Notes on generalization of proof normalization

Marc Aiguier\textsuperscript{a,*}, Delphine Longuet\textsuperscript{b},
\textsuperscript{a}École Centrale Paris (ECP), MAS, Châtenay-Malabry, France
\textsuperscript{b}Centre for Mathematics and Computer Science (CWI), Dpt. of Software Technology, Amsterdam, Netherlands

Abstract

In this paper, we provide a general setting under which results of normalization of proof trees such as, for instance, the logicality result in equational reasoning and the cut-elimination property in sequent or natural deduction calculi, can be unified and generalized. This is achieved by giving simple conditions which are sufficient to ensure that such normalization results hold, and because syntactical, can be implemented. These conditions are based on basic properties of elementary combinations of inference rules which assure that the induced “global” proof tree transformation processes do terminate.

Key words: Formal systems; Proof tree transformation; Weak and strong normalization; Rewriting; Cut-elimination.

1. Introduction

In many situations in logic, to facilitate the use of a logical system, or to obtain consistency results for proof systems, or else to automatically prove theorems, one often uses search (construction) proof strategies. These strategies enable one to bound the search space for proofs to a given class of trees having a specific structure. One of the interests is to reduce proof search space (or even to make proof search feasible). This has been devised in various different contexts. Just a few examples: the so-called “logicality result”, which establishes a correspondence between derivability and convertibility in rewriting for many equational logics (sub-equational [38], mono-sorted [10], multi-sorted, conditional [29], partial [3], etc.); the cut-elimination result which shows that the cut rule is redundant for sequent and natural deduction calculi of many logics (classical first order [23], intuitionistic first order [30], some modal logics [39], linear [24], deduction modulo [19], etc.); the confluence property of rewrite systems which establishes that derivability can be proven by “valleys” for many logics dealing with transitive relations (equational [15,7], preorder [31,35], special relations [34]),\textsuperscript{1} or by using the Curry-Howard’s isomorphism, normalization results in typed $\lambda$-calculi as initiated by D. Prawitz [33]. More recently, the authors also used such proof normalization results to show the correctness of procedures based on axiom unfolding which enable us to generate test data sets from various specification formalisms [1,2,5,6].

In all these cases, the main difficulty is to show that the full derivability (i.e. without any specific proof strategy) coincides with the derivability restricted to a given class of proofs (i.e. with a specific proof strategy). Soundness of a given a strategy of proof search, which means that the restricted derivability is included into the full one, is obvious, because proofs resulting of such a strategy are particular instances of the general class of proofs. Completeness, which is expressed by the converse inclusion, is much more difficult. Indeed, it requires that for any theorem proven by a tree in the general class, there exists some proofs in the restricted one (i.e. built according to the considered strategy). In most logical systems, completeness is the consequence of a stronger result which consists in defining basic transformations to rewrite elementary combinations of inference rules (possibly by duplicating a sub-derivation), and showing that the global proof tree transformation which is naturally induced by these basic transformations are normalizing.\textsuperscript{2} For instance, concerning the logicality result for the classical equational logic, the basic transformations con-

\textsuperscript{*} Corresponding author.

\textit{Email addresses:} marc.aiguier@epc.fr (Marc Aiguier), delphine.longuet@cwi.nl (Delphine Longuet).

\textsuperscript{1} Besides, in the equational setting, G. Dowek has shown that the confluence of a rewrite system can be defined as the cut elimination property of a proof system associated to this rewrite system: asymmetric deduction modulo [16].

\textsuperscript{2} Our paper does not aim to generalize normalization by evaluation approach, the main idea of which is to use the semantics for the purpose of normalization as in [9].
sist in "distributing" substitution and context rules above transitivity, and removing all rules under reflexivity. In sequent calculus, the basic transformations consist, roughly speaking, in erasing the cut rule under axioms and moving it above all other rules, in order to eliminate it.

These basic transformations are usually tedious but easy to define. The difficulty is to show that the induced global normalization of proof trees are established for each underlying logical system in an ad-hoc way.

In this article, we unify and generalize the method of normalization of proof trees used in sequent calculus, natural deduction, equational reasoning, or term rewriting into an abstract setting of arbitrary logical systems. Abstraction is obtained by studying proof tree normalization in the abstract framework of formal systems. This then enables one to be logical system independent.

The article is organized as follows. In Section 2, we recall standard definitions and notations about formal systems, deductions and proof trees. In Section 3, we introduce the class of (proof tree) transformation procedures for which we will express the crucial (shared) arguments of many different normalization results as generic conditions on allowed transitions. In Section 4, we will formalize a first set of conditions that will enable us to prove a strong normalization result by using standard rewriting techniques for proving termination such as recursive path orderings (RPO). We will then show then that the cut-elimination procedure for the sequent calculus \( \text{LK} \) with implicit structural rules and the cut rule of the form \( \Gamma, \Delta \vdash \varphi \) does not satisfy this first set of conditions and then the underlying cut-elimination procedure is not strongly normalizing. A thorough study of reasons for which such a strong normalization result failed will enable us to set off limits of standard rewriting techniques for proving termination such as RPO. Then, we will give a new set of sufficient conditions that enable us to prove a weak normalization result. Finally, in Section 5, as an application of our generalization, we present a cut elimination algorithm for the sequent calculus modulo developed by Dowek, Hardin and Kirchner [17]. This result differs from the original one developed by [27,28] which gives a semantic proof of cut elimination whereas a syntactical proof is given in this paper. Other normalization results (logicality, Newman's lemma and cut-elimination for \( \text{LK} \)) that meet all the requirements given in this paper, can be found in the long version of this paper [4] as well as in our papers on test selection criteria for first-order and modal specifications [1,2,5,6] based on unfolding procedures. Indeed, in these papers, we showed correctness of our unfolding procedures (i.e. no potential test case is lost or added) by using a set of basic proof tree transformation whose the termination can be shown by using the basic conditions given in this paper.

