

TECHNISCHE
UNIVERSITÄT
DRESDEN

Fakultät Informatik

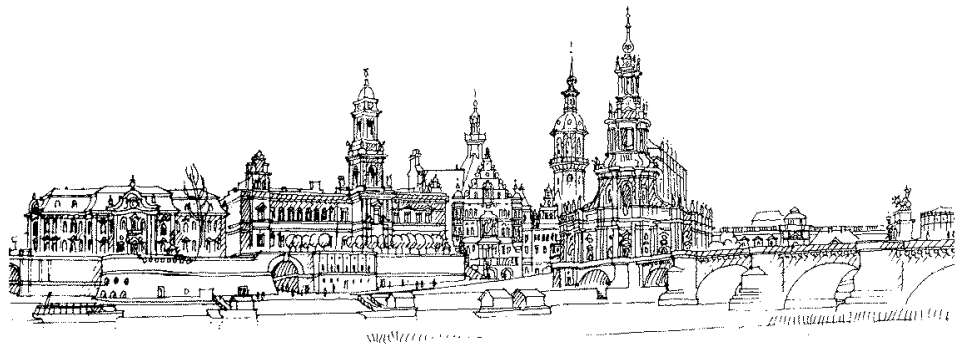
Technische Berichte
Technical Reports

ISSN 1430-211X

FI05-08-Juli 2005

Paola Bruscoli, François Lamarche
and Charles Stewart (Eds.)

**Structures and Deduction –
the Quest for the Essence of Proofs
(satellite workshop of ICALP 2005)**



Technische Universität Dresden
Fakultät Informatik
D-01062 Dresden
Germany
URL: <http://www.inf.tu-dresden.de/>

Foreword

The Structures and Deductions workshop was held in Lisbon on July 16-17, 2005, as a satellite workshop of the ICALP international conference. Citing the official web page will convey what the organizers had in mind regarding its scientific purpose.

This meeting is about new algebraic and geometric methods in proof theory, with the aim of expanding our ability to manipulate proofs, eliminate bureaucracy from deductive systems, and ultimately provide: 1) a satisfying answer to the problem of identity of proofs and 2) tools for improving our ability to implement logics.

Stimulated by computer science, proof theory is progressing at fast pace. However, it is becoming very technical, and runs the risk of splitting into esoteric specialties. The history of science tells us that this has happened several times before, and that these centrifugal tendencies are very often countered by conceptual reunifications, which occur when one is looking at a field after having taken a few steps back.

The organizers were aware that efforts of that kind are happening in several distinct communities, and the workshop was conceived as a point of meeting and exchange; in particular the schedule was designed to encourage impromptu discussions.

Fourteen papers were chosen out of the nineteen submissions. The use of Andrei Voronkov's EasyChair software made it easy to ensure that everything got reviewed by three referees.

This meeting was enough of a success for us to want to have another edition next year; we hope that our better knowledge of the intricacies of international funding agencies will allow us to pay for the expenses of our guest speakers this time.

We wish to thank the following for their help, financial or personal:

- the guest speakers Claude Kirchner and Dale Miller, who came at their own expenses;
- the organizers of the string of ICALP satellite workshops, Antonio Ravara and Vasco Vasconcelos, as well as the whole of the ICALP organization;
- Carolina Silva and the Service de Coopération et d'Action Culturelle de l'Ambassade de France à Lisbonne, as well as Bettina Schneider and the Deutsche Forschungsgemeinschaft for their financial help;
- Sylvia Epp, Alessio Guglielmi, Steve Prestwich and Andrei Voronkov;
- INRIA and ICCL at TU Dresden.

The Organization Committee

Paola Bruscoli, François Lamarche, Charles Stewart

July 2005

Organization

Programme Committee:

Paola Bruscoli (TU Dresden)
Pietro Di Gianantonio (Univ. Udine)
Gilles Dowek (LIX and Ecole Polytechnique, Paris)
Roy Dyckhoff (St Andrews)
Rajeev Goré (NICTA and ANU, Canberra)
François Lamarche (LORIA and INRIA Lorraine, Nancy) – Chair
Luke Ong (Oxford)
Prakash Panangaden (McGill)
Michel Parigot (CNRS, Paris)
Charles Stewart (TU Dresden)
Thomas Streicher (TU Darmstadt)

External Referees

Fabio Alessi
Denis Bechet
Kai Brännler
Agata Ciabattoni
Paolo Coppola
Stéphane Lengrand
Sara Negri
Sylvain Pogodalla
Ivan Scagnetto
Lutz Straßburger
Vitezslav Svejdar

Contents

Claude Kirchner:

Beyond Deduction Modulo (Invited Talk) **p. 1**

Lutz Straßburger:

From Deep Inference to Proof Nets **p. 2**

Richard Iain McKinley:

Classical Categories and Deep Inference **p. 19**

Dale Miller:

A Games Semantics for Proof Search (Invited Talk) **p. 34**

Yves Guiraud:

The Three Dimensions of Proofs **p. 35**

Alessio Guglielmi:

The Problem of Bureaucracy and Identity of Proofs from the Perspective of Deep Inference **p. 53**

Kai Brännler and Stéphane Lengrand:

On two Forms of Bureaucracy in Derivations **p. 69**

Jean-Baptiste Joinet:

Completeness of MLL Proof-Nets w.r.t. Weak Distributivity **p. 81**

Christophe Fouqueré and Virgile Mogbil:

Rewritings in Polarized (Partial) Proof Structures **p. 95**

Estelle Dumoulin and Didier Galmiche:

Labelled Structures and Provability in Resource Logics **p. 110**

Charles Stewart and Robert Hein:

Purity Through Unravelling **p. 126**

Ewen Denney, John Power and Konstantinos Turlas:

Hierarchical Proof Structures **p. 144**

Ozan Kahramanoğulları, Pierre-Etienne Moreau and Antoine Reilles:

Implementing Deep Inference in TOM **p. 158**

Gerard R. Renardel de Lavalette:

Abstract Derivations, Equational Logic and Interpolation **p. 173**

Elaine Pimentel, Simona Ronchi della Rocca and Luca Roversi:

Intersection Types: a Proof-Theoretical Approach **p. 189**

Joao Rasga:

A Cut Elimination in Propositional Based Logics **p. 205**

Beyond Deduction Modulo

Claude Kirchner

INRIA & LORIA
Nancy, France

From Deep Inference to Proof Nets

Lutz Straßburger

Universität des Saarlandes — Informatik — Programmiersysteme
Postfach 15 11 50 — 66041 Saarbrücken — Germany
<http://www.ps.uni-sb.de/~lutz>

Abstract. This paper shows how derivations in (a variation of) *SKS* can be translated into proof nets. Since an *SKS* derivation contains more information about a proof than the corresponding proof net, we observe a loss of information which can be understood as “eliminating bureaucracy”. Technically this is achieved by cut reduction on proof nets. As an intermediate step between the two extremes, *SKS* derivations and proof nets, we will see nets representing derivations in “Formalism A”.

1 Introduction

Through the development of the two concepts of *deep inference* [Gug02] and *proof nets* [Gir87] the quest for the identity of proofs has become fashionable again, and the research on the fundamental question “When are two proofs the same?” seems now to be booming.

Proof nets have been conceived by Girard [Gir87] in order to avoid *bureaucracy*: in formal systems like the sequent calculus two proofs that are “morally the same” are distinguished by trivial rule permutations.

Deep inference has been conceived by Guglielmi in order to obtain a deductive system for a non-commutative logic [Gug02]. In a formalism employing deep inference, like the calculus of structures, one can apply inference rules anywhere deep inside formulae as we know it from term rewriting, instead of decomposing formulae along their main connectives as we know it from traditional formalisms. From the “we-wish-to-eliminate-bureaucracy” point of view, this is a disaster: The number of possible “trivial rule permutations” explodes, compared to the sequent calculus. However, the finer granularity of inference rules (one inference step in the sequent calculus corresponds to many inference steps in the calculus of structures) allows a finer analysis of the inner structure of proofs, which in turn can lead to new notions of proof nets (as happened in [SL04] and [LS05b]).

In this paper we will see how proof nets can be extracted directly from deep inference systems. I will concentrate here only on classical logic, more precisely on (a slight variation of) system *SKS* [BT01,Brü03a], the most popular system for classical logic in the calculus of structures. But it should be clear that the exercise of this paper can in the same way be carried out for any other system, in particular also for linear logic as it is presented in [Str02].

To some extent, one can say that proof nets make as many identifications between proofs as possible (without ending up in a triviality), and derivations

in the calculus of structures makes as few identifications as possible. These two extremes span a whole universe of possible proof identifications. And going from the extreme with few identifications to the extreme with many identification means losing information, namely, the “bureaucratic” information that makes the additional distinctions. I will argue, that this process of losing information can be modelled by cut elimination. In each single cut reduction step some bit of information is lost. Depending on the restrictions on cut elimination one can choose which information to lose.

The question, when this information is bureaucratic and when it is non-bureaucratic (i.e., essential for the proof), must be left unanswered in this paper.

2 Proof Nets for Classical Logic

Proof nets are abstract (graphical) presentations of proofs such that all “trivial rule permutations” are quotiented away. Ideally the notion of proof net should be independent from any syntactic formalism. But due to the almost absolute monopoly of the sequent calculus, most notions of proof nets proposed in the past related themselves to the sequent calculus. Consequently we could observe features like “boxes” and explicit “contraction links”. The latter appeared not only in linear logic [Gir96] but also in classical logic (as sketched in [Gir91] and detailed out in [Rob03]). The slogan of the early proof nets was

Slogan 1: *Every link in the proof net corresponds to a rule application in the sequent calculus.*

with the basic idea that if two rules “trivially permute” in the sequent calculus, then the corresponding links in the proof net are independent. However, more recent proposals for proof nets follow a different slogan:

Slogan 2: *A proof net is a formula tree (or sequent forest) enriched with additional graph structure.*

This additional graph structure is supposed to capture the *essence* of the proof. To our knowledge the first notion of proof net in this more modern setting were [HvG03] for unit-free multiplicative additive linear logic (MALL) and [SL04] for multiplicative linear logic (MLL) with units.¹ Then in [LS05b] proof nets for classical logic obeying Slogan 2 followed. Let me now recall that latter notion of proof nets. (I consider here only the \mathbb{N} -nets of [LS05b].)

The set of *formulae* is generated via the binary connectives \wedge (*conjunction*) and \vee (*disjunction*) from the set $\mathcal{A} \cup \bar{\mathcal{A}} \cup \{\mathbf{t}, \mathbf{f}\}$, where $\mathcal{A} = \{a, b, c, \dots\}$ is a countable set of *propositional variables* and $\bar{\mathcal{A}} = \{\bar{a}, \bar{b}, \bar{c}, \dots\}$ is the set of *negated propositional variables*, and \mathbf{t} and \mathbf{f} are the *constants* representing “true” and

¹ In fact, the first has been [Gir87] (or more precisely [KM71]) simply because for the special case of unit-free MLL both slogans coincide: every connective in the formulae corresponds to an application of a sequent rule, and the axiom links attached to the formulae capture exactly the essence of a proof in unit-free MLL. This very fortunate coincidence is also the reason why proof nets for unit-free MLL behave so remarkably well and were so successful from the very beginning.

“false”, respectively. The elements of the set $\mathcal{A} \cup \bar{\mathcal{A}} \cup \{\mathbf{t}, \mathbf{f}\}$ are called *atoms*. A finite list $\Gamma = A_1, A_2, \dots, A_n$ of formulae is called a *sequent*. I will consider formulae as binary trees (and sequents as forests), whose leaves are decorated by atoms, and whose inner nodes are decorated by the connectives. The negation \bar{A} of a formula A is defined as follows:

$$\bar{\bar{a}} = a \quad \bar{\mathbf{t}} = \mathbf{f} \quad \bar{\mathbf{f}} = \mathbf{t} \quad \overline{(A \wedge B)} = \bar{B} \vee \bar{A} \quad \overline{(A \vee B)} = \bar{B} \wedge \bar{A} \quad (1)$$

Here a ranges over the set \mathcal{A} . However, from now on I will use a to denote an arbitrary atom (including constants). Note that $\bar{\bar{A}} = A$ for all A .

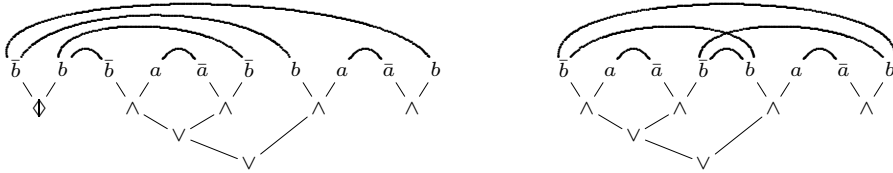
There is a special kind of auxiliary formula, called *cut*, which is of the shape $B \diamond \bar{B}$, where \diamond is called the *cut connective* and is allowed only at the root of a formula tree. A *cut sequent* is a finite list $\Sigma = B_1 \diamond \bar{B}_1, \dots, B_n \diamond \bar{B}_n$ of cuts.

