\texttt{hcut} \to (\overline{suc(n)\gamma} : \texttt{lcut} \mid \overline{n\gamma} : \texttt{rcut})[\overline{A\gamma}/A]\} \end{aligned}$$

The rules in $\mathscr{S}$ are extended with axioms corresponding to the sequents resulting after the application of r_s and r_t on the premises of the cut-rule. Note that those premises are added with and without the height annotations, which can be used in applications of inductive hypothesis with shorter derivations and (non-height preserving) applications of the cut-rule over simpler formulas, respectively. Note also that the set of axioms is closed under applications of the rules in $\mathscr{I}$.

The set of rules in *ind-F* specifies a valid application of the cut-rule with sub-terms of the cut-formula ($t \prec \overline{A\gamma}$). As usual, this rule is assumed to be universally quantified on the remaining variables (t is a ground term). On the other hand, the set of rules in *ind-H* specifies a valid application of the cut-rule with shorter derivations. As discussed previously, two cases need to be considered: when the left and right premises are shorter. In both cases, the height of the derivation is fixed ($\overline{n\gamma}$) as well as the cut-formula ($[\overline{A\gamma}/A]$).

Regarding the base-cases of the induction, if the cut-formula is a constant or an atomic proposition, the set *ind-F* is empty. If the rule r_s is an axiom, then the set $\left\{\overline{n:S_j\gamma}, \overline{S_j\gamma} \mid j \in 1..m\right\}$ is also empty. In this case, an attempt of proving $\overline{\texttt{hcut}\gamma}$ by starting with the first rule in *ind-H* leads to a goal of the form $\overline{n\gamma} : \texttt{lcut}$ where no inference rule $r \in \mathscr{S}$ can be applied: $\overline{n\gamma}$ is a constant of the form %1 and it does not unify with the annotation $suc(n)$ in the conclusion of r. Hence, if r_s is an axiom, a proof of $\overline{\texttt{hcut}\gamma}$ cannot start with *ind-H*. A similar analysis applies for r_t.

Proving the admissibility of cut needs to consider all the possible matchings of the rules of the system and the premises of the cut-rule.

Theorem 3. *Let $\mathscr{S}$ be a sequent system, and $\mathscr{H}, \mathscr{I}$ be sets of rules. If for each r_s and $r_t \in \mathscr{S}$ the cut-rule is admissible relative to r_s and r_t under the assumptions $\mathscr{H}$ and $\mathscr{I}$, then the cut-rule is admissible in $\mathscr{S}$ (relative to $\mathscr{H}$ and $\mathscr{I}$).*

Proof. Consider the annotated cut-rule below (a similar analysis applies for the other cut-rules introduced in Section 8):

$$\frac{\Gamma \vdash_{suc(n)} A \quad \Gamma, A \vdash_{suc(m)} B}{\Gamma \vdash B}\ Cut$$

Assume that there exists a derivation of the premises starting, respectively with the rules r_s and r_t. By the hypothesis and Definition 8, there is a valid derivation of the sequent in the conclusion and the result follows. □

6.2. Identity expansion

The identity axiom states that any atomic formula is a consequence of itself. It has a dual flavor w.r.t. the cut-rule, in the sense that, while in the cut-rule a formula is eliminated, in the identity axiom an atomic formula is introduced. In G3ip, I represents the general schema for the identity axiom (see Fig. 1), where p is an atomic formula.

In Section 6.1, the cut-rule was proved admissible relative to some assumptions. The corresponding dual property w.r.t. the identity axiom is the *identity expansion*: assuming I, is *any* well formed formula a consequence of itself? Or equivalently: Is the general identity axiom – not restricted to atoms – admissible? For example, in G3ip, proving identity expansion requires proving the admissibility of the axiom I_A for an arbitrary formula A:

$$\frac{}{A \vdash A}\ I_A$$

Observe that the cut/identity duality is also reflected during proof-search: while applications of the cut-rule should be avoided because arbitrary formulas need to be produced "out of thin air", applications of general identity axioms are most welcome, since they may make the proof smaller and proof-search simpler.

The proof of identity expansion proceeds by induction on the formula A. Consider a constructor $C = f(\vec{t})$ of type $S_1, ..., S_n \to$ `Formula` and let $\overline{C}$ be the ground term $f(\vec{\overline{t}})$ where each $t_i \in \vec{t}$ is replaced by a fresh constant $\overline{t_i}$ of sort S_i. Hence, the goal is to prove $\overline{C} \vdash \overline{C}$ assuming that $\overline{t_i} \vdash \overline{t_i}$ is provable for each $t_i \in \vec{t}$ of sort `Formula`.

Theorem 4. *Let $\mathscr{S}$ be a propositional sequent system and $\mathscr{R}_{\mathscr{S}}$, with signature $(S, \leq, F)$, be the resulting rewrite theory encoding $\mathscr{S}$. Identity expansion holds in $\mathscr{S}$ iff for each symbol $f \in F$ of type $S_1, ..., S_n \to$ `Formula` $(n \geq 0)$,*

$$\mathscr{R}_{\mathscr{S}} \cup \{t' \vdash t' \mid t' \prec \overline{f(\vec{t})} \text{ and } t' \text{ has sort } \texttt{Formula}\} \Vdash (\overline{f(\vec{t})} \vdash f(\vec{t}))$$

In one-sided systems, the goal is to show that $\vdash A, A^{\perp}$ is provable for any A, where $A^{\perp}$ denotes the *dual* of A (see Section 8.4 for the definition of duality). Theorem 4 can be easily adapted to the one-sided case.

7. Reflective implementation

This section details the design principles behind the `L-Framework` [4], a tool implementing the procedures described in Sections 5 and 6. The `L-Framework` receives as input the object logic sequent system (OL) to be analyzed, as a rewrite theory $\mathscr{R}_{\mathscr{S}}$, and the specification of the properties of interest. Then, it outputs LaTeX files with the results of the analyses of, *e.g.*, the proof reductions needed to establish cut-admissibility. The specification of the OL follows as in Section 4 (details can be found in Section 7.1). As explained in Section 7.2, the implementation of the algorithms heavily relies on the reflective capabilities of rewriting logic. Moreover, the specification of the properties for each kind of analysis follows a similar pattern where a suitable functional theory needs to be instantiated (Section 7.3). The subsequent sections offer further details about the implementation of each kind of analysis: admissibility of rules (Section 7.4), invertibility of rules (Section 7.5), cut-admissibility (Section 7.6), and identity expansion (Section 7.7).

For readers interested in the details of the implementation, pointers to the Maude files and definitions are given in this section. However, readers interested only in the results of the analyses can safely skip this section; several examples and instances of the definitions presented here can be found in Section 8. The file `docs/tutorial.md` contains detailed instructions on how to specify an object logic in the framework and how to configure and run the different analyses.

7.1. Sequent system specification

The starting point for a sequent system specification is the definition of its syntax and inference rules. The file `syntax.maude` contains the functional module (*i.e.*, an equational theory and no rewriting rules) `SYNTAX-BASE`. Such a module defines the sorts `Prop` and `Formula`, as well as the subsort relation `Prop < Formula`. No constructors for these sorts are given since those depend on the OL and hence must be provided. The sort `MSFormula`, for multiset of formulas, is pre-defined here with the constructors presented in Section 4: `op *` denoting the empty multiset and `op _;_` for multiset union. Some auxiliary functions, needed to produce LaTeX outputs, are declared in this module. In particular, the OL may define a mapping to replace symbols of the syntax with LaTeX macros, *e.g.*,

```
eq TEXReplacement = ('|-- |-> '\vdash), ('/\ |-> '\wedge) , ...
```

The `sequent.maude` file defines the functional module `SEQUENT` with the sort `Sequent` and the following constructors:

```
op  |--_ :           MSFormula -> Sequent . --- One sided
op _|--_ : MSFormula MSFormula -> Sequent . --- Two sided
```

This syntax should suffice for most of the sequent-based inference systems. As shown in Section 8, it is also possible to provide more constructors to deal, for instance, with dyadic systems (*i.e.*, one-sided sequents with two separated zones).

Internally, the tool annotates sequents with inductive measures. For this purpose, the constructor

```
op _:_ : INat Sequent -> Sequent .          --- Height annotations
```

is added to the specification. The sort `INat` (file `nat-inf.maude`) extends the natural numbers expressed in Peano-like notation as $s^n(z)$ with a constant `inf` denoting "unknown". This constant is used, *e.g.*, in the cut-admissibility procedure where structural cuts do not preserve/decrease the height of the derivation and, therefore, the resulting sequent does not have any (known) measure.

OLs are also allowed to define equations to complete the definition of the mapping `TEXReplacementSeq` that replaces the name of the rules with suitable LaTeX symbols:

```
eq TEXReplacementSeq = ('AndL |-> '\wedge_L), ('AndR |-> '\wedge_R) ...
```

The sort `Goal`, the subsort relation `Sequent < Goal`, the constructors `op _|_`, and `proved` (for building list of sequents) are also defined in the module `SEQUENT`.

The `sequent.maude` file also specifies the module `SEQUENT-SOLVING` with auxiliary procedures for building derivation trees and outputting LaTeX code. This module uses reflection heavily (more on this in the next section) in order to deal, in a general and uniform way, with the representation of any sequent regardless of its specific syntax. Moreover, Maude's module `LEXICAL` is used for converting between strings and lists of Maude's quoted identifiers (terms of the form `'identifier` with sort `Qid`). As shown below, Qids materialize the meta-representation of any term.

7.2. Reflection and the core system

A reflective logic is a logic in which important aspects of its meta-theory can be represented consistently at the object level. In a nutshell, a reflective logic is a logic that can be faithfully interpreted in itself. In this way, the object-level representation can correctly simulate the relevant deductive aspects of its meta-theory. Maude's language design and implementation make systematic use of the fact that rewriting logic is reflective, making the meta-theory of rewriting logic accessible to the user as a programming module [24].

For the purpose of this paper, the focus is on two meta-theoretic notions, namely, those of theory/module and the deductive entailment relation $\Vdash$ (see Sec. 3). Formally, there is a *universal* rewrite theory $\mathscr{U}$ in which any finitely represented rewrite theory $\mathscr{R}$ can be represented as a term $\widehat{\mathscr{R}}$, including $\mathscr{U}$ itself, any terms t, u in $\mathscr{R}$ as terms $\widehat{t}, \widehat{u}$, respectively, and a pair $(\mathscr{R}, t)$ as a term $\langle \widehat{\mathscr{R}}, \widehat{t} \rangle$ in such a way that the equivalence $\mathscr{R} \Vdash t \to u \quad \Leftrightarrow \quad \mathscr{U} \Vdash \langle \widehat{\mathscr{R}}, \widehat{t} \rangle \to \langle \widehat{\mathscr{R}}, \widehat{u} \rangle$ holds. Since $\mathscr{U}$ is representable in itself, a "reflective tower" can be achieved with an arbitrary number of levels of reflection.

In general, simulating a single step of rewriting at one level involves many rewriting steps one level up. Therefore, in naive implementations, each step up the reflective tower comes at considerable computational cost. In Maude, key functionalities of the universal theory $\mathscr{U}$ have been efficiently implemented in the functional module `META-LEVEL`, providing ways to perform reflective computations.

Additional utilities for manipulating modules and terms in the `L-Framework` are implemented in the `meta-level-ext.maude` file. For instance, all the operations on theories described in Section 5 are contained there (some of them are detailed below).

The module `META-LEVEL` implements the so called *descent* functions that manipulate (meta-) terms. The function `op upTerm : Universal -> Term .` returns the (meta) representation $\widehat{t}$ of a term t. For example, constants are represented as `Id.Type` (*e.g.*, `'proved.Goal`), variables as `Id:Type` (*e.g.*, `'F:Formula`), and functions as `Id[Params]` (*e.g.*, the term `'_;_[Gamma:MSFormula,Delta:MSFormula]` represents the multiset of formulas `Gamma ; Delta`).

From a well-formed (meta-) term $\widehat{t}$ it is possible to recover the term t (one level down). Note, however, that not all meta-terms have a suitable representation in the theory in the level below, for instance, `'_|_[S:Sequent, F:Formula]` is not the image of any valid sequent or formula in the module specifying the system G3ip. The function `op downTerm : Term Universal -> Universal` takes as parameter the (meta) representation of a term $\widehat{t}$ and a term t'. It returns the canonical form of t if it is a term having the same sort of t'; otherwise, it returns t'. Usually, t' is a term used to denote that the descent translation was not possible. For instance, if $\widehat{t}$ is expected to be the meta-representation of a formula, then t' can be the constant `op error : -> Formula .`

At the meta-level, modules in Maude (*i.e.*, rewrite theories) are represented as terms with sort `Module`. The function `upModule` can be used to obtain such a term. All the components of a module (sorts, functions, equations and rules) have a suitable sort and representation in the meta-level, thus making them first-class citizens. For instance, most of the analyses require to extend the sequent theory with new rules. The function below adds to the module `M` a set of rules `RS`:

```
op newModuleRls : Module RuleSet -> Module .
```

```
eq newModuleRls(M, RS)
  = (mod 'NEW-MOD is
      getImports(M)     --- Sorts, imports, equations, etc remain the same
      sorts getSorts(M) .
      getSubsorts(M) getOps(M) getMbs(M) getEqs(M)
      (getRls(M) RS)    --- Union of the rules of M and RS
     endm ) .
```

Rules at the meta-level are terms of sort `Rule` built from the constructors

```
op rl_=>_[_]. :     Term Term           AttrSet -> Rule .
op crl_=>_if_[_]. : Term Term Condition AttrSet -> Rule .
```

The parameter of sort `AttrSet` is a set of attributes of the rule, including, *e.g.*, the label used as identification (`[label('AndR)]`). This representation opens the possibility of manipulating rules in a simple way. For instance, consider a sequent rule $r = (S \to S')$, for some conclusion S and premise S', to be proved height-preserving admissible. Consider also that the current goal is to show that a given ground sequent is provable with height at most `suc(h)`. A call to the function below with parameters `h` and r produces a restricted version of r where it can be applied only on sequents annotated with (the constant height) `h`:

```
op inductive-rule : GroundTerm Rule -> Rule .
eq inductive-rule(gt, rl '_:_['n:FNat , S] => '_:_['n:FNat , S'] [ AS ].)
                  =  rl '_:_[gt ,      S] => '_:_[gt ,      S'] [ AS ].
```

The module `META-LEVEL` offers functions to perform Maude's operations at the meta-level. For instance, the function `metaRewrite` allows for rewriting the meta-representation of a term in a given module and `metaIrredundantDisjointUnify` to solve unification problems. Moreover, given a theory $\mathscr{R}$ and two terms t, u in that theory, `metaSearchPath` allows for checking the entailment $\mathscr{R} \Vdash t \to u$ by testing whether $\mathscr{U} \Vdash \langle \widehat{\mathscr{R}}, \widehat{t} \rangle \to \langle \widehat{\mathscr{R}}, \widehat{u} \rangle$. For instance, in the following assignment:

```
Ans := metaSearchPath(M, SGoal, upTerm(proved), nil, '*, bound-spec, 0) .
```

The term `M` is the meta-representation of a sequent system; `SGoal` is a term representing the goal/sequent to be proved; `nil` specifies that there are no additional conditions to be satisfied by the solution; `'*` means that the reduction may take 0 or more steps; `bound-spec` indicates the maximum depth of the search; and the final 0 is the solution number. The term `Ans` is a term of type `Trace?`. It can be `failure` or a list of trace steps showing how to perform each of the reductions $t \to_r t'$ where the rule r is applied to t leading to t'. It is worth noticing that `metaSearchPath` implements a breadth-first search strategy. Hence, if `SGoal` can be rewritten into `proved` with at most `bound-spec` steps, then `metaSearchPath` will eventually find such proof (if the executability conditions for the rewrite theory are met [8]). Moreover, since `Ans` is a proof term evidencing how to prove the rewriting `SGoal` $\overset{*}{\to}$ `proved`, it can be used to rebuild the needed derivation of `SGoal` in the sequent system at hand.