2. Preliminaries

A formal system (a so-called calculus) \( S = (F,R) \) consists of a set \( F \) whose elements are called formulas, \(^4\) and a set \( R \) of \( n \)-ary relations on \( F \), called inference rules. Thus, a rule with arity \( n \geq 1 \) is a set of tuples \((\varphi_1, \ldots, \varphi_n)\) of formulas of \( F \). Each sequence \((\varphi_1, \ldots, \varphi_n)\) belonging to a rule \( r \) of \( R \) is called an instance of that rule with premises \((\varphi_1, \ldots, \varphi_{n-1})\) and conclusion \( \varphi_n \). It is usually written \( \langle \varphi_1, \ldots, \varphi_{n-1}, \varphi_n \rangle \). If \( n = 1 \), the instance is called an axiom and is written \( \varphi \). A deduction in \( S \) is a finite sequence \((\psi_1, \ldots, \psi_m)\) of formulas such that \( m \geq 1 \) and, for all \( i = 1, \ldots, m \), either \( \psi_i \) is an axiom or there is an instance \( \langle \psi_1, \ldots, \psi_{i-1}, \psi_i \rangle \) of a rule in \( S \) such that \( \varphi_n = \psi_i \) and \((\varphi_1, \ldots, \varphi_{n-1}) \subseteq \{\psi_1, \ldots, \psi_{i-1}\} \).

A theorem in \( S \) is a formula \( \varphi \) such that there exists a deduction in \( S \) with \( \varphi \) as last element. The existence of such a deduction is usually denoted by the meta-statement \( \vdash \varphi \). Instances of rules can also be composed to build proofs trees and proofs. Formally, a proof tree \( \pi \) in a formal system \( S \) is a finite tree whose nodes are labelled with formulas of \( F \) in the following way: if a non-leaf node is labelled with \( \varphi \) and \( \rightarrow \) then it has at least two children, the premise nodes are labelled (from left to right) with \( \varphi_1, \ldots, \varphi_{n-1} \), then \( \langle \varphi_1, \ldots, \varphi_{n-1}, \varphi_n \rangle \) is an instance of a rule of \( S \). \( \pi \) is called a proof when its leaves are axioms. Thus, proofs are another way to denote deductions of theorems in formal systems. Given a proof tree \( \pi \), note \( \mathcal{L}(\pi) \) (resp. \( \mathcal{L}(\pi) \)) the multiset \(^5\) (resp. the set \(^6\)) of leaves of \( \pi \). We will use the notation \( \{a_1, a_2, \ldots, a_n\} \) to denote a finite multiset. A proof tree \( \pi \) with root \( \varphi \) is denoted by \( \pi : \varphi \).

We write \( \pi = (\pi_1, \ldots, \pi_n) \), with \( n \in \mathbb{N} \), the proof tree whose last inference rule is \( i \), and a string \( \omega \) on \( \mathbb{N} \) which represents the path from the root of \( \pi \) to the subtree whose conclusion occurs at that position. This subtree is denoted by \( \pi_\omega \). Given a position \( \omega \in \mathbb{N}^* \) in a proof tree \( \pi \), \( \pi[\pi_\omega] \) is the proof tree obtained from \( \pi \) by replacing the subtree \( \pi_\omega \) by \( \pi' \). The trees \( \pi_\omega \) and \( \pi' \) necessarily have the same root. If \( \pi \) and \( \pi' : \varphi \) are two proof trees and \( \omega \) is a leaf position of \( \pi \) such that \( \pi_\omega = \varphi \), then we use the expression \( \pi \cdot \omega \cdot \pi' \).

---

\(^4\) Classically, \( F \) is a subset of \( A^* \) the set of all finite words on the alphabet \( A \). In this report, this condition will never be used. Therefore, it will not be considered.

\(^5\) In \( \mathcal{L}(\pi) \), all leaves of \( \pi \) are considered. We then have a multiset because several occurrences of a same formula can appear in \( \pi \).

\(^6\) Here, when a formula \( \varphi \) of \( \pi \) has several occurrences, only one is considered in \( \mathcal{L}(\pi) \).
rather than $\pi[\pi']|_\omega$. This operation is called composition of $\pi$ and $\pi'$ on leaf position $\omega$.

3. Proof transformation procedure

In all logical calculi where proof search strategies have been applied, the completeness of restricted derivability with respect to the full one is obtained by defining normalizing proof transformation procedures on the basis of elementary proof tree transformations. When we study most of these procedures, we can observe that they consist in replacing in proofs, some basic patterns, that we will call basic proof trees, of the form $(\iota_1, \ldots, \iota_n, \varphi_i)$, where each $\iota_i$ ($1 \leq i \leq n$) is either an instance of rule in $R$ or a formula in $F$, by proof trees in normal form (i.e. trees that are not reducible by other proof tree transformations). These basic patterns describe critical situation that do not respect the strategy because some rule instances $\iota_i$ ($1 \leq i \leq n$) should not be over $\iota$. For instance, underlying the logicality result for the equational logic, we have the transformation rule:

\[
\begin{align*}
\text{Trans.} & : \quad \frac{\tau_0 \Rightarrow \tau' \Rightarrow \tau'' \Rightarrow \tau'}{\tau(t)} = \sigma(t') \quad \Rightarrow \quad \text{Sub.} \quad \frac{\tau'' = \tau'}{\sigma(t(t')) = \sigma(t'(t'))}
\end{align*}
\]