A *prenet* $P, \Sigma \triangleright \Gamma$ consists of a sequent Γ , a cut sequent Σ , and an undirected multi-graph P whose set of vertices is the set of leaves of Γ and Σ and whose set of edges obeys the following conditions: (i) whenever there is an edge between two leaves, then one is decorated by an atom a and the other by its dual \bar{a} , and (ii) whenever there is an edge connecting a leaf to itself, then this leaf is decorated by \mathbf{t} .

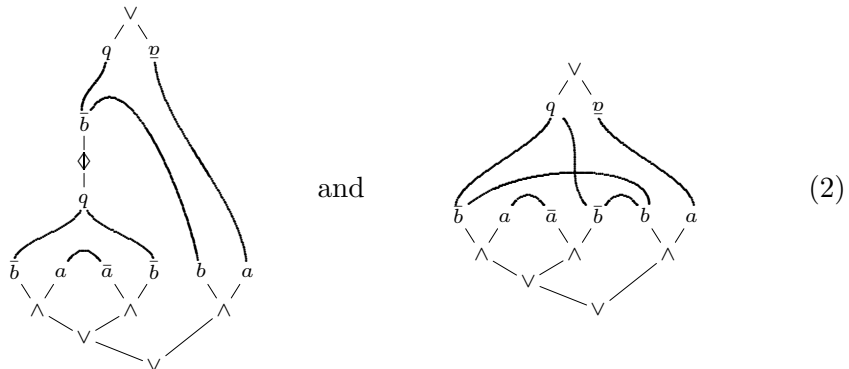
One can think of P also of an undirected graph whose edges are labeled by natural numbers (hence the name N-net in [LS05b]), but here I will draw it as multi-graph, for example:



In the following, I will consider only nets where Γ contains exactly two formulae (there is no restriction on the number of cuts in Σ). Here are two examples, one with and one without cuts (both are variations of examples in [LS05b]):



These can also be drawn as



which will be the preferred way from now on.

The cut reduction procedure for prenets is defined as follows. For cuts on compound formulae we have:

$$\begin{array}{c} \diagup \\ \diagdown \end{array} \text{---} \diamond \text{---} \begin{array}{c} \diagdown \\ \diagup \end{array} \quad \rightsquigarrow \quad \begin{array}{c} \text{---} \\ \text{---} \end{array} \diamond \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (3)$$

(For saving space, the picture is put on the side.) Atomic cuts are reduced as follows:

$$\begin{array}{c} \text{---} \text{---} \\ \vdots \\ \text{---} \text{---} \end{array} \text{---} \diamond \text{---} \begin{array}{c} \text{---} \text{---} \\ \vdots \\ \text{---} \text{---} \end{array} \quad \rightsquigarrow \quad \begin{array}{c} \text{---} \text{---} \\ \vdots \\ \text{---} \text{---} \end{array} \begin{array}{c} \text{---} \text{---} \\ \vdots \\ \text{---} \text{---} \end{array} \quad (4)$$

This means that for every pair of edges, where one is connected to the cut on one side and the other is connected to the other side of the cut, there is in the reduced net an edge connecting the two other ends of the pair (an edge connecting the two ends of the cut disappears with the cut). For the formal details, see [LS05b]. Here, let me only draw attention to the fact that if there is more than one edge between two dual atoms, the number of edges is multiplied, as in the following example:

$$\begin{array}{c} \text{---} \text{---} \\ \vdots \\ \text{---} \text{---} \end{array} \text{---} \diamond \text{---} \begin{array}{c} \text{---} \text{---} \\ \vdots \\ \text{---} \text{---} \end{array} \quad \rightsquigarrow \quad \begin{array}{c} \text{---} \text{---} \\ \vdots \\ \text{---} \text{---} \end{array} \begin{array}{c} \text{---} \text{---} \\ \vdots \\ \text{---} \text{---} \end{array} \quad (5)$$

This causes an exponential increase of the number of edges (in the number of atomic cuts) during the cut reduction process.

Obviously the cut reduction on prenets is terminating. But we have confluency only in the case where the number of edges between two atoms is restricted to one, i.e, where the multi-graph is just a graph (as shown in [LS05b]). In that case also the correctness criterion of [LS05b] is adequate. However, at the current state of the art, there is no suitable criterion yet for the general case. For that reason the objects here are called ‘‘prenets’’. The term ‘‘proof net’’ should be reserved to those objects that actually represent proofs.

3 Deep Inference for Classical Logic

Deep inference is a new paradigm for proof theoretical formalisms. The most prominent example of a formalism employing deep inference is the calculus of structures. It has successfully been employed to give new presentations for many logics, including classical logic [BT01,Brü03a], minimal logic [Brü03b], intuitionistic logic [Tiu05], several modal logics [SS03,Sto04], linear logic [Str03], and various noncommutative logics [DG04,Gug02,GS02].

Let me now recall the deep inference system SKS for classical logic [BT01]. At the same time I modify it slightly: I remove the syntactical equivalence employed by the calculus of structures and represent all the defining equations by inference rules. Nonetheless, I use the ‘‘outfix’’ notation employed by the calculus of structures because derivations are much easier to read that way. That

		$\text{ai}\downarrow \frac{S\{\mathbf{t}\}}{S[a, \bar{a}]}$	$\text{ai}\uparrow \frac{S(a, \bar{a})}{S\{\mathbf{f}\}}$		
		$\text{s} \frac{S(A, [B, C])}{S[(A, B), C]}$			
$\text{aw}\downarrow \frac{S\{\mathbf{f}\}}{S\{a\}}$	$\text{ac}\downarrow \frac{S[a, a]}{S\{a\}}$	$\text{ac}\uparrow \frac{S\{a\}}{S(a, a)}$	$\text{aw}\uparrow \frac{S\{a\}}{S\{\mathbf{t}\}}$		
$\text{nm}\downarrow \frac{S\{\mathbf{f}\}}{S(\mathbf{f}, \mathbf{f})}$	$\text{m} \frac{S[(A, C), (B, D)]}{S([A, B], [C, D])}$		$\text{nm}\uparrow \frac{S\{\mathbf{t}, \mathbf{t}\}}{S\{\mathbf{t}\}}$		
$\sigma\downarrow \frac{S[A, B]}{S[B, A]}$	$\alpha\downarrow \frac{S[A, [B, C]]}{S[[A, B], C]}$	$\alpha\uparrow \frac{S(A, (B, C))}{S((A, B), C)}$	$\sigma\uparrow \frac{S(A, B)}{S(B, A)}$		
$\mathbf{f}\downarrow \frac{S\{A\}}{S[A, \mathbf{f}]}$	$\mathbf{t}\downarrow \frac{S\{A\}}{S(A, \mathbf{t})}$	$\mathbf{t}\uparrow \frac{S\{\mathbf{f}, A\}}{S\{A\}}$	$\mathbf{f}\uparrow \frac{S(\mathbf{t}, A)}{S\{A\}}$		

Fig. 1. The inference rules of system SKS

means I write $[A, B]$ for $A \vee B$ and (A, B) for $A \wedge B$. For example the formula $((\bar{b} \wedge a) \vee (\bar{a} \wedge \bar{b})) \vee (b \wedge a)$ is in this notation written as $[[(\bar{b}, a), (\bar{a}, \bar{b})], (b, a)]$.

Before presenting the rules of the system we need to introduce the notion of a context, which is simply a formula with a hole. It is usually denoted by $S\{ \}$. For example $[[(\bar{b}, a), \{ \}], (b, a)]$ is a context. Let it be denoted by $S\{ \}$, and let $A = (\bar{a}, \bar{b})$. Then $S\{A\} = [[(\bar{b}, a), (\bar{a}, \bar{b})], (b, a)]$. I will omit the context-braces when structural parentheses fill the hole exactly. For example $S(\bar{a}, \bar{b})$ stands for $S\{(\bar{a}, \bar{b})\}$.

Now we are ready to see the inference rules. In this paper, they are all of the shape

$$\rho \frac{S\{A\}}{S\{B\}}$$

and this simply specifies a step of rewriting (via the implication $A \Rightarrow B$) inside a generic context $S\{ \}$. Such a rule is called *deep*, which is the reason for the term “deep inference”. If a rule scheme does not have this generic context (or there are size restrictions to the context), then the rule is called *shallow*.

The inference rules of *system SKS* (which are all deep) are shown in Figure 1. The rules $\text{ai}\downarrow$ and $\text{ai}\uparrow$ are called *atomic identity* and *atomic cut* (the i stands for “interaction”).² The rules s and m are called *switch* and *medial*, respectively.

² Here we can make an important observation: There are two very different notions of “cut” and the two should not be mixed up. On the one side we have the cut as a rule, and cut elimination means that this rule is admissible. In the calculus of structures it means that the whole up-fragment of the system (i.e., all rules with the \uparrow in the name) are admissible. This holds in particular also for system SKS, see [Brü03a],

They are the soul of system SKS. Note that these two rules are self-dual, while all other rules have their dual “co-rule”. For example, the rules $\text{aw}\downarrow$ and $\text{ac}\downarrow$ (called *atomic weakening down* and *atomic contraction down*, respectively) have as duals the rules $\text{aw}\uparrow$ and $\text{ac}\uparrow$, which are called *atomic weakening up* and *atomic contraction up*, respectively.³ The rules $\text{nm}\downarrow$ and $\text{nm}\uparrow$ are called *nullary medial (up and down)*. At this point it might seem rather strange that they are in the system. After all, they are instances of the atomic contraction rules. There are two reasons: The first one is that in a system in the calculus of structures the complete up-fragment should be admissible (i.e., also the rule $\text{ac}\uparrow$), but we need $\text{nm}\downarrow$ for completeness. The second reason is that the two rules $\text{ac}\uparrow$ and $\text{nm}\downarrow$ look similar only from the outside. When we look at the inside of the rules—we will do this in Section 5—we can see that they are of a very different nature.

The other rules ($\alpha\downarrow, \alpha\uparrow, \sigma\downarrow, \sigma\uparrow, \mathbf{t}\downarrow, \mathbf{t}\uparrow, \mathbf{f}\downarrow, \mathbf{f}\uparrow$) just say that \wedge and \vee are associative and commutative and that \mathbf{t} and \mathbf{f} are the units for them. Usually, in systems in the calculus of structures, formulae are considered up to an syntactic equivalence incorporating the associativity and commutativity (and the units) of the binary connectives. For example $[[A, B], [C, \mathbf{f}]]$ and $[B, [(\mathbf{t}, A), C]]$ are considered the same and denoted by $[A, B, C]$. Here, I deviate from this practice because having explicit rules for associativity and commutativity simplifies the translation to derivation nets in Section 5.

But before that, we need to plug our rules together to get *derivations*, which are finite chains of instances of inference rules. The topmost formula in a derivation is called its *premise* and the bottommost formula is called the *conclusion*. A derivation Δ , whose premise is A , whose conclusion is B , and whose inference rules are all contained in the system S , is denoted by

$$\begin{array}{c} A \\ \Delta \parallel_S \\ B \end{array} .$$

Figure 2 shows two examples of derivations in system SKS.

4 The ABC of Bureaucracy

The term “bureaucracy” is used to describe the phenomenon that oftentimes two formal proofs in a certain formalism denote “morally” the same proof but differ due to trivial rule permutations or other syntactic phenomena. Of course, the main problem here is to decide when two proofs should be “morally” the same. I.e., when is a certain syntactic phenomenon an important information about the proof and when is it just “bureaucracy”?

but is not of relevance for this paper. On the other side we have the cut \diamond , and cut elimination means composition of derivations (or proofs, or arrows in a category), and this will play a role in this paper.

³ In system SKS the rules $\text{ai}\downarrow$, $\text{ac}\downarrow$, and $\text{aw}\downarrow$ are atomic because their general counterparts $\text{i}\downarrow$, $\text{c}\downarrow$, and $\text{w}\downarrow$ are derivable. Dually for the up-rules (see [Brü03a] for details).