7.3. The general approach for implementing the algorithms

All the analyses implemented in `L-Framework` are instances of the same template:

1. A module interface (called *theory* in Maude) specifies the input for the analysis. This theory includes parameters such as: the name of the module implementing the OL; the specification, as a rewrite rule, of the theorem to be proved; the extra hypotheses (set of rewrite rules) corresponding to the already proved theorem; the bound for the search depth; etc.
2. A module implementing the decision procedures proposed in Sections 5 and 6. All algorithms follow the same principles:
   (a) A function `generate-cases` uses unification to generate all the proof obligations to be discharged. In each kind of analysis, there are suitable sorts and constructors to represent the proof obligations. Normally, the cases include the terms denoting the premises (that can be assumed to be provable) as well as the goal to be proved.
   (b) Auxiliary definitions to extend the theory with new axioms and the right inductive hypothesis. Take for instance the function `inductive-rule` explained above.
   (c) A function `holds?` that receives as parameter one of the proof obligations, uses the functions in (b) to extend the theory and calls to `metaSearchPath` to check if the goal can be proved.
3. An extra module providing facilities to produce the LaTeX output.

The following sections give some details about the components (1) and (2) for each kind of analysis. Common facilities for all the analyses are implemented in the `theorem-proving-base.maude` file, *e.g.*, generating axioms (`rl [ax] T => proved .`) from the assumptions of the theorem.

7.4. Admissibility analysis

The core procedures for automating the proof of admissibility theorems, following the definitions in Section 5.1, are specified in `admissibility.maude`. The input point is the definition of the *functional theory* below.

```
fth ADMISSIBILITY-SPEC is
 pr META-LEVEL .
 op th-name    : -> String .  --- Name of the theorem
 op mod-name   : -> Qid .     --- Module with the OL specification
 op file-name  : -> String .  --- Output file
 op rule-spec  : -> Rule .    --- Rule to be proved admissible (goal => premise)
 op bound-spec : -> Nat .     --- Depth-search
 --- Already proved admissibility lemmas (set of rules)
 op already-proved-theorems : -> RuleSet .
 --- Identifiers of the height preserving invertible rules
 op inv-rules  : -> QidList .
 --- Mutual inductive rule (for mutual inductive proofs).
 op mutual-ind : GroundTerm -> RuleSet .
endfth
```

Theories in Maude are module interfaces used for declaring parameterized modules. Such interfaces define the syntactic and semantic properties to be satisfied by the actual parameter modules used in an instantiation [24].

The name of the theorem is specified with the string `th-name` (*e.g.*, "Admissibility of weakening."). The identifier of the module `mod-name` (*e.g.*, `'G3ip`) is used to obtain the meta-representation of the OL module defining the syntax and inference rules of the sequent system. The field `file-name` specifies the output (LaTeX) file.

The theorem to be proved is specified via a rewriting rule (a term of sort `Rule`). As an example, the height-preserving admissibility of weakening for G3ip is:

```
eq rule-spec =
 rl '_:_['n:FNat, '_|--_['_;_['Gamma:MSFormula,'F:Formula],'G:Formula]] =>
    '_:_['n:FNat, '_|--_[     'Gamma:MSFormula            ,'G:Formula]]
    [ label('W) ]. ) .
```

This term is the meta-representation of the rule

```
rl [W] n : Gamma, F |-- G => n : Gamma |-- G .
```

Note that the premise and the conclusion have the same height, specifying that if the premise is provable with height at most n so is the conclusion. Since the entailment relation is undecidable in general, all the analyses are performed up to a given search depth (field `bound-spec`). Hence the procedures are sound (in the sense of the theorems in Sections 5 and 6), but not complete.

As already explained, for modularity, the analyses can depend on external lemmas. Those auxiliary results are specified as `already-proved-theorem` (backward reasoning) and `inv-rules` (forward reasoning). Finally, some theorems require mutual induction. For instance, admissibility of contraction for the system of classical logic (Section 8.3) requires a mutual induction on the right and left rule for contraction. The field `mutual-ind` specifies the other theorems that can be applied on shorter derivations. For that, if the parameter of type `GroundTerm` is of the form `'s[h.Nat]`, the mutual-theorem is instantiated with sequents of (constant) height `h`.

The main procedures implementing the analysis of admissibility of rules can be found in the parametric module `ADMISSIBILITY-ALG{SPEC :: ADMISSIBILITY-SPEC}`. Consider, for instance, the task of proving the admissibility of a rule `rs`. Such a rule is specified as the parameter `rule-spec` in `SPEC`. For each rule `rt` of the system, proof obligations are generated by the module's functionality. Following Definition 6, this is done by unifying the premise/body of `rs` with the conclusion/head of `rt`:

```
U :=  metaIrredundantDisjointUnify(module,
         getBody(rule-spec, module) =? getHead(rt, module), '%, N) .
```

Here, `N` is a natural number used to enumerate the unifiers. By identifying the unifier `U`, it is possible to obtain the resulting premises when `rt` is applied on the body/premise of `rs` (see Equation (2)). For that, the descent function `metaXapply` enables the application of a rule to a term according to a given substitution. Computing the ground term that substitutes the variables by fresh constants is a routine exercise by following the inductive definition of the sort `Term`.

The cases for admissibility take the form `adm-case(Q, M, GTC, GTP, GG)`, where: `Q` is the identifier of the rule `rt` in Definition 6; `M` is the module implementing the OL, `GTC` and `GTP` correspond to the conclusion and the premises in Equation (2), and `GG` is the goal to be proved ($\overline{(i : S)\theta}$ in Definition 6).

A useful mnemonic that applies from now on: `G` is for *ground*, `T` for *term*, `C` for *conclusion*, `P` for *premise(s)*, and the last `G` in `GG` for *goal*.

The module `M` above is extended as follows:

```
M' := newModuleRls(M,
  inductive(getHeight(GG))          --- Theorem instantiated on shorter der.
  mutual-ind(getHeight(GG))
  premises(GTP)                     --- All premises in GTP as axioms
  inv-premises(M, inv-rules, GTP)   --- Closing w.r.t invertible rules
  already-proved-theorems ) .
```

The function `inductive` takes as parameter the annotated height of the current goal `GG`. If it is of the form `suc(h)`, `rule-spec` is instantiated with the height `h` (as explained in Section 7.2). Otherwise, `inductive` and `mutual-ind` return `none` (an empty list of rules). The function call `premises(GTP)` generates, for each sequent `S` in the set of premises `GTP`, a new axiom rule (`S => proved`); `inv-premises` generates further axioms by applying the invertible rules `inv-rules` in the premises `GTP`; and `already-proved-theorems` adds already proved theorems (specified as rules).

From the extended theory `M'`, `metaSearchPath` is used to check whether `GG` can reach `proved` and determine the status of the current proof obligation.

7.5. Invertibility of rules

The core procedures are in `invertibilty.maude`. The input to the invertibility analysis is specified as a realization of the functional theory `INV-SPEC`:

```
fth INV-SPEC is
    pr META-LEVEL .
    op th-name : -> String .     --- Name of the theorem
    op mod-name : -> Qid .       --- Module with the OL specification
    op bound-spec : -> Nat .     --- Bound of the search procedure
    op file-name : -> String .   --- File name to write the output
    op already-proved-theorems : -> RuleSet . --- Admissible rules
endfth
```

There is no need to specify the theorem to be proved, because it suffices to take each of the rules of the input module and "flip" it in order to obtain the invertibility lemma to be proved (see Definition 5). This procedure outputs a LaTeX file with the invertibility status of each rule in the system. In the case of rules with two premises, each premise is analyzed separately (see Definition 7). This allows for proving, *e.g.*, that the rule $\supset_L$ in the system G3ip is invertible only in its right premise (see Section 8.1).

Following the same recipe for admissibility, the proof obligations for invertibility are generated by solving unification problems. Assume that the rule being analyzed is `Q` and it is to be tested invertible with respect to the rule `Q'`. If `Q` is a two-premise rule, then it is split into two different rules to be analyzed separately. Assume that the resulting sliced rule with at most one premise is `R`. The `N`-th unifier between the heads/conclusions of the rules is

```
T  := getHead(R)
T' := getHead(Q', module)
U  := metaIrredundantDisjointUnify(module, getHead(T) =? getHead(T'), '%, N)
```

The case to be analyzed is then:

```
inv-case(R, Q',
  apply(T', U), --- The sequent where R and Q' can be applied
  --- Applying Q' and R on the resulting sequent
  applyRule(apply(T, U), Q', module, U),
  applyRule(apply(T, U), R,  module, U))
```

The first two parameters identify the case. The third parameter corresponds to the sequent where both rules can be applied, *i.e.*, the conclusion in Equation (6). The fourth parameter corresponds to the premises after the application of `Q'`, *i.e.*, the premises in Equation (6). The last parameter is the goal being proved, *i.e.*, the premise resulting after the application of `R` on the same sequent ($(\overline{k : S_l})\theta$ in Definition 7). Call `GTC`, `GTP`, and `GG` to the last three parameters of the case once their variables are replaced with fresh constants.

The inductive reasoning consists in applying, when possible, the rule `R` on the sequents in `GTP`. By induction, such application of `R` on a sequent S in `GTP` must preserve the height annotation of S. Hence, a modified version of `R` is needed:

```
op inductive-rule : Rule -> Rule .
eq inductive-rule( ( rl '_:_['s[T],T'] => T'' [ AS ]. ) )
       =           ( rl '_:_[   T, T'] => T'' [ label('IND) ]. ) .
```

Note the use of the height `T` (instead of `s[T]`) in the resulting rule. Call `RI` the rule resulting from the application of this function to the rule `R` being analyzed.

Computing the set of sequents that can be assumed as axioms by induction becomes now a simple task. It suffices to compute one step of rewriting for each sequent `GT` in the set `GTP` as follows:

```
metaSearch(MRI, GT, 'G:Goal, nil, '+, 1, k) .
```

The term `MRI` inherits all the functional description of the module specifying the OL, but it contains only one single rule, namely, `RI`. The (ground) sequent `GT` is rewritten to any possible list of sequents (variable `'G:Goal`) in this theory in exactly one step (`'+` means one or more steps and the bound in the 6th parameter forces it to be exactly one). The last parameter is used to enumerate all the possible solutions.

Finally, the OL theory `M` can be extended before attempting a proof of the goal `GG` of the current case:

```
M' := newModuleRls(M,  premises(GTP)         --- Axioms from GTP
                       inductive(M, GTP, R) --- Inductive reasoning
                       already-proved-theorems)
```

7.6. Admissibility of cut

There are different cut-rules depending on the shape of the sequent system at hand (*e.g.*, one-sided, two-sided, dyadic) and also on the structural rules allowed. The file `cut-elimination-base.maude` defines common facilities for all the cut-admissibility procedures. Impressive enough, minor extensions to this module have been required to prove cut-admissibility theorems for the systems in Section 8.

The common interface is the following functional theory:

```
fth CUT-SPEC is
    pr META-LEVEL .
    op th-name      : -> String .  --- Name of the theorem
    op mod-name     : -> Qid .     --- Module with the OL specification
    op bound-spec   : -> Nat .     --- Bound of the search procedure
    op file-name    : -> String .  --- File name to write the output
    op already-proved-theorems : -> RuleSet . --- Admissible rules
    op inv-rules    : -> QidList .            --- Invertible rules
endfth
```

The parametric module `CUT-BASE{SPEC :: CUT-SPEC}` contains the common definitions. An operator for the cut rule is defined `op cut-rule : -> Rule .` , but no equation for it is provided. Each OL is responsible for completing this definition. For instance, in G3ip, the cut-rule shares the context in the antecedent of the sequent between the two premises. Hence, the file `cut-add-scon.maude` (additive cut for single conclusion systems) extends `CUT-BASE` with the following equation:

```
eq cut-rule =
 (rl '_:_['inf.INat, '_|--_['Gamma:MSFormula, 'F:Formula]] =>
 '_|_[
   '_:_['h1$$:FNat,'_|--_['_;_['Gamma:MSFor., 'FCut$$:Formula],'F:For.]],
   '_:_['h2$$:FNat,'_|--_[     'Gamma:MSFor.             ,'FCut$$:For.]]]
  [ label('\Cut) ]. ) .
```

The conclusion of the rule is annotated with `inf` (the constant of type `INat` denoting "don't know"). This specification is nothing else that the meta-representation of the rule in Equation (8). In all the analyses, the cut-formula is expected to be named as `FCut$$` and the height of the two premises as `h1$$` and `h2$$`.

The module `CUT-BASE` offers mechanisms to generate, in a uniform way, the proof obligations for the cut-admissibility procedures considered here. This is done in two steps. First, one of the rules of the system, identified as `Q1`, is matched against the first premise of the cut-rule (`lcut` below):

```
U :=  metaIrredundantDisjointUnify(module,
                                   lcut =? getHead(Q1, module), '%, N)
```

The unifier `U` must map the variables of `lcut` to some fresh variables. Hence, before unifying a second rule `Q2` against the second premise of the cut-rule, denoted as `rcut`, the substitution `U` must be applied on `rcut` (see Definition 8):

```
U' :=  metaIrredundantDisjointUnify(module,
               apply(rcut, U) =? getHead(Q2, module), '%, M) .
```

A proof obligation takes the following form:

```
cut-case(
        cut-sub-case(TC, TP),    --- Left premise  (conc. and premises)
        cut-sub-case(TC', TP'), --- Right premise (conc. and premises)
```

```
        apply(getHead(cut-rule), (U ; U')),  --- Goal to be proved
        apply('FCut$$:Formula, (U ; U'))     --- The cut formula
  )
```

The notation `U ; U'` is used for composition of substitutions.

Induction on the height of the derivation is also defined in `CUT-BASE` in a general way and independently of the cut-rule.

```
op induct-height : GroundTerm GroundTerm GroundTerm -> RuleSet .
eq induct-height('suc[gh], 'suc[gh'], GF) =
   ( rl getHead(cut-rule) =>
     '_|_[  --- Fixing heights (gh and suc[gh']) and the cut-formula (GF)
       apply(lcut, ('h1$$:FNat <- gh ; 'FCut$$:Formula <- GF)),
       apply(rcut, ('h2$$:FNat <- 'suc[gh'] ; 'FCut$$:Formula <- GF))]
     [ label('\hCut) ]. )
   ( rl getHead(cut-rule) =>
     '_|_[  --- Fixing heights (suc[gh] and gh') and the cut-formula (GF)
       apply(lcut, ('h1$$:FNat <- 'suc[gh] ; 'FCut$$:Formula <- GF)),
       apply(rcut, ('h2$$:FNat <- gh' ; 'FCut$$:Formula <- GF))]
    [ label('\hCut) ]. ) .
```

The syntax `x <- t` is used to denote the substitution $[t/x]$. This specification implements *ind-H* in Definition 8. The first two parameters are the height of the left and the right premises above the cut. The last parameter is the cut-formula considered in the current goal.

Induction on the structure of the formula is partially defined in `CUT-BASE`, but additional work is needed in the specific OL. General definitions include, *e.g.*,

```
op induct-struct : GroundTerm -> RuleSet .
eq induct-struct(Q[LGT]) = $induct-struct(LGT) .
eq induct-struct(GT)     = none [owise] .
```

The first equation applies to constructors for formulas (*e.g.*, `'_/\_[A, B]`, representing $A \wedge B$) and then, induction applies on all the sub-terms in the list of ground terms `LGT` (if they are of sort `Formula`). If the parameter is a constant, *e.g.*, `'False.Formula`, only the second equation applies and no inductive rule is generated. The keyword `owise` (a shorthand for `otherwise`) means "use this equation if all the others defining the corresponding function symbol fail". This is syntactic sugar to simplify the specification and it can be encoded in plain conditional equational theories [8].

The function `$induct-struct` calls to `induct-struct-formula` for each term in `LGT`. The definition of `induct-struct-formula` is specific for each OL. Here the one defined in `cut-add-scon.maude`:

```
op induct-struct-formula : GroundTerm -> RuleSet .
eq induct-struct-formula(GTA)
  =  if getType(GTA) == 'Formula      --- Only sub-terms of sort Formula
     then ( rl  getHead(cut-rule) =>  --- Head unchanged
        '_|_[  --- premises with height 'inf and the cut-f. fixed to GTA
          apply(lcut, ('h1$$:FNat <- 'inf.INat ; 'FCut$$:Formula <- GTA)),
          apply(rcut, ('h2$$:FNat <- 'inf.INat ; 'FCut$$:Formula <- GTA)) ]
        [ label('\sCut) ]. )
      else none --- No rule if GTA is not a formula
      fi .
```

The parameter `GTA` is a ground term denoting a proper sub-formula of the cut-formula. The cut-rule is instantiated as follows: the head of the rule is unchanged, the height of the premises is `'inf.INat`, and the cut-formula is fixed to be (the ground term) `GTA`. The change in the height of the premises is due to the fact that cuts on smaller formulas do not preserve necessarily the height of the derivation.

