Available online at www.sciencedirect.com
SCIENCE DIRECT®

ELSEVIER Electronic Notes in Theoretical Computer Science 125 (2005) 115–147

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Issues in the Analysis of Proof-Search Strategies in Sequential Presentations of Logics

Tatjana Lutovac [1] James Harland[2]

School of Computer Science and Information Technology, RMIT University GPO Box 2476V, Melbourne, 3001, Australia

Abstract

Many proof search strategies can be expressed as restrictions on the order of application of the rules of the sequent calculus. Properties of these strategies are then shown by permutation arguments, such as showing that a given strategy is complete by transforming an arbitrary proof into one which obeys the strategy. Such analyses involve some very tedious manipulations of proofs, and are potentially overwhelming for humans. In this paper we investigate the development of systematic techniques for the analysis of sequent calculi. We show how a particular specification of inference rules leads to a detailed analysis of permutation properties for these rules, and we also investigate how to detect redundancies in proofs resulting from these rules.

Keywords: Proof search, affine logic, linear logic, loop detection.

1 Introduction

There have been a variety of proof-theoretic techniques used to design and analyze proof-search strategies for theorem proving and logic programming [2,1,6,7,10,15,20]. One lesson that can be drawn from these various approaches is that it is usually insufficient just to find a proof; mostly, once a proof is found, it is desirable to extract information from the proof, such as identifying which strategy or tactic lead to success, recognizing structures common to

[1] Email: tlutovac@eunet.yu
[2] Email: jah@cs.rmit.edu.au

doi:10.1016/j.entcs.2005.01.002

other proofs, finding all proofs or all essentially different proofs, generating answer substitutions, minimizing unnecessary parts of the proof, and recognizing unused formulae. Hence proof-search is not simply a matter of determining whether a given sequent is provable, but of providing the appropriate "contextual" information about provability.

It is notable that many of the above analysis are all rather sophisticated and involve complex manipulations of proofs. Many are restricted to particular logic or classes of formulae. Almost all are designed for analysis on paper by a human and many of them are ripe for automation, being formally defined in precise detail, and yet somewhat overwhelming for humans.

In this paper we focus on the development of systematic techniques for the analysis of sequent proofs in order to extract useful information. In particular, we propose a more precise specification of sequent calculus inference rules that we use for the generalization and automation-oriented specification of the permutation process. Our work is motivated by the fact that there are some inference rules of interest which cannot be analyzed by the existing framework ([4,6,12]). One such situation is illustrated by the example below.

$$\frac{\vdash \phi \wp \psi, \Gamma}{\vdash \phi \wp \psi, \Gamma, ?\delta}$$

Whilst this is a combination of the $\wp$ and contraction rules for linear logic, similar rules can be found in LM, a multiple-conclusioned system for intuitionistic logic (see, for example, [22]) .

We also propose a mechanism for distinguishing between the necessary and unnecessary formulas in a proof. The mechanism is incorporated into the sequent rules and is independent of the search strategy used.

The results of our analysis can be implemented and utilized by means of an automated proof assistant. Our work is a contribution to a library of automatic support tools for extracting useful information about proofs in order to design and analyze proof-search strategies.

This paper is organized as follows. In Section 2 we discuss our motivations for using permutations as a tool for strategy analysis and for extending the existing framework. In Section 3 we give our specification of sequent calculus rules, and in Section 4 we show how this refined specification can be used to prove various properties of permutations. In Section 5 we address the issue of redundancy in proofs and in Section 6 we describe an algorithm for elimination of redundancies. Finally in Section 7 we present our conclusions.

2 Permutations and Strategy Analysis

2.1 Motivations

The *permutation* of two adjacent inference rules of a given proof is reversing their order in the proof but without disturbing the rest of the proof (the part above and below the inferences modulo duplication of some proof branches and renaming certain free variables) as a result of which we get an *equivalent* proof to the given one:

$$\dfrac{\dfrac{\vdots \atop \vdash a, c, d, \Gamma \qquad \vdots \atop \vdash b, \Delta}{\vdash c, d, a \otimes b, \Gamma, \Delta}\otimes}{\vdash c\wp d, a \otimes b, \Gamma, \Delta \atop \vdots}\wp \quad \Longleftrightarrow \quad \dfrac{\dfrac{\vdots \atop \vdash c, d, a, \Gamma}{\vdash c\wp d, a, \Gamma}\wp \qquad \vdots \atop \vdash b, \Delta}{\vdash c\wp d, a \otimes b, \Gamma, \Delta \atop \vdots}\otimes$$

Results from permutation analyses have a strong influence on the design of proof-search strategies. Many examples of such analyses can be found in the proof-search strategies defined in [2,1,6,7,10,15,20]. A key example of the relationship between the permutation properties and the execution model of the language is given by a comparison of Lygon[23] and Forum[16]; Lygon is based on the search strategy that *some* permutations of right-hand side rules will lead to a proof, whereas Forum is based on the search strategy that *any* permutation of right-hand rules will lead to a proof. Note that the strategy in question is defined by the restrictions placed on the order in which the inference rules may be applied.

For such strategies, permutation properties can be used to show that for a given set of inference rules either a given search strategy will find all possible proofs, or to construct an example of a provable sequent which cannot be proved with the given strategy. Strategies which satisfy the constraint on the order of inference rules may then be amenable to further analysis, such as minimizing the amount of branching in the proof, or delaying certain choices until the optimum amount of information is available.

Here we give a brief overview of some applications of permutations for the analysis of particular logic programming strategies.

Generating equivalent proofs which obey different strategies

A logic program for which there are many proofs of the same goal (and hence there are goals which return the same answer substitution many times) is generally considered to be somewhat deficient. Different proofs are generally only considered interesting if they lead to different answers.

For example, consider the sequent $p(a), \forall y\, q(y) \to p(y), q(b) \vdash \exists x p(x)$ in intuitionistic logic. All uniform proofs[3] can be classified into two groups, depending on the answer substitution for x. A representative of each class is below (the others are variations, depending on the order of application of the rules $\to$L, wL and $\forall$L).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{q(b)\vdash q(b)}\,Ax \qquad \dfrac{\dfrac{}{p(b)\vdash p(b)}\,Ax}{p(a),p(b)\vdash p(b)}\,wL}{p(a),q(b)\to p(b),q(b)\vdash p(b)}\,\to L}{p(a),\forall yq(y)\to p(y),q(b)\vdash p(b)}\,\forall L}{p(a),\forall yq(y)\to p(y),q(b)\vdash \exists xp(x)}\,\exists R \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{}{p(a)\vdash p(a)}\,Ax}{p(a),q(b)\vdash p(a)}\,wL}{p(a),\forall yq(y)\to p(y),q(b)\vdash p(a)}\,wL}{p(a),\forall yq(y)\to p(y),q(b)\vdash \exists xp(x)}\,\exists R$$

Clearly, the question of finding all proofs which lead to different answers have some overlap with the question of finding all equivalent proofs modulo inference permutations. Since proofs from the same equivalence class lead to the same answer, it is sufficient to generate just one of them, which then will be representative for the whole class.

Note also that some non-uniform permutations may also be useful. In the first example above, the reasons for choosing $x \leftarrow b$ are not obvious at the time the rule is applied:

$$\dfrac{\begin{array}{c}\vdots\\ p(a),\forall y\; q(y)\to p(y),q(b)\vdash p(b)\end{array}}{p(a),\forall y\; q(y)\to p(y),q(b)\vdash \exists xp(x)}\,\exists R$$

This substitution arises from the formulas $\forall yq(y) \to p(y)$, $q(b)$. If asked by a user to explain why that substitution was generated, an implementation could choose to produce the permutation below, which is not uniform, but demonstrates the origin of the substitution more directly, as indicated by the step $(*)$.

$$\dfrac{\dfrac{\dfrac{}{q(b)\vdash q(b)}\,Ax \qquad \dfrac{\dfrac{\dfrac{}{p(b)\vdash p(b)}\,Ax}{p(a),p(b)\vdash p(b)}\,wL}{p(a),p(b)\vdash \exists xp(x)}\,\exists R}{p(a),q(b)\to p(b),q(b)\vdash \exists xp(x)}\,\to L}{p(a),\forall y\; q(y)\to p(y),q(b)\vdash \exists xp(x)}\,\forall L \qquad (*)$$

[3] A uniform proof [17] of a sequent $\mathcal{P} \vdash G$ means a goal-directed proof, in a sense that the goal G is decomposed uniformly, based only on its structure and without reference to the program $\mathcal{P}$, until atomic goals are reached. The program is only consulted when atomic goals are to be proved.

Design of a new proof-search strategy

Permutation properties of the inference rules of a given logical fragment have a direct impact on defining efficient proof search strategies which deal with the order of inference rules. We explain this impact through the so-called *"normal proof"* [6] strategy from linear logic. In fact, we illustrate many ideas through examples from linear logic. Linear logic can be seen as a refinement of classical logic, in that there is a fragment of linear logic which has precisely the same properties as classical logic; at the same time however, linear logic contains features which are not present in classical logic. In essence, these features are due to removing the rules for contraction and weakening and re-introducing them in a controlled manner. Hence formulae have to be used exactly once in a linear proof. The unary connectives ? and ! allow a controlled application of weakening and contraction. Two different traditions for writing the sequent rule for classical conjunction result in two different conjunctions $\otimes$ and & and in two different disjunctions $\wp$ and $\oplus$.

A particular search strategy typically imposes constraints on the choice of formula to be decomposed in the next step of the proof construction. These constraints are based on the permutability of inference rules and reduces some sources of non-determinism during the proof construction. In a *bottom-up proof construction* (in which search starts at the root of the tree) this approach always first applies rules for which there is a "guarantee" that if the sequent is provable, there is a proof with that rule in that particular place in the proof tree. For instance, consider the sequent $\vdash A \otimes B, C \wp D, \Gamma$. During proof construction we must have a complete strategy for selecting the next formula to be decomposed. This selection is crucial for bottom-up proof construction. On the basis of the permutation results given in [6] we have the following table, where the case (t_1, t_2) in the table contains p iff inference t_1 *permutes down* over inference t_2, and, np iff there exists a proof in which inferences t_1 and t_2 are not permutable.

	t_1					
t_2	$\wp$	$\forall$	$\otimes$	$\oplus$	$\exists$	$w?$
$\wp$	p	p	np	p	p	p
$\forall$	p	p	p	p	np	p
$\otimes$	p	p	p	p	p	p
$\oplus$	p	p	p	p	p	p
$\exists$	p	p	p	p	p	p
$w?$	p	p	p	p	p	p

Table 1.