As another example, we have in the cut-elimination result for sequent calculi:

\[
\begin{align*}
\text{Cut} & : \quad \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma, \varphi \Rightarrow \Delta} \quad \Gamma, \varphi \Rightarrow \Delta
\end{align*}
\]

And, in the cut-elimination result for natural deduction:

\[
\begin{align*}
\wedge\text{-elim} & : \quad \frac{\Gamma, \varphi \Rightarrow \Delta, \psi \Rightarrow \Delta'}{\Gamma \Rightarrow \Delta \wedge \Delta'}
\end{align*}
\]

Definition 3.1 (Basic proof tree) Let $S = (F, R)$ be a formal system. A basic proof tree is a proof tree of the form $(\iota_1, \ldots, \iota_n, \varphi)$, such that for every $i$, $1 \leq i \leq n$, either $\iota_i \in F$ or there exists $r \in R$ with $\iota_i \in r$.

Definition 3.2 (Transformation procedure) Let $S = (F, R)$ be a formal system. A (proof tree) transformation procedure for $S$ is a binary relation on proof trees $\Rightarrow$ such that for every $\pi \Rightarrow \pi'$, $\pi$ is a basic proof tree, $\pi'$ is a proof tree in normal form, $\mathcal{L}S(\pi') \subseteq \mathcal{L}S(\pi)$, and they have the same root.

Note $\Rightarrow_S$, the closure of $\Rightarrow$ under proof tree context and composition on leaf position.

By using the standard terminology available in rewriting, $\Rightarrow$ is strongly normalizing (or terminating) if every deduction sequence is finite, and weakly normalizing if every proof has a normal form.

Normalization of the “global” transformation procedure obtained by repeated application of transformation rules cannot be ensured without supplementary conditions. Moreover, there are some well-known examples of transformation procedures that meet all the requirements of Definition 3.2 but are weakly normalizing while some others are strongly normalizing (see the next section). In order to take into account the larger family of transformation procedures, in the next section, we will study conditions on basic transformation rules which are easy to check, yet powerful enough to ensure normalization.

4. Abstract normalization theorems

This section is devoted to (strong and weak) proof trees normalization theorems. Their proofs are abstract in the sense that they proceed on the structure of proof trees of any formal system.

A basic and powerful idea underlying most of termination methods such as recursive path ordering (RPO) consists in comparing terms of rewriting rules by first comparing their root symbols, and recursively comparing the collections of their immediate subterms. Therefore, start by assuming a well-founded ordering on atomic elements of proof trees (i.e. rule instances). Hence, given a transformation procedure $\Rightarrow$, a well-founded ordering $\preceq$ on $\bigcup_{r \in R} r$ is supplied. This order is often simple to define for most pairs of rule instances. Indeed, it is obvious to impose that every instance $\iota$ of any rule that has never to be underneath some instances $\iota'$ of other rules in proof trees, has a thinner weight for this order than $\iota'$. At times, this order is defined in a more ad-hoc way for the instances of a same inference rule. For instance, in the proof of the termination for the cut-elimination result, this order is then defined as follows:

\[
\forall \varphi \in \{\lor, \land, \exists\}, \quad \varphi_1 \preceq \varphi_2 \Rightarrow \varphi_1 \preceq \varphi_2
\]

and

\[
\begin{align*}
\Gamma_1 \Rightarrow \Delta_1, \varphi_1 \Rightarrow \Delta'_1 \quad & \Rightarrow \quad \frac{\Gamma_2 \Rightarrow \Delta_2, \varphi_2 \Rightarrow \Delta'_2, \varphi_1 \Rightarrow \Delta'_1}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta'_2}
\end{align*}
\]

where $|\varphi|$ is the depth of formula defined to be $\text{sup}_k(1+|\varphi_k|)$ if the $\varphi_i$ are the direct sub-formulae of $\varphi$.

In most examples of strong normalization results, transformation procedures $\Rightarrow$ consist more or less to “distribute” some rules over others in order to respect the search proof strategy (see for instance the above case of the rule $\text{Sub}$.}

\begin{align*}
\end{align*}
over the rule $Trans.$). Condition 1 generalizes this notion of distributivity of some rules over others.

**Condition 1** For each $(t_1, \ldots, t_n, \varphi) \leadsto \pi$ and for each $(\pi_1', \ldots, \pi_m', \varphi')$, subtree of $\pi$:  

(i) $\iota \succeq \iota'$ if each $\pi_i' \in F \cup \bigcup_{r \in R} r$  

\[ \iota \triangleleft \iota' \]  

(ii) If $\iota \sim \iota'$, then $\{(t_1, \ldots, t_n) \}\succ \{(t_1', \ldots, t_m')\}$ where $\succ$ extends $\triangleright$ to multisets on $F \cup \bigcup_{r \in R} r$.

Notice that Condition 1 is a particular case of RPO which can be easily implemented and then automatically checked given a well-founded ordering on rule instances. Obviously, we then have the following result:

**Theorem 4.1** Every transformation procedure $\leadsto$ that satisfies Condition 1 is strongly normalizing.

**Proof** Using proof terms for proofs, with a recursive path ordering $\triangleright^{\text{RPO}}$ to order proofs induced by the well-founded relation (precedence) $\succeq$ on rule instances, we show that $\leadsto \subseteq \triangleright^{\text{RPO}}$. Let $(t_1, \ldots, t_n, \varphi) \leadsto \pi$ be a transformation rule. By mathematical induction on $n$ we let us show that $(t_1, \ldots, t_n, \varphi) \triangleright^{\text{RPO}} \pi$.