The above definition seems to be general enough for any cut-rule and, in theory, it would be possible to define it once and for all the systems. However, the generated rule is problematic from the point of view of proof search. Note that both premises are annotated with `'inf.INat`. Hence, even if the rule does not need to "guess" the cut-formula (since it is fixed to a ground term `A`), it is always possible to rewrite the goal $\Gamma \vdash G$ into $\Gamma, A \vdash G$ and later into $\Gamma, A, A \vdash G$, etc. For this reason, in some systems, extra conditions are needed to restrict the application of this rule and reduce the search space (more about this in Section 8). Of course, a bad choice of such conditions may render the analysis inconclusive. A natural rule of thumb that worked in most of the cases was to restrict the application of this rule when the sub-formula A is not already in the context Γ.

Finally, the main search procedure must also be tailored for each OL. There is a template that, with little modifications, can be used in all systems reported here. In particular, OLs may define different conditions for the application of the rules with the aim of reducing the search space. The main definition in `cut-add-scon.maude` is:

```
ceq holds$?(cut-case( ... ))
 = output ...
```

```
 if      --- Premises with implicit weakening
  RS := premises-W(GTC) premises-W(GTC') premises-W(GTP) premises-W(GTP')
        --- Invertibility lemmas on the premises
        inv-premises(M, inv-rules, GTP) inv-premises(M, inv-rules, GTP')
        already-proved-theorems --- adding  admissibility  lemmas
        --- Induction on the height of the derivation
        induct-height(getHeight(gtc), getHeight(gtc'), GF)
        --- Induction on the formula only in principal cases
        if numFormulas(GG) <= 1 then induct-struct(GF) else none fi
```

In the code above, `RS` is the set of rules used to extend the theory of the OL before calling the search procedure. This extension is similar to the ones presented for admissibility and invertibility. There are, however, two new ingredients: (a) the way the axioms are generated and (b) the additional condition deciding whether the rule for structural induction is added or not to solve the current goal `GG`.

(a) `premises-W(S)` converts into an axiom the sequent `S`. Unlike `premises(S)` used in the previous sections, the resulting rule in `premises-W(S)` internalizes weakening: if S is the (ground) sequent $\Gamma \vdash F$, then $\Delta \vdash F$ is provable for any $\Delta \supseteq \Gamma$. This avoids the need for adding the rule W into `already-proved-theorems`, thus reducing the search space. This simplification cannot be used, of course, in substructural logics, such as linear logic (Section 8.4).

(b) As already explained, the rule for structural induction is problematic from the point of view of proof search. In some OLs it is possible to control its use. For example, in G3ip, the application of *ind-F* can be restricted to solve only the principal cases (and the non-principal cases will run faster). Such cases can be identified by counting the number of constants of type `Formula` in the current goal `GG` (see the last line in the code above). In other systems, however, this simplification does not work as in the case of the system mLJ (Section 8.2) where structural induction is also needed in some of the non-principal cases.

Wrapping up, configuring the cut-elimination procedure requires adjusting and tuning some parameters. As shown in Section 8, some logics that share the same structural properties may reuse a common infrastructure. For instance, the cut-rule and procedures defined in file `cut-add-mcon.maude` (two-sided and multi-conclusion systems where weakening and contraction are allowed) can be used to prove cut-admissibility for the system G3cp (Sec. 8.3), mLJ (Sec. 8.2), and some systems for modal logics (Sec. 8.5). However, cut-admissibility is a non-trivial property and hence, full automation is impossible for 'the' general case. In each of these systems, the user must determine the invertibility lemmas that will be considered during the search procedure. This is done by simply modifying the input parameter `inv-rules` in `CUT-SPEC`.

7.7. Identity expansion

This analysis uses the following functional theory as interface (`id-expand.maude`):

```
fth ID-EXP-SPEC is
 op mod-name : -> Qid .
 op file-name : -> String .
 op bound-spec : -> Nat .
 op goal : GroundTerm -> GroundTerm . --- E.g., F |-- F  or  |- F, dual(F)
 op already-proved-theorems : -> RuleSet .
 --- Types different from 'Formula to be analyzed
 op types-formula : -> TypeList .
endfth
```

Given a ground term F denoting a formula, the call `goal(F)` returns the sequent to be proved. This definition allows to consider id-expansion theorems for one-sided and two-sided systems. The last field is used to define other sorts that need to be considered in the analysis. For instance, the specification of modal logics includes the sort `BFormula < Formula` for boxed formulas (see Section 8.5). By adding `BFormula` in `types-formula`, the case $\Box F \vdash \Box F$ is generated.

8. Case studies

This section explains how structural meta-properties of several propositional sequent systems can be specified and proved with the approach presented in this paper. The site hosting the `L-Framework` includes all the logical systems described here, the proof-search implementation of the strategies, and the PDFs generated by the proof-search algorithms. The chosen methodology for proving the (meta-)theorems in this section is modular: first, it attempts to build a proof without any external lemma; second, when needed, it analyzes the failing cases and adds already proved theorems for completing the proof. This methodology allows for analyzing interdependencies between the different results inside various logical systems.

Since there are no similar tools to compare the performance of the `L-Framework`, the main benchmark pursued is to show that it is flexible enough to deal with several proof-theoretic properties with little effort in defining the object logic and the properties of interest. In all the cases, but those reported in Section 8.4 (where the proof of cut-admissibility requires induction on two different rules), implementing the analyses amounts only to instantiate the interfaces/theories described in Section 7.

In order to show the feasibility of the `L-Framework`, time-related observations are reported. In all the cases, the depth bound (`bound-spec`) is set to 15 and the experiments are performed on a MacBook Pro, 4 cores 2.3 GHz, and 8 GB of RAM, running Maude 3.2.1. On average, the admissibility analyses are completed in about 5 seconds. Once all the auxiliary lemmas are added, the time needed to complete the proof of cut-admissibility depends on whether explicit structural rules are used (about 30 seconds) or not (about 15 seconds).

The results of this section, together with the outputs of the tool, are a valuable companion material to classical books such as [11,12], covering all the (rutinary) cases and bringing into light the most interesting ones that require auxiliary lemmas. In each of the following subsections, a different logical system is introduced and its properties analyzed at length. The proof of each theorem explicitly states the interdependencies between the different meta-theoretical properties. Moreover, due to the flexibility of the framework, it is not difficult to attempt proofs of cut-admissibility when considering different cut rules. This is exemplified in the system G3ip (Section 8.1), where an additive and a multiple cut rule are considered.

8.1. System G3ip for propositional intuitionistic logic

The system G3ip and its specification as a rewrite theory are presented in Section 5. The proof of meta-theoretical properties is next described in detail.

Weakening. Lemma 1 (see rule H in Equation (5)) and height-preserving admissibility of weakening can be proved without auxiliary lemmas.

Theorem 5 *(Weakening and weak-height). If* G3ip $\blacktriangleright_n\ \Gamma \vdash C$, *then* G3ip $\blacktriangleright_{suc(n)}\ \Gamma \vdash C$. *Moreover, the rule* W *(Equation* (1)*) is height-preserving admissible in* G3ip.

Proof. See the specification of the theorems in `g3i/prop-W.maude` and `g3i/prop-H.maude`. The resulting proofs are in `g3i/g3i.pdf`. Both properties are proved in less than 1 second without any additional lemma. □

Invertibility. All the rules, but $\vee_{R_i}$ and $\supset_L$, are invertible in G3ip. However, the tool initially fails (in 7.6 s) to prove the invertibility of $\supset_R$, $\wedge_R$, $\wedge_L$, $\vee_L$, and $\top_L$.

Consider the rule $\supset_R$. Recall that the invertibility of a rule is proven by testing the local invertibilities relative to all possible rules (see Definition 7). Hence, the tool proves, *e.g.*, that $\supset_R$ is invertible w.r.t $\wedge_L$ (the symbol $\bullet$ denotes successor):

$$\frac{h_3 : \Delta_6, F_4, F_5 \vdash F_1 \supset F_2}{\bullet h_3 : \Delta_6, F_4 \wedge F_5 \vdash F_1 \supset F_2}\ \wedge_L \quad \rightsquigarrow \quad \frac{\dfrac{}{h_3 : \Delta_6, F_1, F_4, F_5 \vdash F_2}\ \texttt{ax/ind}}{\bullet h_3 : \Delta_6, F_1, F_4 \wedge F_5 \vdash F_2}\ \wedge_L$$

The derivation on the left of the symbol $\rightsquigarrow$ represents the current case. On the right, the result of the proof search procedure applied to the goal in the conclusion. Note the application of the inductive hypothesis on the shorter derivation of height h_3. All the other cases of local invertibility of $\supset_R$ are similar, but the cases w.r.t. $\supset_R$ and $\supset_L$ fail when `already-proved-theorems` is empty. The proof obligation w.r.t $\supset_R$ that cannot be rewritten to `proved` is in red:

$$\frac{h_1 : \Delta_2, F_3 \vdash F_4}{\bullet h_1 : \Delta_2 \vdash F_3 \supset F_4}\ \supset_R \quad \rightsquigarrow \quad \frac{}{\bullet h_1 : \Delta_2, F_3 \vdash F_4}\ \texttt{fail}$$

This is the dummy case where the same rule is applied on the same formula and the proof transformation should be trivial. If H (Theorem 5) is added to the set of already proved theorems, this case is solved:

$$\frac{h_1 : \Delta_2, F_3 \vdash F_4}{\bullet h_1 : \Delta_2 \vdash F_3 \supset F_4}\ \supset_R \quad \rightsquigarrow \quad \frac{\dfrac{}{h_1 : \Delta_2, F_3 \vdash F_4}\ \texttt{ax}}{\bullet h_1 : \Delta_2, F_3 \vdash F_4}\ \texttt{H}$$

Regarding the failure of the invertibility of $\supset_R$ w.r.t $\supset_L$

$$\frac{h_3 : \Delta_6, F_4 \supset F_5 \vdash F_4 \quad h_3 : \Delta_6, F_5 \vdash F_1 \supset F_2}{\bullet h_3 : \Delta_6, F_4 \supset F_5 \vdash F_1 \supset F_2}\ \supset_L \quad \rightsquigarrow \quad \frac{}{\bullet h_3 : \Delta_6, F_1, F_4 \supset F_5 \vdash F_2}\ \texttt{fail}$$

If $\supset_L$ is applied on the sequent $\Delta_6, F_1, F_4 \supset F_5 \vdash_{suc(h_3)} F_2$, two premises are obtained: $\Delta_6, F_1, F_4 \supset F_5 \vdash_{h_3} F_4$ and $\Delta_6, F_1, F_5 \vdash_{h_3} F_2$. The second premise is provable by induction. The proof of the first premise requires weakening on F_1:

$$\frac{h_3 : \Delta_6, F_4 \supset F_5 \vdash F_4 \quad h_3 : \Delta_6, F_5 \vdash F_1 \supset F_2}{\bullet h_3 : \Delta_6, F_4 \supset F_5 \vdash F_1 \supset F_2}\ \supset_L \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{}{h_3 : \Delta_6, F_4 \supset F_5 \vdash F_4}\ \texttt{ax}}{h_3 : \Delta_6, F_1, F_4 \supset F_5 \vdash F_4}\ W \quad \dfrac{}{h_3 : \Delta_6, F_1, F_5 \vdash F_2}\ \texttt{ax/ind}}{\bullet h_3 : \Delta_6, F_1, F_4 \supset F_5 \vdash F_2}\ \supset_L$$

Once W is added, the proofs of invertibility of $\wedge_R$, $\wedge_L$, $\vee_L$ and $\top_L$ are also completed.

Some cases are vacuously discharged. For instance, there are no proof obligations for the invertibility of $\supset_R$ w.r.t. $\wedge_R$ (it is impossible to unify the conclusion of these rules). Moreover, the cases of the axioms I, $\top_R$, and $\bot_L$ are trivial since the only proof obligation is to reduce `proved` into `proved`:

$$\frac{\mathtt{h}_1 : \Delta_4, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \mathtt{F}_2 \quad \mathtt{h}_1 : \Delta_4, \mathtt{F}_3 \vdash \top}{\bullet \mathtt{h}_1 : \Delta_4, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \top} \supset_L \quad \rightsquigarrow \quad \texttt{trivial}$$

The rules $\vee_{R_1}$, $\vee_{R_2}$, and the left premise of $\supset_L$ are clearly not invertible. The following failures provide good evidences that these cases do not succeed:

$$\frac{\mathtt{h}_1 : \Delta_2 \vdash \mathtt{F}_4}{\bullet \mathtt{h}_1 : \Delta_2 \vdash \mathtt{F}_3 \vee \mathtt{F}_4} \vee_2 \quad \rightsquigarrow \quad \frac{}{\bullet \mathtt{h}_1 : \Delta_2 \vdash \mathtt{F}_3}\ \texttt{fail} \qquad \text{and}$$

$$\frac{}{\mathtt{h}_3 \bullet : \Delta_4, \mathtt{F}_1 \supset \mathtt{F}_2 \vdash \top} \top_R \quad \rightsquigarrow \quad \frac{}{\mathtt{h}_3 \bullet : \Delta_4, \mathtt{F}_1 \supset \mathtt{F}_2 \vdash \mathtt{F}_1}\ \texttt{fail}$$

Theorem 6 *(Invertibility). All the rules, but $\vee_{R_i}$ and $\supset_L$, are height-preserving invertible in* G3ip. *Moreover, the right premise of $\supset_L$ is height-preserving invertible.*

Proof. The invertibility of $\wedge_R$, $\wedge_L$, $\vee_L$, and $\top_L$ depends on the admissibility of W (Theorem 5). The invertibility of $\supset_R$ and the invertibility of the right premise of $\supset_L$ require Theorem 5 (W and H). The specification of the property is in `g3i/prop-inv.maude`. The analysis is completed in 7.7 seconds. □

Contraction. When attempting a proof of admissibility of contraction, the local admissibility cases w.r.t $\supset_L$, $\wedge_L$, $\vee_L$, and $\top_L$ fail. Here is the failing case for $\vee_L$:

$$\frac{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4 \quad \mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \vee \mathtt{F}_3, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4} \vee_L \quad \rightsquigarrow \quad \frac{}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4}\ \texttt{fail}$$

Note that the inductive hypothesis cannot be used neither on the left nor on the right premise. After adding `'OrL` to the set of already-proved invertible rules (field `inv-rules`), the `L-Framework` completes this case as follows:

$$\frac{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4 \quad \mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \vee \mathtt{F}_3, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4} \vee_L \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2, \mathtt{F}_2 \vdash \mathtt{F}_4}\ \texttt{inv-th/ax}}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \vdash \mathtt{F}_4}\ \texttt{IH} \quad \dfrac{\dfrac{}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_3 \vdash \mathtt{F}_4}\ \texttt{inv-th/ax}}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_3 \vdash \mathtt{F}_4}\ \texttt{IH}}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4} \vee_L$$

$$\frac{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2, \mathtt{F}_6, \mathtt{F}_6 \vdash \mathtt{F}_4 \quad \mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_6, \mathtt{F}_6 \vdash \mathtt{F}_4}{\bullet \mathtt{h}_1 : (\Delta_5, \mathtt{F}_2 \vee \mathtt{F}_3), \mathtt{F}_6, \mathtt{F}_6 \vdash \mathtt{F}_4} \vee_L \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2, \mathtt{F}_6, \mathtt{F}_6 \vdash \mathtt{F}_4}\ \texttt{ax}}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2, \mathtt{F}_6 \vdash \mathtt{F}_4}\ \texttt{IH} \quad \dfrac{\dfrac{}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_6, \mathtt{F}_6 \vdash \mathtt{F}_4}\ \texttt{ax}}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_6 \vdash \mathtt{F}_4}\ \texttt{IH}}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_6, \mathtt{F}_2 \vee \mathtt{F}_3 \vdash \mathtt{F}_4} \vee_L$$

Due to unification, there are indeed two cases: one in which the disjunctive formula is contracted and one of the copies is principal, and another case with the disjunction not being contracted (instead, another formula F_6 is). The second case follows without using the invertibility lemma.