As a consequence of np for the case $(\otimes, \wp)$ and p for the case $(\wp, \otimes)$, for any provable sequent $\vdash A \otimes B, C \wp D, \Gamma$ there is a proof where $\wp$ is applied before (closer to the root than) the $\otimes$ rule, but not necessary the proof with $\otimes$ applied before $\wp$. For instance, the proof construction of the sequent $\vdash a \otimes b, a^\perp \wp b^\perp$ terminates successfully only when $\wp$ precedes $\otimes$:

$$\dfrac{\dfrac{\dfrac{}{\vdash a, a^\perp}Ax \qquad \dfrac{}{\vdash b, b^\perp}Ax}{\vdash a \otimes b, a^\perp, b^\perp}\otimes}{\vdash a \otimes b, a^\perp \wp b^\perp}\wp \qquad \dfrac{\dfrac{\dfrac{?}{\vdash a, a^\perp, b^\perp}}{\vdash a, a^\perp \wp b^\perp}\wp \qquad \dfrac{?}{\vdash b}\wp}{\vdash a \otimes b, a^\perp \wp b^\perp}\otimes \qquad \dfrac{\dfrac{?}{\vdash a}\wp \qquad \dfrac{\dfrac{?}{\vdash b, a^\perp, b^\perp}}{\vdash b, a^\perp \wp b^\perp}\wp}{\vdash a \otimes b, a^\perp \wp b^\perp}\otimes$$

while in a proof construction of the sequent $\vdash a \otimes b, a^\perp \wp ?b^\perp, b^\perp$,the order of applied rules does not matter:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash a, a^\perp}Ax}{\vdash a, a^\perp, ?b^\perp}w? \qquad \dfrac{}{\vdash b, b^\perp}Ax}{\vdash a \otimes b, a^\perp, ?b^\perp, b^\perp}\otimes}{\vdash a \otimes b, a^\perp \wp ?b^\perp, b^\perp}\wp \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash a, a^\perp}Ax}{\vdash a, a^\perp, ?b^\perp}w?}{\vdash a, a^\perp \wp ?b^\perp}\wp \qquad \dfrac{}{\vdash b, b^\perp}Ax}{\vdash a \otimes b, a^\perp \wp ?b^\perp, b^\perp}\otimes$$

Thus, in the proof construction of a sequent $\vdash A \otimes B, C \wp D, \Gamma$ the choice to decompose $C \wp D$ first has a greater chance to be successful *i.e.* to be the choice that lead to a proof.

Completeness of some strategies

We explain this on a simplified version of *"normal proof"* strategy whose full version is given in [6]. Let assume that, in the logical fragment we consider, the sequents are one-sided: $\vdash \Gamma$. Γ is a multiset of goal formulas, denoted G and defined by the following grammar: $G := A | G \otimes G | G \wp G | G \oplus G | \forall x G | \exists x G | ?G$

The strategy is defined by the following algorithm:

```
If $\vdash \Gamma$ contains a formula of the form $?F$
  then $\vdash \Gamma$ is the conclusion of $w?$ rule
  else
      if $\vdash \Gamma$ contains a formula whose top-most connective is $\wp$ or $\forall$
      then $\vdash \Gamma$ is the conclusion of rule which introduces that particular formula
      else $\vdash \Gamma$ contains a formula whose top-most connective is $\otimes$ or $\oplus$ or $\exists$ and
           $\vdash \Gamma$ is the conclusion of rule which introduces that particular formula
```

The completeness properties of the outlined strategy follow directly from the permutability properties of the concerned linear logic fragment, which are given in Table 1. Any proof, from this logical fragment, can be transformed, by permutations of its inferences, to a proof that obeys the given strategy:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash p(t), p^{\perp}(t)}\,Ax \qquad \dfrac{\dfrac{\dfrac{}{\vdash a, a^{\perp}}\,Ax}{\vdash a, a^{\perp}, ?b}\,w?}{\vdash a, a^{\perp}\wp ?b}\,\wp}{\vdash p(t)\otimes a, p^{\perp}(t), ?b\wp a^{\perp}}\,\otimes}{\vdash p(t)\otimes a, \exists y p^{\perp}(y), ?b\wp a^{\perp}}\,\exists}{\vdash \forall x(p(x)\otimes a), \exists y p^{\perp}(y), ?b\wp a^{\perp}}\,\forall \quad \longmapsto \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash a, a^{\perp}}\,Ax \qquad \dfrac{\dfrac{}{\vdash p(t), p^{\perp}(t)}\,Ax}{\vdash p(t), \exists y p^{\perp}(y)}\,\exists}{\vdash p(t)\otimes a, \exists y p^{\perp}(y), a^{\perp}}\,\otimes}{\vdash p(t)\otimes a, \exists y p^{\perp}(y), ?b, a^{\perp}}\,w?}{\vdash p(t)\otimes a, \exists y p^{\perp}(y), ?b\wp a^{\perp}}\,\wp}{\vdash \forall x(p(x)\otimes a), \exists y p^{\perp}(y), ?b\wp a^{\perp}}\,\forall$$

2.2 The Existing Framework

The problem of permutation analysis is not new ([4,12,6]). In this subsection we give a brief description of the terminology and definitions given in [6].

The *active* formulae of an inference are the formulae which are present in the premise(s), but not in the conclusion. The *principal* formula of an inference is the formula which is present in the conclusion, but not in the premise(s). Intuitively, the inference converts the active formulae into the principal formula.

When looking to permute the order of two inferences, it is necessary to check that the principal formula of the upper inference is not an active formula of the lower one. When this property occurs, the two inferences are said to be in *permutation position* (Definition 3.1 of [6].) We will refer to this as *GP permutation position.*

For example, consider the two inferences below:

$$\dfrac{\dfrac{\dfrac{q \vdash p, q, r}{q \vdash p, q\wp r}\,\wp R}{\neg p, q \vdash q\wp r}\,\neg L}{\neg p \otimes q \vdash q\wp r}\,\otimes L \qquad \longmapsto \qquad \dfrac{\dfrac{\dfrac{q \vdash p, q, r}{\neg p, q \vdash q, r}\,\neg L}{\neg p, q \vdash q\wp r}\,\wp R}{\neg p \otimes q \vdash q\wp r}\,\otimes L$$

In either inference, we have the following:

Rule	Principal Formula	Active Formulae
$\otimes L$	$\neg p \otimes q$	$\neg p, q$
$\neg L$	$\neg p$	p
$\wp R$	$\neg p \wp q$	$\neg p, q$

Note that in the left-hand subproof $\neg L$ and $\otimes L$ are not in permutation position, while $\neg L$ and $\wp R$ are in permutation position.

As illustrated by the following example, two adjacent rules in GP permutation position are not necessary permutable. In the left hand proof below the rules ? and ! are in GP permutation position but they are not permutable (the LL rule ! requires all non active formulae from the premise to be prefixed by !):

$$\dfrac{\dfrac{\vdash A, B, ?\Delta}{\vdash A, ?B, ?\Delta}\,?}{\vdash !A, ?B, ?\Delta}\,! \qquad \not\longrightarrow \qquad \dfrac{\vdash A, B, ?\Delta}{\times}\,!$$

Definition 2.1 (Definition 3.2 of [6]) (inference permutability)

An inference I is permutable over an inference J of a given proof $\begin{array}{c}\vdots\ \mathcal{D}\\ I\\ J\end{array}$ iff they satisfy the conditions:

$\mathcal{A}$. I and J are in the permutation position;

$\mathcal{B}_1$. J is applicable on the appropriate premise(s) of I;

$\mathcal{B}_2$. The conclusion-sequent of $\begin{array}{c}\vdots\ \mathcal{D}\\ J\end{array}$ can be a premise of I;

$\mathcal{C}$. Subproofs $\begin{array}{c}\vdots\ \mathcal{D}\\ I\\ J\end{array}$ and $\begin{array}{c}\vdots\ \mathcal{D}\\ J\\ I\end{array}$ have the same conclusion and the same hypothesis modulo duplication of some of them and renaming certain free variables.

In the other cases, we say that I is *not permutable over* J.

2.3 Some Problems With the Existing Framework

The existing techniques for permutation analysis have been adapted to particular sets of inference rules. There are many situations where the permutations can not be explained by the above definitions. In particular the above analysis of permutations via the notions of active and principal formulae is sometimes too coarse to capture when permutations are possible.

Also, the existing techniques are all designed for analysis on paper by a human and many of them are ripe for automation. Our aim is to make explicit and systematic permutation analysis of the existing sequent calculi and to generalize and extend the existing approach.

At first, it is our contention that a more refined syntax for sequent calculus rules is needed. There are some inference rules of interest which cannot be analyzed by the existing framework. Here we give a few examples.

Example 2.2 Consider the following rule from LJ as well as from a multiple-conclusioned LJ ([5]) calculus:

$$\frac{\Gamma, a, F \vdash G}{\Gamma, a, a \rightarrow F \vdash G} \rightarrow L$$

Clearly, we have the following:

Principal Formula	Active Formulae	Context
$a \rightarrow F$	F	Γ, G

But, what can be stated about the formula a? According to existing framework formula a is a context formula. But, it has a specific 'role' which should not characterize any context formula. Namely, the occurrence of a in the premise of the rule $\rightarrow L$ is necessary for the rule application and hence, a cannot be neither omitted nor freely replaced with another formula.

Another possibility is to interpret the occurrences of formula a in the premise and in the conclusion as active and principal formula, respectively. Under such an interpretation, for example, in the derivation below, among formulae $a \rightarrow F_1$, $a \rightarrow F_2$ the choice of formula to be decomposed next will not be arbitrary. This means that some possible permutations cannot be detected: the rule instances are not in a permutation position as a is both a principal formula of the upper $\rightarrow L$ inference and an active formula of the lower $\rightarrow L$ inference.

$$\cfrac{\cfrac{\begin{matrix}\vdots\\ \Gamma, a, F_1, F_2 \vdash G\end{matrix}}{\Gamma, a, a \rightarrow F_1, F_2, \Delta \vdash G} \rightarrow L}{\begin{matrix}\Gamma, a, a \rightarrow F_1, a \rightarrow F_2, \Delta \vdash G\\ \vdots\end{matrix}} \rightarrow L \quad \longleftrightarrow \quad \cfrac{\cfrac{\begin{matrix}\vdots\\ \Gamma, a, F_1, F_2 \vdash G\end{matrix}}{\Gamma, a, F_1, a \rightarrow F_2, \Delta \vdash G} \rightarrow L}{\begin{matrix}\Gamma, a, a \rightarrow F_1, a \rightarrow F_2, \Delta \vdash G\\ \vdots\end{matrix}} \rightarrow L$$

The intuition behind this is that a formula occurrence whose presence in a premise is necessary (but not sufficient) for the rule application, and which is copied unchanged into the conclusion should be considered and treated separately from the context, active and principal part of the rule. We call such formulae *quasi active* formulae.

A similar situation appears in the following example from a triple-context calculus TC [11] for intuitionistic linear logic with strong negation $ILL^{\sim}$ [11].

Example 2.3 Consider the rules:

$$\frac{\alpha, \Sigma\,;\,\Gamma\,;\,\Delta \;\vdash\; \gamma}{\delta, \Sigma\,;\,\Gamma\,;\,\Delta, !\sim\alpha \wedge \beta \;\vdash\; \gamma}\,\star \qquad\qquad \frac{\alpha, \Sigma\,;\,\Gamma\,;\,\sim\alpha, \Delta \;\vdash \gamma}{\alpha, \Sigma\,;\,\Gamma\,;\,\Delta \;\vdash \gamma}\,absorb2$$

where the rule $\star$ presents a mix of $\wedge L$, $sim!W$, and $\sim!move$ rules from TC calculus for $ILL^{\sim}$.