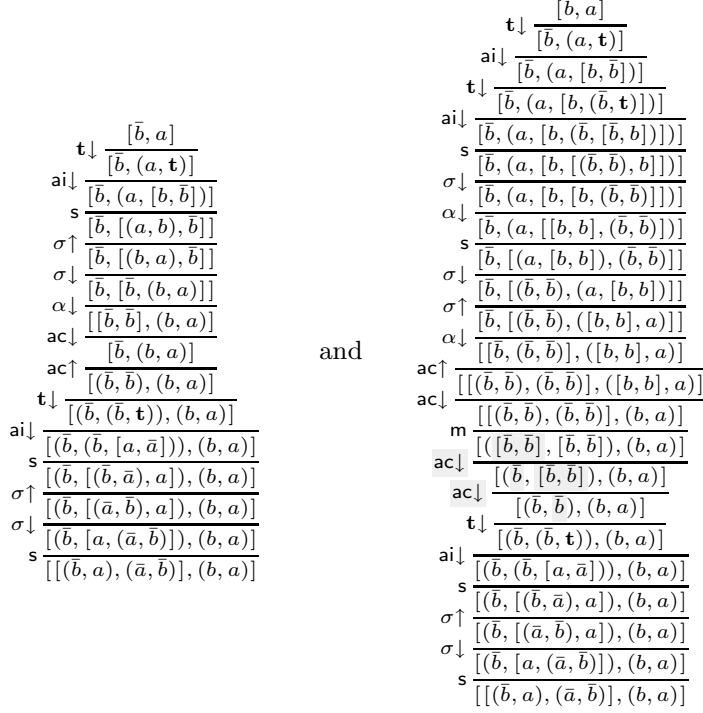


Fig. 2. Two examples of derivations

Consider now the right derivation in Figure 2. It is intuitively clear that we would not change the essence of the derivation if we exchanged the two $\mathbf{ac}\downarrow$ between the \mathbf{m} and the $\mathbf{t}\downarrow$ in the lower half of the derivation. That the two $\mathbf{ac}\downarrow$ are ordered one above the other can be considered to be an act of “bureaucracy”. In fact, following this intuition, we can permute the first $\mathbf{ac}\downarrow$ almost all the way down in the derivation, as shown in Figure 3. This kind of “bureaucracy” has been dubbed *bureaucracy of type A* by Guglielmi [Gug04a]. More generally whenever there is a derivation Δ from A to B and a derivation Δ' from C to D , then

$$\begin{array}{ccc}
(A, C) & & (A, C) \\
\Delta \parallel & & \Delta' \parallel \\
(B, C) & \text{and} & (A, D) \\
\Delta' \parallel & & \Delta \parallel \\
(B, D) & & (B, D)
\end{array} \tag{6}$$

as well as any other “merge” of Δ and Δ' should be considered to be the same. Guglielmi calls a formalism which *per se* makes these identifications *Formalism A*. However, Formalism A does not allow the identification of the two derivations in Figure 4. where the $\mathbf{ac}\downarrow$ is not “next to” another derivation but “inside” another derivation. This phenomenon is called *bureaucracy of type B* [Gug04b]. Then *Formalism B* is a formalism that avoids this kind of bureaucracy.

Besides bureaucracy of type A and B, we can observe another kind of bureaucracy, which we will call here *bureaucracy of type C*. Consider the two derivations in Figure 5. They can be considered to be essentially the same, because in both the

$$\begin{array}{c}
\text{ac}\downarrow \frac{[(\bar{b}, \bar{b}), [\bar{b}, \bar{b}]), (b, a)]}{[(\bar{b}, [\bar{b}, \bar{b}]), (b, a)]} \\
\text{ac}\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{t}\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{ai}\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\sigma\uparrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\sigma\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]}
\end{array}
\rightsquigarrow
\begin{array}{c}
\text{ac}\downarrow \frac{[(\bar{b}, \bar{b}), [\bar{b}, \bar{b}]), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{t}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{ai}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\sigma\uparrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\sigma\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{ac}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}
\end{array}$$

Fig. 3. Example for type-A bureaucracy

$$\begin{array}{c}
\text{ac}\downarrow \frac{[(\bar{b}, \bar{b}), [\bar{b}, \bar{b}]), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{t}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{ai}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\sigma\uparrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\sigma\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{ac}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}
\end{array}
\rightsquigarrow
\begin{array}{c}
\text{t}\downarrow \frac{[(\bar{b}, \bar{b}), [\bar{b}, \bar{b}]), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{ai}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\sigma\uparrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\sigma\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{ac}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]} \\
\text{ac}\downarrow \frac{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), \bar{b}), (b, a)]}
\end{array}$$

Fig. 4. Example for type-B bureaucracy

$$\begin{array}{c}
\text{t}\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{ai}\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\sigma\uparrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\sigma\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]}
\end{array}
\rightsquigarrow
\begin{array}{c}
\sigma\uparrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{t}\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{ai}\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\sigma\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\text{s} \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\sigma\uparrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]} \\
\sigma\downarrow \frac{[(\bar{b}, \bar{b}), (b, a)]}{[(\bar{b}, \bar{b}), (b, a)]}
\end{array}$$

Fig. 5. Example for type-C bureaucracy

same a and \bar{a} in the conclusion are “brought together” and disappear in an identity. The difference is that the derivation on the right contains two more applications of commutativity in which the two \bar{b} are exchanged. But neither Formalism A nor Formalism B can identify the two. Let us call *Formalism C* a formalism that is able to avoid this kind of bureaucracy.

So far, none of the three formalisms mentioned above has been formalized as a deductive system.⁴ Nonetheless, from the intuition given above one can translate the forced identifications in category theoretical terms. This is briefly done in Figure 6, which might be helpful for understanding the differences between the various kinds of bureaucracy. But the reader should be warned that this comparison is only very rough.⁵ There are various issues which are still unclear an subject to future research. Most important are the questions: How does the

⁴ But compare [BL05] for work on term calculi for Formalisms A and B.

⁵ See also [Hug04] and [McK05] for work relating deep inference and categorical logic.

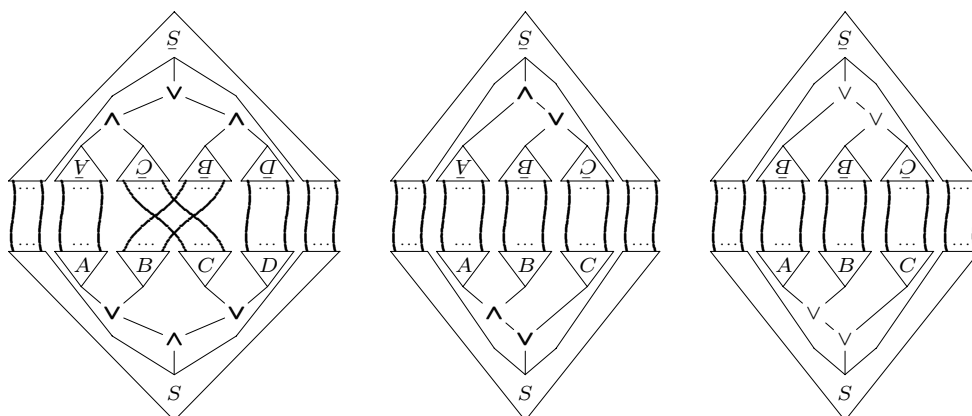


Fig. 7. The shape of m -nets, s -nets, and $\alpha\downarrow$ -nets

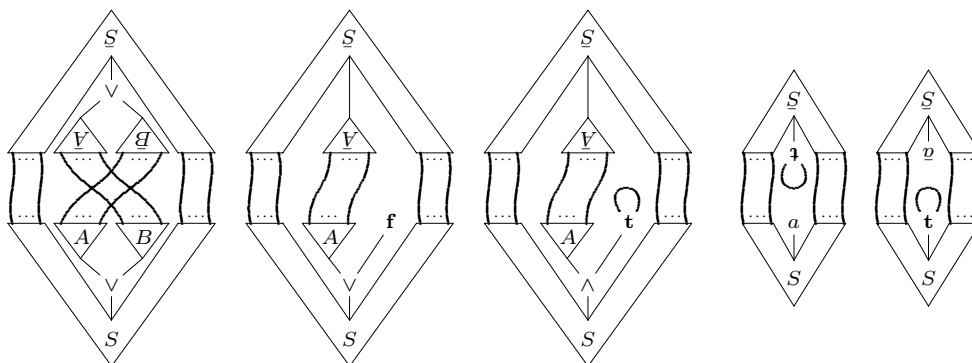


Fig. 8. The shape of $\sigma\downarrow$ -nets, $f\downarrow$ -nets, $t\downarrow$ -nets, $aw\downarrow$ -nets, and $aw\uparrow$ -nets

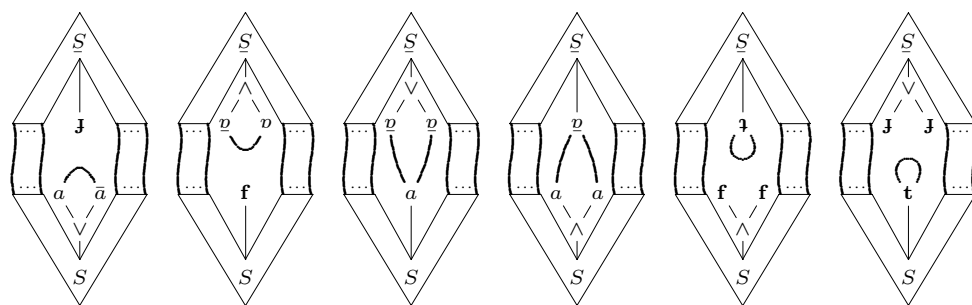


Fig. 9. The shape the nets for the rules $ai\downarrow$, $ai\uparrow$, $ac\downarrow$, $ac\uparrow$, $nm\downarrow$, and $nm\uparrow$

given in Figure 1. For the rules s , m , $\alpha\downarrow$, $\alpha\uparrow$, $\sigma\downarrow$, and $\sigma\uparrow$, it is intuitively clear what should happen: every atom in the premise is connected to its counterpart in the conclusion via an edge in in the linking; and there are no other edges. Note that the nets for $\alpha\downarrow$ and $\alpha\uparrow$ are the same; one written as the upside-down version of the other. The same holds for all other pairs of dual rules. For $\alpha\downarrow$, $\sigma\downarrow$, $f\downarrow$, and

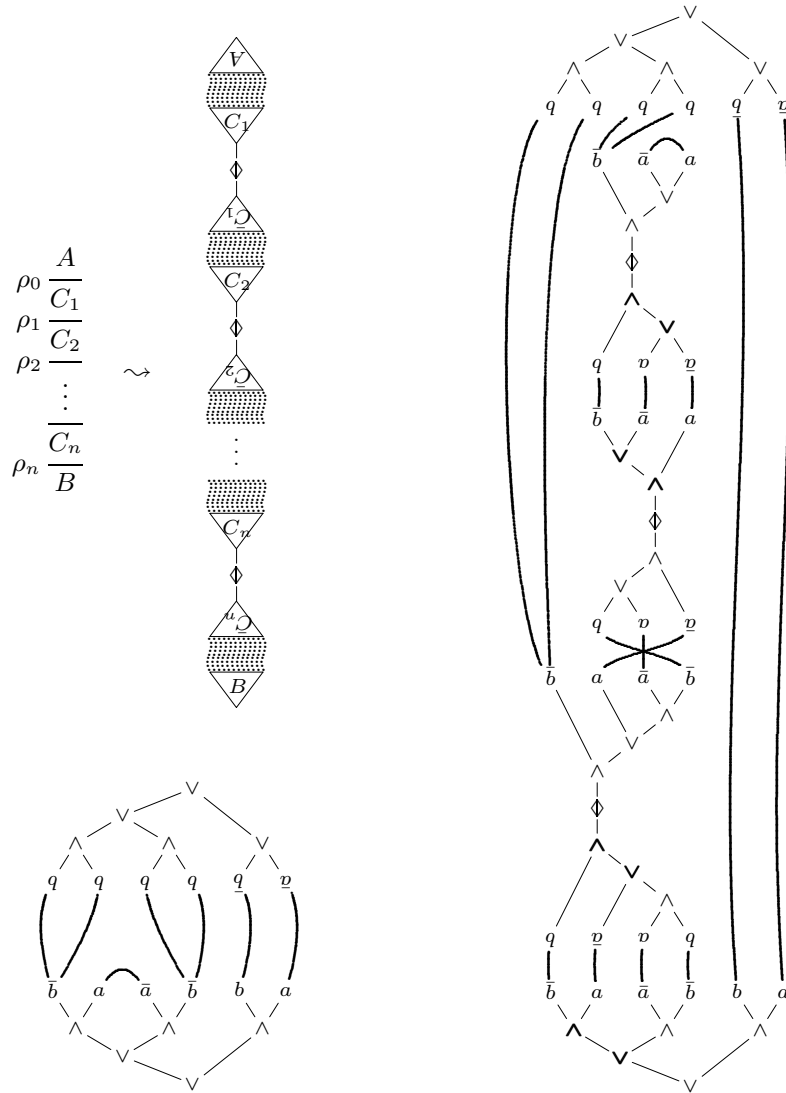


Fig. 10. Upper left: From derivations to derivation nets. **Right:** Example of an A-reduced net. **Lower left:** Result of applying level-C cut elimination to it.

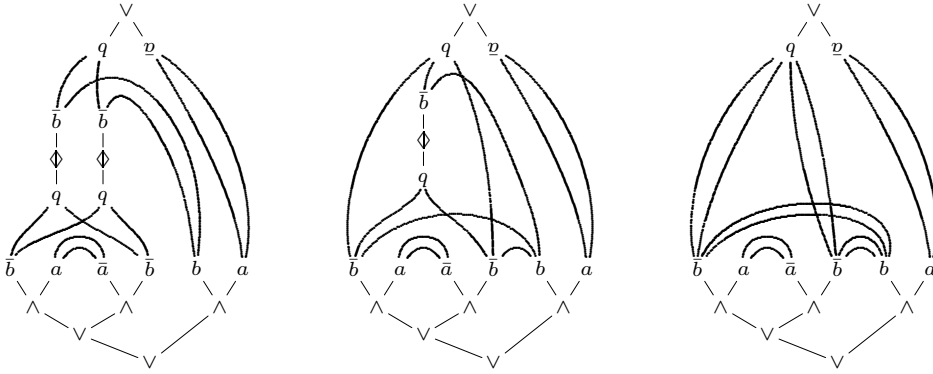
In order to give a formal proof, it would be necessary to give a formal definition of formalism A. The reader will agree that at this point it would be an easy exercise to come up with a technical definition such that the claim holds. In fact, one could use the A-reduced nets as the defining criterion. However, a more interesting and not so trivial problem is to come up with a deductive formalism for A-reduced nets. Another problem is to establish the relation between the A-reduced nets and the term calculus for Formalism A presented in [BL05]. The conjecture would be that level-A cut elimination is “the same” as the term normalization of [BL05].