Theorem 7 *(Contraction). The contraction rule* C *(Equation* (1)*) is height-preserving admissible in* G3ip.

Proof. The cases $\vee_L$, $\wedge_L$, and $\top_L$ require the invertibility of the respective rules (Theorem 6). The case $\supset_L$ requires invertibility of the right premise of this rule (specified in `inv-rules` as `'impL$$1`). The proof takes 1.3 seconds. □

Cut-admissibility. Controlling the structural rules is one of the key points to reduce the search space in sequent systems. The system G3ip embeds weakening in its initial rule (Γ can be an arbitrary context) and contraction in rules with two premises (all rules are additive and contraction is explicit in the left premise of $\supset_L$). Thus, W and C are admissible in G3ip, and the proof search procedure does not need to guess how many times a formula must be contracted or whether some of them need to be weakened. The cut-rule considered for G3ip is additive (see Equation (8)), hence also carrying an implicit contraction.

The proof of cut-admissibility for G3ip implements two strategies for reducing the search space (see Section 7.6):

1. If `GTP` is the resulting set of sequents that can be assumed to be provable, then such sequents are added as axioms that internalize weakening. For instance, if $\Gamma \vdash F$ is a (ground) sequent in `GTP`, then, $\Delta \vdash F$ is assumed to be provable whenever $\Delta \supseteq \Gamma$. This avoids the need for adding the rule W to the set of already proved theorems, thus reducing the non-determinism during proof-search: weakening is "delayed" until the leaves of the derivation are reached.
2. The inductive hypothesis on the structure of the formula is only used in the principal cases. Hence, less alternatives are explored when proving the non-principal cases.

The strategy (1) considerably affects the performance in both failing and successful attempts as explained below. The strategy (2) saves few seconds when all the needed auxiliary lemmas are added and the proof succeeds. In failing attempts, this strategy has an important impact.

Due to (1), W and C are not added to `already-proved-theorems` and, for the moment, consider also that none of the invertibility lemmas is added. This experiment leads to proofs for the trivial cases ($\top_R$, $\bot_L$, I and $\top_L$) and fails for the other rules in almost 15 seconds.[3]

Non-principal cases. Usually the non-principal cases are easily solved by permuting down the application of a rule and reducing the height of the cut. Some of these cases are already proved in this first iteration. For instance, the case ($\wedge_L$, $\vee_1$) — $\wedge_L$ applied on the left premise and $\vee_1$ on the right premise — is solved as follows:

$$\dfrac{\dfrac{h_1 : \Delta_{11}, F_6, F_7 \vdash F_{12}}{\bullet h_1 : \Delta_{11}, F_6 \wedge F_7 \vdash F_{12}}\wedge_L \quad \dfrac{h_8 : \Delta_{11}, F_{12}, F_6 \wedge F_7 \vdash F_9}{\bullet h_8 : (\Delta_{11}, F_6 \wedge F_7), F_{12} \vdash F_9 \vee F_{10}}\vee_1}{- : \Delta_{11}, F_6 \wedge F_7 \vdash F_9 \vee F_{10}}\,\mathtt{Cut} \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{}{\bullet h_1 : \Delta_{11}, F_6 \wedge F_7 \vdash F_{12}}\,\mathtt{ax/W} \quad \dfrac{}{h_8 : \Delta_{11}, F_{12}, F_6 \wedge F_7 \vdash F_9}\,\mathtt{ax/W}}{- : \Delta_{11}, F_6 \wedge F_7 \vdash F_9}\,\mathtt{hCut}}{- : \Delta_{11}, F_6 \wedge F_7 \vdash F_9 \vee F_{10}}\vee_1$$

In the right derivation, `hCut` is an application of the cut-rule with shorter derivations. Moreover, `ax/W` finishes the proof due to the sequents assumed to be provable (on the left derivation), possibly applying W. In this particular case, weakening is not needed. Some other similar cases, however, fail. Take for instance the case ($\wedge_L$, $\wedge_L$):

$$\dfrac{\dfrac{h_1 : \Delta_{12}, F_6, F_7 \vdash F_9 \wedge F_{10}}{\bullet h_1 : \Delta_{12}, F_6 \wedge F_7 \vdash F_9 \wedge F_{10}}\wedge_L \quad \dfrac{h_8 : \Delta_{12}, F_9, F_{10}, F_6 \wedge F_7 \vdash F_{11}}{\bullet h_8 : (\Delta_{12}, F_6 \wedge F_7), F_9 \wedge F_{10} \vdash F_{11}}\wedge_L}{- : \Delta_{12}, F_6 \wedge F_7 \vdash F_{11}}\,\mathtt{Cut} \quad \rightsquigarrow \quad \dfrac{}{- : \Delta_{12}, F_6 \wedge F_7 \vdash F_{11}}\,\mathtt{fail}$$

What is missing here is the invertibility of $\wedge_L$ on the assumption $\Delta_{12}, F_9, F_{10}, F_6 \wedge F_7 \vdash_{h_8} F_{11}$. If this invertibility lemma is added, the tool completes the case:

$$\rightsquigarrow \quad \dfrac{\dfrac{}{h_1 : \Delta_{12}, F_6, F_7 \vdash F_9 \wedge F_{10}}\,\mathtt{ax/W} \quad \dfrac{\dfrac{}{h_8 : \Delta_{12}, F_{10}, F_6, F_7, F_9 \vdash F_{11}}\,\mathtt{inv\text{-}th/ax}}{\bullet h_8 : \Delta_{12}, F_6, F_7, F_9 \wedge F_{10} \vdash F_{11}}\wedge_L}{- : \Delta_{12}, F_6, F_7 \vdash F_{11}}\,\mathtt{hCut}}{- : \Delta_{12}, F_6 \wedge F_7 \vdash F_{11}}\wedge_L$$

Inspecting similar failing cases suggests the need for including also the invertibility of the rules $\wedge_R$ and $\vee_L$, and also the invertibility of the right premise of $\supset_L$. This solves some missing cases but still, the cases for $\supset_L$, $\wedge_L$ and $\vee_L$ are not complete. One of the failures for ($\supset_L$, $\supset_L$) is the following:

$$\dfrac{\dfrac{h_1 : \Delta_{10}, F_7 \supset F_8 \vdash F_7 \quad h_1 : \Delta_{10}, F_8 \vdash F_{11}}{\bullet h_1 : \Delta_{10}, F_7 \supset F_8 \vdash F_{11}}\supset_L \quad \dfrac{h_6 : \Delta_{10}, F_{11}, F_7 \supset F_8 \vdash F_7 \quad h_6 : \Delta_{10}, F_8, F_{11} \vdash F_9}{\bullet h_6 : (\Delta_{10}, F_7 \supset F_8), F_{11} \vdash F_9}\supset_L}{- : \Delta_{10}, F_7 \supset F_8 \vdash F_9}\,\mathtt{Cut} \quad \rightsquigarrow \quad \dfrac{}{- : \Delta_{10}, F_7 \supset F_8 \vdash F_9}\,\mathtt{fail}$$

The cut-formula is F_{11} and it is not principal in any of the premises. Once $\supset_L$ is applied on the goal $\Delta_{10}, F_7 \supset F_8 \vdash F_9$, the resulting left premise is already proved (see the left-most sequent in the left derivation). The right premise can be proved with H (Theorem 5) if it is added to `already-proved-theorems`.

Principal cases. Note that in all the above derivations, due to unification, there is always more than one constant of sort `Formula` in the goal: besides the formula in the succedent of the sequent, there are formulas in the antecedent that are needed for the application of a left rule in the left premise of the cut. Due to the strategy (2), cuts on smaller formulas are not considered during the proof search for these cases. The situation is different in the principal cases. Consider for instance the case ($\supset_R$, $\supset_L$):

$$\dfrac{\dfrac{h_1 : \Delta_9, F_6 \vdash F_7}{\bullet h_1 : \Delta_9 \vdash F_6 \supset F_7}\supset_R \quad \dfrac{h_5 : \Delta_9, F_6 \supset F_7 \vdash F_6 \quad h_5 : \Delta_9, F_7 \vdash F_8}{\bullet h_5 : \Delta_9, F_6 \supset F_7 \vdash F_8}\supset_L}{- : \Delta_9 \vdash F_8}\,\mathtt{Cut} \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{}{\bullet h_1 : \Delta_9 \vdash F_6 \supset F_7}\,\mathtt{ax/W} \quad \dfrac{}{h_5 : \Delta_9, F_6 \supset F_7 \vdash F_6}\,\mathtt{ax/W}}{- : \Delta_9 \vdash F_6}\,\mathtt{hCut} \quad \dfrac{\dfrac{}{- : \Delta_9, F_6 \vdash F_7}\,\mathtt{ax/W} \quad \dfrac{}{- : \Delta_9, F_6, F_7 \vdash F_8}\,\mathtt{ax/W}}{- : \Delta_9, F_6 \vdash F_8}\,\mathtt{sCut}}{- : \Delta_9 \vdash F_8}\,\mathtt{sCut}$$

Rule `sCut` corresponds to a cut on a sub-formula. Note that the antecedent of the goal is just a constant of sort `MSFormula` (Δ_9).

Theorem 8 *(Cut-admissibility). The cut-rule in Equation* (8) *is admissible in* G3ip.

[3] In the same experiment, if the strategy (2) is not considered, the case $\supset_R$ does not finish after 15 min.

Proof. See the specification and dependencies in `g3i/prop-cut.maude`. The proof requires Theorem 5. All the cases –except for $\top_R$, $\top_L$, $\bot_L$, and I– require Theorem 6. The proof is completed in 13.8 sec. □

Multiplicative cut in G3ip. Observe that, by adopting the additive version of the cut-rule, several common proof search problems are avoided, *e.g.*, loops created by the uncontrolled use of contraction. But what if the following *multiplicative* cut is considered in G3ip instead, where the context is split between the two premises?

$$\frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ Cut \tag{9}$$

The search tree now is considerably bigger since all the alternatives on how to split the context Γ, Δ need to be considered (see `cut-mul-scon.maude`). This is an interesting question, and the discussion presented next will serve to pave the way to the analysis of sequent systems with linear contexts (see Section 8.4).

Note that, if only the rule H (Theorem 5) is added, then all the cases but the principal cases for $\supset$ and $\wedge$ succeed in 26 seconds. The failure on $(\wedge_R, \wedge_L)$ is:

$$\dfrac{\dfrac{\mathtt{h}_1 : \Delta_2 \vdash \mathtt{F}_6 \quad \mathtt{h}_1 : \Delta_2 \vdash \mathtt{F}_7}{\bullet \mathtt{h}_1 : \Delta_2 \vdash \mathtt{F}_6 \wedge \mathtt{F}_7}\ \wedge_R \quad \dfrac{\mathtt{h}_5 : \Delta_9, \mathtt{F}_6, \mathtt{F}_7 \vdash \mathtt{F}_8}{\bullet \mathtt{h}_5 : \Delta_9, \mathtt{F}_6 \wedge \mathtt{F}_7 \vdash \mathtt{F}_8}\ \wedge_L}{- : \Delta_2, \Delta_9 \vdash \mathtt{F}_8}\ \mathtt{Cut} \quad \rightsquigarrow \quad \dfrac{}{- : \Delta_2, \Delta_9 \vdash \mathtt{F}_8}\ \mathtt{fail}$$

This case is solved by cutting with F_6 and F_7. However, since the cut-rule is multiplicative, the contexts Δ_2 and Δ_9 need to be contracted first. Adding contraction, on terms of sort `MSFormula` makes infeasible the proof search procedure: any subset of the antecedent context can be chosen for contraction and such rule can be applied on any sequent/goal. Instead, a more controlled version of contraction can be added: contract the whole context only if there are no duplicated elements in it. This more restricted rule cannot be applied twice on the same goal, thus reducing the number of alternatives leading to the following proof transformation:

$$\dfrac{\dfrac{\mathtt{h}_1 : \Delta_2, \mathtt{F}_6 \vdash \mathtt{F}_7}{\bullet \mathtt{h}_1 : \Delta_2 \vdash \mathtt{F}_6 \supset \mathtt{F}_7}\ \supset_R \quad \dfrac{\mathtt{h}_5 : \Delta_9, \mathtt{F}_6 \supset \mathtt{F}_7 \vdash \mathtt{F}_6 \quad \mathtt{h}_5 : \Delta_9, \mathtt{F}_7 \vdash \mathtt{F}_8}{\bullet \mathtt{h}_5 : \Delta_9, \mathtt{F}_6 \supset \mathtt{F}_7 \vdash \mathtt{F}_8}\ \supset_L}{- : \Delta_2, \Delta_9 \vdash \mathtt{F}_8}\ \mathtt{Cut} \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\bullet \mathtt{h}_1 : \Delta_2 \vdash \mathtt{F}_6 \supset \mathtt{F}_7}\ \mathtt{ax/W} \quad \dfrac{}{\mathtt{h}_5 : \Delta_9, \mathtt{F}_6 \supset \mathtt{F}_7 \vdash \mathtt{F}_6}\ \mathtt{ax/W}}{- : \Delta_2, \Delta_9 \vdash \mathtt{F}_6}\ \mathtt{hCut} \quad \dfrac{\dfrac{}{- : \Delta_2, \mathtt{F}_6 \vdash \mathtt{F}_7}\ \mathtt{ax/W} \quad \dfrac{}{- : \Delta_9, \mathtt{F}_7 \vdash \mathtt{F}_8}\ \mathtt{ax/W}}{- : \Delta_2, \Delta_9, \mathtt{F}_6 \vdash \mathtt{F}_8}\ \mathtt{sCut}}{- : \Delta_2, \Delta_2, \Delta_9, \Delta_9 \vdash \mathtt{F}_8}\ \mathtt{sCut}}{- : \Delta_2, \Delta_9 \vdash \mathtt{F}_8}\ C$$

Theorem 9 *(Multiplicative cut). The rule in Eq. (9) is admissible in* G3ip.

Proof. The specification is in `g3i/prop-cut-mul.maude`. See the contraction rule used in the definition of `already-proved-theorems`. No invertibility lemma is needed for this proof. The analysis is completed in 25.6 seconds. □

If W is not embedded in the initial axioms and C is allowed on arbitrary contexts, there is little hope to conclude these proofs in reasonable time.

Identity expansion. Finally, the dual property of cut-admissibility, identity expansion, is easily proved in the `L-Framework`.

Theorem 10 *(Identity-expansion). If F is a formula, then $F \vdash F$ is provable in* G3ip.

Proof. See `g3i/prop-ID.maude`. The rule W needs to be added to the set of already proved theorems. It is used, *e.g.*, in the following case:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{- : \mathtt{F}_0 \vdash \mathtt{F}_0}\ \mathtt{IH}}{- : \mathtt{F}_0, \mathtt{F}_0 \supset \mathtt{F}_1 \vdash \mathtt{F}_0}\ W \quad \dfrac{\dfrac{}{- : \mathtt{F}_1 \vdash \mathtt{F}_1}\ \mathtt{IH}}{- : \mathtt{F}_0, \mathtt{F}_1 \vdash \mathtt{F}_1}\ W}{- : \mathtt{F}_0, \mathtt{F}_0 \supset \mathtt{F}_1 \vdash \mathtt{F}_1}\ \supset_L}{- : \mathtt{F}_0 \supset \mathtt{F}_1 \vdash \mathtt{F}_0 \supset \mathtt{F}_1}\ \supset_R \qquad \square$$

8.2. Multi-conclusion propositional intuitionistic logic (mLJ)

Maehara's mLJ [25] is a multiple conclusion system for intuitionistic logic. The rules in mLJ have the exact same shape as in G3ip, except for the right rules for disjunction and implication (see Fig. 2). The disjunction right rule in mLJ matches the corresponding rule in classical logic where the disjunction is interpreted as the comma in the succedent of sequents. The

$$\frac{\Gamma, A \supset B \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \supset B \vdash \Delta}\ \supset_L \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B, \Delta}\ \supset_R \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ \vee_R$$

Fig. 2. The multi-conclusion intuitionistic sequent system mLJ.

right implication, on the other hand, forces all formulas in the succedent of the premise to be weakened. This guarantees that, when the $\supset_R$ rule is applied on $A \supset B$, the formula B should be proved assuming *only* the pre-existent antecedent context extended with the formula A. This creates an interdependency between A and B.