The formula α in the $absorb2$ rule cannot be considered as a *context* formula, as it is necessary for the rule application. It cannot belong to the active or principal part either, since in the left derivation below, the permutation position (and hence the permutation) will not be identified:

$$\frac{\dfrac{\begin{array}{c}\vdots\\ \varphi, \Sigma\,;\,\Gamma\,;\,\sim\varphi, \Delta \;\vdash\gamma\end{array}}{\varphi, \Sigma\,;\,\Gamma\,;\,\Delta \;\vdash\; \gamma}\,absorb2}{\varphi, \Sigma\,;\,\Gamma\,;\,\Delta, !\sim\varphi \wedge \beta \;\vdash\; \gamma}\,\star \quad\longleftrightarrow\quad \frac{\dfrac{\begin{array}{c}\vdots\\ \varphi, \Sigma\,;\,\Gamma\,;\,\sim\varphi, \Delta \;\vdash\gamma\end{array}}{\varphi, \Sigma\,;\,\Gamma\,;\,\sim\varphi, !\sim\varphi \wedge \beta, \Delta \;\vdash\; \gamma}\,\star}{\varphi, \Sigma\,;\,\Gamma\,;\,\Delta, !\sim\varphi \wedge \beta \;\vdash\; \gamma}\,absorb2$$

Example 2.4 Consider now an example from a sequent calculus for the logic of bunched implications (BI) [19]. Let us replace the left implication $-\!\!*\, L$ rule with the rule

$$\frac{\Gamma_1 \vdash \phi \qquad \Gamma_2, a \vdash a}{\Gamma_1, \Gamma_2, \phi -\!\!*\, a \;\vdash\; a}\; -\!\!*\, L'$$

(According to Lemma 3.7.4 of [19] the sequent system resulting from this replacement is equivalent to the original one.) Only the analysis of the $-\!\!*$ rule in *quasi active* terms (the occurrence of a in the succedent is a quasi active formula) enables the identification and justification of the following permutation:

$$\frac{\dfrac{\begin{array}{c}\vdots\\ \Gamma_1 \vdash \varphi\end{array} \qquad \dfrac{}{\Gamma_2, a \vdash a}\,Ax}{\Gamma_1, \Gamma_2, \varphi -\!\!*\, a \;\vdash\; a}\,-\!\!*\, L' \qquad \dfrac{}{\Gamma_3, a \;\vdash\; a}\,Ax}{\Gamma_1, \Gamma_2, \Gamma_3, \varphi -\!\!*\, a, a -\!\!*\, a \;\vdash\; a}\,-\!\!*\, L' \rightarrow \left\{\begin{array}{l} \dfrac{\begin{array}{c}\vdots\\ \Gamma_1 \;\vdash\; \varphi\end{array} \qquad \dfrac{\dfrac{}{\Gamma_2, a \vdash a}\,Ax \qquad \dfrac{}{\Gamma_3, a \vdash a}\,Ax}{\Gamma_2, \Gamma_3, a -\!\!*\, a, a \;\vdash\; a}\,-\!\!*\, L'}{\Gamma1, \Gamma_2, \Gamma_3, \varphi -\!\!*\, a, a -\!\!*\, a \;\vdash\; a}\,-\!\!*\, L' \\ \\ \text{or} \\ \\ \dfrac{\begin{array}{c}\vdots\\ \Gamma_1 \;\vdash\; \varphi\end{array} \qquad \dfrac{}{\Gamma_2, \Gamma_3, a -\!\!*\, a, a \;\vdash\; a}\,Ax}{\Gamma1, \Gamma_2, \Gamma_3, \varphi -\!\!*\, a, a -\!\!*\, a \;\vdash\; a}\,-\!\!*\, L' \end{array}\right.$$

Example 2.5 Consider now the rule $\wp'$, which is a combination of the $\wp$ and w rules from linear logic, and the permutation below:

$$\dfrac{\vdash \phi, \psi, \Gamma}{\vdash \phi\wp'\psi, \Gamma, ?\delta}\,\wp' \qquad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash ?A, ?A, B, \Gamma\end{array}}{\vdash ?A, B, \Gamma}\,c?}{\vdash ?A\wp' B, \Gamma, ?A}\,\wp' \quad \leftrightsquigarrow \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash ?A, ?A, B, \Gamma\end{array}}{\vdash ?A, ?A\wp' B, \Gamma, ?A}\,\wp'}{\vdash ?A\wp' B, \Gamma, ?A}\,c?$$

schema-rule

The rules $c?$ and $\wp'$ are permutable although they are not in a permutation position in the left subproof. The reason for this is that the formula $?A$ is at the same time the active and principal formula of the $c?$ rule and an active formula of the $\wp'$ rule.

Thus, in the left subproof above, the rule $\wp'$ consumes formula $?A$ which is a *quasi* active formula of the $C?$ rule and as such is present in the premise and available for the rule $\wp'$, after eventual inversion of the rules. Also, a very important detail here is the 'restoration' of formula $?A$ through the principal part of the $\wp'$ rule. So, after the inversion, $?A$ is available again for the $c?$.

Let us now concentrate on the above 'restoration'. Consider the current definition of a *principal* formula. Consider the current categorization and note the difference between the principal formula(e) of the rules shown in the following Table:

Logic	Rule	***Active***	***Principal***	***Context***
LL	$\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash ?A, \Delta}\,W?$		$?A$	Γ, Δ
LL	$\dfrac{\vdash \phi, \psi, \Gamma}{\vdash \phi\wp'\psi, \Gamma, ?\delta}\,\wp'$	$\phi,\ \psi$	$\phi\wp'\psi,\ ?\delta$	Γ
LJT multiple–conclusioned	$\dfrac{A, \Gamma \vdash B}{\Gamma \vdash A \supset B, \Delta}\,\supset_R$	$A,\ B$	$A \supset B,\ \Delta$	Γ
LL	$\dfrac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\,\otimes R$	$A,\ B$	$A \otimes B$	$\Gamma, \Delta, \Gamma', \Delta'$
LL	$\dfrac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A\wp B, \Delta}\,\wp R$	$A,\ B$	$A\wp B$	$\Gamma,\ \Delta$

Note that among principal formulae of the rules $W?$, $\supset_R$ and $\wp'$ there are formulae which are not result of a disappearance of some active formulae ($?A$, $?\delta$ and Δ respectively). It is our contention that such formulae should be distinguished from those principal formulae that are result of a conversion of certain active formulas. We call such formulae *extra* formulae. An *extra* formula can be used in a permutation (as we have seen in the last example), to 'restore' some formula(e) being already consumed.

In the next example we point out another reason for distinguishing *extra* from ordinary *principal* formulae. Namely, some impossible permutations can be 'resolved' if an admissible rule (a structural rule or an inversion) is allowed.

Example 2.6 Consider, once again, the rule

$$\frac{\Gamma \vdash \phi \supset \psi}{\Gamma, \phi \vdash \psi, \Delta} \supset_R$$

from LM, a multiple-conclusioned sequent calculi for intuitionistic logic, given in [22], and the impossible permutation in the left-hand subproof below.

$$\cfrac{\cfrac{A \vdash B}{\vdash A \supset B, \Delta, C} \supset R}{\neg C \vdash A \supset B, \Delta} \neg L \qquad \rightsquigarrow \qquad \cfrac{\cfrac{\cfrac{A \vdash B}{A \vdash B,\ C} wR}{\neg C, A \vdash B} \neg L}{\neg C \vdash A \supset B,\ \Delta} \supset R$$

Clearly, the rules on the left are not in the permutation position (the active formula of the lower rule is a principal formula of the upper rule). An analysis of these situation in the terms of *extra* formulae gives the following:

$$Extra_{\supset_R} = \{\Delta, C\}, \qquad Active_{\neg L} = \{C\}, \qquad \text{and} \quad Active_{\neg L} \subseteq Extra_{\supset_R}$$

The last relation indicates that impossible permutation can be overcome by inserting a weakening rule whose *extra* part is equal to $Active_{\neg L}$, as shown in the right subproof above.

It should also be noted that once we have *quasi active* formulae, this allows us to give a more natural (and arguably simpler) analysis of the contraction rule. Thus, for example, for the contraction rule (presented below in its form in linear logic) we will have the categorization as follows:

$$\frac{\underbrace{\Delta}_{Context} \vdash \underbrace{?F}_{active\ formula}, \underbrace{?F}_{quasi\ active\ formula}, \underbrace{\Gamma}_{Context}}{\underbrace{\Delta}_{Context} \vdash \underbrace{?F}_{quasi\ active\ formula}, \underbrace{\Gamma}_{Context}} C?$$

Thus the contraction rule now has one active and one quasi active formula, but no principal formula.

3 A Refined Structure of Sequent Rules

It is our contention that a more refined syntax for sequent calculus rules is needed, as it is proposed by the following definition (we assume, for simplicity, one-sided inference systems, i.e. that the antecedents are always empty):

Definition 3.1 The structure of sequent calculus inference rules:

- An *active formula* of rule I is a formula in a premise that does not exist in the conclusion.
- A *quasi-active formula* of an inference I is a formula occurrence whose presence in a premise is necessary (but not sufficient) for the rule application, and which is copied unchanged into the conclusion of the rule.[4]
- A *principal formula* of rule I is a formula occurrence of a conclusion that does not exist in premise(s) and that is a result of the disappearance of some active formulae of I.
- A formula occurrence in the conclusion of rule I, that does not exist in a premise and that is not a principal formula, is called an *extra formula.*
- The *active part* (denoted by $\mathcal{A}_I^i$) and the *quasi-active part* (denoted by $\mathcal{QA}_I^i$) of the i-th premise of an inference I are the (possibly empty) multisets of its active and quasi-active formulae, respectively. The *principal part* (denoted by $\mathcal{P}_I$) and the *extra part* of an inference I (denoted by $\mathcal{E}_I$) are the (possibly empty) multisets of its principal and extra formulae, respectively. The *context* of the i-th premise of an inference I (denoted by $Context_I^i$) is the (possibly empty) multiset - complement of its active and quasi-active part.

The above definition is illustrated through examples, from linear logic, in the Table below:

[4] We make an exception for the exchange rule. However, all of the examples used in this paper are for commutative logics, and hence we can consider antecedents and succedents as multisets, which obviates the need for this rule.