**Basic case.** $\pi$ is the formula $\varphi$. According to Definition 3.2, $\mathcal{L}_\mathcal{S}(\pi) \subseteq \mathcal{L}_\mathcal{S}((t_1, \ldots, t_n, \varphi))$. Therefore, because reduction orderings have the subterm property, we conclude $(t_1, \ldots, t_n, \varphi) \triangleright^{\text{RPO}} \pi$.

**General case.** $\pi$ is of the form $(\pi_1, \ldots, \pi_n, \varphi)$. Here two cases have to be considered:

(i) $\iota \succeq \iota'$. But, by the induction hypothesis, for every $i$, $1 \leq i \leq n$, $(t_1, \ldots, t_n, \varphi_i) \triangleright^{\text{RPO}} \pi_i$, and then by RPO, $(t_1, \ldots, t_n, \varphi) \triangleright^{\text{RPO}} \pi$.

(ii) $\iota \sim \iota'$. By Condition 1.(i), each $\pi_i$ is in $F \cup \bigcup_{r \in R} r$.

\[ (1 \leq i \leq n) \]  

By Condition 1.(ii), we have $\{(t_1, \ldots, t_n)\} \succ \{(t_1', \ldots, t_m')\}$, and then $\{(t_1, \ldots, t_n)\} \triangleright^{\text{RPO}} \{(t_1', \ldots, t_m')\}$. By RPO, we then conclude $(t_1, \ldots, t_n, \varphi) \triangleright^{\text{RPO}} \pi$.

$\square$

Problems may occur with transformation rules of the form

\[ (t_1, \ldots, t_n, \varphi) \leadsto \pi \]

such that there exists $\pi' = (t_1', \ldots, t_m', \varphi')$ subtree of $\pi$ with $t_j' \in F \cup \bigcup_{r \in R} r$ for every $j$, $1 \leq j \leq m$, satisfying:

- $\iota \sim \iota'$, and
- $\mathcal{L}_\mathcal{M}(t_i') = \mathcal{L}_\mathcal{M}(t_i, \varphi_i)$

Indeed, such transformation rules may prevent from using the standard techniques to prove termination such as RPO, and then from obtaining a strong normalization result.

An example of such a transformation rule is Rule R1 defined in the cut-elimination procedure for the sequent calculus LK that considers the following cut rule:

\[ \Gamma \Rightarrow \Delta, \varphi \quad \Gamma' \Rightarrow \Delta' \]  

\[ \text{Cut} \]  

\[ \Gamma \Rightarrow \Delta \]

Indeed, with such a cut rule, we need weakening as explicit rule in order to allow such basic transformations:

R1

\[ \frac{\Gamma \Rightarrow \Delta, \varphi \Rightarrow \Delta' \text{Cut} \Rightarrow \Gamma' \Rightarrow \Delta'}{\Gamma' \Rightarrow \Delta} \]

The only measure that decreases with such a rule is the number of occurrences of rules $\@Left$ and $\@Right$ (i.e. rules that do not belong to Elim - see Definition 4.2) that occur above Cut in the right-hand side of the rule. Therefore by applying a strategy that eliminates, in proof trees, maximal proof trees (see the abstract definition of maximality in the proof of Theorem 4.3 just below) which are the nearest to leaves, this kind of measure will be sufficient to obtain a result of weak normalization.

**Definition 4.2** Let $\leadsto$ be a transformation procedure. Let us define the set $\text{Elim}$ as follow:

\[ \text{Elim} = \{ \iota \exists (t_1, \ldots, t_n, \varphi) \iota \leadsto \pi \} \]

Hence, $\text{Elim}$ contains rule instances we have to make go over or eliminate to respect the underlying strategy. For the cut elimination procedure, $\text{Elim}$ then contains all the rule instances of Cut, Weak, and Sub.

Now, we resume the above discussion by the following condition expressed in our abstract framework:

**Condition 2** For each transformation rule $(t_1, \ldots, t_n, \varphi) \leadsto \pi$:

(i) there exists $i$, $1 \leq i \leq n$, such that $i \notin \text{Elim}$

(ii) $\mathcal{L}_\mathcal{M}(\pi) \subseteq \mathcal{L}_\mathcal{M}((t_1, \ldots, t_n, \varphi))$

(iii) for every rule instance $\iota'$ occurring in $\pi$, $\iota \succeq \iota'$

(iv) if $\pi$ is of the form $(\pi_1, \ldots, \pi_m, \varphi)$, then every rule instance different of $\iota'$ in $\pi$ belongs to $\text{Elim}$.

Condition 2 being purely syntactical, can also be automatically checked, given a transformation procedure and a well-founded ordering on rule instances.

Now we arrive at the main theorem of this paper.

**Theorem 4.3** Suppose that $\leadsto$ satisfies Condition 2. Then $\leadsto$ is weakly normalizing.

**Proof** The theorem is obtained by generalizations of Tait’s proof [36].

The length of a proof $\pi = (\pi_1, \ldots, \pi_n, \varphi)$, noted $|\pi|$, is inductively defined to be $\sum_{\iota \in \text{Elim}} |\pi_\iota|$ if $\iota \notin \text{Elim}$ and $\sum_{\iota \in \text{Elim}} |\pi_\iota| + 1$ otherwise, if the $\pi_\iota$ are the direct subtrees of $\pi$, where $|\varphi| = 0$ if $\varphi$ is a leaf. Hence, the length of a proof $\pi$ is the number of rule instances that do not belong to $\text{Elim}$.