7 Some remarks on the induced categories

We can define two categories: Let **Pre** be the category whose objects are the formulae and whose arrows between formulae A and B are the C -reduced prenets with $\Gamma = \bar{A}, B$. The category **Deri** is the wide subcategory of **Pre** in which the Hom-sets contain only those C -reduced prenets which are obtained from an SKS-derivation in the way described in the previous sections. Note that the X -reduced nets do not form a category, because composition defined via cut elimination is not associative. The A -reduced nets do also not form a category because the identity nets do not behave as identities.

The category **Deri** can be called “Boolean” in the sense of [LS05a]. The forgetful functor from the category of (small) categories to the category of posets maps it to a Boolean algebra. For the category **Pre** this is not the case because it contains morphisms that do not correspond to implications in Boolean logic.

I cannot give here a full characterisation of the two categories **Pre** and **Deri**, but I will compare them to the three different axiomatisations given in [FP04], [DP04], and [LS05a]. All of them have in common that the Hom-sets are equipped with an idempotent semigroup structure. This semigroup structure is also present for **Pre** and **Deri**, but it is *not* idempotent. In the case of **Pre** the sum of two nets is given by their union. This is best understood by seeing an example. Let f be the left net in (2) and g the right one. Then we can form $f + f$, $f + g$, and $g + g$ as follows:



In the case of **Deri** we can also form the “sum” of two derivations by using contraction, i.e., the two rules

$$c\downarrow \frac{S[A, A]}{S\{A\}} \quad \text{and} \quad c\uparrow \frac{S\{A\}}{S(A, A)} ,$$

which are both derivable in SKS (see [BT01] for details), and the rule

$$\text{mix} \frac{S(A, B)}{S[A, B]} ,$$

which is also derivable in SKS, for example via

$$\begin{array}{c} \mathbf{f}\downarrow \frac{S(A, B)}{S([A, \mathbf{f}], B)} \\ \mathbf{aw}\downarrow \frac{S([A, \mathbf{f}], B)}{S([A, \mathbf{t}], B)} \\ \mathbf{s} \frac{S([A, \mathbf{t}], B)}{S[A, (\mathbf{t}, B)]} \\ \mathbf{f}\uparrow \frac{S[A, (\mathbf{t}, B)]}{S[A, B]} \end{array} .$$

For any two derivations Δ_1, Δ_2 from A to B we can now form their sum by taking

$$\begin{array}{c} \mathbf{c}\uparrow \frac{A}{(A, A)} \\ (\Delta_1, \Delta_2) \Big\|_{\text{SKS}} \\ \mathbf{mix} \frac{(B, B)}{[B, B]} \\ \mathbf{c}\downarrow \frac{[B, B]}{B} \end{array} ,$$

where (Δ_1, Δ_2) is some “merge” of Δ_1 and Δ_2 ; compare (6). Note, that this sum of derivations could also be obtained in a different way, for example by first mixing (A, A) and then taking $[\Delta_1, \Delta_2]$, or by using a different derivation for mix. However, the important observation to make here is that no matter which one we choose, the translation into a C-reduced prenet yields the same result for all of them; and we obtain the same result if we first translate the derivations Δ_1 and Δ_2 into C-reduced prenets, and then taking their sum as nets (as described above). Hence, the semigroup structure on the Hom-sets is the same for **Pre** and **Deri**.

Furthermore, we can equip the Hom-sets of our categories with a partial order, defined by cut elimination: We say $f \leq g$ if g is obtained from f by eliminating some of the remaining cuts⁸, as it is the case in our example above for f and g . Then we also have $f + f \leq f + g \leq g + g$. The important observation about the semigroup and the partial order structure is, that *they are independent*. Although this seems to be natural from the viewpoint of our nets, it is not the case in the “classical categories” of [FP04] which are based on the proof nets in [Rob03]. In a “classical category” the sum-of-proofs-semigroup structure and the cut-elimination-partial-order structure on the Hom-sets determine each other uniquely via $f \leq g$ iff $f + g = g$. (In [DP04] and [LS05a] there is also a partial order structure on the Hom-sets, simply because the semigroup structure is idempotent. But this partial order structure has nothing to do with cut elimination, simply because everything is *a priori* cut-free.)

The category **Pre** follows quite closely the axiomatisation given in [LS05a]: it is *-autonomous (with weak units), it has monoids and comonoids, and it is

⁸ Even if this process is not confluent, we stay in the realm of C-reduced nets.

“graphical”. But it does not obey the equation

$$\begin{array}{ccc}
 A \vee A & \xrightarrow{\Delta_{A \vee A}} & (A \vee A) \wedge (A \vee A) \\
 \nabla_A \downarrow & & \downarrow \nabla_{A \wedge A} \\
 A & \xrightarrow{\Delta_A} & A \wedge A
 \end{array} \tag{9}$$

However, the two maps $A \vee A \rightarrow A \wedge A$ in (9) are ordered according to the cut-elimination-partial-order defined above, as it is the case in [FP04].

For the category **Deri** it is almost the same. The difference is that I have no proof showing that it is *-autonomous. Furthermore, I do not know whether **Deri** is closed under cut-elimination: Let Δ be an SKS derivation and let f_Δ be its corresponding C-reduced net. Now let f' be a net obtained from f_Δ by reducing some of the remaining cuts. Is there an SKS-derivation Δ' corresponding to f' ?

Solving these two open problems (and, in case of a negative answer, find a better deep inference deductive system for classical logic) is an important pre-requisite for starting to look for a decent geometrical correctness criterion for proof nets. Only with a good behaved deductive system one can ask for a “sequentialization theorem” for proof nets.

References

- [BL05] Kai Brännler and Stéphane Lengrand. On two forms of bureaucracy in derivations. In *Structures and Deduction 2005 (Satellite Workshop of ICALP'05)*, 2005.
- [Brü03a] Kai Brännler. *Deep Inference and Symmetry for Classical Proofs*. PhD thesis, Technische Universität Dresden, 2003.
- [Brü03b] Kai Brännler. Minimal logic in the calculus of structures. note, 2003.
- [BT01] Kai Brännler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 347–361. Springer-Verlag, 2001.
- [Bus91] Samuel R. Buss. The undecidability of k -provability. *Annals of Pure and Applied Logic*, 53:72–102, 1991.
- [Car97] Alessandra Carbone. Interpolants, cut elimination and flow graphs for the propositional calculus. *Annals of Pure and Applied Logic*, 83:249–299, 1997.
- [DG04] Pietro Di Gianantonio. Structures for multiplicative cyclic linear logic: Deepness vs cyclicity. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, CSL 2004*, volume 3210 of *Lecture Notes in Computer Science*, pages 130–144. Springer-Verlag, 2004.
- [DP04] Kosta Došen and Zoran Petrić. *Proof-Theoretical Coherence*. KCL Publications, London, 2004.
- [FP04] Carsten Führmann and David Pym. Order-enriched categorical models of the classical sequent calculus. To appear in *Journal of Pure and Applied Algebra*, 2004.
- [Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
- [Gir91] Jean-Yves Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1:255–296, 1991.

- [Gir96] Jean-Yves Girard. Proof-nets : the parallel syntax for proof-theory. In Aldo Ursini and Paolo Agliano, editors, *Logic and Algebra*. Marcel Dekker, New York, 1996.
- [GS02] Alessio Guglielmi and Lutz Straßburger. A non-commutative extension of MELL. In Matthias Baaz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2002*, volume 2514 of *LNAI*, pages 231–246. Springer-Verlag, 2002.
- [Gug02] Alessio Guglielmi. A system of interaction and structure. To appear in *ACM Transactions on Computational Logic*, 2002.
- [Gug04a] Alessio Guglielmi. Formalism A. note, April 2004.
- [Gug04b] Alessio Guglielmi. Formalism B. note, December 2004.
- [Gui05] Yves Guiraud. The three dimensions of proofs. In *Structures and Deduction 2005 (Satellite Workshop of ICALP'05)*, 2005.
- [Hug04] Dominic Hughes. Deep indereference proof theory equals categorical proof theory minus coherence. preprint, 2004.
- [HvG03] Dominic Hughes and Rob van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 1–10, 2003.
- [KM71] Gregory Maxwell Kelly and Saunders Mac Lane. Coherence in closed categories. *Journal of Pure and Applied Algebra*, 1:97–140, 1971.
- [LS05a] François Lamarche and Lutz Straßburger. Constructing free Boolean categories. In *Proceedings of the Twentieth Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, 2005.
- [LS05b] François Lamarche and Lutz Straßburger. Naming proofs in classical propositional logic. In Paweł Urzyczyn, editor, *Typed Lambda Calculi and Applications, TLCA 2005*, volume 3461 of *Lecture Notes in Computer Science*, pages 246–261. Springer-Verlag, 2005.
- [McK05] Richard McKinley. Classical categories and deep inference. In *Structures and Deduction 2005 (Satellite Workshop of ICALP'05)*, 2005.
- [Rob03] Edmund P. Robinson. Proof nets for classical logic. *Journal of Logic and Computation*, 13:777–797, 2003.
- [SL04] Lutz Straßburger and François Lamarche. On proof nets for multiplicative linear logic with units. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, CSL 2004*, volume 3210 of *LNCS*, pages 145–159. Springer-Verlag, 2004.
- [SS03] Charles Stewart and Phiniki Stouppa. A systematic proof theory for several modal logics. Technical Report WV-03-08, Technische Universität Dresden, 2003. To appear in proceedings of *Advances in Modal Logic 2004*, published by King’s College Publications.
- [Sto04] Finiki Stouppa. The design of modal proof theories: the case of S5. Master’s thesis, Technische Universität Dresden, 2004.
- [Str02] Lutz Straßburger. A local system for linear logic. In Matthias Baaz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2002*, volume 2514 of *LNAI*, pages 388–402. Springer-Verlag, 2002.
- [Str03] Lutz Straßburger. *Linear Logic and Noncommutativity in the Calculus of Structures*. PhD thesis, Technische Universität Dresden, 2003.
- [Tiu05] Alwen Fernanto Tiu. A local system for intuitionistic logic: Preliminary results. preprint, 2005.

Classical Categories and Deep Inference

Richard M^cKinley*

University of Bath

Abstract. Deep inference is a proof-theoretic notion in which proof rules apply arbitrarily deeply inside a formula. We show that the essence of deep inference is the bifunctoriality of the connectives. We demonstrate that, when given an equational theory that models cut-reduction, a deep inference calculus for classical logic (SKSg) is a categorical model of the classical sequent calculus LK in the sense of Führmann and Pym. We observe that this gives a notion of cut-reduction for derivations in SKSg, for which the usual notion of cut in SKSg is a special case. Viewing SKSg as a model of the sequent calculus uncovers new insights into the Craig interpolation lemma and intuitionistic provability.

1 Introduction

In recent years, the received wisdom that classical logic is uninteresting from a proof-theoretic point of view has been seriously challenged by a number of successful attempts to give a denotational semantics to classical proofs. The difficulties these approaches overcome are illustrated by the example below.

Given two proofs Φ_1 and Φ_2 of the same sequent, the proof

$$\begin{array}{c}
 \begin{array}{c} \Phi_1 \\ \vdots \\ \Gamma \vdash \Delta \end{array} \quad \begin{array}{c} \Phi_2 \\ \vdots \\ \Gamma \vdash \Delta \end{array} \\
 \hline
 \begin{array}{c} \Gamma \vdash \phi, \Delta \end{array} \quad \begin{array}{c} \Gamma, \phi \vdash \Delta \end{array} \quad \text{WR} \quad \text{WL} \\
 \hline
 \begin{array}{c} \Gamma, \Gamma \vdash \Delta, \Delta \\ \Gamma \vdash \Delta \end{array} \quad \text{Cut} \\
 \hline
 \begin{array}{c} \Gamma, \Gamma \vdash \Delta, \Delta \\ \Gamma \vdash \Delta \end{array} \quad \text{CL, CR,}
 \end{array} \tag{1}$$

(usually attributed to Lafont [1]), reduces (essentially) to either Φ_1 or Φ_2 ; the choice is non-deterministic. In a model which admits cut-reduction as equality, Φ_1 and Φ_2 acquire the same denotation. Note that this example does not rely on negation.

Certain previous attempts to avoid this collapse have relied on moving to a sub-logic (i.e., intuitionistic or linear logic). Others rely on a restricted cut-reduction system which makes a systematic choice of the left- or right-hand reduction in (1): for example, classical natural deduction systems [2, 3]. None of these supply what we want of a semantics: a faithful representation of the structure of cut-reduction, with concrete models that illuminate the proof theory.