Weakening. The proof of admissibility of weakening in mLJ is similar to G3ip, only noting that, in the former, weakening is also height-preserving admissible in the succedent of sequents.

Theorem 11 *(Weakening and weak-height). If* mLJ $\blacktriangleright_n \Gamma \vdash \Delta$, *then* mLJ $\blacktriangleright_{suc(n)} \Gamma \vdash \Delta$ (H). *Moreover, the following rules are height-preserving admissible in* mLJ*:*

$$\frac{\Gamma \vdash \Delta}{\Gamma, F \vdash \Delta}\ \mathsf{W}_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, F}\ \mathsf{W}_R$$

Proof. The three properties are specified in `mLJ/prop-WH.maude`. No auxiliary lemmas are needed. This theorem is proved in less than 3 seconds. □

Invertibility. All the rules in mLJ are invertible, with the exception of $\supset_R$.

Theorem 12 *(Invertibility). All the rules but $\supset_R$ are height-preserving invertible in* mLJ.

Proof. See the specification in `mLJ/prop-inv.maude`. H (Theorem 11) is needed for all the cases but $\top_R$, $\bot_L$ and I. This proof takes 15.1 seconds. □

Contraction. Contraction is also admissible, on both sides of sequents.

Theorem 13 *(Contraction). The rules*

$$\frac{\Gamma, F, F \vdash \Delta}{\Gamma, F \vdash \Delta}\ \mathsf{C}_L \qquad \frac{\Gamma \vdash F, F, \Delta}{\Gamma \vdash F, \Delta}\ \mathsf{C}_R$$

are both height-preserving admissible in mLJ.

Proof. The specification is in `mLJ/prop-C.maude`. The proof of admissibility of C_L (resp. C_R) requires the invertibility of $\top_L$, $\vee_L$, $\wedge_L$, and $\supset_L$ (resp., $\bot_R$, $\vee_R$ and $\wedge_R$). □

Cut-admissibility and identity expansion. In the proof of cut-admissibility for G3ip, the application of the rule for structural induction (`sCut`) is restricted to goals with at most one term of sort `Formula` (corresponding to the principal cases). That simplification is not possible in mLJ: with that restriction, the cases for $\wedge_R$ and $\vee_R$ fail w.r.t. $\supset_R$. Here is the case for $(\vee_R, \supset_R)$:

$$\dfrac{\dfrac{\mathtt{h}_1 : \Delta_{12} \vdash (\Delta_{11}, \mathtt{F}_9 \supset \mathtt{F}_{10}), \mathtt{F}_6, \mathtt{F}_7}{\bullet \mathtt{h}_1 : \Delta_{12} \vdash (\Delta_{11}, \mathtt{F}_9 \supset \mathtt{F}_{10}), \mathtt{F}_6 \vee \mathtt{F}_7}\ \vee_R \quad \dfrac{\mathtt{h}_8 : \Delta_{12}, \mathtt{F}_9, \mathtt{F}_6 \vee \mathtt{F}_7 \vdash \mathtt{F}_{10}}{\bullet \mathtt{h}_8 : \Delta_{12}, \mathtt{F}_6 \vee \mathtt{F}_7 \vdash \Delta_{11}, \mathtt{F}_9 \supset \mathtt{F}_{10}}\ \supset_R}{- : \Delta_{12} \vdash \Delta_{11}, \mathtt{F}_9 \supset \mathtt{F}_{10}}\ \mathtt{Cut} \qquad \rightsquigarrow$$

$$\frac{}{- : \Delta_{12} \vdash \Delta_{11}, \mathtt{F}_9 \supset \mathtt{F}_{10}}\ \mathtt{fail}$$

Note that the cut formula $F_6 \vee F_7$ is principal on the left premise, but it is not on the right premise. This case cannot be solved by reducing the height of the cut by first applying $\supset_R$ since this would remove the context Δ_{11} (needed in the left premise).

The cut-admissibility procedure for mLJ is based on the module defined in the file `cut-add-mcon.maude` (additive multiple-conclusion) where `sCut` is added in cases where the goal sequent has at most two terms of sort `Formula`. Hence, `sCut` is allowed also in non-principal cases. This solves the previous case and the search procedure finds the following proof transformation:

$$\rightsquigarrow \quad \dfrac{\dfrac{\dfrac{}{- : \Delta_{12} \vdash \Delta_{11}, \mathtt{F}_6, \mathtt{F}_7, \mathtt{F}_9 \supset \mathtt{F}_{10}}\ \mathtt{ax/W} \quad \dfrac{\dfrac{}{- : \Delta_{12}, \mathtt{F}_7, \mathtt{F}_9 \vdash \mathtt{F}_{10}}\ \mathtt{inv\text{-}th/ax}}{- : \Delta_{12}, \mathtt{F}_7 \vdash \Delta_{11}, \mathtt{F}_6, \mathtt{F}_9 \supset \mathtt{F}_{10}}\ \supset_R}{- : \Delta_{12} \vdash \Delta_{11}, \mathtt{F}_6, \mathtt{F}_9 \supset \mathtt{F}_{10}}\ \mathtt{sCut} \quad \dfrac{\dfrac{}{- : \Delta_{12}, \mathtt{F}_6, \mathtt{F}_9 \vdash \mathtt{F}_{10}}\ \mathtt{inv\text{-}th/ax}}{- : \Delta_{12}, \mathtt{F}_6 \vdash \Delta_{11}, \mathtt{F}_9 \supset \mathtt{F}_{10}}\ \supset_R}{- : \Delta_{12} \vdash \Delta_{11}, \mathtt{F}_9 \supset \mathtt{F}_{10}}\ \mathtt{sCut}$$

Theorem 14 *(Cut-admissibility and ID-expansion). The following cut rule*

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}\ Cut$$

is admissible in mLJ. *Moreover, for any F, the sequent $F \vdash F$ is provable.*

Proof. For cut-admissibility (`mLJ/prop-cut.maude`), H as well as the invertibility of the rules $\wedge_L$, $\wedge_R$, $\vee_L$, $\vee_R$, and $\supset_L$ are needed. For identity-expansion (`mLJ/prop-ID.maude`), the admissibility of W_R and W_L (Theorem 11) is needed. The proof takes 35.2 seconds. □

8.3. System G3cp for propositional classical logic

G3cp [11] is a well-known two-sided sequent system for classical logic, where the structural rules are implicit and all the rules are invertible. The rules are similar to those of mLJ with the exception of $\supset_R$:

$$\frac{\Gamma, F \vdash \Delta, G}{\Gamma \vdash \Delta, F \supset G}\ \supset_R$$

Weakening. Admissibility of weakening of G3cp follows the same lines as in mLJ.

Theorem 15 *(Weakening and weak-height). If* G3cp $\blacktriangleright_n\ \Gamma \vdash \Delta$, *then* G3cp $\blacktriangleright_{suc(n)}\ \Gamma \vdash \Delta$ (H). *The rules* W_L *and* W_R *(see Theorem 11) are height-preserving admissible in* G3cp.

Proof. The three properties are specified in `g3c/prop-WH.maude`. No auxiliary lemmas are needed. □

Besides weakening and contraction, the procedures for checking admissibility of one-premise rules have been used for proving the admissibility of the rule H. Proposition 2 shows two further applications of such procedures.

Proposition 2. *The following two rules are admissible in* G3cp*:*

$$\frac{\Gamma \vdash \Delta, A \wedge A}{\Gamma \vdash \Delta, A}\ \wedge_w \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \wedge A}\ \wedge_c$$

Proof. The specification is in `g3c/prop-and.maude`. Rule $\wedge_w$ is height-preserving admissible. In $\wedge_c$, what it is proved is: If G3cp $\blacktriangleright_n\ \Gamma \vdash \Delta, A$ then G3cp $\blacktriangleright_{suc(n)}\ \Gamma \vdash \Delta, A \wedge A$, *i.e.*, the conclusion is provable with at most one extra derivation step. Here a representative case:

$$\frac{\mathtt{h}_1 : \Delta_2, \mathtt{F}_4 \vdash \Delta_3, \mathtt{F}_5}{\bullet \mathtt{h}_1 : \Delta_2 \vdash \Delta_3, \mathtt{F}_4 \supset \mathtt{F}_5}\ \supset_R \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{}{\mathtt{h}_1 : \Delta_2, \mathtt{F}_4 \vdash \Delta_3, \mathtt{F}_5}\ \mathtt{ax}}{\bullet \mathtt{h}_1 : \Delta_2 \vdash \Delta_3, \mathtt{F}_4 \supset \mathtt{F}_5}\ \supset_R \quad \dfrac{\dfrac{}{\mathtt{h}_1 : \Delta_2, \mathtt{F}_4 \vdash \Delta_3, \mathtt{F}_5}\ \mathtt{ax}}{\bullet \mathtt{h}_1 : \Delta_2 \vdash \Delta_3, \mathtt{F}_4 \supset \mathtt{F}_5}\ \supset_R}{\bullet\bullet \mathtt{h}_1 : \Delta_2 \vdash \Delta_3, (\mathtt{F}_4 \supset \mathtt{F}_5) \wedge (\mathtt{F}_4 \supset \mathtt{F}_5)}\ \wedge_R \quad \square$$

Invertibility. All the rules in G3cp are invertible.

Theorem 16 *(Invertibility). All the rules in* G3cp *are height-preserving invertible.*

Proof. See the specification in `g3c/prop-inv.maude`. H (Theorem 15) is needed. This proof takes 15.1 seconds. □

Contraction. An attempt of proving admissibility of contraction on the left side of the sequent fails due to $\supset_L$ (and on the right due to $\supset_R$). Here the failing case for C_L:

$$\frac{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \Delta_4, \mathtt{F}_2 \quad \mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \Delta_4}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \supset \mathtt{F}_3, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \Delta_4}\ \supset_L \quad \rightsquigarrow \quad \frac{}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \Delta_4}\ \mathtt{fail}$$

Due to invertibility of $\supset_L$, the sequent $\Delta_5 \vdash \Delta_4, F_2, F_2$ is provable. However, induction does not apply on this sequent since F_2 is on the right side of the sequent.

Hence, the proof of admissibility of contraction is by mutual induction on C_L and C_R. For instance, the case of $\supset_L$ is solved by applying C_R on a shorter derivation:

$$\frac{\mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \Delta_4, \mathtt{F}_2 \quad \mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \Delta_4}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \supset \mathtt{F}_3, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \Delta_4}\ \supset_L \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{}{\mathtt{h}_1 : \Delta_5 \vdash \Delta_4, \mathtt{F}_2, \mathtt{F}_2}\ \mathtt{inv\text{-}th/ax}}{\mathtt{h}_1 : \Delta_5 \vdash \Delta_4, \mathtt{F}_2}\ \mathtt{IH\text{-}Mutual} \quad \dfrac{\dfrac{}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_3, \mathtt{F}_3 \vdash \Delta_4}\ \mathtt{inv\text{-}th/ax}}{\mathtt{h}_1 : \Delta_5, \mathtt{F}_3 \vdash \Delta_4}\ \mathtt{IH}}{\bullet \mathtt{h}_1 : \Delta_5, \mathtt{F}_2 \supset \mathtt{F}_3 \vdash \Delta_4}\ \supset_L$$

$$\frac{}{\vdash p^{\perp}, p}\ I \qquad \frac{}{\vdash 1}\ 1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\ \perp \qquad \frac{}{\vdash \Gamma, \top}\ \top$$

$$\frac{\vdash \Gamma_1, A \quad \vdash \Gamma_2, B}{\vdash \Gamma_1, \Gamma_2, A \otimes B}\ \otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ \parr \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\ \& \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\ \oplus_1 \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ \oplus_2$$

Fig. 3. One-sided multiplicative-additive linear logic (MALL).

$$\frac{\mathrm{h}_1 : \Delta_5, \mathrm{F}_6, \mathrm{F}_6 \vdash \Delta_4, \mathrm{F}_2 \quad \mathrm{h}_1 : \Delta_5, \mathrm{F}_3, \mathrm{F}_6, \mathrm{F}_6 \vdash \Delta_4}{\bullet \mathrm{h}_1 : (\Delta_5, \mathrm{F}_2 \supset \mathrm{F}_3), \mathrm{F}_6, \mathrm{F}_6 \vdash \Delta_4}\ \supset_L \qquad \rightsquigarrow \qquad \frac{\dfrac{\dfrac{}{\mathrm{h}_1 : \Delta_5, \mathrm{F}_6, \mathrm{F}_6 \vdash \Delta_4, \mathrm{F}_2}\ \mathtt{ax}}{\mathrm{h}_1 : \Delta_5, \mathrm{F}_6 \vdash \Delta_4, \mathrm{F}_2}\ \mathtt{IH} \quad \dfrac{\dfrac{}{\mathrm{h}_1 : \Delta_5, \mathrm{F}_3, \mathrm{F}_6, \mathrm{F}_6 \vdash \Delta_4}\ \mathtt{ax}}{\mathrm{h}_1 : \Delta_5, \mathrm{F}_3, \mathrm{F}_6 \vdash \Delta_4}\ \mathtt{IH}}{\bullet \mathrm{h}_1 : \Delta_5, \mathrm{F}_6, \mathrm{F}_2 \supset \mathrm{F}_3 \vdash \Delta_4}\ \supset_L$$

In the second proof transformation, contraction is applied on a formula different from the implication and mutual induction is not needed.

Theorem 17 *(Contraction). The rules* C_L *and* C_R *are height-preserving admissible in* G3cp.

Proof. The specification is in `g3c/prop-C.maude`. The proof of admissibility of C_L (resp., C_R) requires the invertibility of $\top_L$, $\vee_L$, $\wedge_L$, and $\supset_L$ (resp., $\perp_R$, $\vee_R$, $\wedge_R$, and $\supset_R$). For C_L, the specification includes the following definition for mutual induction:

```
eq mutual-ind('suc[GT]) =
  rl '_:_[GT, '_|--_['Gm:MSFormula, '_;_['F:Formula, 'Dt:MSFormula]]] =>
     '_:_[GT, '_|--_['Gm:MSFormula, '_;_['_;_['F:Formula,'F:Formula],
                      'Dt:MSFormula]]]   [ label('\IHMutual) ]. ) .
eq mutual-ind (GT) = none [owise] .
```

This means that C_R can be applied on shorter derivations of height `GT` (due to the pattern "`'suc[GT]`" in the definition of the equation). Similarly, the proof of C_R requires mutual induction on C_L (specifically for the case $\supset_R$). □

Cut-admissibility and identity expansion. The proof of these properties follow the same recipe as in mLJ.

Theorem 18 *(Cut-admissibility and ID-expansion). The cut-rule in Theorem 14 is admissible in* G3cp. *Moreover, for any* F, *the sequent* $F \vdash F$ *is provable.*

Proof. For cut-admissibility (`g3c/prop-cut.maude`), H as well as the invertibility of the rules $\supset_L$, $\supset_R$, $\wedge_L$, $\wedge_R$, $\vee_L$, and $\vee_R$ are needed. For identity-expansion (`g3c/prop-ID.maude`), the admissibility of W_R and W_L is needed. The proof of cut-admissibility takes 22.5 seconds and id-expansion less than one second. □

8.4. Propositional linear logic

Linear logic (LL) [26] is a resource aware logic. Formulas are consumed when used during proofs, unless they are marked with the exponential ? (whose dual is !), in which case they can be weakened and contracted. Besides the exponentials, propositional LL connectives include the additive conjunction & and disjunction $\oplus$, their multiplicative versions $\otimes$ and $\parr$, and the unities $1, 0, \top, \perp$. These connectives form actually pairs of *dual* operators:

$$(A \,\&\, B)^{\perp} \stackrel{\text{def}}{=} A^{\perp} \oplus B^{\perp} \qquad (A \otimes B)^{\perp} \stackrel{\text{def}}{=} A^{\perp} \parr B^{\perp} \qquad (!\,A)^{\perp} \stackrel{\text{def}}{=} ?A^{\perp} \qquad 0^{\perp} \stackrel{\text{def}}{=} \top \qquad 1^{\perp} \stackrel{\text{def}}{=} \perp$$

where $A^{\perp}$ denotes the negation of the formula A. All negations in LL can be pushed inwards and restricted to the atomic scope. For an atomic formula A, $(A^{\perp})^{\perp} \stackrel{\text{def}}{=} A$.