It is well-known that all well-founded sets are isomorphic to a unique ordinal. Note $\alpha : \left( \bigcup_{r \in R} r, \leq \right) \rightarrow \alpha$ where $\alpha$ is an ordinal, this isomorphism.

A proof $\pi = (\pi_1, \ldots, \pi_n, \varphi)$ is said maximal if and only if for every $i$, $1 \leq i \leq n$, $\pi_i$ is in normal form but $\pi$ is not.
Let $\pi = (\pi_1, \ldots, \pi_n, \varphi)_i$ be a maximal proof (i.e. this implies that $i \in E\text{lim}$). Then, Define the rank of $\pi$, noted $rk(\pi)$ to be $d(i) + |\pi|$.

To prove the theorem, start by proving the following lemma

**Lemma 4.4** Suppose that a formula $\varphi$ is provable and has a proof $\pi : \varphi$ which is maximal. Then, there is a proof $\pi' : \varphi$ in normal form.

The proof of the lemma is by induction on the rank of maximal proofs.

Let $\pi = (\pi_1, \ldots, \pi_n, \varphi)_i$ be a maximal proof and let $(\iota_1, \ldots, \iota_n, \varphi)_i \Rightarrow \pi'$ be a transformation rule that transforms $\pi$ into $\pi'$. Either $\pi'$ is in normal form or there exists a maximal subproof $\pi'' = (\pi_1', \ldots, \pi_m', \varphi')$ in $\pi$. By Conditions 2.(ii) and 2.(iv), no rule instance of $\bigcup r \in R \setminus \text{Elim}$ has been introduced in $\pi''$, and by Condition 2.(i) one of them has been removed. Therefore, we have $|\pi''| < |\pi|$. Now, by Condition 2.(iii), we have $d(i) > d(i')$. Therefore, we can write that $rk(\pi'') \leq rk(\pi) - 1$. Hence, by the induction hypothesis, every maximal subproof of $\pi'$ can be transformed into a normal proof. This leads to a new proof $\pi''$ that can contain some maximal subproofs. But, by following the same steps that for $\pi''$, we can conclude that the rank of these other maximal subproofs is lower than the rank of $\pi$. This enables us to conclude that $\pi$ can be transformed into a proof in normal form. This ends the proof of Lemma 4.4.

The proof of Theorem 4.3 is then continued by induction on the structure of proofs. The basic case is obvious because proofs are restricted to instances of axioms and then are in normal form.

Now, let $\pi = (\pi_1, \ldots, \pi_n, \varphi)_i$ be a proof which is not in normal form and not maximal. By the induction hypothesis, each $\pi_i$ can be transformed into a proof $\pi_i'$ in normal form. This leads to a proof $\pi' = (\pi_1', \ldots, \pi_n', \varphi)$. $\pi'$ is either in normal form or maximal. In the last case, by Lemma 4.4, $\pi'$ can be transformed into a proof $\pi' : \varphi$ in normal form. This ends the proof of Theorem 4.3.

In cut-elimination procedures for sequent calculi with explicit structural rules (i.e. contractions and weakening and then sequents are defined by lists rather than sets of formulæ), it is well-known that infinite reduction sequences can occur. Indeed, in a proof tree, when contractions are over cuts, this leads to the following transformation:

**R2**

$\frac{\Gamma, \varphi \vdash \Delta \quad \varphi, \Delta \rightarrow \Delta'}{\Gamma \vdash \Delta'}$

Notice that both Conditions 1 and 2 do not hold for **R2**. The reason is that the sequent $\Gamma \vdash \varphi, \Delta$ and Cut are duplicated, and Contr. $\prec$ Cut.

For such sequent calculus, some normalizing cut-elimination procedures have been developed. The first proof of strong normalization has been given by Dragalin in 1979 [21]. More recently, other proofs of strong normalization of cut-elimination procedures have been given in [11,12,26,37]. All of these strong results are obtained either by “camouflaging” contractions into introduction or cut rules, or by transforming the cut rule in order to prevent contractions on cut rules. For the latter, this is how J. Kahle and T. Skotowski in [11] enter upon strong normalization proofs for cut-elimination. The cut rule in [11], called Mix rule, is defined as follows:

$\frac{\Gamma, \varphi \vdash \Delta \quad \Gamma', \varphi' \vdash \Delta'}{\Gamma, \varphi \vdash \Delta', \varphi' \vdash \Delta'\text{Mix}}$

where $\varphi'$ means $\varphi, \ldots, \varphi$. For such a calculus, Rule R2 above is then transformed as follows:

$\frac{\Gamma, \varphi \vdash \Delta \quad \Gamma', \varphi' \vdash \Delta'}{\Gamma, \varphi \vdash \Delta', \varphi' \vdash \Delta'\text{Mix}}$

By using the precedence order:

$\forall \forall \in \{\land, \lor, \exists, \text{Contr.}, \text{Weak}\}, \text{Mix} \rightarrow \forall\text{Right}, \forall\text{Left}, \text{Axiom}$

We easily show that this cut-elimination procedure satisfies above Condition 1. Of course, by considering the Mix rule with the form $\frac{\Gamma, \varphi \vdash \Delta \quad \Gamma, \varphi \vdash \Delta'}{\Gamma, \varphi \vdash \Delta', \varphi' \vdash \Delta'\text{Mix}}$ leads to a weak normalization result.