	$\mathcal{A}^1$	$\mathcal{A}^2$	$\mathcal{QA}^1$	$\mathcal{QA}^2$	$\mathcal{P}$	$\mathcal{E}$	$Context^1$	$Context^2$
$\dfrac{\vdash \Gamma \quad \vdash \Delta}{\vdash \Gamma, \Delta}\ mix$							Γ	Δ
$\dfrac{\vdash A, \Gamma \quad \vdash B, \Delta}{\vdash A \otimes B, \Gamma, \Delta}\ \otimes$	A	B			$A \otimes B$		Γ	Δ
$\dfrac{\vdash A, \Gamma \quad \vdash B, \Gamma}{\vdash A \& B, \Gamma}\ \&$	A	B			$A\&B$		Γ	Γ
$\dfrac{\vdash A, \Gamma \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\ cut$	A	$A^{\perp}$					Γ	Δ
$\dfrac{\vdash \Gamma}{\vdash \Gamma, ?F}\ w?$						$?F$	Γ	
$\dfrac{\vdash ?F, ?F, \Gamma}{\vdash ?F, \Gamma}\ c?$	$?F$		$?F$				Γ	

Note that we can categorise the binary rules according to how the contexts of the two rules are combined. We can identify *multiplicative* rules, where the context of each premise is copied unchanged into the conclusion (rules $\otimes, mix, cut$ in above table), and *strong additive* rules, where the context of each premise and of the conclusion are identical (rule &). Note that there are also *weak additive* rules where some part of the context of the premise and of the conclusion are identical.

4 Some applications of the refined sequent structure

As discussed above, the existing permutation analysis techniques do not cover all possible situations. Also, they are too implicit from the automation point of view. The proposed finer specification of sequent rules allows the development of more precise and more general notions connected with permutation analysis of sequent rules.

In the rest of this section we develop a more detailed, generalized and automation-oriented specification of a permutation process.

Remark 4.1 In the considerations below for any two adjacent inferences I and J, whenever J is a binary rule and directly follows I we will consider, unless otherwise stated, that I is above the left (*i.e.* first) premise $\vdash \mathcal{A}^1_J, \mathcal{QA}^1_J, Context^1_J,$ of J.

4.1 A More Precise Specification of Permutation Position

Definition 4.2 (weak and strong permutation position)
Two inferences I and J of a given proof Π are:

1. in the *weak permutation position* iff:
   i) J follows directly I in Π,
   ii) $\mathcal{A}_J, \mathcal{QA}_J \subseteq Context_I, \mathcal{QA}_I$ [5]
2. in the *strong permutation position* iff:
   i) J follows directly I in Π,
   ii) If I is a unary rule then
   $\mathcal{A}_J, \mathcal{QA}_J \subseteq Context_I, \mathcal{QA}_I$ and $(\mathcal{A}_J \not\subseteq Context_I) \Rightarrow (\mathcal{A}_J \backslash Context_I \subseteq \mathcal{E}_J)$
   else
   $\exists k \in \{1,2\}$ $\mathcal{A}_J, \mathcal{QA}_J \subseteq Context_I^k, \mathcal{QA}_I^k$ and $(\mathcal{A}_J \not\subseteq Context_I^k) \Rightarrow (\mathcal{A}_J \backslash Context_I^k \subseteq \mathcal{E}_J)$
   The value $k = 1$ (respectively $k = 2$) corresponds to *strong-left* (respectively *strong-right*) permutation position.

For example, in the first of the proofs below, the $\otimes$ and $\wp$ rules are in the weak permutation position only, because $\mathcal{A}_\wp, \mathcal{QA}_\wp = \{a^\perp, b^\perp\} \subseteq Context_\otimes, \mathcal{QA}_\otimes = \{a^\perp, b^\perp, ?c\}$ and the active formulae of the $\wp$ rule (formulae $a^\perp$ and $b^\perp$) belong to different premises of the $\otimes$ rule. In the second proof, the active formulas of the $\wp$ rule belong to same premise of the $\otimes$ rule, and hence the rules are, at the same time, in the weak and strong permutation position.

$$\dfrac{\dfrac{\vdash a, a^\perp \qquad \dfrac{\vdash b, b^\perp}{\vdash ?c, b, b^\perp}\,w?}{\vdash a \otimes b, a^\perp, ?c, b^\perp}\,\otimes}{\vdash b^\perp \wp a^\perp, a \otimes b, ?c}\,\wp$$

weak perm pos. for $\otimes$ and $\wp$

$$\dfrac{\dfrac{\vdash a, a^\perp \qquad \dfrac{\vdash b, b^\perp}{\vdash ?c, b, b^\perp}\,w?}{\vdash a \otimes b, a^\perp, ?c, b^\perp}\,\otimes}{\vdash b^\perp \wp ?c, a \otimes b, a^\perp}\,\wp$$

strong and weak permutation pos. for $\otimes$ and $\wp$

The strong permutation position takes into account more carefully the distribution of formulae from $\mathcal{A}_J, \mathcal{QA}_J$ between the multiset $Context_I, \mathcal{QA}_I$. The requirement that $\mathcal{A}_J, \mathcal{QA}_J \subseteq Context_I^k, \mathcal{QA}_I^k$ prevents the situation where active and/or quasi-active formulae of J belong to different premises of I. Such a situation can not be "recognized" in the weak permutation position:

[5] For a binary rule I $Context_I$ (respectively $\mathcal{QA}_I$) is the union of the $Context_I^k$ (respectively $\mathcal{QA}_I^k$) for each premise k.

i)

$$\dfrac{\dfrac{\vdash a, a^{\perp} \qquad \vdash b, b^{\perp}}{\vdash a \otimes b, a^{\perp}, b^{\perp}}\otimes}{\vdash a^{\perp}\wp b^{\perp}, a \otimes b}\wp$$

weak ($\not\Rightarrow$ strong)
not permutable rules

ii)

$$\dfrac{\dfrac{\vdash a, a^{\perp} \qquad \dfrac{\vdash b, b^{\perp}}{\vdash ?c, b, b^{\perp}}w?}{\vdash a \otimes b, a^{\perp}, ?c, b^{\perp}}\otimes}{\vdash b^{\perp}\wp ?c, a \otimes b, a^{\perp}}\wp$$

strong ($\Rightarrow$ weak)
permutable rules

iii)

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ \vdash A, B, C\end{matrix}}{\vdash A, B, C \oplus D}\oplus}{\vdash A\wp B, C \oplus D}\wp$$

strong ($\Leftrightarrow$ weak)
permutable rules

The condition $(\mathcal{A}_J \not\subseteq Context_I^k) \Rightarrow (\mathcal{A}_J \backslash Context_I^k \subseteq \mathcal{E}_J)$ ensures that the situations in which quasi-active formula(s) of upper inference I, which has(have) been "consumed" by lower J (*i.e.* used as the active formulae(s) for J) are acceptable (for the further permutation analysis) iff such formula(s) can be restored again through the extra part $\mathcal{E}_J$ (in that way it(they) remains available for I, after the eventual reversing of the order of I and J). As example, consider permutation from example 4 of subsection 2.3:

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ \vdash ?A, ?A, B, \Gamma\end{matrix}}{\vdash ?A, B, \Gamma}c?}{\vdash ?A\wp' B, \Gamma, ?A}\wp' \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{matrix}\vdots\\ \vdash ?A, ?A, B, \Gamma\end{matrix}}{\vdash ?A, ?A\wp' B, \Gamma, ?A}\wp'}{\vdash ?A\wp' B, \Gamma, ?A}c?$$

We use the notion of *strong permutation position* as a precondition for further checking of permutabilities of given rules. We could say that the notion of *strong permutation position* is dedicated to the analysis of possible permutations in a given proof *i.e.* when we have the concrete instances of a given inference rule. Recall that such permutations and inference movements in a given proof are essential for rearranging proofs to satisfy particular strategies. It also helps us to detect other proofs of a given sequent which differ only in the order of inference rules.

The notion of *weak permutation position* is suitable for recognition of non-permutable rules in a given proof tree (in regard to part 3. of Lemma 4.3 below), as well as for recognition of pairs of schema-rules which represent rules that can never be permutable.

The condition for *GP permutation position* corresponds to the condition $\mathcal{A}_J, \mathcal{QA}_J \subseteq Context_I$. In the absence of quasi-active formulae, the GP permutation position corresponds to the weak permutation position. Like the weak permutation position, the GP permutation position is more suitable for permutation analysis on the level of given set of schema-rules than in the analysis of possible permutations directly in a given proof.

We can establish the following dependencies among the different permutation positions:

Lemma 4.3 *For any two adjacent inference rules* I *and* J *of a given proof* Π*:*

1. *Strong permutation position* $\Rightarrow$ *weak permutation position.*
2. *Weak permutation position* $\not\Rightarrow$ *strong permutation position.*
3. I *is not in weak permutation position with* J $\Rightarrow$ I *is not permutable over* J*.*
4. GP *permutation position* $\Rightarrow$ *weak permutation position,*
5. GP *permutation position* $\not\Rightarrow$ *strong permutation position,*
6. *weak permutation position* $\not\Rightarrow$ GP *permutation position,*
7. *If* $\mathcal{QA}_J = \mathcal{QA}_I = \emptyset$ *then weak permutation position* $\Leftrightarrow$ GP *permutation position,*
8. *strong permutation position* $\not\Rightarrow$ GP *permutation position.*

Proof. 1., 4., 7. obvious from Definitions 2.1 and 4.2;
2. see above examples;
3. corollary of 1.;
5. see example i) above;
6., 8. see example 4 of subsection 2.3. □

Hence in order to study possible permutations, we adopt the Definition 2.1 but with condition $\mathcal{A}$. ('being in *permutation position*') changed to $\mathcal{A}^s$: I and J are in the '*strong permutation position*'.

4.2 Automating Permutations

We use the proposed characterisation of the structure of sequent calculi rules for the *automated-oriented specification* of the permutation process. Here we outline some results which have been described in detail and proved formally in[13].

- Our framework characterizes sets of inference rules, and is intended to be as general as possible. Whilst it is difficult to state with confidence that the permutation properties described here apply to arbitrary logical fragments, we believe that the classes of inference rules covered here includes all well-known sequent calculi and many lesser-known ones.
- The active, quasi-active, principal and extra part of an inference rule are not changed by its permutation up or down, in a given proof. The context part of an inference can change during inference permutation. The appropriate changes in the context have been specified in detail.

- Some "pre-conditions" for a permutation have been analysed in detail and restricted to particular parts of inference rules, with minimally sufficient tests.
- Some dependencies between the ways of permuting rules and their generic properties and the characteristic of a given proof have been specified and proved. For example, the *upward permutation* of a strong-additive rule requires a set of particular conditions. Also, the form of the permutation result depends on the generic properties of the rules in question and therefore can be determined independently of direct permutation.