First, consider the fragment of LL without the exponentials, that is, (classical) propositional multiplicative-additive linear logic (MALL). The one-sided proof system for MALL is depicted in Fig. 3. As expected, the structural rules for weakening and contraction are not admissible in this system.

Invertibility. The well known invertibility results for MALL are easily proved in the `L-Framework`.

Theorem 19 *(Weak-height and invertibility). If* MALL $\blacktriangleright_n \vdash \Gamma$, *then* MALL $\blacktriangleright_{suc(n)} \vdash \Gamma$ (H). *Moreover, all the rules but* $\otimes$ *and* $\oplus_i$, $i \in \{1, 2\}$, *are height-preserving invertible.*

Proof. See `MALL/prop-H.maude` and `MALL/prop-inv.maude`. Due to the splitting of the context, clearly $\otimes$ is not invertible. Here two (failing) cases showing that the left premise of this rule is not invertible. The first one corresponds to invertibility w.r.t. $\top$ and the second one w.r.t. $\otimes$:

$$\frac{}{\bullet \mathrm{h}_3 :\vdash \top, \Delta_4, \Delta_5, \mathrm{F}_1 \otimes \mathrm{F}_2}\ \top \qquad \rightsquigarrow \qquad \frac{}{\bullet \mathrm{h}_3 :\vdash \mathrm{F}_1, \Delta_4}\ \mathtt{fail}$$

$$\frac{\mathrm{h}_3 :\vdash \mathrm{F}_4, \Delta_6, \Delta_7, \mathrm{F}_1 \otimes \mathrm{F}_2 \quad \mathrm{h}_3 :\vdash \mathrm{F}_5, \Delta_8, \Delta_9}{\bullet \mathrm{h}_3 :\vdash (\Delta_6, \Delta_7, \mathrm{F}_1 \otimes \mathrm{F}_2), (\Delta_8, \Delta_9), \mathrm{F}_4 \otimes \mathrm{F}_5}\ \otimes \qquad \rightsquigarrow \qquad \frac{}{\bullet \mathrm{h}_3 :\vdash \mathrm{F}_1, \Delta_6, \Delta_8, \mathrm{F}_4 \otimes \mathrm{F}_5}\ \mathtt{fail}$$ □

$$\frac{\vdash ?A_1, \ldots, ?A_n, A}{\vdash ?A_1, \ldots, ?A_n, !A}\ !\qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ ?\qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ ?_W\qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ ?_C$$

Fig. 4. Exponential rules for LL.

Cut-admissibility and identity expansion. Since the system under consideration is one-sided, the cut-rule has a one-sided presentation. The theory in `cut-lin-osided.maude` (linear cut, one sided) specifies the operation `op dual : Formula -> Formula .` and the OL must define equations for it reflecting the De Morgan dualities of the connectives. Here some examples for the LL connectives:

```
eq dual(p(i)) = perp(i) . eq dual(perp(i)) = p(i) . eq dual(1) = bot .
eq dual(F ** G) = dual(F) $ dual(G) . eq dual(F $ G) = dual(F) ** dual(G).
```

Theorem 20 *(Cut-admissibility and id-expansion). The rule* $\dfrac{\vdash \Gamma, F \quad \vdash \Delta, F^{\perp}}{\vdash \Gamma, \Delta}\ Cut$ *is admissible in* MALL. *Moreover, for any formula* F*, the sequent* $\vdash F, F^{\perp}$ *is provable.*

Proof. The notation $F^{\perp}$ corresponds to `dual(F)`. The proof of cut-admissibility (specification in `MALL/prop-cut.maude`) relies only on the admissibility of H. Here the principal case $(\otimes, \parr)$:

$$\dfrac{\dfrac{\dfrac{h_1 :\vdash F_6, \Delta_4 \quad h_1 :\vdash F_7, \Delta_5}{\bullet h_1 :\vdash F_6 \otimes F_7, \Delta_4, \Delta_5}\otimes \quad \dfrac{h_8 :\vdash \Delta_9, dual(F_6), dual(F_7)}{\bullet h_8 :\vdash dual(F_6 \otimes F_7), \Delta_9}\parr}{- :\vdash (\Delta_4, \Delta_5), \Delta_9}\ \text{Cut}}{\rightsquigarrow}$$

$$\dfrac{\dfrac{}{- :\vdash \Delta_4, F_6}\ \text{ax} \quad \dfrac{\dfrac{}{- :\vdash \Delta_5, F_7}\ \text{ax} \quad \dfrac{}{- :\vdash \Delta_9, dual(F_6), dual(F_7)}\ \text{ax}}{- :\vdash \Delta_5, \Delta_9, dual(F_6)}\ \text{sCut}}{- :\vdash \Delta_4, \Delta_5, \Delta_9}\ \text{sCut}$$

□

Exponentials. Consider the one-sided system for linear logic obtained by adding the exponential ? and its dual !. The system LL results from the inclusion of the inference rules in Fig. 4 to those in Fig. 3. Note the explicit rules for weakening and contraction on formulas marked with ?.

The specification of the rule ! (called *promotion*) requires a new sort to guarantee that the context only contains formulas marked with ? (see `LL/LL.maude`):

```
--- Formulas and multisets of formulas marked with ?
sorts ?Formula ?MSFormula .
op ?_ : Formula -> ?Formula .     --- a ?-Formula is built with ?
subsort ?Formula < ?MSFormula .
subsort ?MSFormula < MSFormula .
--- In Rule !,  the context must contain only ?-Formulas
var CLq : ?MSFormula .
rl [bang]  : |-- CLq ; ! F => |-- CLq ; F .
```

Theorem 21 *(Weak-height, weak-! and inv.). If* LL $\blacktriangleright_n \vdash \Gamma$*, then* LL $\blacktriangleright_{suc(n)} \vdash \Gamma$ (H)*. The rule* $\dfrac{\vdash \Gamma, !F}{\vdash \Gamma, F}\ \text{W!}$ *is height-preserving admissible. Moreover, none of the rules in Fig. 4, but !, is height-preserving invertible.*

Proof. See `LL/prop-H.maude`, `LL/prop-WB.maude` and `LL/prop-inv.maude`. The invertibility results and the admissibility of W! depend on H. Note that the rule $?_C$ is invertible: it is always possible to apply it to later use $?_W$ on the same contracted formula. However, such procedure does not preserve the height of the derivation. If the admissibility of W! is added to the set of already proved theorems, the invertibility of ! can be concluded. Note that the rule ! is context sensitive. This means that it cannot be eagerly applied, even though provability of the conclusion implies provability of the premise (and hence, invertible according to Definition 5). □

The cut-admissibility procedure for this system is certainly more involved. An attempt of proving this result fails in the case $(!, ?_C)$:

$$\dfrac{\dfrac{h_1 :\vdash F_4, ?\Gamma_3}{\bullet h_1 :\vdash !F_4, ?\Gamma_3}\ ! \quad \dfrac{h_5 :\vdash \Delta_6, ?dual(F_4), ?dual(F_4)}{\bullet h_5 :\vdash dual(!F_4), \Delta_6}\ ?_C}{- :\vdash ?\Gamma_3, \Delta_6}\ \text{Cut} \qquad \rightsquigarrow \qquad \dfrac{}{- :\vdash ?\Gamma_3, \Delta_6}\ \texttt{fail}$$

This is the principal case where the cut formula is $!F$ and it is promoted, and its dual contracted. This case is solved by using the rule below that cuts $!F$ with n copies of the formula $?F^{\perp}$:

$$\frac{\vdash \Gamma, !F \quad \vdash \Delta, (?F^{\perp})^{n}}{\vdash \Gamma, \Delta}\ mCut \tag{10}$$

Hence, the proof of cut-admissibility for this system must mutually eliminate the cut-rules in Theorem 20 and Rule (10). More precisely, the elimination of one of the cases of *Cut* relies on the application of *mCut* on shorter derivations and one of the cases in the elimination of *mCut* requires the application of *Cut* on smaller formulas. Here are the two relevant cases:

$$\frac{\dfrac{\mathtt{h}_1 :\vdash \mathtt{F}_4, ?\Gamma_3}{\bullet\mathtt{h}_1 :\vdash !\mathtt{F}_4, ?\Gamma_3}\ ! \quad \dfrac{\mathtt{h}_5 :\vdash \Delta_6, ?dual(\mathtt{F}_4), ?dual(\mathtt{F}_4)}{\bullet\mathtt{h}_5 :\vdash dual(!\mathtt{F}_4), \Delta_6}\ ?_C}{- :\vdash ?\Gamma_3, \Delta_6}\ \mathtt{Cut} \quad \rightsquigarrow \quad \frac{\dfrac{}{\bullet\mathtt{h}_1 :\vdash ?\Gamma_3, !\mathtt{F}_4}\ \mathtt{ax} \quad \dfrac{}{\mathtt{h}_5 :\vdash \Delta_6, ?dual(\mathtt{F}_4), ?dual(\mathtt{F}_4)}\ \mathtt{ax}}{- :\vdash ?\Gamma_3, \Delta_6}\ \mathtt{mCut}$$

$$\frac{\dfrac{\mathtt{h}_1 :\vdash \mathtt{F}_4, ?\Gamma_3}{\bullet\mathtt{h}_1 :\vdash !\mathtt{F}_4, ?\Gamma_3}\ ! \quad \dfrac{\mathtt{h}_6 :\vdash \Delta_7, dual(\mathtt{F}_4), ctr(\mathtt{n}_5, ?dual(\mathtt{F}_4))}{\bullet\mathtt{h}_6 :\vdash ctr(s\mathtt{n}_5, ?dual(\mathtt{F}_4)), \Delta_7}\ ?}{- :\vdash ?\Gamma_3, \Delta_7}\ \mathtt{mCut} \quad \rightsquigarrow$$

$$\frac{\dfrac{}{- :\vdash ?\Gamma_3, \mathtt{F}_4}\ \mathtt{ax} \quad \dfrac{\dfrac{}{\bullet\mathtt{h}_1 :\vdash ?\Gamma_3, !\mathtt{F}_4}\ \mathtt{ax} \quad \dfrac{}{\mathtt{h}_6 :\vdash \Delta_7, dual(\mathtt{F}_4), ctr(\mathtt{n}_5, ?dual(\mathtt{F}_4))}\ \mathtt{ax}}{- :\vdash ?\Gamma_3, \Delta_7, dual(\mathtt{F}_4)}\ \mathtt{hCut}}{- :\vdash ?\Gamma_3, \Delta_7}\ \mathtt{Cut}$$

In the last derivation, `ctr(s n5, ? dual(F4))` denotes $(?F_4^{\perp})^{suc(n_5)}$ and the application of *Cut* is on a smaller formula (F_4 instead of $!F_4$).

Theorem 22 *(Cut-adm. and id-exp.). The following rules are admissible in* LL*:*

$$\frac{\vdash \Gamma, F \quad \vdash \Gamma, F^{\perp}}{\vdash \Gamma}\ Cut \qquad \frac{\vdash \Gamma, !F \quad \vdash \Delta, (?F^{\perp})^{n}}{\vdash \Gamma, \Delta}\ mCut.$$

Moreover, for all formulas F, the sequent $\vdash F, F^{\perp}$ is provable.

Proof. The procedures using mutual-induction are defined in `LL/cut-ll.maude` and `LL/cut-ll-cc.maude` (for *mCut*). The properties are specified in `LL/prop-cut.maude` and `LL/prop-cut-cc.maude`. Discharging the cases for *Cut* (resp., *mCut*) takes 13.2 seconds (resp. 115.6 seconds). In `LL/cut-ll.maude`, the definition

```
op mutual-induct : GroundTerm GroundTerm GroundTerm -> RuleSet .
eq mutual-induct('suc[gh], 'suc[gh'], '!_[GTA]) = ... --- spec. of mCut
eq mutual-induct('suc[gh], 'suc[gh'], '?_[GTA]) = ... --- spec. of mCut
eq mutual-induct(gh, gh',  GTA) = none [owise] .
```

receives as parameter the height of the two premises of the cut and the cut-formula F. If F is not a formula marked with ! or ?, no additional rule is generated (the `[owise]` case in the definition). Also, the pattern matching on the heights of the derivation allows for controlling the application of the cut-rule on shorter derivations. A similar definition can be found in `LL/cut-ll-cc.maude`.

The elimination of *Cut* assumes as auxiliary lemmas H and also a generalization of $?_W$ on terms of sort `?MSFormula`: $\dfrac{\vdash \Gamma}{\vdash \Gamma, ?\Delta}\ ?_{GW}$. Note that this rule is not height-preserving admissible, but it is clearly admissible (and hence, it can be used only on sequents marked with `'inf.INat`). Currently, the `L-Framework` does not support inductive proofs on the size of lists/multisets as needed for this auxiliary lemma. This rule is used in the following case (where the height of the derivation is irrelevant):

$$\frac{\dfrac{\mathtt{h}_1 :\vdash \mathtt{F}_4, ?\Gamma_3}{\bullet\mathtt{h}_1 :\vdash !\mathtt{F}_4, ?\Gamma_3}\ ! \quad \dfrac{\mathtt{h}_5 :\vdash \Delta_6}{\bullet\mathtt{h}_5 :\vdash dual(!\mathtt{F}_4), \Delta_6}\ ?W}{- :\vdash ?\Gamma_3, \Delta_6}\ \mathtt{Cut} \quad \rightsquigarrow \quad \frac{\dfrac{}{- :\vdash \Delta_6}\ \mathtt{ax}}{- :\vdash ?\Gamma_3, \Delta_6}\ ?_G W$$

The elimination of *mCut* assumes H and the admissibility of W! (Th. 21), *e.g.*,

$$\frac{\dfrac{\mathtt{h}_2 :\vdash \mathtt{F}_3, ?\Gamma_4, !\mathtt{F}_5}{\bullet\mathtt{h}_2 :\vdash !\mathtt{F}_5, ?\Gamma_4, ?\mathtt{F}_3}\ ? \quad \dfrac{\mathtt{h}_7 :\vdash \Delta_8, dual(\mathtt{F}_5), ctr(\mathtt{n}_6, ?dual(\mathtt{F}_5))}{\bullet\mathtt{h}_7 :\vdash ctr(s\mathtt{n}_6, ?dual(\mathtt{F}_5)), \Delta_8}\ ?}{- :\vdash (?\Gamma_4, ?\mathtt{F}_3), \Delta_8}\ \mathtt{Cut} \quad \rightsquigarrow$$

$$\frac{\dfrac{\dfrac{}{- :\vdash ?\Gamma_4, !\mathtt{F}_5, ?\mathtt{F}_3}\ \mathtt{ax}}{- :\vdash ?\Gamma_4, \mathtt{F}_5, ?\mathtt{F}_3}\ W! \quad \dfrac{\dfrac{}{\bullet\mathtt{h}_2 :\vdash ?\Gamma_4, !\mathtt{F}_5, ?\mathtt{F}_3}\ \mathtt{ax} \quad \dfrac{}{\mathtt{h}_7 :\vdash \Delta_8, dual(\mathtt{F}_5), ctr(\mathtt{n}_6, ?dual(\mathtt{F}_5))}\ \mathtt{ax}}{- :\vdash ?\Gamma_4, \Delta_8, ?\mathtt{F}_3, dual(\mathtt{F}_5)}\ \mathtt{hCut}}{- :\vdash ?\Gamma_4, \Delta_8, ?\mathtt{F}_3}\ \mathtt{mCut}$$

Also, a generalization of $?_C$ on terms of sort `?MSFormula` is used here:

$$\frac{\vdash \Gamma, F : \Delta}{\vdash \Gamma : \Delta, ?F}\ ? \qquad \frac{\vdash \Gamma : F}{\vdash \Gamma : !F}\ ! \qquad \frac{\vdash \Gamma, F : \Delta, F}{\vdash \Gamma, F : \Delta}\ ?_C \qquad \frac{\vdash \Gamma : \Delta_1, A \quad \vdash \Gamma : \Delta_2, B}{\vdash \Gamma : \Delta_1, \Delta_2, A \otimes B}\ \otimes$$