Of the numerous recent candidates, Führmann and Pym [4–6] seem to give the best synthesis of those two requirements. Their *classical categories* model classical proofs

* Supported by a University of Bath studentship

as morphisms in a special kind of poset-enriched linearly distributive category. The poset enrichment models cut-reduction, so that whenever a proof Φ cut-reduces to a proof Ψ , we have $[\Phi] \leq [\Psi]$, where $[\Phi]$ is the denotation of Φ in the category. Classical categories admit all commuting conversions of the sequent calculus as equalities; the ordering represents certain cuts against structural rules. The semantics sheds new light on the status of the MIX law in classical logic: the two obvious ways of defining it (as a cut against either \perp or \top) acquire the same denotation, and the rule can be eliminated.

Meanwhile, work by Brünnler [7], building on the calculus of structures of Guglielmi [8, 9] has led to a new proof system (SKSg) for classical logic that promises a finer-grained analysis of proofs. The calculus of structures uses “deep inference” (inference rules operating arbitrarily deeply inside formulæ) to dispense with the tree-like structure of sequent proofs. The result is a precise duality: inference rules are unary, and come in dual pairs (or are self dual). A derivation can be dualized by “inverting” each inference rule to obtain a proof of the contrapositive derivation

Much of the work in deep inference at present lies in finding new proof-theoretic systems (“Formalism A” [10] and “Formalism B” [11]) that generalize the calculus of structures by eliminating “bureaucracy”: proofs which are essentially the same but differ syntactically. For example,

$$\frac{(A, [B, C])}{[(A, B), C]}_s \quad \text{and} \quad \frac{(A, [B, C])}{[(A', B), C]}_\Phi \quad \frac{(A', [B, C])}{[(A', B), C]}_s \quad (2)$$

would, in Formalism B, be represented by the same syntactic structure.

Table 1. System LK

$\frac{}{\varphi \vdash \varphi} Ax_{LK}$	$\frac{\Gamma \vdash \Delta, \varphi \quad \Gamma', \varphi \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} CUT_{LK}$	$\frac{\Gamma, \varphi, \varphi \vdash \Delta}{\Gamma, \varphi \vdash \Delta} CL$	$\frac{\Gamma \vdash \Delta, \varphi, \varphi}{\Gamma \vdash \Delta, \varphi} CR$
$\frac{}{\vdash \top} \top L$	$\frac{}{\perp \vdash} \perp R$	$\frac{\Gamma \vdash \Delta}{\Gamma, \varphi \vdash \Delta} WL$	$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \varphi} WR$
$\frac{\Gamma \vdash \Delta, \varphi \quad \Gamma' \vdash \Delta', \psi}{\Gamma, \Gamma' \vdash \Delta, \Delta', \varphi \wedge \psi} \wedge R$	$\frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, \varphi \wedge \psi \vdash \Delta} \wedge L$	$\frac{\Gamma, \varphi \vdash \Delta}{\Gamma \vdash \neg \varphi, \Delta} \neg L$	$\frac{\Gamma \vdash \varphi, \Delta}{\Gamma, \neg \varphi \vdash \Delta} \neg R$
$\frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash \varphi \vee \psi, \Delta} \vee R$	$\frac{\Gamma, \varphi \vdash \Delta \quad \Gamma', \psi \vdash \Delta'}{\Gamma, \Gamma', \varphi \vee \psi \vdash \Delta, \Delta'} \vee L$	$\frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, \psi, \varphi \vdash \Delta} EL$	$\frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash \psi, \varphi, \Delta} ER$

This paper sets out to show, using classical categories as a case study, that the well established notions of proof equality arising from categorical logic may be applied with

great ease to deep inference formalisms. The author hopes that these insights will act as a guide to those designing future formalisms.

- In §4 we give an inequational theory on proofs in SKSg, making it a classical category. This category:
 - Captures the essence of Formalism A as bifunctionality of the connectives;
 - Captures the essence of Formalism B by requiring that certain inference rules are (lax) natural;
 - Does not collapse to a boolean algebra;
 - Is a model of the two-sided classical sequent calculus, LK.
- We identify two distinct forms of cut in SKSg (§5.1):
 1. A cut equivalent to that of a one-sided sequent system; and
 2. A cut equivalent to that of a two-sided sequent system.
 We show that the first is well-definable in terms of the second.
- We give a characterization of intuitionistically valid calculus of structures derivations (Lemma 3). Using this, we give a refinement of the Craig interpolation lemma for propositional classical logic (Corollary 1).

2 Classical categories

First, we introduce *classical categories* [4–6]. We do not give details of the diagrams required for coherence (although many can be inferred later from the equational parts of the theory given below for SKSg). Classical categories are a sound and complete semantics of the classical sequent calculus, developed by Führmann and Pym, that unlike previous attempts can distinguish between proofs and model all cut-reductions. A classical category is a poset-enriched category with extra structure, including two symmetric monoidal products \otimes and \oplus , with units 1 and 0. Mediating between these functors is a natural transformation $\delta : A \otimes (B \oplus C) \rightarrow (A \otimes B) \oplus C$, making it a symmetric linearly distributive category ([12]). Various coherence conditions hold for these categories, which in addition to giving coherence model the structure of cut-reduction in a two-sided sequent calculus. The category also has for each object A a complement A^\perp , and morphisms $A^\perp \otimes A \rightarrow 0$ (contradiction) and $1 \rightarrow A \oplus A^\perp$ (excluded middle) which, with certain coherence conditions make it a symmetric linearly distributive category with negation. It is, by virtue of this, equivalent to a $*$ -autonomous category, and a model of multiplicative linear logic, in the sense that objects model propositions and morphisms model proofs. We model $\phi \wedge \psi$ inductively as $[\phi] \otimes [\psi]$, and similarly disjunction is modelled by \oplus .

How the cut rule is modelled is of particular interest. Given a proof Φ of $\Gamma \vdash \Delta$, ϕ and a proof Ψ of $\phi, \Gamma' \vdash \Delta'$, with denotations $\mathcal{C}[\Phi]$, $\mathcal{C}[\Psi]$ in a classical category \mathcal{C} , we denote the cutting together of these two proofs by:

$$[\Gamma] \otimes [\Gamma'] \xrightarrow{\mathcal{C}[\Phi] \otimes \mathbf{id}} ([\Delta] \oplus [\phi]) \otimes [\Gamma'] \xrightarrow{\delta'} [\Delta] \oplus ([\phi] \otimes [\Gamma']) \xrightarrow{\mathbf{id} \oplus \mathcal{C}[\Psi]} [\Delta] \oplus [\Delta'], \quad (3)$$

where δ' is the evident morphism obtained from δ and symmetric monoidal isomorphisms, and \mathbf{id} is identity. In this setting, cut is a generalized composition.

In addition to this structure, a classical category carries the structure necessary to model weakening and contraction on the right. Every object has a *symmetric monoid* — a multiplication $\nabla_A : A \oplus A \rightarrow A$ and unit $[]_A : 0 \rightarrow A$, satisfying equations that state the associativity and symmetry of ∇_A and the neutrality of $[]_A$. In addition, we require that $\nabla_{A \oplus B}$ is definable pointwise; that $[]_{A \oplus B}$ is definable pointwise; and that $[]_0 = \text{id}$. We say a symmetric monoidal category *has symmetric monoids* provided it satisfies these three laws. Dually, a classical category has symmetric comonoids given by $\Delta_A : A \rightarrow A \otimes A$ and $\langle \rangle_A : A \rightarrow 1$ for weakening and contraction on the right.

Table 2. Inequalities of a classical category

$$\begin{array}{ccc}
\begin{array}{ccc}
A \oplus C & \xrightarrow{\Delta} & (A \oplus C) \otimes (A \oplus C) \\
\text{id} \oplus \Delta \downarrow & \leq & \hat{\delta} \downarrow \\
A \oplus (C \otimes C) & \xleftarrow{\nabla \oplus \text{id}} & (A \oplus A) \oplus (C \otimes C)
\end{array} & &
\begin{array}{ccc}
A \oplus C & \xrightarrow{\langle \rangle} & 1 \\
\text{id} \oplus \langle \rangle \downarrow & \leq & \cong \downarrow \\
A \oplus 1 & \xleftarrow{[] \oplus \text{id}} & 0 \oplus 1
\end{array} \\
\Delta \nabla & & \langle \rangle []
\end{array}$$

$$\begin{array}{ccc}
\begin{array}{ccc}
A \otimes C & \xleftarrow{\nabla} & (A \otimes C) \oplus (A \otimes C) \\
\text{id} \otimes \nabla \uparrow & \leq & \check{\delta} \uparrow \\
A \otimes (C \oplus C) & \xrightarrow{\Delta \oplus \text{id}} & (A \otimes A) \otimes (C \oplus C)
\end{array} & &
\begin{array}{ccc}
A \otimes C & \xleftarrow{[]} & 0 \\
\text{id} \otimes [] \uparrow & \leq & \cong \uparrow \\
A \otimes 0 & \xrightarrow{\langle \rangle \oplus \text{id}} & 1 \otimes 0
\end{array} \\
\nabla \Delta & & [] \langle \rangle
\end{array}$$

Definition 1. A classical category is an order-enriched symmetric linearly distributive category with negation such that:

1. The symmetric monoidal category $(\mathcal{C}, \oplus, 0)$ has symmetric monoids;
2. The symmetric monoidal category $(\mathcal{C}, \otimes, 1)$ has symmetric comonoids;
3. The object indexed families of maps $\Delta_A, \nabla_A, \langle \rangle_A$ and $[]_A$ are lax natural transformations, in the sense that for every morphism f we have

$$\begin{array}{ccc}
\Delta \circ f \leq (f \otimes f) \circ \Delta & & f \circ \nabla \leq \nabla \circ (f \oplus f) \\
\langle \rangle \circ f \leq \langle \rangle & & f \circ [] \leq []
\end{array}$$

4. The inequalities in Table 2 hold, where $\hat{\delta}$ and $\check{\delta}$ are the evident morphisms obtained from δ and symmetric monoidal isomorphisms
5. Composition of morphisms, and the functors \oplus, \otimes are monotonic in all arguments.

Example 1. \mathbf{Rel}_\otimes is a classical category with objects sets and morphisms binary relations, in which both \otimes and \oplus are given by the set theoretic product. Both 0 and 1 are given by the singleton set $\{*\}$. Negation is identity on objects, and the excluded middle on a set A is the relation $\{(*, (x, x)) : x \in A\}$ from $\{*\}$ to $A \times A$. The map ∇_A is $\{((x, x), x) : x \in A\}$ and $[]_A$ is $\{(*, x) : x \in A\}$. The order on hom-sets is set-theoretic inclusion of relations.

Example 2. A boolean lattice \mathbf{B} is a classical category, with meet as \otimes and join as \oplus .

Example 3. If \mathcal{C} and \mathcal{C}' are classical categories, then so are \mathcal{C}^{op} and $\mathcal{C} \times \mathcal{C}'$. In particular, the product of a classical category with non-trivial hom-sets (e.g. \mathbf{Rel}_{\otimes}) and a boolean algebra \mathcal{B} is a non-compact, non-trivial classical category.

Before stating soundness and completeness, we need a notion of theory. For the sake of simplicity we consider only pure logic.

Definition 2. A sequent theory over a collection of atoms \mathcal{A} is a set of inequalities $\Phi \preceq \Psi$, where both Φ and Ψ are proofs of a sequent $\Gamma \vdash \Delta$ over \mathcal{A} , such that:

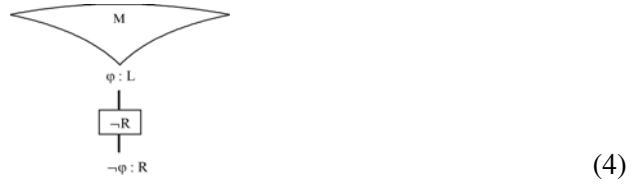
1. The relation \preceq is reflexive, transitive, and compatible (i.e. all inference rules are “monotonic w.r.t. \preceq ”);
2. The relation holds for both directions of the usual cut-reduction rules for eliminating logical cuts. It also holds in both directions for a number of coherence rules, including axiom expansions (for details, see [4]);
3. The usual rules for eliminating cut against weakening and contraction hold in only one direction: from redex to reduct.

Führmann and Pym prove the following [4]:

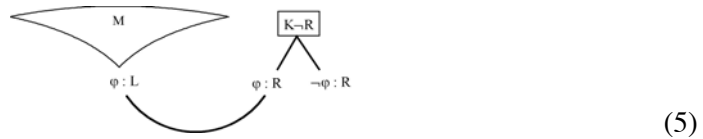
Theorem 1 (Soundness [4]). Let \mathcal{P} be a set of proofs over a set of atoms \mathcal{A} . Then for every interpretation $\mathcal{C}[_]$ in a classical category \mathcal{C} , the judgements $\Phi \preceq \Psi$ such that $[\Phi] \leq [\Psi]$ form a sequent theory.

Theorem 2 (Completeness [4]). Let \mathcal{T} be a sequent theory, and suppose that $[\Phi] \leq [\Psi]$ holds for every interpretation in a classical category \mathcal{C} . Then $\Phi \preceq \Psi$ is in \mathcal{T} .