We will explain some of our results on the example of a unary rule I and a strong-additive rule J :

$$\frac{\vdash\ \mathcal{A}_I,\ \mathcal{QA}_I,\ Context_I}{\vdash\ \mathcal{P}_I,\ \mathcal{QA}_I,\ Context_I,\ \mathcal{E}_I}\ I \qquad \frac{\vdash\ \mathcal{A}^1_J,\ \mathcal{QA}^1_J,\ Context_J \qquad \vdash\ \mathcal{A}^2_J,\ \mathcal{QA}^2_J,\ Context_J}{\vdash\ \mathcal{P}_J,\ \mathcal{QA}^1_J, \mathcal{QA}^2_J,\ Context_J,\ \mathcal{E}_J}\ J$$

Proposition 4.4 *After the permutation, the contexts of the permuted rules I and J are changed as follows:*

$$\begin{array}{rcl} Context_I & := & [Context_I, \mathcal{E}_J] \backslash \mathcal{A}^1_J,\ \ \mathcal{P}_J,\ \ \mathcal{QA}^2_J \\ Context_J & := & Context_J \backslash [\mathcal{P}_I, \mathcal{E}_I]\ ,\ \ \mathcal{A}_I \end{array}$$

Proof. Assuming the strong-left permutation position, the subproofs before and after the permutation would be respectively:

$$\dfrac{\dfrac{\vdash \mathcal{A}_I,\ \ \mathcal{QA}_I,\ \ Context_I}{\vdash \mathcal{P}_I;\ \ \underbrace{\underbrace{\mathcal{QA}_I}_{\mathcal{A}''_J, \mathcal{QA}''_J, q\mathcal{A}_I}\ \ ,\ \ \underbrace{Context_I}_{\mathcal{A}'_J, \mathcal{QA}'_J, \mathcal{C}_I},\ \ \mathcal{E}_{\mathcal{I}}}_{\vdash \underbrace{\mathcal{A}'_J, \mathcal{A}''_J}_{\mathcal{A}^1_J},\ \underbrace{\mathcal{QA}'_J, \mathcal{QA}''_J}_{\mathcal{QA}^1_J},\ \underbrace{\mathcal{P}_I, \mathcal{E}_I, \mathcal{C}_I, q\mathcal{A}_I}_{Context_J}}}\ I \qquad \vdash \mathcal{A}^2_J,\ \ QA^2_J,\ \ Context_J}{\vdash \mathcal{P}_J,\ \ \mathcal{QA}^1_J, \mathcal{QA}^2_J,\ \ Context_J,\ \ \underbrace{\mathcal{E}_J}_{\mathcal{A}''_J, e_J}}\ J$$

$$\dfrac{\vdash \underbrace{\mathcal{A}_I, \quad \underbrace{\mathcal{QA}_I,}_{\mathcal{A}''_J,\mathcal{QA}''_J,q\mathcal{A}_I} \quad \underbrace{Context_I}_{\mathcal{A}'_J,\mathcal{QA}'_J,\mathcal{C}_I}}_{\mathcal{A}^1_J,\ \mathcal{QA}^1_J,\ \mathcal{A}_I,C_I,q\mathcal{A}_I} \qquad \vdash \mathcal{A}^2_J,\ \mathcal{QA}^2_J,\ Context_J}{\vdash \underbrace{\mathcal{P}_J, \quad \underbrace{\mathcal{QA}^1_J}_{\mathcal{QA}'_J,\mathcal{QA}''_J} \ , \mathcal{QA}^2_J, \quad q\mathcal{A}_I, \mathcal{A}_I, \mathcal{C}_I, \quad \underbrace{\mathcal{E}_J}_{\mathcal{A}''_J,e_J}}} \ J$$

$$\dfrac{\vdash \mathcal{A}_I,\ \underbrace{\mathcal{A}''_J, \mathcal{QA}''_J, q\mathcal{A}_I,}_{\mathcal{QA}_I}\ \underbrace{\mathcal{QA}'_J, \mathcal{QA}^2_J, \mathcal{C}_I, \mathcal{P}_J, e_J}_{Context_I}}{\vdash\ \mathcal{P}_I,\ \ \mathcal{QA}_I,\ \ Context_I,\ \ \mathcal{E}_I,} \ I$$

The proof is straightforward from a simple inspection of distribution of formulae in the above subproofs. □

Theorem 4.5 *A unary rule* I *is permutable over a strong additive rule* J *iff they satisfy the conditions:*

i) I *and* J *are in the strong permutation position;*

ii) $\mathcal{P}_I = \emptyset, \quad \mathcal{E}_I = \mathcal{A}_I;$

iii) The multiset $[Context_I, \mathcal{E}_J] \backslash \mathcal{A}^1_J, \quad \mathcal{P}_J,\ \mathcal{QA}^2_J$ *can be (i.e. satisfies all restrictions for) the context of rule* I.

Proof. On the base of Proposition 4.4 and Definition 2.1, the following equivalences are straightforward:

$$\begin{aligned} \text{i)} &\iff \mathcal{A}^s \\ \text{iii)} &\iff \mathcal{B}_2 \end{aligned}$$

$\boxed{\Rightarrow:}$

As J is a strong additive rule, the context of the premisses must be identical. Thus, it must be: $Context_J = Context_J \backslash [\mathcal{P}_I, \mathcal{E}_I]\ ,\ \mathcal{A}_I \iff \mathcal{P}_I, \mathcal{E}_I = \mathcal{A}_I$.
As $\mathcal{P}_I \cap \mathcal{A}_I = \emptyset$, we have $\mathcal{P}_I = \emptyset, \quad \mathcal{E}_I = \mathcal{A}_I$.

$\boxed{\Leftarrow:}$

According to Proposition 4.4, after the permutation, the new context in the left premise of J would be $Context_J \backslash [\mathcal{P}_I, \mathcal{E}_I]\ ,\ \ \mathcal{A}_I$.
The condition ii) $\mathcal{P}_I = \emptyset, \quad \mathcal{E}_I = \mathcal{A}_I$ gives the following:
$Context_J \backslash [\mathcal{P}_I, \mathcal{E}_I]\ ,\ \ \mathcal{A}_I = Context_J$. That ensures the applicability of J on the appropriate premise(s) of I (condition $\mathcal{B}_1$ of Definition 2.1). The equality of the end-sequents (*i.e.* condition $\mathcal{C}$ of Definition 2.1) is trivially true (and can be checked directly). □ □

For example, consider the & rule from linear logic, which is a strong additive rule, and the *cw* rule, which is a combination of weakening and contraction:

$$\frac{\vdash \phi,\ \Sigma \qquad \vdash \psi,\ \Sigma}{\vdash \phi\&\psi,\ \Sigma}\ \& \qquad\qquad \frac{\vdash ?F,\ ?F,\ \Delta}{\vdash\ ?F,\ \Delta,\ G}\ cw$$

Consider the following transformation:

$$\dfrac{\dfrac{\vdash ?C, ?C, A}{\vdash ?C, A, ?C}\,cw \qquad \vdash B, ?C, ?C}{\vdash A\&B, ?C, ?C}\,\& \quad\Longrightarrow\quad \dfrac{\dfrac{\vdash A, ?C, ?C \qquad \vdash B, ?C, ?C}{\vdash A\&B, ?C, ?C}\,\&}{\vdash ?C, A\&B, ?C}\,cw$$

Note that instances of a rule which satisfy the condition *ii*) of Theorem 4.5 are actually redundant steps in a proof; in other words, all such potential permutations can be removed entirely from the proof. Note that in the above example, the *cw* rule is redundant (both before and after the permutation). Hence it is not necessary to permute these rules; we can simply eliminate the *cw* instance. If we adopt the strategy of *“eliminate redundant steps in a given proof prior to any other proof transformation”*, then as a consequence of Theorem 4.5, we come to the statement:
It is never possible to permute one of the unary rules over a strong-additive rule.

In other words, when such a combination is found, we should eliminate the redundant inference, rather than permute it.

As noted in Subection 4.2 of [6], there are some non-permutabilities which can be overcome by taking special cases into account. One such case is when the unary rule is not just above the left premise of the lower rule, but above both of them. The idea is to notice the subproof as shown on the left (if it exists) and to (try to) transform it into the right-hand subproof:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \alpha\end{array}}{\gamma_1}\,I \qquad \dfrac{\begin{array}{c}\vdots\\ \beta\end{array}}{\gamma_2}\,I}{\delta}\,J \qquad\longmapsto\qquad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \alpha\end{array} \quad \begin{array}{c}\vdots\\ \beta\end{array}}{\gamma}\,J}{\delta}\,I$$

We call such permutations *special permutations.*

As example, consider the following special permutation from linear logic.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash A, C, D\end{array}}{\vdash A, C\wp D}\,\wp \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash B, C, D\end{array}}{\vdash B, C\wp D}\,\wp}{\vdash A\&B, C\wp D}\,\& \quad\Longrightarrow\quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash A, C, D\end{array} \qquad \begin{array}{c}\vdots\\ \vdash B, C, D\end{array}}{\vdash A\&B, C, D}\,\&}{\vdash A\&B, C\wp D}\,\wp$$

In the following theorem we give the necessary and sufficient conditions for a special-permutation.

Theorem 4.6 *A unary rule I is special-permutable over a strong-additive rule J in a subproof*

$$\dfrac{\dfrac{\overset{\vdots}{\alpha}}{\gamma_1}\,I \qquad \dfrac{\overset{\vdots}{\beta}}{\gamma_2}\,I}{\delta}\,J$$

iff they satisfy the conditions:

i) *Each occurrence of I is in the strong-permutation position with J;*

ii) $Context_J \backslash [\mathcal{P}_I, \mathcal{E}_I]$, $\mathcal{A}_I$ *can be (i.e. satisfies all restrictions for) the context of rule J;*

iii) $[Context_I, \mathcal{E}_J] \backslash \mathcal{A}^1_J$, $\mathcal{P}_J$, $\mathcal{Q}\mathcal{A}^2_J$ *can be the context of rule I;*

iv) $Context_J \backslash [\mathcal{P}_{I_L}, \mathcal{E}_{I_L}]$, $\mathcal{A}_{I_L} = Context_J \backslash [\mathcal{P}_{I_R}, \mathcal{E}_{I_R}]$, $\mathcal{A}_{I_R}$

where I_L (respectively I_R) denotes the occurrence of I in the left (respectively in the right) premise of J.

Proof. Similar to proof of Theorem 4.5. □

4.3 Form of the permutation result

As mentioned above, the form of the permutation result can be determined independently of direct permutation in a given proof. For example, the *permutation result* for a binary rule J and a unary I can have, as shown below, different forms in regard to possible duplication of some proof branches or duplication of I. Let us denote their initial structure by F_0 and possible forms of permutation results respectively by F_1, F_2 and F_3:

$$Fo: \dfrac{\dfrac{\overset{\vdots}{\gamma_1} \quad \overset{\vdots}{\gamma_2}}{\beta}\,J}{\alpha}\,I \qquad F_1: \dfrac{\dfrac{\overset{\vdots}{\gamma_1}}{\beta_1}\,J \quad \overset{\vdots}{\gamma_2}}{\alpha}\,J \qquad F_2: \dfrac{\overset{\vdots}{\gamma_1} \quad \dfrac{\overset{\vdots}{\gamma_2}}{\beta_2}\,I}{\alpha}\,J \qquad F_3: \dfrac{\dfrac{\overset{\vdots}{\gamma_1}}{\beta_1}\,I \quad \dfrac{\overset{\vdots}{\gamma_2}}{\beta_2}\,I}{\alpha}\,J$$

The form of the permutation result can be determined in advance, before the direct, final permutation. As a consequence of Theorem 4.5 we have the following assertion:

Theorem 4.7 *If we eliminate redundant steps in a given proof prior to any other proof transformation (i.e. prior to any permutation), then a permutation of a strong additive rule J over a unary rule I results only in the permutation result of the form*

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \gamma_1\end{array}}{\beta_1}\,I \qquad \dfrac{\begin{array}{c}\vdots\\ \gamma_2\end{array}}{\beta_2}\,I}{\alpha}\,J$$

The result is appropriate for a process such as adapting proofs to obey certain strategies, identifying cases where this is not possible, minimizing the amount of branching in the proof, delaying certain choices until the availability of an optimum amount of information, and controlling the degree of non-determinism (if possible, not to duplicate rules or branches with high level of non-determinism).