Fig. 5. Some rules of the dyadic system DyLL ([13]).

$$\frac{\dfrac{h_2 :\vdash F_3, ?\Gamma_4, !F_5}{\bullet h_2 :\vdash !F_5, ?\Gamma_4, ?F_3}\ ? \qquad \dfrac{h_7 :\vdash \Delta_{10}, F_8, ctr(n_6, ?dual(F_5)) \quad h_7 :\vdash \Delta_{11}, F_9, ?dual(F_5)}{\bullet h_7 :\vdash ctr(sn_6, ?dual(F_5)), \Delta_{10}, \Delta_{11}, F_8 \otimes F_9}\ \otimes}{- :\vdash (?\Gamma_4, ?F_3), \Delta_{10}, \Delta_{11}, F_8 \otimes F_9}\ \text{Cut} \quad \rightsquigarrow$$

$$\frac{\dfrac{\dfrac{\bullet h_2 :\vdash ?\Gamma_4, !F_5, ?F_3}{}\ \text{ax} \qquad \dfrac{h_7 :\vdash \Delta_{10}, F_8, ctr(n_6, ?dual(F_5))}{}\ \text{ax}}{- :\vdash ?\Gamma_4, \Delta_{10}, F_8, ?F_3}\ \text{hCut} \qquad \dfrac{\dfrac{\bullet h_2 :\vdash ?\Gamma_4, !F_5, ?F_3}{}\ \text{ax} \qquad \dfrac{h_7 :\vdash \Delta_{11}, F_9, ?dual(F_5)}{}\ \text{ax}}{- :\vdash ?\Gamma_4, \Delta_{11}, F_9, ?F_3}\ \text{hCut}}{- :\vdash ?\Gamma_4, ?\Gamma_4, \Delta_{10}, \Delta_{11}, ?F_3, ?F_3, F_8 \otimes F_9}\ \otimes}{- :\vdash ?\Gamma_4, \Delta_{10}, \Delta_{11}, ?F_3, F_8 \otimes F_9}\ ?_{GC}$$

As explained in Sec. 8.1, arbitrary applications of contraction are problematic for proof search. Hence, in `LL/prop-cut-cc.maude`, $?_{GC}$ is introduced as a conditional rule that can be applied on a given multiset only if there is exactly one occurrence of it in the current sequent. As in the case of $?_{GW}$, the admissibility of $?_{GC}$ cannot be proved in `L-Framework`. Discharging the cases of *Cut* (resp., *mCut*) takes 30 s (resp. 124 s). □

Dyadic system for linear logic. Since formulas of the form $?F$ can be contracted and weakened, such formulas can be treated as in classical logic. This rationale is reflected in the syntax of the so called *dyadic sequents* of the form $\vdash \Gamma : \Delta$, interpreted as the linear logic sequent $\vdash ?\Gamma, \Delta$ where $?\Gamma = \{?A \mid A \in \Gamma\}$. It is then possible to define a proof system without explicit weakening and contraction (system DyLL in Fig. 5). The complete dyadic proof system for linear logic can be found in [13].

Theorem 23 *(Weak-height, weak-! and invertibility). If* DyLL $\blacktriangleright_n \vdash \Gamma : \Delta$, *then* DyLL $\blacktriangleright_{suc(n)} \vdash \Gamma : \Delta$ (H). *The following rules are height-preserving admissible:*

$$\frac{\Gamma, F, F : \Delta}{\Gamma, F : \Delta}\ \mathsf{C} \qquad \frac{\Gamma : \Delta}{\Gamma, F : \Delta}\ \mathsf{W} \qquad \frac{\vdash \Gamma : \Delta, !F}{\vdash \Gamma : \Delta, F}\ \mathsf{W!}.$$

All the rules of the system but $\oplus_i$, $\otimes$ and $?_C$ are height-preserving invertible.

Proof. For the admissibility results, see the files `DyLL/prop-H.maude`, `DyLL/prop-C.maude`, `DyLL/prop-W.maude` and `DyLL/prop-WB.maude`. The invertibility results are specified in `DyLL/prop-inv.maude`. H and the admissibility of C and W do not require any additional lemma. The admissibility of W! depends on H. The invertibility results rely on H, W, and W!. □

As in the system LL, cut-admissibility in DyLL requires mutual induction on two different rules. The rule *Cut*! below internalizes the storage of formulas marked with ? into the classical context.

Theorem 24 *(Cut-elim. and id-exp.). The following rules are admissible in* LL*:*

$$\frac{\vdash \Gamma : \Delta_1, F \quad \vdash \Gamma : \Delta_2, F^\perp}{\vdash \Gamma : \Delta_1, \Delta_2}\ Cut \qquad \frac{\vdash \Gamma : !F \quad \vdash \Gamma, F^\perp : \Delta}{\vdash \Gamma : \Delta}\ Cut!.$$

Moreover, for any formula F, the sequent $\vdash \cdot : F, F^\perp$ is provable.

Proof. The procedures are defined in the files `DyLL/cut-dyadic.maude` and, for *Cut*!, `DyLL/cut-dyadic-cc.maude`. The properties are specified in `DyLL/prop-cut.maude` and `DyLL/prop-cut-cc.maude`. Discharging the cases for *Cut* (resp., *Cut*!) takes 32.4 seconds (resp., 4 seconds). The admissibility of *Cut* (resp., *Cut*!) relies on H and W (resp., H, W and W!). □

Intuitionistic linear logic. The directory `ILL` contains the specification and analyses for intuitionistic linear logic (ILL). In this system, the multiplicative disjunction $\parr$ is not present and the linear implication $\multimap$ needs to be added (in classical LL, $F \multimap G$ is a shorthand for $F^\perp \parr G$). The resulting system is two-sided and single-conclusion. Formulas marked with ! on the left of the sequent can be weakened and contracted. The proof of cut-admissibility follows the same principles that the one for LL and the multicut rule below is required where $!\Gamma$ is a multiset of formulas marked with ! (specified as the sort `!MSFormula`)

$$\frac{!\Gamma \vdash !F \quad \Delta, (!F)^n \vdash G}{!\Gamma, \Delta \vdash G}\ mCut$$

$$\frac{\Gamma \vdash A}{\Gamma', \Box\Gamma \vdash \Box A, \Delta}\ \mathsf{k} \qquad \frac{\Gamma, \Box A, A \vdash \Delta}{\Gamma, \Box A \vdash \Delta}\ \mathsf{T} \qquad \frac{\Box\Gamma \vdash A}{\Gamma', \Box\Gamma \vdash \Box A, \Delta}\ \mathsf{4}$$

Fig. 6. The modal sequent rules for K (k) and S4 (k + T + 4).

8.5. Normal modal logics: K and S4

Modal logics extend classical logic with the addition of modal connectives (*e.g.*, the unary modal connective $\Box$), used to qualify the truth of a judgment, widening the scope of logical-conceptual analysis. The alethic interpretation of $\Box A$ is "the formula A is necessarily true". A modal logic is *normal* if it contains the axiom

$$(\mathsf{k})\ \ \Box(A \supset B) \supset \Box A \supset \Box B$$

and it is closed under *generalization*: if A is a theorem then so it is $\Box A$.

The smallest normal modal logic is called K, and S4 extends K by assuming the axioms $(\mathsf{T})\Box A \supset A$ and $(4)\Box A \supset \Box\Box A$. The sequent systems considered here for the modal logics K and S4 are extensions of G3cp with the additional rules for the modalities depicted in Fig. 6.

Structural rules. The admissibility of the structural rules follows as in Section 8.3.

Theorem 25 *(Structural rules). If* $\mathsf{K} \blacktriangleright_n \Gamma \vdash \Delta$, *then* $\mathsf{K} \blacktriangleright_{suc(n)} \Gamma \vdash \Delta$ (H). *Similarly for* S4. *Moreover, the rules* W_L, W_R, C_L, *and* C_R *are height-preserving admissible in both* K *and* S4.

Invertibility. As in G3cp, all the rules for the propositional connectives are invertible. Furthermore, T is invertible, but k and 4 are not (due to the implicit weakening).

Theorem 26 *(Invertibility). Only the rule* T *in Fig. 6 is invertible.*

Proof. See the specification in `K/prop-inv.maude` and `S4/prop-inv.maude`. H is required in this proof. Below, one of the trivial cases showing that the rule k is not invertible:

$$\frac{}{\bullet \mathrm{h}_2 : \Box\Delta_4, \Delta_5 \vdash \top, \Delta_3, \Box \mathrm{F}_1}\ \top_R \quad \rightsquigarrow \quad \frac{}{\bullet \mathrm{h}_2 : \mathit{unbox}(\Box\Delta_4) \vdash \mathrm{F}_1}\ \mathtt{fail}$$

Multisets of boxed formulas belong to the sort `MSBFormula` and the operation

`op unbox : MSBFormula -> MSFormula .` removes the boxes. □

Cut-admissibility and identity expansion. The proof of these properties for K and S4 uses the same infrastructure developed for G3cp.

Theorem 27 *(Cut-admissibility and ID-expansion). The cut-rule in Theorem 14 is admissible in both* K *and* S4. *Moreover, for any* F, *the sequent* $F \vdash F$ *is provable in* K *and* S4.

Proof. The specifications are in `K/prop-cut.maude` and `S4/prop-cut.maude`. In both cases, H is required as well as the invertibility of the propositional rules. □

Modal logic S5. Some extensions of the modal logic K do not have (known) cut-free sequent systems. In particular, consider the system S5, obtained by extending K with the axiom T and the rule below:

$$\frac{\Box\Gamma \vdash A, \Box\Delta}{\Gamma', \Box\Gamma \vdash \Delta', \Box A, \Box\Delta}\ \mathsf{45}$$

(see `KT45/KT45.maude`). Using the same strategy as in Theorem 27, the tool is able to discharge some of the proof obligations for cut-admissibility. However, some subcases involving the rule 45 and the other two modal rules (T and k) fail. Here, one example:

$$\dfrac{\dfrac{\dfrac{\mathrm{h}_1 : \Box\Delta_{13} \vdash \Box\Delta_{11}, \mathrm{F}_8, \Box \mathrm{F}_{10}}{\bullet \mathrm{h}_1 : \Box\Delta_{13}, \Delta_{14} \vdash (\Box\Delta_{11}, \Delta_{12}, \Box \mathrm{F}_8), \Box \mathrm{F}_{10}}\ A45 \quad \dfrac{\mathrm{h}_9 : \Box\Delta_{13}, \mathrm{F}_{10}, \Delta_{14}, \Box \mathrm{F}_{10} \vdash \Box\Delta_{11}, \Delta_{12}, \Box \mathrm{F}_8}{\bullet \mathrm{h}_9 : (\Box\Delta_{13}, \Delta_{14}), \Box \mathrm{F}_{10} \vdash \Box\Delta_{11}, \Delta_{12}, \Box \mathrm{F}_8}\ AT}{- : \Box\Delta_{13}, \Delta_{14} \vdash \Box\Delta_{11}, \Delta_{12}, \Box \mathrm{F}_8}\ \mathtt{Cut}}{- : \Box\Delta_{13}, \Delta_{14} \vdash \Box\Delta_{11}, \Delta_{12}, \Box \mathrm{F}_8}\ \mathtt{fail} \quad \rightsquigarrow$$

This case is clearly not provable: the cut-formula $\Box F_{10}$ is not decomposed in the left premise. Hence, cutting with F_{10} will not finish the proof. The only alternative is to reduce the height of the cut. The rule T cannot be applied (on F_{10}) on

the last sequent. If 45 is applied on the last sequent, either F_8 or one of the formulas in the boxed context Δ_{11} will lose the box. In both cases, none of the leaves of the left derivation can be used.

The failing cases are interesting, because they spot the point where the cut elimination procedure fails: if the goal is to propose a cut-free sequent system for a logic, certain shapes of rules should be avoided.

9. Concluding remarks

Related work. Tailoring adequate logical frameworks that are able to both, specify a representative class of logical systems and uniformly verify meta-level properties about the specified systems, is a challenge. Consider, for instance, the LF framework [27,28] based on intuitionistic logic. Since the left context is handled by the framework as a set, elaborated mechanisms for specifying sequent systems based on multisets are required. As a consequence, encodings become neither natural nor straightforward. One way of naturally specifying multisets is by adopting frameworks based on resource conscious logics. This is the case of LLF [29,30], grounded on (intuitionistic) linear logic. It turns out that linear logic is not adequate for naturally specifying contexts with different structural properties, as in the case of mLJ (Section 8.2). This can be partially fixed by adding subexponentials [31] to linear logic (*e.g.* as in SELL [32,33]). However, the resulting encodings are often non-trivial and difficult to automate. Moreover, since SELL is not adequate for specifying modalities behaving differently from its own, several logical systems cannot be naturally encoded in SELL, such as the modal sequent system K [34].

Combining the type system of LF [27] with deduction modulo theories leads to the $\lambda\Pi$-Calculus Modulo Theory [35]. Different logics and type systems can be expressed in this calculus, where connectives and quantifiers can be naturally defined with the aid of rewrite rules [36]. Dedukti [37] is an implementation of the $\lambda\Pi$-Calculus Modulo Theory and it can be used to check proofs produced by other systems. Different from what is pursued in this paper, no automatic proofs can be developed in that system.

Proof assistants (Coq, Lean, Isabelle/HOL, Agda, etc.), grounded on different formal systems (set theory, higher order logic or different type systems) provide a robust infrastructure to build proofs. These tools have been extensively used in the formalization of logical systems and their meta-properties. See for instance the work in [38] that combines shallow and deep embeddings to reason, in a general way, about sequent systems in Isabelle. Proof assistants are usually coupled with mechanisms (*e.g.*, tactic based languages) to interact with the user and automate routinary tasks. However, most of the proofs remain manual and the resulting procedure is usually tailored for some specific logic at hand. In this context, the `L-Framework` can be seen as a companion tool to quickly spot the problematic cases before facing a more time consuming proving task in a proof assistant.

A completely different approach is presented in [39], where cut-elimination is obtained by considering the relation between the cut-rule in Gentzen's system LK and the resolution rule for (propositional) classical logic. But, again, this method is restricted to systems having a (classical) semantics as a starting point.

All frameworks mentioned above have, at least, one common characteristic: the distance between the inference rules of the logical system and their specification as meta-level formulas. This justifies the fact that a framework is more or less adequate for specifying a specific class of logical systems. In [40], the web-based application Sequoia was proposed for specifying sequent calculi and performing basic reasoning about them. The specification is acquainted via translations of object-level trees into tree datatypes, using unification strategies. In this case, the specification distance is smaller, but proving meta-level properties becomes harder. Like Sequoia, the present work adopts the so-called ε-distance between the inference rules of the logical system and their specification as rewrite rules, using rewriting logic as a framework.

This paper presents many evidences that rewriting logic is an innovative and elegant framework for both specifying and reasoning in and about logical systems. Indeed, while approaches using logical frameworks depend heavily on the specification method and/or the implicit properties of the meta and object logics, rewriting logic enables the specification of the rules as they are actually written in text and figures [6,7]. Notions as derivability and admissibility of rules can be neatly formulated in the context of rewriting systems [41]. Moreover, rewriting modulo axioms greatly simplifies the definition of contexts in sequents, which would otherwise be represented as lists as it is the case in many proof assistants. As evidenced in the previous section, these features allow to achieve an impressive amount of automation in proofs.

Summary and future perspectives. Checking structural properties of proof systems is usually done via a case-by-case analysis, where all the possible combinations of rule applications in a system are exhausted. The advent of automated reasoning has changed completely the landscape, since theorems can nowadays be proved automatically in meta-logical frameworks (see *e.g.* [38,42]). This approach has brought a fresh perspective to the field of proof theory: useless proof search steps, usually singular to a specific logic, have been disregarded in favor of developing universal methods for providing general automation strategies. These developments have ultimately helped in abstracting out conceptual characteristics of logical systems, as well as in identifying effective frameworks that can capture (and help in reasoning about) them in a natural way.

This work moves forward towards such a direction: it proposes a general, natural, and uniform way of proving key structural properties of sequent systems by using the rewriting logic logical and meta-logical framework [7]. It ultimately enables modular and incremental proofs of meta-level properties of propositional sequent systems, both specified and mechanized in the language of Maude [8]. The approach builds on top of core algorithms that are combined and used for proving admissibility and invertibility of rules, as well as cut-admissibility and identity expansion of sequent systems.