The proof in [4] of the latter relies heavily on the use of proof nets for classical logic, introduced by Robinson [13], to construct a term model. These proof-nets are two-sided, and correspond very closely to the sequent calculus. (In [6], equivalent proof nets in style of Blute et. al. [14] are used, since they are closer to the categorical structure). The calculations involved are significantly simplified, since proof nets admit commuting conversions as syntactic equalities. One other very useful refinement of the presentation of LK (which is best presented in proof nets) is the use of cut against constants instead of rules to present $\neg L$, $\neg R$, and a variety of other proof rules: the proof net



which corresponds to an application of the LK rule $\neg R$ to a derivation of a sequent $\Gamma, \varphi \vdash \Delta \neg R$, is taken to be shorthand for



which corresponds to that same LK derivation followed by cut against the (canonical) proof of the sequent $\vdash \varphi, \neg\varphi$. The $\wedge R$ rule is modelled by cut against a proof net for $A, B \vdash A \wedge B$, and similarly for $\vee L$.

3 The calculus of structures for classical logic

We summarize Brünnler’s SKSg [7], a deep symmetric system in the style of the calculus of structures for classical propositional logic. We give translations into a single-sided sequent system. Finally, we discuss Guglielmi’s notions of “Formalism A” and “Formalism B” [10, 11], which generalize the calculus of structures.

3.1 System SKSg

The *calculus of structures* [8, 9] is a formalism that employs deep inference. The sequent calculus LK does not exhibit deep inference. Consider the sequent $\vdash C \wedge (A \vee A), B, B$. We can apply contraction across the comma, but not across the disjunction. Semantically, comma on the right is the same as disjunction: the syntactic distinction is required for completeness of the sequent calculus. The calculus of structures removes this distinction, and any calculus of structures inference rule operates arbitrarily deeply in a formula: this is deep inference. Proofs in the calculus of structures are not trees as in the sequent calculus, but are linear.

We now present the syntax for classical logic in a deep inference setting:

Definition 3. *Given a set V of propositional variables, we consider formulae given by the grammar*

$$F ::= f \mid \mathfrak{t} \mid v \mid [F, \dots, F] \mid (F, \dots, F) \mid \bar{v},$$

where v is a variable, \mathfrak{t} and f are true and false, $[..]$ and $(..)$ are disjunction and conjunction, and \bar{v} is the negation of v (negation on general formulae being inductively defined). We use $\{\}$ to denote a hole in a formula: a context $S\{\}$ is a formula with one occurrence of the hole, and $S\{R\}$ is that context with the hole filled with a formula R . We will omit the notation for the empty context where it can be inferred, for example writing $S(A, B)$ for $S\{(A, B)\}$

We take associativity and commutativity of connectives and the usual behaviour of units, to hold at the level of syntactic equivalence. The set S of structures is the quotient of the set F by the smallest relation containing these syntactic equivalences and closed under formation of formulae from contexts.

The deep symmetric system SKSg for classical propositional logic is given in Table 3. Each rule is unary, and is either self dual or comes as one of a dual pair. A *derivation* is a sequence of applications of rules. A *proof* of A is a derivation from \mathfrak{t} to A .

The rules correspond closely to those of the one-sided sequent system GS1p. (GS1p differs from LK in its axiom and cut rules, as given in Table 4. The other rules of GS1p are the rules $\wedge R$, $\vee L$, CL and WR of LK, restricted to right handed sequents.) For example:

Table 3. System SKSg

$$\begin{array}{c}
\frac{S\{t\}}{S[R, \bar{R}]} \text{i}\downarrow \qquad \frac{S(R, \bar{R})}{S\{f\}} \text{i}\uparrow \\
\frac{S[R, R]}{S\{R\}} \text{c}\downarrow \qquad \frac{S([R, U], T)}{S[(R, T), U]} \text{s} \qquad \frac{S\{R\}}{S(R, R)} \text{c}\uparrow \\
\frac{S\{f\}}{S[R]} \text{w}\downarrow \qquad \frac{S(R)}{S\{t\}} \text{w}\uparrow
\end{array}$$

$$\frac{\frac{\vdash \Gamma, \varphi \quad \vdash \Gamma', \psi}{\vdash \Gamma, \Gamma', \varphi \wedge \psi} \wedge R \rightarrow \frac{([\Gamma, \varphi], [\Gamma', \psi]) \text{s}}{[\Gamma, (\varphi, [\Gamma', \psi])] \text{s}}}{[\Gamma, \Gamma', (\varphi, \psi)] \text{s}} \text{s}$$

and

$$\frac{\frac{\vdash \Gamma, \varphi \quad \vdash \Gamma', \neg \varphi}{\vdash \Gamma, \Gamma'} \text{CUT} \rightarrow \frac{([\Gamma, \varphi], [\Gamma', \bar{\varphi}]) \text{s}}{[\Gamma, (\varphi, [\Gamma', \bar{\varphi}])] \text{s}} \text{s}}{\frac{[\Gamma, \Gamma', (\varphi, \bar{\varphi})] \text{s}}{[\Gamma, \Gamma', f]} \text{i}\uparrow} \text{i}\uparrow} = \frac{[\Gamma, \Gamma']}{[\Gamma, \Gamma']} =$$

The cases for axiom, weakening and contraction are similar. We now have the essential tools for the proof of the following theorem from [7]:

Theorem 3 (GS1p to SKSg). *For every proof Φ of $\vdash \Delta$ in GS1p there exists an SKSg proof of Δ that does not involve $\text{c}\uparrow$ or $\text{w}\uparrow$, and has the same number of occurrences of $\text{i}\uparrow$ as there are occurrences of cut in Φ .*

Remark 1. Only the translation of the cut rule requires an application of $\text{i}\uparrow$: hence the preservation result. Since the $\text{i}\uparrow$ rule is so closely related to the cut rule in GS1p, it is often referred to as the cut rule for the calculus of structures.

The following theorem (from [7]) gives us a partial converse — partial, since the number of cuts is not preserved:

Theorem 4 (SKSg to GS1p). *For every proof of a structure S in SKSg there exists a proof of $\vdash S$ in GS1p.*

Given a proof in SKSg which begins with t , we can translate it into a GS1p proof. Since cut is admissible in GS1p, we can obtain a cut-free proof. Translating that proof back into SKSg, the proof we obtain contains no instance of $\text{i}\uparrow$. This proves the following, (for which there is also a direct proof [15, 7]):

Theorem 5 (Cut-elimination). *Any structure provable from t in SKSg is provable without the use of $\text{i}\uparrow$ (and without $\text{w}\uparrow$ or $\text{c}\uparrow$).*

3.2 “Formalism A” and “Formalism B”

The calculus of structures allows more freedom in the application of inference rules. As a result, it lays bare more bureaucracy than many other systems. Formalisms A [10] and B [11] are suggestions for ways to design new systems which lack this bureaucracy. Here we will describe the ideas behind these systems and the types of bureaucracy they avoid. Later, we will ask what light a categorical semantics sheds on the issues involved.

When dealing with the sequent calculus, one has to deal with an enormous number of commuting conversions; this is why proof nets, which validate these conversions as identities, are so useful. In the calculus of structures, the situation is made worse. For example, in the sequent calculus derivation,

$$\frac{\frac{\vdots \Phi}{\vdash \Gamma, \varphi} \quad \frac{\vdots \Psi}{\vdash \Gamma', \psi}}{\vdash \Gamma, \Gamma', \varphi \wedge \psi} \wedge R \quad (6)$$

the two sub-proofs act in parallel. In the calculus of structures, the same derivation can be written by applying the sub-proofs sequentially, as either

$$\frac{\frac{\frac{\frac{t}{(t, t)}}{([\Gamma, \varphi], t)} \Phi}{([\Gamma, \varphi], [\Gamma, \psi]) } \Psi}{[\Gamma, (\varphi, [\Gamma', \psi])] } S}{[\Gamma, \Gamma', (\varphi, \psi)] } S \quad \text{or} \quad \frac{\frac{\frac{\frac{t}{(t, t)}}{(t, [\Gamma, \psi]) } \Psi}{([\Gamma, \varphi], [\Gamma, \psi]) } \Phi}{[\Gamma, (\varphi, [\Gamma', \psi])] } S}{[\Gamma, \Gamma', (\varphi, \psi)] } S, \quad (7)$$

or by interleaving the inference rules from each proof, in a number of possible ways. All of these should, morally, represent the same proof. Formalism A would be a system in which applications of inference rules can be made in parallel, as they are in the sequent calculus or proof nets.

Formalism B caters to another, more subtle, form of bureaucracy. Suppose we have a derivation Φ of A' from A in the calculus of structures. The syntactic objects

$$\frac{(A, [B, C])}{[(A, B), C]} S \quad \text{and} \quad \frac{(A, [B, C])}{[(A', B), C]} \Phi \quad (8)$$

should denote the same proof. The solution posed by Guglielmi [11] is to have inference rules acting not on formulæ, or on structures, but on derivations. Thus

$$\frac{(\Delta, [\Delta', \Delta''])}{[(\Delta, \Delta'), \Delta'']} S \quad (9)$$

would be the canonical expression for the both expressions in (8). Recent progress has also been made on a version of the formalism called *wired deduction*.

Table 4. System GS1p: Axiom and Cut

$$\frac{}{\vdash \varphi, \bar{\varphi}} Ax_{GS1} \quad \frac{\vdash \Gamma, \varphi \quad \vdash \Gamma', \bar{\varphi}}{\vdash \Gamma, \Gamma'} CUT_{GS1p}$$

4 SKSg forms a classical category

In this section, we show that SKSg admits an inequational theory such that it forms a classical category.

We should note that deep inference proof theory regards each formula/structure as its own identity derivation.

Definition 4 (The theory \mathcal{T}).

The theory \mathcal{T} is a set of expressions $\Phi \leq \Psi$, where Φ and Ψ are derivations in the calculus of structures. We write \equiv for the symmetric closure of \leq . We give the inequations in a shallow form, with the understanding that the theory is closed under formation of contexts.

We deal first with permutation of non-interfering rules. Given two inference rules p and q , the following holds:

$$\frac{\frac{(A, A')}{(A, B')} q}{(B, B')} p \equiv \frac{\frac{(A, A')}{(B, A')} p}{(B, B')} q \quad (:= (p, q)), \quad (10)$$

and similarly for disjunction.

The nesting of derivations and switch is also part of our theory:

$$\frac{\frac{(A, [B, C])}{(E, [F, G])} (p, [q, r])}{[(E, F), G]} s \equiv \frac{\frac{(A, [B, C])}{[(A, B), C]} s}{[(E, F), G]} [(p, q), r], \quad (11)$$

along with several equalities relating different ways of nesting switches, for example:

$$\frac{(A, B, [C, D])}{[(A, B, C), D]} s \equiv \frac{\frac{(A, B, [C, D])}{(A, [(B, C), D])} s}{[(A, B, C), D]} s. \quad (12)$$

The following rule and its dual govern interactions between negations:

$$\frac{\frac{\frac{A}{(A, \mathbf{t})}}{(A, [\bar{A}, A])} i\downarrow}{[(A, \bar{A}), A]} s \equiv \frac{\frac{A}{A} \mathbf{id}}{[f, A]} i\uparrow \quad (13)$$

The remaining equations are given in Table 5 and the inequations of the theory are given in Table 6.

Table 5. Equalities: weakening and contraction

$$\begin{array}{c}
\frac{S[[P, Q], [P, Q]]}{S[[P, P], [Q, Q]]} = \\
\frac{\frac{S[[P, P], [Q, Q]]}{S[[P, P], Q]} \downarrow c}{S[P, Q]} \downarrow c \equiv \frac{S[[P, Q], [P, Q]]}{S[P, Q]} \downarrow c
\end{array}
\qquad
\begin{array}{c}
\frac{[[A, A], A]}{[A, A]} \text{c}\downarrow \equiv \frac{[A, [A, A]]}{[A, A]} \text{c}\downarrow \\
\frac{[A, A]}{A} \text{c}\downarrow \equiv \frac{[A, A]}{A} \text{c}\downarrow
\end{array}$$

$$\begin{array}{c}
\frac{A}{[A, f]} = \\
\frac{[A, A]}{A} \text{w}\downarrow \equiv - \mathbf{id} \\
\frac{[A, A]}{A} \text{c}\downarrow
\end{array}
\qquad
\begin{array}{c}
\frac{f}{[f, f]} = \\
\frac{[f, f]}{[A, B]} [\text{w}\downarrow, \text{w}\downarrow] \equiv \frac{f}{[A, B]} \text{w}\downarrow
\end{array}
\qquad
\begin{array}{c}
\frac{f}{- \text{w}\downarrow} \equiv - \mathbf{id} \\
\frac{f}{f}
\end{array}$$

We now prove the main technical theorem of the paper.

Since we have an identity derivation on structures, we can form a category from SKSg, with structures as objects and derivations between two structures as morphisms; in particular, each inference rule is a morphism. Composition is given by concatenation. We have extended conjunction and disjunction to the inference rules (Equation 10) and we can extend that definition inductively to all derivations. We have required that the connectives preserve identity derivations. By induction on the length of derivation, we can show that both are bifunctorial. That they are monoidal follows from the syntactic equalities of associativity, symmetry and units (i.e, they are *strong* monoidal products). SKSg is a symmetric linearly distributive category, δ being given by the switch rule, which is natural by (11). The required coherences are those typified by (12) (for further details of the coherences see [12]). Along with (13), $i\uparrow$ and $i\downarrow$ are precisely what is required to model negation in a linearly distributive category. The equations in Table 5 show that $c\downarrow$ and $w\downarrow$ form a symmetric monoid $(\nabla, [])$ (and dually). Finally, the inequations in Table 6 plus their duals give SKSg the structure of a classical category.