5. Cut elimination in Deduction modulo

Recently, sequent calculi for first-order logic have been extended by G. Dowek, Th. Hardin and C. Kirchner [17] to some computations on formulæ. This extension is based on the fact that some axioms can be successfully replaced by rewrite rules on formulæ. This has permitted to have a faster proof-search and more readable proofs. The cut-elimination result given here differs from Hermant’s one [27,28]. Indeed, Hermant gives a semantic proof of cut elimination whereas a syntactical proof is given in this paper by defining an algorithm that transforms a proof into a cut-free one.

5.1. Sequent calculus modulo

First, let us introduce the formal system for the sequent calculus modulo for classical first order predicate logic. We restrict ourselves to the logical connectives ($\land, \lor$) and the quantifier $\exists$. It is well-known that other classical connectives and quantifiers can be defined from this restricted set. First, we recall the basic notions and notations of first-order logic and sequent calculus modulo [17].

In the case of first-order logic, a signature $\Sigma = (\mathcal{O}, P)$ contains two sets of operation names and predicate names, and each operation and predicate name is equipped with an arity in $N$. $\Sigma$-atoms are formulæ $p(t_1, \ldots, t_n)$ where $p \in P$ and every $t_i$ ($1 \leq i \leq n$) is a term in $\mathcal{T}_\varSigma(V)$. Well-formed $\Sigma$-formulæ are then either atoms or sentences of form $\neg \varphi$, $\varphi \lor \psi$, and $\exists x. \varphi$ where $\varphi$ and $\psi$ are well-formed $\Sigma$-formulæ and $x$ is a variable in $V$. The notions of free and bound
variable are defined as usual, and write $\varphi[x/t]$ for clash of variables-avoiding substitution of $t$ for $x$ in $\varphi$.

In the formulation of the sequent calculus modulo presented here, a $\Sigma$-sequent is any pair $\Gamma \Rightarrow \Delta$ where $\Gamma$ and $\Delta$ are two finite sets of well-formed $\Sigma$-formulae. In the following, we will write $\Gamma$, $\varphi$ and $\Gamma, \Delta$ to mean $\Gamma \cup \{\varphi\}$ and $\Gamma \cup \Delta$, respectively. Our choice to consider sets rather than multisets (or even lists) of formulae in the definition of sequents eliminates contraction and exchange as explicit structural rules which, as already shown in the last section, lead to infinite reduction sequences.

In the sequel, we suppose that the reader is accustomed with the elementary definitions of rewriting theory as found in the introductory chapters of textbooks on the subject [7,15]. We only introduced the following definition and notations:

**Definition 5.1.** Let $\Sigma$ be a signature. A formula rewrite rule is a pair of formulae $\varphi \rightarrow \psi$ such that $\varphi$ is a $\Sigma$-atom and all free variables of $\psi$ occur in $\varphi$. A rewrite system $R$ is a set of formula rewrite rules.

For a rewrite system $R$, we note $\rightarrow_R$ the rewriting relation induced by $R$, $\rightarrow_R$ the transitive and reflexive closure of $\rightarrow_R$ and $\equiv_R$ its transitive, reflexive and symmetrical closure.

Given a signature $\Sigma = (Op,P)$, the formal system $S = (F,R)$ for $\Sigma$ associated to Gentzen-style sequent calculi modulo is defined as follows:

- $F$ is the set of $\Sigma$-sequents
- $R$ contains all the instances of the following rule schemas:

Let $R$ be a set of formula rewrite rules

\[
\frac{\Gamma, \varphi \Rightarrow \Delta, \psi \text{ if } \varphi =_R \psi}{\Gamma, \varphi \Rightarrow \Delta, \psi} \text{ Ax.}
\]

\[
\frac{\Gamma, \varphi \Rightarrow \Delta, \psi \Gamma, \varphi' \Rightarrow \Delta \text{ if } \varphi \vee \varphi' =_R \psi}{\Gamma, \varphi \vee \varphi' \Rightarrow \Delta} \text{ vLeft}
\]

\[
\frac{\Gamma \Rightarrow \Delta, \psi \text{ if } \neg \varphi =_R \psi}{\Gamma \Rightarrow \Delta, \neg \psi} \text{ \negLeft}
\]

\[
\frac{\Gamma \Rightarrow \Delta, \varphi[x/t]}{\Gamma, \psi \Rightarrow \Delta} \text{ \existsLeft}
\]

\[
\frac{\Gamma \Rightarrow \Delta, \varphi, \psi \text{ if } \varphi =_R \psi}{\Gamma, \varphi \Rightarrow \Delta} \text{ \varRight}
\]

\[
\frac{\Gamma \Rightarrow \Delta, \neg \varphi \text{ if } \varphi =_R \psi}{\Gamma \Rightarrow \Delta, \neg \psi} \text{ \negRight}
\]

\[
\frac{\Gamma \Rightarrow \Delta, \exists x. \varphi \text{ if } \exists x. \varphi =_R \psi}{\Gamma, \psi \Rightarrow \Delta} \text{ \existsRight}
\]

\[
\frac{\Gamma \Rightarrow \Delta, \varphi \text{ if } \varphi =_R \psi}{\Gamma \Rightarrow \Delta, \varphi} \text{ \Cut}
\]

where the $\exists$Left rule obeys to the usual eigenvariable condition, stating that $x$ is not free in $\Gamma, \Delta$.

In the cut rule, the pair $(\varphi, \psi)$ is called the cut formula.