4.4 Inference permutability

The proposed structure of inference rules allows the specification of some connections between the structure of rules and their permutability properties. Thus, for example, the following theorem holds:

Theorem 4.8 *Let J be a strong additive rule and I_1 and I_2 be unary rules. If rules I_1 and I_2 satisfy either of conditions:*

i) $\mathcal{P}_{I_1} = \mathcal{P}_{I_2}$ and $\mathcal{QA}_{I_1} = \mathcal{QA}_{I_2}$

ii) $\mathcal{P}_{I_1} = \emptyset$ and $\mathcal{E}_{I_1} = \mathcal{P}_{I_2}$ and $\mathcal{QA}_{I_1} = \mathcal{QA}_{I_2}$

then the rules I_k $k \in \{1, 2\}$ and J are not permutable.

As an illustration consider the strong additive rule $\&R$ and the unary rules $w?$ and $?R$ from linear logic:

$$\dfrac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A\&B, \Delta}\,\& \qquad \dfrac{\Gamma \vdash \Delta}{\Gamma \vdash ?F, \Delta}\,w? \qquad \dfrac{\Gamma \vdash F, \Delta}{\Gamma \vdash ?F, \Delta}\,?R$$

5 Redundancy Analysis of Sequent Proofs

As we have seen, permutation of inferences can be used to ‘improve’ a given proof (or proof fragment). As shown by the analysis of Theorem 4.5, this can also involve removing redundant inferences. In this section we study how some other forms of redundancy may be removed from a proof.

Success in a proof search is not simply a matter of finding a proof, but of providing more information about provability. Parts of a constructive proof may be irrelevant to the actual computation which results. It is also obvious that a more efficient implementation of proof search strategies demands the finer control and management of the formulae involved in the proof-search.

We consider the problem of detection of unnecessary parts of a sequent proof and elimination of redundant formulae that does not alter the search strategy applied. There are several potential benefits of such knowledge. For the logic programming strategies this knowledge is particularly useful when composing programs (and hence proofs), for debugging purposes and also for teaching purposes. For a given search strategy, we may thus consider this work as an initial requirements analysis of the properties of proofs.

Below we briefly illustrate some ideas about detection and elimination of unused formulae and redundant parts of a proof through examples in linear logic.

For example, it is straightforward to find a proof of a sequent containing $\top$, as such a sequent is an axiom in the linear sequent calculus. Consider the provable sequents $p \vdash q, \top$ and $p \vdash p, \top$. For the first sequent there is no meaningful information that can be extracted from the proof, apart from the presence of $\top$. For the later, $\top$ is clearly redundant, and hence it is useful to be able to deduce this.

Making use of past successful proof-search experience

Example 5.1 Let us consider the successful proof below. We denote by $\mathcal{P}$ and $\mathcal{G}$ respectively the antecedent and succedent part of the sequent $\underbrace{!s, r\wp p}_{\mathcal{P}} \vdash \underbrace{(?r\wp t) \wp (((\top \otimes p) \oplus q) \otimes s}_{\mathcal{G}}$

$$\dfrac{\dfrac{\dfrac{\overline{s \vdash s}\,Ax}{!s \vdash s}\,!L \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{r \vdash r}\,Ax \qquad \dfrac{\overline{p \vdash p}\,Ax \qquad \overline{\vdash \top, t}\,\top}{p \vdash p \otimes \top, t}\,\otimes R}{\dfrac{r\wp p \vdash r, t,\ \top \otimes p}{r\wp p \vdash ?r, t,\ \top \otimes p}\,?R}\,\wp L}{r\wp p \vdash ?r, t,\ ((\top \otimes p) \oplus q)}\,\oplus R}{r\wp p \vdash ?r\wp t,\ ((\top \otimes p) \oplus q)}\,\wp R}{!s, r\wp p \vdash ?r\wp t,\ ((\top \otimes p) \oplus q)\ \otimes\ s}\,\otimes R}{!s, r\wp p \vdash (?r\wp t) \wp (((\top \otimes p) \oplus q)\ \otimes\ s)}\,\wp R$$

As t and q are *unused* subformulae, both can be omitted or replaced with another formula. Hence we may think of the underlined parts of the formula $\mathcal{G} : \underline{(?r \wp} t) \underline{\wp (((\top \otimes p)} \oplus q) \underline{\otimes s)}$ as necessary parts of $\mathcal{G}$, in that the search process establishes that $(?r \wp ((\top \otimes p) \otimes s))$ succeeds, and hence deducing the success of $\mathcal{G}$. Furthermore, we can refine this process by omitting (redundant) constant $\top$ as well as connective ?, resulting in the formula $\mathcal{G}' = r\wp(p \otimes s)$. In this way an analyser could find a formula $\mathcal{G}'$ such that both $\mathcal{P} \vdash \mathcal{G}'$ and $\mathcal{G}' \vdash \mathcal{G}$ are provable. This may be thought of as calculating an *interpolant formula*

$\mathcal{G}'$ from $\mathcal{P}$ and $\mathcal{G}$. Note that the transformation from $\mathcal{G}$ to $\mathcal{G}'$ does not alter the search strategy used, in that the order of application of the rules is not changed:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{s \vdash s}\,Ax}{!s \vdash s}\,!L \qquad \dfrac{\dfrac{}{r \vdash r}\,Ax \qquad \dfrac{}{p \vdash p}\,Ax}{r \wp p \vdash r, p}\,\wp L}{!s, r \wp p \vdash r,\ p \otimes s}\,\otimes R}{!s, r \wp p \vdash r \wp (p \otimes s)}\,\wp R$$

This formula $\mathcal{G}'$ can be thought of as a representative of a family of formulae whose derivations, for the given formula $\mathcal{P}$, will require no effort to establish. For the above example, the obligatory part [6] of $\mathcal{G}$ is $r \wp (p \otimes s)$ while a general template for successful formulae based on $\mathcal{G}$ could be

$$(\,[?]\, r \wp [F]\,)\ \wp\ (\,(([\top] \otimes p) \oplus [Q]) \otimes s\,)$$

where F and Q are arbitrary formulae, and $[\,]$ denotes parts of the original formula $\mathcal{G}$ that can be omitted.

This knowledge allows later computations to make use of earlier work. So a proof-search strategy can retain the results of a previous successful search and to apply and combine them to a new situation.

Example 5.2 For a given proof, we distinguish unused formulae that can be freely eliminated from the proof and unused formulae whose elimination will cause the proof to 'crash'. For example, consider the proof Π on the left-hand side below. (Sub)formulae p, q and s are *unused*, but only s can be freely deleted from the proof while formulae p and q cannot be simultaneously eliminated. Note that p, q and s are subformulae of the active formula $(?q \wp ?p) \oplus s$ of the multiplicative rule $\otimes R$. Elimination of the whole formula $(?q \wp ?p) \oplus s$ will disable proof branching *i.e.* distribution of formulae across the multiplicative branches of the proof. Elimination of the subformula $?q \wp ?p$ will also lead to the unprovable sequent (on the right-hand side below).

$$\Pi :$$

$$\dfrac{\dfrac{}{t \vdash t}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{r \vdash r}\,Ax}{r \vdash ?p, r}\,?wR}{r \vdash ?q, ?p, r}\,?wR}{r \vdash ?q \wp ?p, r}\,\wp R}{r \vdash (?q \wp ?p) \oplus s, r}\,\oplus R}{r, t \vdash t \otimes ((?q \wp ?p) \oplus s), r}\,\otimes R \qquad \dfrac{?}{r, t \vdash t, r} \qquad \dfrac{t \vdash t \quad r \vdash s, r}{r, t \vdash t \otimes s, r}$$

So, we have that p, q and s are unused and that p and q cannot be simultaneously eliminated from the proof. For each unused atom we have three possibilities:

[6] i.e., the minimal information which must be present in $\mathcal{G}$

- to omit it from the proof;
- to leave the atom unchanged and
- to replace it with an arbitrary formula.

So proof Π can be thought of as a template for $(3^2 - 1) \cdot 3$ proofs (*i.e.* some variations of the given proof) which can be generated by alterations of p, q and s. All that proofs do not alter the search strategy used, in that the order of application of the rules is not changed.

Reducing the amount of work that must be done

Let us illustrate how to take advantage of the detection of unused formulae in one branch during a proof construction.

Example 5.3 Detection of unused formulae of a subproof can reduce the branching factor at some search node and hence enables the search strategy to immediately terminate the proof-search with success. For example consider the following situation:

$$\begin{array}{c} \dfrac{\overset{\Pi_1}{\Gamma_1 \vdash A, \Delta_1} \quad \Gamma_1 \vdash B, \Delta_1}{\Gamma_1 \vdash A\&B, \Delta_1}\ \&R \\ \vdots \\ \Gamma \vdash A\&B, \Delta \end{array}$$

If A is unused in the subproof Π_1 (and hence $\Gamma_1 \vdash \Delta_1$ is provable), an analyser could conclude (without any examination) provability for the sequents $\Gamma \vdash A, \Delta$ and $\Gamma \vdash \Delta$. Also, if B is *similar enough* to A, it could be concluded (on the basis of possible replacement of A with formula B and hence without examination of the right branch) that sequent $\Gamma_1 \vdash B, \Delta_1$ is provable too.

Example 5.4 It is not uncommon that a given logical fragment is simply too weak to directly support a number of features that programmers demand. To solve this problem some logic programming systems provides extra-logical language features, such as module systems and the dynamic predicates *assert/retract*. A common use of dynamic predicates is to temporarily assert information needed for a proof. For instance, suppose that the proof below involves *assert*'ing the fact, say S, while solving the left branch.

$$\dfrac{\dfrac{\dfrac{}{\boxed{\text{S}}, \Gamma, p \vdash p}\,Ax \qquad \overset{\Sigma}{\boxed{\text{S}}, \Gamma \vdash F}}{\boxed{\text{S}}, \Gamma, F \supset p \vdash p}\,\supset_L \qquad \dfrac{\dfrac{}{\Gamma, p \vdash p}\,Ax \qquad \overset{\Sigma_1}{\Gamma \vdash F}}{\Gamma, F \supset p \vdash p}\,\supset_L}{\Gamma, F \supset p \vdash p \wedge p}\,\wedge_R$$

If S is unused in the subproof Σ (and hence $\Gamma \vdash F$ is provable) we are confronted with the equivalent subproofs (Σ and Σ_1). The search strategy might recognize this situation and terminate the proof construction with success at the proof step denoted by $*?*$ below. If the proof Σ_1 is large then the savings may be considerable.

$$\dfrac{\dfrac{\dfrac{}{\boxed{\mathrm{S}}, \Gamma, p \vdash p}\,Ax \qquad \dfrac{\Sigma}{\boxed{\mathrm{S}}, \Gamma \vdash F}}{\boxed{\mathrm{S}}, \Gamma, F \supset p \vdash p}\supset_L \qquad \dfrac{\dfrac{}{\Gamma, p \vdash p}\,Ax \qquad \dfrac{*?*}{\Gamma \vdash F}}{\Gamma, F \supset p \vdash p}\supset_L}{\Gamma, F \supset p \vdash p \wedge p}\wedge_R$$

6 Automated detection of unused formulae

6.1 Related Work

The immediate inspiration for our algorithm was the work on labelled deduction for resource distribution by Harland and Pym [9], and Harland [8]. Their work actually presents a characterization of a range of strategies for distributing and selecting resources in linear sequent calculus proof-search. It is based on a sequent calculus annotated with Boolean constraints. Proof-search strategies are characterized by calculations of solutions of sets of Boolean equations generated by searches.