Theorem 6. *SKSg with \mathcal{T} forms a classical category.*

Remark 2. The structure we have given to SKSg models precisely the equalities of Formalisms A and B, and we would expect these formalisms (when they appear) to form a classical category. Conversely, by looking at classical categories we can infer what additional structure these formalisms will need to include. Categorically, Formalism A amounts to bifunctoriality of disjunction and conjunction. (Deep inference in the calculus of structures is the ordinary functoriality of the connectives with respect to each argument.) The behaviour of Formalism B is modelled by naturality of switch (and lax naturality of the structural rules).

Table 6. Inequalities: weakening and contraction

$$\begin{array}{ccc}
\frac{A}{(A, A)} \text{c}\uparrow & \geq & \frac{A}{B} p \\
\frac{(p, p)}{(B, B)} & & \frac{A}{(B, B)} \text{c}\uparrow \\
\frac{A}{(A, A)} \text{c}\uparrow & & \frac{A}{B} p \\
\frac{(p, p)}{(B, B)} & & \frac{A}{(B, B)} \text{c}\uparrow
\end{array}
\qquad
\begin{array}{ccc}
\frac{f}{A} \text{w}\downarrow & \leq & \frac{f}{B} \text{w}\downarrow \\
\frac{A}{B} p & & \frac{f}{B} \text{w}\downarrow
\end{array}$$

$$\begin{array}{ccc}
\frac{[A, C]}{[A, (C, C)]} \text{c}\uparrow & \leq & \frac{[A, C]}{([A, C], [A, C])} \text{c}\uparrow \\
\frac{[A, C]}{[A, (C, C)]} \text{c}\uparrow & & \frac{([A, C], [A, C])}{[A, A, (C, C)]} s' \\
\frac{[A, C]}{[A, (C, C)]} \text{c}\uparrow & & \frac{[A, A, (C, C)]}{[A, (C, C)]} \text{c}\downarrow
\end{array}
\qquad
\begin{array}{ccc}
\frac{(A, f)}{(t, f)} \text{w}\uparrow & & \frac{(A, f)}{(A, B)} \text{w}\downarrow \\
\frac{(A, f)}{(t, f)} \text{w}\uparrow & = & \frac{(A, f)}{(A, B)} \text{w}\downarrow \\
\frac{(A, f)}{(t, f)} \text{w}\uparrow & & \frac{f}{(A, B)} \text{w}\downarrow
\end{array}$$

Theorem 7. *SKSg + T is not equivalent to a boolean algebra; it is a non-trivial.*

Proof. The quotient of the category \mathbf{Rel}_\otimes by the equivalence relation \mathcal{R} , which identifies objects along the symmetric monoidal isomorphisms, is a nontrivial classical category. Define a functor \mathcal{G} from $\mathbf{Rel}_\otimes/\mathcal{R}$ inductively on objects by mapping each propositional variable to a unique set, and inductively on morphisms by mapping each inference rule to the corresponding classical category morphism in \mathbf{Rel}_\otimes . This is well defined, as \mathbf{Rel}_\otimes is monoidal. Then, for any proof Φ of SKSg, $\mathcal{G}(f \circ i\downarrow) \neq i\downarrow$.

Remark 3. The need to quotient \mathbf{Rel}_\otimes arises from the equalities in SKSg. If, instead, we add new (invertible) rules corresponding to the symmetric monoidal isomorphisms, we obtain a category equivalent to the category of proof nets for classical logic.

5 Insights into SKSg and the sequent calculus

5.1 The meaning of cut in SKSg

We have seen that when we refer to cut in SKSg, we mean the rule $i\uparrow$. This corresponds well to the definition of cut in a one-sided sequent system such as GS1p, and can be eliminated, either directly or via a translation into GS1p. However, in the previous section we saw SKSg as a classical category, and thereby a model of the two sided system LK. The order given on proofs is a model of cut-reduction, and yet it is independent of the rule $i\uparrow$. In this section we explore the relationship between these two notions of cut.

First, notice that any sequent in GS1p is a sequent in LK; we show that we can embed any GS1p proof into LK.

Lemma 1. *Any proof Φ in GS1p can be transformed into a proof in LK.*

Proof. Suppose we have a GS1p proof Φ . Since GS1p and LK differ only on cut and axiom, we transform only instance of these rules.

We replace each occurrence of an axiom

$$\frac{}{\vdash \neg\varphi, \varphi} Ax_{GS1p} \quad \text{with} \quad \frac{\frac{}{\varphi \vdash \varphi} Ax_{LK}}{\vdash \neg\varphi, \varphi} \neg R \quad (14)$$

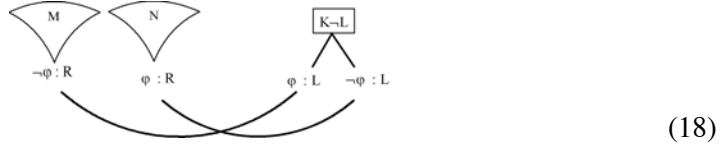
The case of cut is rather more complicated. There are three obvious ways of defining the GS1p cut rule in LK:

$$\frac{\frac{\frac{}{\varphi \vdash \varphi} Ax_{LK}}{\neg\varphi, \varphi \vdash} \neg L}{\vdash \neg\varphi, \Delta'} \text{Cut}_{LK}, \quad \frac{\vdash \varphi, \Delta}{\vdash \Delta, \Delta'} \text{Cut}_{LK}, \quad (15)$$

$$\frac{\frac{\frac{\vdash \varphi, \Delta}{\neg\varphi, \Delta'} \neg L}{\vdash \neg\varphi, \Delta'} \text{Cut}_{LK}, \quad \frac{\vdash \varphi, \Delta}{\neg\varphi \vdash \Delta} \neg L}{\vdash \Delta, \Delta'} \text{Cut}_{LK}, \quad (16)$$

$$\frac{\frac{\frac{\vdash \neg\varphi, \Delta' \quad \vdash \varphi, \Delta}{\vdash (\neg\varphi \wedge \varphi), \Delta, \Delta'} \wedge R}{\vdash \Delta, \Delta'} \text{Cut}_{LK}, \quad \frac{\frac{\frac{}{\varphi \vdash \varphi} Ax_{LK}}{\neg\varphi, \varphi \vdash} \neg L}{\neg\varphi \wedge \varphi \vdash} \wedge L}{\vdash \Delta, \Delta'} \text{Cut}_{LK}. \quad (17)$$

These definitions are coherent, in the sense that they are identified in our semantics. Eliminating the logical cut in (17), and shifting to proof nets, with negation as cut against a constant as in (5, §2), all three are represented by the proof net:



As SKSg is a classical category, the usual notion of interpretation gives us a translation from LK to SKSg. The following is a corollary of soundness:

Theorem 8 (From LK to SKSg). *For every proof Φ of a sequent $\phi \vdash \psi$ in LK, there is a derivation $[\Phi]$ from ϕ to ψ in SKSg. For any cut-reduction $\Phi \preceq \Psi$ in LK, there is a corresponding inequality in SKSg + \mathcal{T}*

Remark 4. This translation maps a proof ψ , containing no occurrences of $\neg L$, to a “cut-free” proof in the sense of SKSg, regardless of the number of instances of cut in ψ . In particular, consider the following lemma:

Lemma 2. *The sequent $\vdash \Gamma$, if provable in LK, is provable without recourse to rules operating on the left hand side of the sequent.*

The SKSg cut-elimination theorem (Theorem 5) is a corollary of this result.

Each inference rule in SKSg, when read from top to bottom, is a valid entailment $\phi \rightarrow \psi$ in classical logic, and there is therefore a cut-free proof of $\phi \vdash \psi$. Composing these proofs using the cut rule, we obtain:

Theorem 9 (From SKSg to LK). *For every derivation Φ from A to B in SKSg there exists an LK proof of the sequent $A \vdash B$, with a number of instances of cut equal to the number of rule applications in Φ minus one.*

Remark 5. This theorem embodies the notion of cut as composition. A proof is cut free in this sense only if it is an instance of an inference rule: in particular $i\uparrow$ translates to a cut-free proof in LK.

Remarks 4 and 5 show that the translations between LK and SKSg respect the notion of cut given by composition and the ordering on proofs, but not that given by instances of the $i\downarrow$ rule. The presence of this rule is clearly not sufficient to identify non-normal derivations of SKSg. We can, however, generalise the notion of a normal proof to a normal derivation in the following way:

Definition 5. *A derivation is normal if it contains no \uparrow rule below a \downarrow rule.*

(This observation was also made recently (and independently) by Brünnler.) This definition agrees with the equivalent notion for LK, in the sense that any normal LK derivation (or proof net) translates to a normal SKSg derivation (taking care to note that since we have (implicit) cuts against constants in our proof nets, a normal proof net is one without *essential cuts*). The notion of a normal SKSg proof is easily seen to be a special case of this definition, since any \uparrow rule with premise t has conclusion equivalent to t . Notice also that each equational law in \mathcal{T} involves only \uparrow or \downarrow rules: That is, application of these rules to a normal derivation yields another normal derivation. Meanwhile, the lax naturalities clearly show that moving an $w\uparrow$ or $c\uparrow$ rule above some other derivation generates some change in denotation: The change corresponds to a move closer to a normal form. By switching to a local presentation of the inequalities on proofs (i.e. an inequational theory on SKSg) the author hopes in the future to remove these inequalities from the theory and understand cut-reduction purely as moving \uparrow rules above \downarrow rules.

5.2 Intuitionistic validity and decomposition

Which proofs in SKSg are intuitionistically valid? Considering each inference rule as an implication, we find that only $i\downarrow$ is invalid. (Recall that negation is derived, and therefore all the formulae in a derivation are assumed to be in negation normal form. This excludes de Morgan duality from the list of syntactic equivalences we include. All the other equivalences are intuitionistically valid). We have, therefore:

Lemma 3. *A calculus of structures derivation is intuitionistically valid if it contains no application of the $i\downarrow$ rule.*

We call a derivation which has this property *intuitionistic*, and a derivation which has no instance of $i\uparrow$ *co-intuitionistic*. (This terminology derives from the fact that if there is a co-intuitionistic derivation S from A to B , the negation normal form (*nnf*) of \bar{B}

intuitionistically entails the nnf of \bar{A} , and the de-Morgan dual of S is such an intuitionistic derivation.) In particular, the \downarrow fragment of SKSg is co-intuitionistic. Consider the following decomposition lemma from [7]:

Theorem 10. *For every derivation of B from A in SKSg there is a derivation that is of the form*

$$\begin{array}{c} A \\ \vdots \\ SKSg \setminus \{i\downarrow, w\downarrow\} \\ C \\ \vdots \\ SKSg \setminus \{i\uparrow, w\uparrow\} \\ B \end{array}$$

Using that the first phase of the above proof introduces no new propositional variables, and the second phase introduces no new propositional variables when viewed bottom up, Brännler obtains the Craig interpolation lemma [7]; using Lemma 3, we may now strengthen that result:

Corollary 1 (Refined Craig interpolation). *For all propositional formulæ ϕ and ψ in nnf, if ϕ implies ψ then there is an interpolant γ such that ϕ intuitionistically implies γ , γ co-intuitionistically implies ψ , and γ contains only propositional variables common to A and B . Given a normal derivation S from ϕ to ψ , we can choose the intuitionistic and co-intuitionistic derivations such that their composition equals S*

Proof. A normal derivation is already $\uparrow - \downarrow$ factored, and clearly the composition of these factors is the original derivation.

McKinley and Brännler have observed this behaviour in the sequent calculus (where it can be demonstrated by an additional observation on top of the usual structural induction), but the property is (in the opinion of the author) perspicuous in the calculus of structures.

Acknowledgements. The author is grateful to David Pym and Carsten Führmann for their guidance, encouragement and criticism and to Kai Brännler and Alessio Guglielmi for clarifications and the motivation for this work. Special thanks to Brännler for repairing a serious error from the first draft of this paper. The author is funded by a studentship from the University of Bath. Figures were typeset using Paul Taylor's *diagrams* and *prooftree* packages.