Finally, we use the standard terminology of principal formula with respect to a rule $r$ as follows: any $\psi$ such that $r \cdot \psi =_R \neg \varphi$ is principal with respect to $\neg$Left and $\neg$Right, $r \cdot \psi =_R \varphi \vee \varphi'$ is principal with respect to $\vee$Left and $\vee$Right, and

\[
\cdot \psi =_R \exists x. \varphi \text{ is principal with respect to } \exists \text{Left and } \exists \text{Right}.
\]

### 5.2. Cut-elimination algorithm

It has been shown in [20,28] that the cut elimination property for the sequent calculus modulo for the first-order logic 10 depends on the considered rewrite system. Indeed, rewriting systems with rewrite rules on formulæ define theories, and it is well-known that cut-elimination is not satisfied in general for theories. For instance, suppose the signature $\Sigma$ only composed of the constant $R$ and the binary predicate $\in$. Then, consider the theory defined by the confluent rewrite system $R$ composed of the unique rewrite rule: $R \in R \Rightarrow \neg R \in R$ which is Russell’s paradox. Obviously, we can prove the sequent $\vdash$ (which is a tautology for the sequent calculus modulo $R$):

\[
\frac{\vdash \neg R \in R}{\vdash \neg R \in R}
\]

**Exercise:** For a signature $\Sigma$ composed of the unique rewrite rule $R \in R$, is it possible to build a sequent calculus modulo $\Sigma$ that is cut-elimination free?

\[
\frac{\vdash R \in R}{\vdash \neg R \in R}
\]

It has been shown in [18] that the empty sequent $\vdash$ can also be proved and then the theory has not the cut elimination property. Hermant in [28] has then shown by semantical arguments that for every confluent rewrite system compatible with a well-founded order $\succ$ on formulæ 11 which satisfies the subformula property, the cut elimination property holds. Here, we show that this condition, also algorithmically entails the cut elimination property. We then assume a well-founded order $\succ$ on formulæ that have the subformula property:

- $\varphi_1 \vee \varphi_2 \succ \varphi_i$
- $\neg \varphi \succ \varphi$
- $\exists x. \varphi \succ \varphi[x/t]$

**Definition 5.2.** A rewrite system $R$ is said compatible with $\succ$ if and only if $\neg_R \subseteq \succ$.

Hence, a confluent rewrite system $R$ compatible with $\succ$, is strongly normalizing, and then the normal form of a formula $\varphi$ exists and is unique. In the following, we will note the normal form of $\varphi$, $\varphi_1$.

**Lemma 5.3.** (i) $(\varphi \vee \varphi_2)_1 = \varphi_1 \vee \varphi_2$

(ii) $(\@ \varphi)_1 = \@ \varphi_1$ where $\@ \in \{\exists x. \| \} \cup \{\neg\}$

**Proof** Result from the form of formula rewrite rules whose the left-hand side is a $\Sigma$-atom.

---

10 This is also true for the propositional and intuitionistic fragment. 11 and then terminating.
To give a syntactical proof of the cut-elimination property, we need to add to the set $R$ of inference rules the three following admissible rules:

$$\frac{\Gamma \Rightarrow \Delta}{\sigma(\Gamma) \Rightarrow \sigma(\Delta)} \text{ Sub.}$$

where $\sigma : V \rightarrow T_2(V)$ is a substitution and $\sigma(\Gamma) = \sigma(\varphi_1), \ldots, \sigma(\varphi_n)$ when $\Gamma = \varphi_1, \ldots, \varphi_n$.

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma' \Rightarrow \Delta'} \text{ Weak}$$

where $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$.

The intuitive reason of introducing the two first rules is that, in usual cut elimination proofs for calculi with implicit structural rules given in the literature [22,30], these two supplementary rules are replaced by intermediary lemmas, which are situated at a meta-theoretical level. Since we aim to see the cut-elimination procedure as a rewriting procedure, we do need to shift them at the object level. The reason of the third rule is because we are dealing with confluent rewrite systems compatible with a well-founded order $\succ$ on formulæ.

Let us call the resulting extended system of rules $R'$.

We will see just below that our method will be such that cut-free proofs with respect to $R'$ are indeed cut-free proofs using only rules in $R$.

**Lemma 5.4** $\varphi[x/t] = \varphi_1[x/t]$ 

**Proof** By induction on the structure of the formula $\varphi$. $\Box$

Therefore, Gentzen’s result (and, in particular, the termination of the cut-elimination procedure, which is the difficult part) can be shown by using the basic transformation rules described below. Such a set of transformation rules can be organized in four cases:

(i) Case where the cut formula is not principal with respect to at least one of the inferences leading immediately to the premises of the cut. Here, we give the transformations for the case where left rules are applied so as to get the left premise of the cut rule, but one can generalize such transformations to the symmetric case in a standard way.

$$\frac{\Gamma_1 \Rightarrow \Delta \varphi \quad \Gamma_1, \varphi \vdash \Delta_1}{\Gamma \vdash \Delta \varphi \quad \Gamma \vdash \Delta_1} \text{ Left} \quad \frac{\Gamma_1 \Rightarrow \Delta \varphi \quad \Gamma_1, \varphi \vdash \Delta_1}{\Gamma \vdash \Delta \varphi \quad \Gamma \vdash \Delta_1} \text{ Right}$$

where $@ \in \{v, \neg, \exists\}$ and $\Gamma_1' = \Gamma_1$ except for the $\neg$-Right rule, where $\Gamma_1' = \Gamma_1, \neg \psi$ for some formula $\psi$.