The idea used in [9,8], was to attach a Boolean expression to each formula in the proof, and to use the information thus recorded to keep track of the status of each formula, and in particular which choice of formula has been made. The values of Boolean expressions are typically defined by the leaves of the proof (*i.e.* the axioms) and the choice of rules made in the construction of the proof.

For example, appyling this technique [9,8], to Example 2.3 above gives the following labelling:

$$\dfrac{\dfrac{}{r[x_1], t[x_2] \vdash t, r[y_1]}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{r[\overline{x}_1], t[\overline{x}_2] \vdash r[\overline{y}_1]}\,Ax}{r[\overline{x}_1], t[\overline{x}_2] \vdash ?p[z], s[\overline{z}], r[\overline{y_1}]}\,?wR}{r[\overline{x_1}], t[\overline{x_2}] \vdash ?q[z], ?p[z], s[\overline{z}], r[\overline{y_1}]}\,?wR}{r[\overline{x}_1], t[\overline{x}_2] \vdash (?q\wp?p)[z], s[\overline{z}], r[\overline{y}_1]}\,\wp R}{r[\overline{x}_1], t[\overline{x}_2] \vdash (?q\wp?p) \oplus s, r[\overline{y}_1]}\,\oplus R}{r, t \vdash t \otimes ((?q\wp?p) \oplus s), r}\,\otimes R \qquad \longmapsto \qquad \begin{array}{c} y_1 = 0,\ z = 1 \\ x_1 = 0,\ x_2 = 1 \end{array}$$

$$\longmapsto \quad \dfrac{\dfrac{}{r[0],t[1] \vdash t, r[0]}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{r[1],t[0]\vdash r[1]}\,Ax}{r[1],t[0]\vdash ?p[1],s[0],r[1]}\,?wR}{r[1],t[0]\vdash ?q[1],?p[1],s[0],r[1]}\,?wR}{r[1],t[0]\vdash (?q\wp ?p)[1],s[0],r[1]}\,\wp R}{r[1],t[0]\vdash (?q\wp ?p)\oplus s, r[1]}\,\oplus R}{r,t\vdash t\otimes((?q\wp ?p)\oplus s), r}\,\otimes R$$

Note that by the technique being developed in [9,8] we can detect that s is unused in the above proof by inspecting the value of the Boolean variable upon completion of the search, and noting that it is not set to 1. Thus, we cannot conclude that formulae $?p$ and $?q$ are not used, and that cannot be simultaneously eliminated from the proof.

6.2 Automated Partial Redundancy Elimination

We present an algorithm for *partial* elimination of redundant formulae (Algorithm PRE) from a given proof. Our intention is not to find all different proofs of a given sequent but to generate all the concrete simplifications which are instances of a generated proof. By partial elimination of redundant formulae we mean:

- elimination independent of the search strategy used;
- elimination which does not alter the search strategy applied;
- no additional proof search *i.e.* redundant formulae remaining in the resulting proof cannot be eliminated without additional proof search;
- preserving the multiplicative branching structure of the proof.

Our technique is independent of proof-search strategy used and implies a comparison of sets of atoms from the root-sequent and from the leaves of a proof tree as well as supervision of (only) those formulae which in some sense 'preserve' multiplicative branching of the proof.

We will briefly, informally explain our approach on the propositional fragment of linear logic excluding contraction and binary additive rules. These rules may be incorporated into what follows without much difficulty. Also, in the rest of this section we restrict our attention to detection and elimination of unused formulae that appear in the succedent only.

We will briefly, informally explain *Algorithm PRE* on the Example 2.3 above.

1. **At first, the sequent to be proved has to be rectified so that distinct occurrences of a same atom have distinct indices**: $r, t \vdash t_1 \otimes ((?q\wp ?p)\oplus s), r_1$. The set of atoms at the root sequent is $\mathcal{A} = \{r, t, t_1, q, p, s, r_1\}$.
2. **Keeping track of formulae being relevant for proof branching.**

Our sequents assume the form

$$A_1^{x_1},\ A_2^{x_2},\ \ldots\ A_k^{x_k}\ \vdash\ B_1^{y_1},\ B_2^{y_2},\ \ldots B_n^{y_n}\ -\mathcal{H},$$

where superscript labels $x_1, \ldots x_k, y_1, \ldots y_n$ are variables and $\mathcal{H}$ is the set whose elements are sets of formulae $\{F_1, F_2, \ldots F_n\}$, $n \geq 1$. As search proceeds, we record in $\mathcal{H}$ only formulae which are occurred and relevant for the branching on the current path.

On the successful completion of the search, each set from the $\mathcal{H}$ place the constraints on the elimination of the corresponding formulae: formulae recorded in the same set cannot be eliminated simultaneously in order to preserve proof branching.

We use superscripts to tag formulae which are currently recorded in $\mathcal{H}$. Formulae recorded in the same set $\{F_1, F_2, \ldots F_n\}$ are labelled with the same superscript. A formula already recorded will be replaced with its subformulae encountered (if any) during a search.

The tag ϵ denotes that there is no restriction on elimination of the corresponding formula. At the beginning, all formulae of a sequent to be proved are labelled with ϵ.

Informally, the algorithm includes simultaneous (bottom-up) construction of a proof tree and maintaining of labels and set $\mathcal{H}$.

Below we give the specification for those rules in linear logic which may change the labels (of succedent's formulae) and/or set $\mathcal{H}$. All other rules propagate up labels and $\mathcal{H}$ unchanged. For simplicity, we only give one premise for the $\otimes^2 R$ rule; to recover the full rule, we set i to 1 or 2 as appropriate, and replace ϕ with ψ.

$$\frac{\Gamma_1 \vdash \phi^x, \Delta_1 \quad -\mathcal{H}, \{\phi^x\} \qquad \Gamma_2 \vdash \psi^y, \Delta_2 \quad -\mathcal{H}, \{\psi^y\}}{\Gamma_1, \Gamma_2\ \vdash\ (\phi \otimes \psi)^\epsilon, \Delta_1, \Delta_2 \quad -\mathcal{H}}\ \otimes^1 R$$

$$\frac{\Gamma_i \vdash \phi_i^{zi}, \Delta_i, \Delta_{i+2}^\epsilon \quad -\mathcal{H}[\{(\phi_1 \otimes \phi_2)^x, F_1, \ldots, F_k\} \leftarrow \{\phi_i^{zi}\}]}{\Gamma_1, \Gamma_2\ \vdash\ (\phi_1 \otimes \phi_2)^x, \Delta_1, \Delta_2, \Delta_3^x, \Delta_4^x \quad -\mathcal{H}}\ \otimes^2 R$$

$$\frac{\Gamma \vdash \phi^\epsilon, \psi^\epsilon, \Delta \quad -\mathcal{H}}{\Gamma\ \vdash\ (\phi \wp \psi)^\epsilon, \Delta \quad -\mathcal{H}}\ \wp^1 R$$

$$\frac{\Gamma \vdash \phi^x, \psi^x, \Delta \quad -\mathcal{H}[\{(\phi \wp \psi)^x, F_1, \ldots, F_k\} \leftarrow \{\phi^x, \psi^x, F_1, \ldots, F_k\}]}{\Gamma\ \vdash\ (\phi \wp \psi)^x, \Delta \quad -\mathcal{H}}\ \wp^2 R$$

$$\frac{\Gamma \vdash \phi^\epsilon, \Delta \quad -\mathcal{H}}{\Gamma\ \vdash\ (\phi \oplus \psi)^\epsilon, \Delta \quad -\mathcal{H}}\ \oplus^1 R$$

$$\frac{\Gamma \vdash \phi^x, \Delta \quad -\mathcal{H}[\{(\phi \oplus \psi)^x, F_1, \ldots F_k\} \leftarrow \{\phi^x, F_1, \ldots F_k\}]}{\Gamma\ \vdash\ (\phi \oplus \psi)^x, \Delta \quad -\mathcal{H}}\ \oplus^2 R$$

$$\frac{\Gamma \vdash \Delta \quad - \mathcal{H}}{\Gamma \vdash ?\phi^x, \Delta \quad - \mathcal{H}}\ ?wR \qquad \frac{\Gamma \vdash \phi^x, \Delta \quad - \mathcal{H}, \{\phi^x\} \qquad \Gamma_1, \psi \vdash \Delta_1 \quad - \mathcal{H}}{\Gamma, \Gamma_1, \phi \multimap \psi \vdash \Delta, \Delta_1 \quad - \mathcal{H}}\ \multimap L$$

$$\frac{\Gamma, \phi \vdash \psi^x, \Delta \quad - \mathcal{H}, \{\psi^x\}}{\Gamma \vdash (\phi \multimap \psi)^\epsilon, \Delta \quad - \mathcal{H}}\ \multimap^1 R \qquad \frac{\Gamma, \phi \vdash \psi^y, \Delta \quad - \mathcal{H}}{\Gamma \vdash (\phi \multimap \psi)^x, \Delta \quad - \mathcal{H}}\ \multimap^2 R\ \boxed{x := \epsilon}$$

where $\mathcal{H}[\Omega \leftarrow \Upsilon]$ denotes replacing of set Ω from list $\mathcal{H}$ with set Υ.

Note that for each application of a multiplicative rule the superscript labels of each active formula are assigned a fresh variable, reflecting the fact that neither of these formulae cannot be totally eliminated. Also the superscript variable of the principal formula must be assigned the value ϵ, in order to cancel the constraint valid for formula(e) being labelled with that variable.

Note that we will often identify an unlabelled formula ϕ with the labelled formula ϕ^ϵ. It will always be possible to disambiguate such annotations from the context.

Thus, for the proof Π we have:

$$\dfrac{\dfrac{}{t \vdash t_1^y \quad - \{t^y\}}\ Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{r \vdash r_1 \qquad - \{?q^x, ?p^x\}}\ Ax}{r \vdash ?p^x, r_1 \qquad - \{?q^x, ?p^x\}}\ ?wR}{r \vdash ?q^x, ?p^x, r_1 \qquad - \{?q^x, ?p^x\}}\ ?wR}{r \vdash (?q \wp ?p)^x, r_1 \qquad - \{(?q \wp ?p)^x\}}\ \wp R}{r \vdash ((?q \wp ?p) \oplus s)^x, r_1 \qquad - \{((?q \wp ?p) \oplus s)^x\}}\ \oplus R}{r, t \vdash t_1 \otimes ((?q \wp ?p) \oplus s), r_1 \qquad - \cdot}\ \otimes R$$

3. **Calculation of unused atoms.**

On the successful completion of the search, we calculate the set of atoms which appear in the *axioms* : $\mathcal{U} = \{t, t_1, r, r_1\}$.