The usual Gentzen's cut-elimination proof strategy can be summarized by the following steps: (i) transforming a proof with cuts into a proof with principal cuts; (ii) transforming a proof with principal cuts into a proof with atomic cuts; and (iii) transforming a proof with atomic cuts into a cut-free proof. While step (ii) is not difficult to solve (see, *e.g.*, [30]), steps (i) and (iii) can be problematic to mechanize. The results presented in this work suggest that such steps can be automated to a certain degree. Once the object system has been defined, the `L-Framework` automatizes most of the tasks: the proof obligations are generated and proof search (automatic) procedures strive at discharging such obligations. In the failing cases, the user needs to interact with the system by manually adding, in the module interfaces, the already proved theorems she considers necessary to complete the proof. The proofs in Section 8 make explicit the dependencies between the results and a possible venue for further automation emerges: add automatically the structural rules already proved (backward reasoning) and also all the already proved invertible rules (forward reasoning). This strategy will suffice to complete, automatically, the proof of cut-elimination for some of the systems in Section 8.

Other case studies can be certainly considered. Only mild adjustments to the proposed approach and algorithms are needed in order to reason about (multi-)modal [43] and paraconsistent [44] sequent systems. More interesting, it seems also possible to have an extension to handle variants of sequent systems themselves, like nested [45] or linear nested [2] systems. This ultimately widens the scope of logics that can be analyzed in the `L-Framework`, such as normal and non-normal modal logics [46]. It is also an interesting future research path to consider first-order sequent systems. In order to handle such systems, more research needs to be pursued. For instance, dealing with variable instantiation and abstraction requires special care. These problems, in the context of rewriting logic, have been studied in [47]. In this work, first-order sequent systems are mechanized, including proof-search heuristics at the meta-level.

A usual concern when a new sequent system is proposed is to implement it. Few implementational efforts have provided tools for emerging sequent systems for logics such as epistemic, lax, deontic, knotted, linear-modal, etc., logics. It would not require much effort to reuse some of the algorithms presented here to implement a procedure that, given the inference rules of a sequent system, outputs an implementation of such a system in Maude. More interestingly, using invertibility lemmas would also enable the generation of a weak-focus [13] version of the original systems, thus eliminating part of the non-determinism during proof-search. It should be noted, however, that this depends on a deeper investigation of the role of invertible rules as equational rules in rewriting logic. While this idea sounds more than reasonable, it is necessary to check whether promoting invertible rules to equations preserves completeness of the system (*e.g.*, the resulting equational theory needs to be Church-Rosser and terminating modulo the structural axioms of the operators). If the answer to this question is in the affirmative for a large class of systems, then the approach presented here also opens the possibility to the automatic generation of focused systems. These developments together with the advanced search mechanisms and strategies [48] in Maude, should provide a robust infrastructure for implementing proof search procedures and for testing different hypotheses/ideas without much modeling/programming effort.

The interaction of the user with the `L-Framework` is currently limited to modify the module interfaces and write properties using the meta-level representation of rewrite rules – which can be cumbersome. The tool can be leveraged to support interactive-theorem-proving in the lines of [49], and become a component of the Maude Formal Environment in [50]. Also, implementing new induction principles on lists/multisets may help in obtaining new automatic proofs (see Theorem 22).

A more ambitious endeavor would be to export the proof objects generated by the `L-Framework` to other proof assistants. This is far from being trivial precisely due to the differences in the representation of contexts and sequents in different theorem provers, and also to the use of unification algorithms modulo axioms for producing proof obligations. A promising starting point for building this bridge is the interoperability capabilities of the system Dedukti [51].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

The authors thank the anonymous reviewers for their detailed comments and suggestions that helped in improving this paper. The work of Olarte and Rocha was funded by the Minciencias (Ministerio de Ciencia Tecnología e Innovación, Colombia) project PROMUEVA (BPIN 2021000100160). The work of Pimentel has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101007627.

References

[1] G. Gentzen, Investigations into logical deduction, in: The Collected Papers of Gerhard Gentzen, North-Holland, 1969, pp. 68–131.

[2] B. Lellmann, Linear nested sequents, 2-sequents and hypersequents, in: H. de Nivelle (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods - 24th International Conference, TABLEAUX 2015, Wrocław, Poland, September 21-24, 2015, Proceedings, in: LNCS, vol. 9323, Springer, 2015, pp. 135–150.

[3] S. Burns, R. Zach, Relational hypersequents for modal logics, CoRR, arXiv:1805.09437 [abs], 2018.

[4] C. Olarte, E. Pimentel, C. Rocha, L-Framework, https://archive.softwareheritage.org/swh:1:rev:9aac4aa067420d353df24ce9ef8b4d440968c307, https://github.com/carlosolarte/L-framework/releases/tag/jlamp.

[5] J. Meseguer, Conditioned rewriting logic as a united model of concurrency, Theor. Comput. Sci. 96 (1) (1992) 73–155, https://doi.org/10.1016/0304-3975(92)90182-F.

[6] N. Martí-Oliet, J. Meseguer, Rewriting logic as a logical and semantic framework, in: Handbook of Philosophical Logic, Springer, Dordrecht, 2002, pp. 1–87.

[7] D.A. Basin, M. Clavel, J. Meseguer, Reflective metalogical frameworks, ACM Trans. Comput. Log. 5 (3) (2004) 528–576, https://doi.org/10.1145/1013560.1013566.

[8] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C.L. Talcott (Eds.), All About Maude - a High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic, LNCS, vol. 4350, Springer, 2007.

[9] C. Olarte, E. Pimentel, C. Rocha, Proving structural properties of sequent systems in rewriting logic, in: V. Rusu (Ed.), Rewriting Logic and Its Applications - 12th International Workshop, WRLA 2018, Held as a Satellite Event of ETAPS, Thessaloniki, Greece, June 14-15, 2018, Proceedings, in: LNCS, vol. 11152, Springer, 2018, pp. 115–135.

[10] N. Francez, R. Dyckhoff, A note on harmony, J. Philos. Log. 41 (3) (2012) 613–628, https://doi.org/10.1007/s10992-011-9208-0.

[11] A.S. Troelstra, H. Schwichtenberg, Basic Proof Theory, second edition, Cambridge Tracts in Theoretical Computer Science, vol. 43, Cambridge University Press, 2000.

[12] S. Negri, J. von Plato, Structural Proof Theory, Cambridge University Press, 2001.

[13] J. Andreoli, Logic programming with focusing proofs in linear logic, J. Log. Comput. 2 (3) (1992) 297–347, https://doi.org/10.1093/logcom/2.3.297.

[14] C.C. Liang, D. Miller, Focusing and polarization in linear, intuitionistic, and classical logics, Theor. Comput. Sci. 410 (46) (2009) 4747–4768, https://doi.org/10.1016/j.tcs.2009.07.041.

[15] N. Dershowitz, J. Jouannaud, Rewrite systems, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, Elsevier and MIT Press, 1990, pp. 243–320.

[16] R. Bruni, J. Meseguer, Semantic foundations for generalized rewrite theories, Theor. Comput. Sci. 360 (1–3) (2006) 386–414, https://doi.org/10.1016/j.tcs.2006.04.012.

[17] C. Rocha, J. Meseguer, Constructors, sufficient completeness, and deadlock freedom of rewrite theories, in: C.G. Fermüller, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010, Proceedings, in: LNCS, vol. 6397, Springer, 2010, pp. 594–609.

[18] P. Viry, Equational rules for rewriting logic, Theor. Comput. Sci. 285 (2) (2002) 487–517, https://doi.org/10.1016/S0304-3975(01)00366-8.

[19] J. Hendrix, J. Meseguer, Order-sorted equational unification revisited, Electron. Notes Theor. Comput. Sci. 290 (2012) 37–50, https://doi.org/10.1016/j.entcs.2012.11.010.

[20] F. Durán, S. Eker, S. Escobar, N. Martí-Oliet, J. Meseguer, C.L. Talcott, Built-in variant generation and unification, and their applications in Maude 2.7, in: N. Olivetti, A. Tiwari (Eds.), Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings, in: LNCS, vol. 9706, Springer, 2016, pp. 183–192.

[21] F. Durán, S. Eker, S. Escobar, N. Martí-Oliet, J. Meseguer, R. Rubio, C.L. Talcott, Equational unification and matching, and symbolic reachability analysis in maude 3.2 (system description), in: J. Blanchette, L. Kovács, D. Pattinson (Eds.), Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings, in: Lecture Notes in Computer Science, vol. 13385, Springer, 2022, pp. 529–540.

[22] J. Meseguer, Twenty years of rewriting logic, J. Log. Algebraic Methods Program. 81 (7–8) (2012) 721–781, https://doi.org/10.1016/j.jlap.2012.06.003.

[23] A. Ciabattoni, N. Galatos, K. Terui, From axioms to analytic rules in nonclassical logics, in: Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24–27 June 2008, Pittsburgh, PA, USA, IEEE Computer Society, 2008, pp. 229–240.

[24] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, J.F. Quesada, Maude: specification and programming in rewriting logic, Theor. Comput. Sci. 285 (2) (2002) 187–243, https://doi.org/10.1016/S0304-3975(01)00359-0.

[25] S. Maehara, Eine Darstellung der intuitionistischen Logik in der klassischen, Nagoya Math. J. (1954) 45–64.

[26] J. Girard, Linear logic, Theor. Comput. Sci. 50 (1987) 1–102, https://doi.org/10.1016/0304-3975(87)90045-4.

[27] R. Harper, F. Honsell, G.D. Plotkin, A framework for defining logics, J. ACM 40 (1) (1993) 143–184, https://doi.org/10.1145/138027.138060.

[28] F. Pfenning, Structural cut elimination: I. Intuitionistic and classical logic, Inf. Comput. 157 (1–2) (2000) 84–141, https://doi.org/10.1006/inco.1999.2832.

[29] I. Cervesato, F. Pfenning, A linear logical framework, Inf. Comput. 179 (1) (2002) 19–75, https://doi.org/10.1006/inco.2001.2951.

[30] D. Miller, E. Pimentel, A formal framework for specifying sequent calculus proof systems, Theor. Comput. Sci. 474 (2013) 98–116, https://doi.org/10.1016/j.tcs.2012.12.008.

[31] V. Danos, J. Joinet, H. Schellinx, The structure of exponentials: uncovering the dynamics of linear logic proofs, in: G. Gottlob, A. Leitsch, D. Mundici (Eds.), Computational Logic and Proof Theory, Third Kurt Gödel Colloquium, KGC'93, Brno, Czech Republic, August 24-27, 1993, Proceedings, in: LNCS, vol. 713, Springer, 1993, pp. 159–171.

[32] V. Nigam, E. Pimentel, G. Reis, An extended framework for specifying and reasoning about proof systems, J. Log. Comput. 26 (2) (2016) 539–576, https://doi.org/10.1093/logcom/exu029.

[33] V. Nigam, G. Reis, L. Lima, Quati: an automated tool for proving permutation lemmas, in: S. Demri, D. Kapur, C. Weidenbach (Eds.), Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014, Proceedings, in: LNCS, vol. 8562, Springer, 2014, pp. 255–261.

[34] C. Olarte, E. Pimentel, B. Xavier, A fresh view of linear logic as a logical framework, in: C. Nalon, G. Reis (Eds.), Proceedings of the 15th International Workshop on Logical and Semantic Frameworks with Applications, LSFA 2020, Online, September 15, 2020, in: ENTCS, vol. 351, Elsevier, 2020, pp. 143–165.

[35] G. Dowek, T. Hardin, C. Kirchner, Theorem proving modulo, J. Autom. Reason. 31 (1) (2003) 33–72, https://doi.org/10.1023/A:1027357912519.

[36] F. Blanqui, G. Dowek, É. Grienenberger, G. Hondet, F. Thiré, Some axioms for mathematics, in: N. Kobayashi (Ed.), 6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, in: LIPIcs, vol. 195, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Argentina, 2021, pp. 20:1–20:19.

[37] G. Hondet, F. Blanqui, The new rewriting engine of Dedukti (system description), in: Z.M. Ariola (Ed.), 5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference), in: LIPIcs, vol. 167, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 35:1–35:16.

[38] J.E. Dawson, R. Goré, Generic methods for formalising sequent calculi applied to provability logic, in: C.G. Fermüller, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010, Proceedings, in: LNCS, vol. 6397, Springer, 2010, pp. 263–277.

[39] R. Zach, Cut elimination and normalization for generalized single and multi-conclusion sequent and natural deduction calculi, Rev. Symb. Log. 14 (3) (2021) 645–686, https://doi.org/10.1017/S1755020320000015.

[40] G. Reis, Z. Naeem, M. Hashim, Sequoia: a playground for logicians - (system description), in: N. Peltier, V. Sofronie-Stokkermans (Eds.), Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II, in: LNCS, vol. 12167, Springer, 2020, pp. 480–488.
[41] C. Grabmayer, From abstract rewriting systems to abstract proof systems, CoRR, arXiv:0911.1412 [abs], 2009, arXiv:0911.1412.
[42] G. Reis, Facilitating meta-theory reasoning (invited paper, LFMTP'21), in: E. Pimentel, E. Tassi (Eds.), Proceedings of the Sixteenth Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, LFMTP 2021, Pittsburgh, USA, 16th July 2021, in: EPTCS, vol. 337, 2021, pp. 1–12.
[43] B. Lellmann, E. Pimentel, Proof search in nested sequent calculi, in: M. Davis, A. Fehnker, A. McIver, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings, in: LNCS, vol. 9450, Springer, 2015, pp. 558–574.
[44] O. Lahav, J. Marcos, Y. Zohar, Sequent systems for negative modalities, Log. Univers. 11 (3) (2017) 345–382, https://doi.org/10.1007/s11787-017-0175-2.
[45] K. Brünnler, Deep sequent systems for modal logic, Arch. Math. Log. 48 (6) (2009) 551–577, https://doi.org/10.1007/s00153-009-0137-3.
[46] B. Lellmann, E. Pimentel, Modularisation of sequent calculi for normal and non-normal modalities, ACM Trans. Comput. Log. 20 (2) (2019) 7:1–7:46, https://doi.org/10.1145/3288757.
[47] C. Rocha, J. Meseguer, Theorem proving modulo based on boolean equational procedures, in: R. Berghammer, B. Möller, G. Struth (Eds.), Relations and Kleene Algebra in Computer Science, 10th International Conference on Relational Methods in Computer Science, and 5th International Conference on Applications of Kleene Algebra, RelMiCS/AKA 2008, Frauenwörth, Germany, April 7-11, 2008, Proceedings, in: LNCS, vol. 4988, Springer, 2008, pp. 337–351.
[48] F. Durán, S. Eker, S. Escobar, N. Martí-Oliet, J. Meseguer, R. Rubio, C.L. Talcott, Programming and symbolic computation in Maude, J. Log. Algebraic Methods Program. 110 (2020), https://doi.org/10.1016/j.jlamp.2019.100497.
[49] M. Clavel, F. Durán, J. Hendrix, S. Lucas, J. Meseguer, P.C. Ölveczky, The Maude formal tool environment, in: T. Mossakowski, U. Montanari, M. Haveraaen (Eds.), Algebra and Coalgebra in Computer Science, Second International Conference, CALCO 2007, Bergen, Norway, August 20-24, 2007, Proceedings, in: Lecture Notes in Computer Science, vol. 4624, Springer, 2007, pp. 173–178.
[50] F. Durán, C. Rocha, J.M. Álvarez, Towards a Maude formal environment, in: G. Agha, O. Danvy, J. Meseguer (Eds.), Formal Modeling: Actors, Open Systems, Biological Systems - Essays Dedicated to Carolyn Talcott on the Occasion of Her 70th Birthday, in: Lecture Notes in Computer Science, vol. 7000, Springer, 2011, pp. 329–351.
[51] G. Dowek, Analyzing individual proofs as the basis of interoperability between proof systems, in: C. Dubois, B.W. Paleo (Eds.), Proceedings of the Fifth Workshop on Proof eXchange for Theorem Proving, PxTP 2017, Brasília, Brazil, 23–24 September 2017, in: EPTCS, vol. 262, 2017, pp. 3–12.