(ii) Case where the cut formula is principal with respect to both the inferences leading immediately to the premises of the cut. In this case we have the following basic proof tree transformations:

$$\frac{\Gamma \vdash \Delta_1 \varphi_1 \quad \Gamma \vdash \Delta_2 \varphi_2 \quad \varphi_1 \Rightarrow \varphi_2}{\Gamma \vdash \Delta_1 \varphi_1 \Rightarrow \varphi_2 \quad \varphi_1 \Rightarrow \varphi_2} \text{ Cut} \succ$$

$$\frac{\Gamma_1 \vdash \Delta_1 \varphi_1 \quad \Gamma_1 \vdash \Delta_2 \varphi_2 \quad \varphi_1 \Rightarrow \varphi_2}{\Gamma_1 \vdash \Delta_1 \varphi_1 \Rightarrow \varphi_2 \quad \varphi_1 \Rightarrow \varphi_2} \text{ Cut} \succ$$

This transformation is correct because we have respectively $\psi_1 = \neg \psi_2$, $\psi_1 = \neg \varphi_1 \lor \varphi_2'$ and $\psi_2 = \neg \varphi_2 \lor \varphi_2'$.

Therefore, we have $\varphi_1 \lor \varphi_2' = \neg \varphi_2 \lor \varphi_2'$, and then by Lemma 5.3 $\varphi_1 \lor \varphi_1' = \varphi_2' \lor \varphi_2'$, whence we conclude that $\varphi_1' = \varphi_2'$ and $\varphi_1' = \varphi_2'$.

$\frac{\Gamma \vdash \Delta \varphi \quad \varphi \Rightarrow \Delta_1}{\Gamma \vdash \Delta \varphi \Rightarrow \Delta_1} \text{ Cut} \succ$
\[ \frac{\Gamma \vdash \phi_1 \Rightarrow \Delta_1 \Gamma \Rightarrow \Delta_1 \text{Cut}}{\Gamma \Rightarrow \Delta_1} \]

where \(\Rightarrow\) is the extension of the order on formulae to multisets of formulæ.

\[ \{\phi_1, \psi_1\} \Rightarrow \{\psi_2, \phi_2\} \]

where \(\Rightarrow\) is the extension of the order on formulae to multisets of formulæ.

where \(\Rightarrow\) is the extension of the order on formulae to multisets of formulæ.

\[ \{\phi_1, \psi_1\} \Rightarrow \{\psi_2, \phi_2\} \]

\[ \frac{\Gamma \vdash \phi \Rightarrow \Delta, \psi_1 \Rightarrow \Delta \text{Left}}{\Gamma, \psi_1 \Rightarrow \Delta \text{Left} \iff \psi \Rightarrow \phi} \]

\[ \frac{\Gamma \vdash \phi \Rightarrow \Delta, \psi_1 \Rightarrow \Delta \text{Right}}{\Gamma, \psi_1 \Rightarrow \Delta, \psi \Rightarrow \Delta \text{Right} \iff \psi \Rightarrow \phi} \]

\[ \frac{\sigma(\Gamma), \sigma(\psi) \Rightarrow \sigma(\Delta), \sigma(\phi) \text{Sub.}}{\sigma(\Gamma), \sigma(\phi) \Rightarrow \sigma(\Delta), \sigma(\psi) \text{Sub.} \iff \psi \Rightarrow \phi} \]

\[ \frac{\Gamma, \psi \Rightarrow \Delta, \Delta \text{Weak}}{\Gamma \Rightarrow \Delta, \psi \Rightarrow \Delta \text{Weak} \iff \psi \Rightarrow \phi} \]

\[ \frac{\Gamma \Rightarrow \Delta, \Delta \text{Weak}}{\Gamma \Rightarrow \Delta, \psi \Rightarrow \Delta \text{Weak} \iff \psi \Rightarrow \phi} \]

By this well-founded order, we can easily check that the above transformation rules satisfy Condition 2. Therefore, by Theorem 4.3, the syntactical cut-elimination procedure presented here is weakly normalizing. By considering the cut rule with the form \(\frac{\Gamma, \psi \Rightarrow \Delta, \phi \Rightarrow \Delta}{\Gamma, \psi \Rightarrow \Delta, \phi \Rightarrow \Delta}\text{Cut}\) leads to a strong normalization result.

6. Conclusions and Perspectives

In this report, we have given a method to obtain results of normalization of proof trees independently of the underlying formal system. This has been achieved by generalizing the observation that some inference rules pass over or are canceled out when they occur under any other rules. This generalization is expressed by a set of basic proof tree transformations which, when the induced global proof transformation is terminating, ensures the completeness of the corresponding proof strategy. However, termination cannot be ensured in general. Therefore, two conditions on the structure of basic proof tree transformations have been given, which are sufficient to ensure such a result of termination. These two conditions allow us to obtain an abstract strong and an abstract weak normalization result, respectively.

In order to validate our approach, we plan to continue this work in at least two directions:

(i) Check “by hand” that we can indeed cover several already known normalization results, for instance: cut-elimination for “standard” modal propositional sequent calculi for the logics \(K, \text{K}_4, T, S_4\); cut-elimination for linear logic sequent calculus and/or proof-nets (when proof-nets are presented under the form of proof structure which are built according to the rules of linear sequent calculus as in [25]); cut-elimination for display calculi [8,14], etc.

(ii) Submit our “meta proofs” to a generic theorem proof assistant, as, for instance, Isabelle [32] or Coq [13], as it is done for a specific proof of strong normalization for the case of display calculi in [14]. This enables one to certify, as a result, all existing and future normalization results. Indeed, it would be only enough to check that the transformation rules of proof trees which underlie the normalization result meet the sufficient conditions of Section 4.

References