The difference of sets $\mathcal{A}$ and $\mathcal{U}$ determines all unused atoms: $\mathcal{A} \backslash \mathcal{U} = \{p, q, s\}$

4. **Constraints on elimination of unused atoms** *i.e.* **formulae.**

Among the sets recorded in $\mathcal{H}$ at the leaves of a proof tree, we exclude those sets which contain at least one formula with at least one atom from the set $\mathcal{U}$. Each remaining set determines a set of formulae that cannot be eliminated simultaneously from the given proof. For the proof Π that is the set $\{?q^x, ?p^x\}$.

5. **Elimination of unused formulae.**

According to the constraints generated in step 4, select atoms that can be deleted simultaneously from the proof. Delete every appearance of the selected atoms *i.e.* every appearance of (sub)formulae made up of the selected atoms. Delete any rule inferences in which at least one active formula is deleted.

Example 6.1 For the sequent $m \otimes n, t, b \ \vdash \ (?p\wp(m \otimes n)) \otimes (b\wp?s), (?r \otimes t)\wp?l$ *i.e.* its 'rectification' $m_1 \otimes n_1, t_1, b_1 \ \vdash \ (?p\wp(m \otimes n)) \otimes (b\wp?s), (?r \otimes t)\wp?l$ we have the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{t_1 \vdash t^{z1} \ - \ \{?p^x, (m \otimes n)^x\},\ \{t^{z1}\}}\,Ax}{t_1 \vdash ?p^x, t^{z1} \ - \ \{?p^x, (m \otimes n)^x\},\ \{t^{z1}\}}\,wR \qquad P_1}{m_1 \otimes n_1, t_1 \ \vdash ?p^x, (m \otimes n)^x, ?r \otimes t, ?l \ \ - \{?p^x, (m \otimes n)^x\}}\,\otimes^1 R}{m_1 \otimes n_1, t_1 \ \vdash \ (?p\wp(m \otimes n))^x, ?r \otimes t, ?l \ \ - \{(?p\wp(m \otimes n))^x\}}\,\wp R}{m_1 \otimes n_1, t_1 \ \vdash \ (?p\wp(m \otimes n))^x, (?r \otimes t)\wp?l \ \ - \{(?p\wp(m \otimes n))^x\}}\,\wp R \qquad P_2}{m_1 \otimes n_1, t_1, b_1 \ \vdash \ (?p\wp(m \otimes n)) \otimes (b\wp?s), (?r \otimes t)\wp?l \ \ - \cdot}\,\otimes^1 R$$

where the proof P_1 is as follows:

$$\dfrac{\dfrac{\dfrac{}{m_1 \vdash m^{v1} \ - \ \{?r^{z2}\}, \{m^{v1}\}}\,Ax \qquad \dfrac{\dfrac{}{n_1 \vdash n^{v2} \ - \ \{?r^{z2}\},\ \{n^{v2}\}}\,Ax}{n_1 \vdash n^{v2}, ?r^{z2}, ?l \ - \ \{?r^{z2}\},\ \{n^{v2}\}}\,wR}{m_1, n_1 \vdash (m \otimes n)^x, ?r^{z2}, ?l \ - \{?p^x, (m \otimes n)^x\},\ \{?r^{z2}\}}\,\otimes^2 R \ \boxed{x := \epsilon}}{m_1 \otimes n_1 \vdash (m \otimes n)^x, ?r^{z2}, ?l \ - \{?p^x, (m \otimes n)^x\},\ \{?r^{z2}\}}\,\otimes L$$

and the proof P_2 is as follows:

$$\dfrac{\dfrac{\dfrac{}{b_1 \vdash b^y \ \ - \{b^y, ?s^y\}}\,Ax}{b_1 \vdash b^y, ?s^y \ \ - \{b^y, ?s^y\}}\,wR}{b_1 \vdash (b\wp?s)^y \ \ - \{(b\wp?s)^y\}}\,\wp R$$

On the successful completion of the search we have the following 'calculations':

$$\mathcal{A} \ = \ \{m_1, n_1, t_1, b_1, p, m, n, b, s, r, t, l\} \qquad\qquad \mathcal{U} \ = \ \{m_1, m, n_1, n, t_1, t, b, b_1\}$$

$$\mathcal{H} \ = \ \{\ \{?p^\epsilon, (m \otimes n)^\epsilon\},\ \ \{t^{z1}\},\ \ \{r^{z2}\},\ \ \{m^{v1}\},\ \ \{n^{v2}\},\ \ \{b^y, ?s^y\}\ \}$$

The set $\mathcal{U}$ 'produces' the following refinement $\mathcal{H} \ = \ \{\ \{r^{z2}\}\ \}$. So we have just the constraint that formula $?r$ cannot be eliminated from the proof. As $\mathcal{A} \backslash \mathcal{U} \ = \ \{p, s, r, l\}$, atoms p, s and l can be freely eliminated.

It should be stressed that our technique can be easily extended to an arbitrary set of sequent rules. It is envisaged that the results of this analysis can then be implemented and utilized by means of an automated proof assistant. Our technique is limited to sequent proofs and thereby differs from dead-code elimination in functional languages. Developing more general techniques for program slicing and dead-code elimination in advanced logic programming languages is research still in progress.

7 Conclusions and Future Work

We have discussed two issues in the analysis of search strategies in sequent systems: permutations and elimination of redundant formulae.

We have proposed a more detailed specification of inference rules, in order to enable a more precise analysis of their permutation properties. This analysis can be used, among others, to reduce the proof search space.

We have also shown how some sequent rules with constraints can be used to extract information about the necessary and unnecessary formulae in a proof. This knowledge can contribute, among others, to a further reduction of the search space. The proposed technique for the detection of unnecessary formulae is quite general and it can be applied more broadly than just linear logic calculi.

We may thus consider this work as an initial requirements analysis of the properties of proofs of interest to (logic programming, at least) proof-search strategies.

It is envisaged that the results of this analysis can then be implemented and utilized by means of an automated proof assistant such as Twelf [18,21], possibly in conjunction with constraint logic programming techniques [14].

Another topic of research is further reduction of redundancy in a proof search, such as loops triggered off by cyclic combinations of inference figures. It is well known that for many logics, backward proof search in the usual sequent calculi does not terminate in general.

Systems for preventing and detecting loops during a proof construction have been studied for a long time and many different approaches have been proposed. It is interesting to note that in spite of expansion of the proof systems based on (fragments of) resource sensitive logics (such as, for example, affine and linear logic) no loop detection mechanism (apart from the naive loop checker) has been developed or described in the literature.

We are interested in developing automated, strategy independent loop detection mechanisms for resource sensitive logics. Work is already underway on a terminating proof search mechanism for affine logics.

Acknowledgements

The authors are grateful to some anonymous referees for some very helpful comments on drafts of this paper.

References

[1] J-M. Andreoli and R. Pareschi *Logic programming with sequent systems: A linear Logic Approach.*, in P.Schröder-Heister ed., Proceedings of Workshop to Extensions of Logic Programming 1-30, Tübingen, 1989. Published by Springer-Verlag as Lecture Notes in Artificial Intelligence 475.

[2] J-M. Andreoli, *Logic Programming with focusing proofs in linear logic*, Journal of Logic and Computation, 2(3):297-347, 1992.

[3] P. Armelin *Programming with Bunched Implications*, PhD Thesis, Queen Mary, University of London, 2002.

[4] H.B. Curry, *The Permutability of Rules in the Classical Inferential Calculus*, Journal of Symbolic Logic 17:245-8, 1952.

[5] R. Dyckhoff, *Contraction-free Sequent Calculi for Intuitionistic Logic*, The Journal of Symbolic Logic, Vol. 57(3):795-807, Sept. 1992.

[6] D. Galmiche and G. Perrier, *On proof normalisation in Linear Logic*, Theoretical Computer Science 135:67-110, 1994.

[7] J. Harland, *A proof-theoretic Analysis of Goal-directed Provability*, Journal of Logic and Computation 4(1):69-88, 1994.

[8] J. Harland, *An Algebraic Approach to Proof Search in Sequent Calculi*, short paper presented at the International Joint Conference on Automated Reasoning, Siena, July, 2001.

[9] J. Harland and D. Pym, *Resource-distribution via Boolean constraints*, ACM Transactions on Computational Logic 4(1)56-90 January, 2003.

[10] J. Hodas and D. Miller, *Logic programming in a Fragment of Intuitionistic Linear Logic*, Journal of Information and Computation 110(2):327-365, 1994.

[11] N. Kamide *Sequent Calculi for Intuitionistic Linear Logic with Strong Negation*, Logic Journal of the IGPL 10(6):653-687, 2002.

[12] S.C. Kleene, Permutability of Inferences in Gentzen's Calculi LK and LJ, *Memoirs of the American Mathematical Society* 10, 1952.

[13] T. Lutovac and J. Harland, *Towards the Automation of the Design of Logic Programming Languages*, Technical Report TR-97-30, Department of Computer Science, RMIT University, 1997.

[14] K. Marriot and P. Stuckey, *Programming with Constraints*, MIT Press, 1998.

[15] D. Miller, *A logical analysis of modules in logic programming*, Journal of Logic Programming 6:79-108, 1989.

[16] D. Miller, *Forum: A multiple-conclusion specification-logic*, Theoretical Computer Science 165(1): 201-232, 1996.

[17] D. Miller, G. Nadathur, F. Pfenning and A. Ščedrov, *Uniform proofs as a Foundation for Logic Programming*, Annals of Pure and Applied Logic 51, 1991.

[18] F. Pfenning and C. Schürmann, *Twelf — a meta-logical framework for deductive systems*, H. Ganzinger, editor, Proceedings of the 16th International Conference on Automated Deduction (CADE-16) 202–206, Trento, Italy, July 1999. Published by Springer-Verlag as Lecture Notes in Artificial Intelligence 1632.

[19] D. Pym, *The Semantics and Proof Theory of the Logic of Bunched Implications*, Kluwer Applied Logic Series Volume 26, 2002.

[20] D. Pym and J. Harland,*A Uniform Proof-theoretic Investigation of Linear Logic Programming*, Journal of Logic and Computation 4(2):175-207, 1994.

[21] C. Schürmann, *Automating the Meta-Theory of Deductive Systems*, PhD thesis, Carnegie-Mellon University, 2000.

[22] L. Wallen, *Automated Proof Search in Non-classical Logic*, MIT Press, 1990.

[23] M. Winikoff and J. Harland, *Implementing the Linear Logic Programming Language Lygon*, Proceedings of the International Logic Programming Symposium 66-80, Portland, December, 1995.