References

1. Girard, J.Y., Lafont, Y., Taylor, P.: Proofs and Types. Cambridge University Press (1989)
2. Parigot, M.: $\lambda\mu$ -calculus: An algorithmic interpretation of classical natural deduction. In LPAR'92, Springer (1992) 190–201
3. Pym, D., Ritter, E.: On the semantics of classical disjunction. JPAA **159** (2001) 315–338
4. Führmann, C., Pym, D.: Order-enriched categorical models of the classical sequent calculus. To appear, JPAA, 2005
5. Führmann, C., Pym, D.: On the Geometry of Interaction for Classical Logic. In Proc. 19th LICS, Turku, 2004. 211-220 IEEE

6. Führmann, C., Pym, D.: On categorical models of classical logic and the geometry of interaction. (*In preparation*. Manuscript available at <http://www.cs.bath.ac.uk/~pym/dj.pdf>)
7. Brünnler, K.: Deep Inference and Symmetry in Classical Proofs. Logos Verlag, Berlin (2004)
8. Guglielmi, A.: A system of interaction and structure. To appear ACM Transactions on Computational Logic. Available at <http://iccl.tu-dresden.de/~guglielm/p/SystIntStr.pdf>.
9. Guglielmi, A.: (Deep inference and the calculus of structures) Project web page: Available at <http://alessio.guglielmi.name/res/cos/index.html>.
10. Guglielmi, A.: Formalism A. (Available at <http://iccl.tu-dresden.de/~guglielm/p/AG11.pdf>)
11. Guglielmi, A.: Formalism B. (Available at <http://iccl.tu-dresden.de/~guglielm/p/AG13.pdf>)
12. Cockett, J.R.B., Seely, R.A.G.: Weakly distributive categories. In Applications of Categories in Computer Science: Proceedings LMS Symp., Durham, UK, 20–30 July 1991. Volume 177. Cambridge University Press, Cambridge (1992) 45–65
13. Robinson, E.: Proof Nets for Classical Logic. *J. Logic Computat.* **13** (2003) 777–797
14. Blute, R., Cockett, J.R.B., Seely, R.A.G., Trimble, T.H.: Natural deduction and coherence for weakly distributive categories. *JPA* **13** (1996) 229–296
15. Brünnler, K.: Atomic cut elimination for classical logic. In Volume 2803 LNCS, Springer-Verlag (2003) 86–97 Available at <http://www.iam.unibe.ch/~kai/Papers/AtomicCutElimination-short.pdf>.

A Game Semantics for Proof Search: Preliminary Results

Dale Miller

INRIA-Futurs and École Polytechnique
Palaiseau, France

We describe an ongoing project in which we attempt to describe a neutral approach to proof and refutation. In particular, we present a language of *neutral expressions* which contains one element for each de Morgan pair of connectives in (linear) logic. Our goal is then to describe, in a neutral fashion, what it means to prove or refute. For this, we use games where moves are described as transitions between positions built with neutral expressions. We can then relate winning a game with provability. More precisely we relate winning strategies for an expression N with proofs for a linear logic formula obtained from N : to winning $\exists\forall$ -strategies for N we associate proofs of the positive (synchronous) translation of N into logic. On the other hand, to winning $\forall\exists$ -strategies for N we associate proofs of the negative (asynchronous) translation of N into logic.

This work is joint with Alexis Saurin. This abstract is based on a paper presented at Mathematical Foundations of Programming Semantics (MFPS) 2005.

The three dimensions of proofs

Yves Guiraud

Institut de Mathématiques de Luminy, Marseille, France
guiraud@iml.univ-mrs.fr

Abstract. This document outlines a 3-categorical translation of the proofs of SKS, a deep inference formal system for classical propositional logic. This interpretation identifies two proofs that only differ by bureaucratic considerations: two proofs using the same inference rules, but in different order. Then, the notion of Penrose diagrams is extended to provide 3-dimensional representations for proofs. Finally, the 3-categorical setting allows one to express local transformations of proofs as 4-dimensional computations; this yields the first concrete example of 4-dimensional rewriting and promises new tools to prove normalization and factorisation results.

1 The two dimensions of formulas

This section is about a 2-dimensional translation of SKS formulas, heavily inspired by the one already known for terms, described and studied in (Guiraud 2004a, 2004b).

1.1 The formulas of system SKS

System SKS is a deep inference formal system for proofs of propositional logic (Brünnler 2004). It is one of the formalisms expressed in the calculus of structures style, an alternative to sequent calculus where inference rules can be applied at any depth inside formulas (Guglielmi 2005). Here a slightly alternative definition of SKS is used.

Definition 1. Let us consider a countable set A . The set \mathcal{A} of *SKS atoms* is the set of terms built on the variables in A and the signature Σ_A made of one unary operator $\overline{(\cdot)}$, *modulo* the congruence \equiv_A generated by the relation $\overline{\overline{a}} \equiv a$. Then, given another countable set V , the set \mathcal{F} of *SKS formulas* is the set of terms built on V and the signature Σ_F made of two constants \top and \perp , one unary operator from \mathcal{A} to \mathcal{F} and two binary operators \wedge and \vee , *modulo* the congruence \equiv_F generated by the following relations:

$$\begin{array}{ll} (f \wedge g) \wedge h \equiv f \wedge (g \wedge h) & (f \vee g) \vee h \equiv f \vee (g \vee h) \\ f \wedge g \equiv g \wedge f & f \vee g \equiv g \vee f \\ \top \wedge f \equiv f & \perp \vee f \equiv f \\ \perp \wedge \perp \equiv \perp & \top \vee \top \equiv \top \end{array}$$

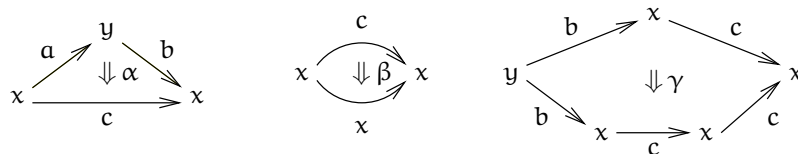
◆

Note that in the original description of system SKS, only closed terms are considered for formulas. However, in order to reduce deduction rules to a *finite* number, it is more convenient to consider all terms. Furthermore, it does not raise any complication, neither does the fact of taking non-linear terms in account.

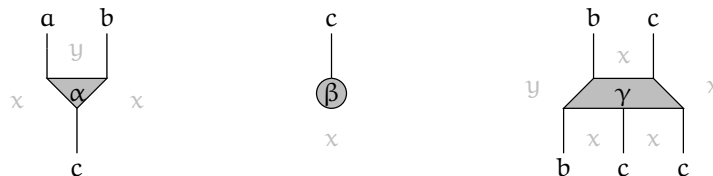
1.2 The 2-polygraph of logical connectives

Here a translation of the aforesaid SKS signature into a 2-polygraph is given. Defining the structure of polygraph can be quite cumbersome, but giving an idea is rather simple: it is a cellular presentation of a 2-graph with no limitation on the shapes of the 2-cells. Such an object is built inductively:

- First, one considers a set Σ_0 of 0-cells, pictured as points in the plane.
- Then, one takes a set Σ_1 of 1-cells, pictured as arrows between these points; formally, the pair (Σ_0, Σ_1) is equipped with a structure of (1-)graph.
- From these data, one can build a (1-)category, which is the graph $(\Sigma_0, \langle \Sigma_1 \rangle)$ of finite paths in (Σ_0, Σ_1) , equipped with the concatenation operation. Two paths starting at the same point and ending at the same point are *parallel*.
- Finally, one chooses a set of 2-cells on the category $(\Sigma_0, \langle \Sigma_1 \rangle)$: this is a set Σ_2 of arrows between parallel paths. They are pictured as directed surfaces, like in the following examples:



For a formal definition of 2-polygraph, see (Burroni 1993), where these objects were introduced, or the first pages of (Métayer 2003). There is another, more useful representation of 2-cells: *Penrose diagrams*, already used in (Lafont 2003; Guiraud 2004a, 2004b), make 2-cells appear as *circuits*. For example, the three aforesaid 2-cells become:



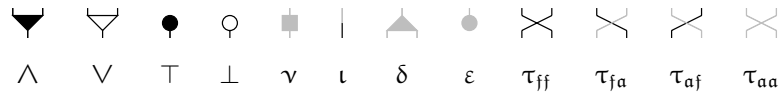
To build these diagrams, the following correspondance is used:

- Each 0-cell is pictured as a label (or colour) for parts of the plane; in the given representations, these labels have been grayed out for sake of clarity.
- Each 1-cell is drawn using a vertical wire, separating the plane in two parts, the leftmost one labelled with the name of its source and the rightmost one with the name of its target; the wire itself is labelled (or coloured) with the name of the 1-cell.
- Each 2-cell is pictured by a small circuit component, a bit like logical gates, with inputs and outputs corresponding to its source and its target.

In most of the concrete examples encountered so far, there has been only one 0-cell and at most two 1-cells. Therefore, the labels in the parts of the plane can be dropped (they are all equal) and one uses only one or two colours for the wires; this simplifies diagrams, but considering the general case does not yield any difficulty since it is easily handled in pictures.

This being defined, let us build a 2-polygraph Σ from the signature defining SKS formulas. We follow the same idea as the one used in (Guiraud 2004a, 2004b) to translate signatures of terms:

- The 2-polygraph Σ has only one 0-cell, denoted by $*$ (no label in parts of the plane).
- Then, we add two 1-cells, denoted by \mathfrak{a} and \mathfrak{f} , respectively corresponding to the sort of atoms (gray wires) and to the sort of formulas (black wires).
- Finally, we consider the following twelve 2-cells:



The interpretation of each 2-cell is given below each diagram. Each one is seen as an operation on formulas, such as $(f, g) \mapsto f \wedge g$ for \wedge . The cells ι and ν stand respectively for the inclusion of atoms into formulas and for the negation on atoms. To these ones, some so-called *ressource management operators* are added; their role is to locally handle duplication, erasement and permutation operations:

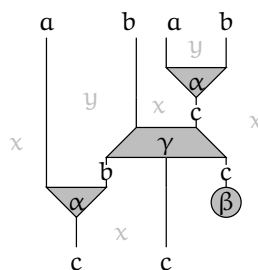
- The 2-cell δ is a local duplicator of atoms. Note that, in what follows, duplication of formulas is not necessary; however, one could add a local duplicator of formulas, with no increased difficulty, but at the cost of a few more equations; in that case, the duplicators would be, if necessary, distinguished by notations such as $\delta_{\mathfrak{a}}$ and $\delta_{\mathfrak{f}}$.
- The 2-cell ε is a local eraser of atoms. The same remark applies, eventually leading to the addition of another local eraser $\varepsilon_{\mathfrak{f}}$ for formulas, the first one becoming $\varepsilon_{\mathfrak{a}}$.
- The four remaining 2-cells are local permutations; there is one for each possible pair of colours, but generally all of them are simply denoted by τ .

1.3 The 2-category of formulas

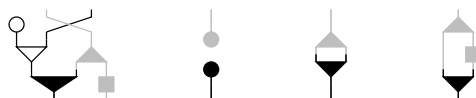
In the same way as the signature operators are building blocks for formulas, these circuit components are used to craft something bigger, in which SKS formulas can be interpreted.

From a 2-polygraph, one considers all the circuits one can build, by plugging together any number of copies of its 2-cells. This is the same idea as the one leading from logical gates to boolean circuits, which are all the constructions one can make using any number of AND-gates, OR-gates, etc.

For example, with the three aforesaid 2-cells α , β and γ , one can build the following circuit:



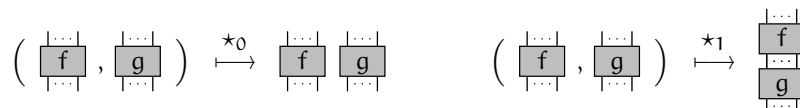
In the case of the polygraph Σ , built from the signature of SKS, one can get the following circuits:



In what follows, we give the following interpretations to these circuits:

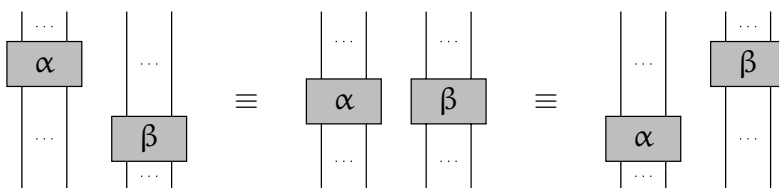
- The first one corresponds to the map $\mathcal{A} \times \mathcal{F} \rightarrow \mathcal{F} \times \mathcal{A}$ sending (a, f) to $((\perp \vee f) \wedge a, \bar{a})$.
- The second one is the constant map $\mathcal{A} \rightarrow \mathcal{F}$ sending every atom to \top .
- The third one is the map $\mathcal{A} \rightarrow \mathcal{F}$ sending a to the formula $a \wedge a$.
- The fourth one is the map $\mathcal{A} \rightarrow \mathcal{F}$ sending a to the formula $a \wedge \bar{a}$.

There are two kinds of *pastings* used to build these circuits: either horizontally or vertically. These operations, called *compositions*, are pictured this way:



The first one, \star_0 , is only defined if the plane labels match (this is always the case here, since there is only one 0-cell). The second one, \star_1 , is only defined if the wires labels match (here, there must be the same number of wires, with the same colours, in the same order). The fact that there are two different ways of pasting circuits together, and only two, is the reason why we say that they are 2-dimensional objects.

These two dimensions are not totally independent. Indeed, the circuits are seen as genuine topological objects. This means that they are to be considered modulo *isotopy*, or homeomorphical deformation: one can deform wires and move components, provided no crossing is created. This identification is generated by the following *local isotopy relations*, given for each pair of components α and β :



Definition 2. The object $\langle \Sigma \rangle$ consisting of all circuits *modulo* isotopy is called the *free 2-category* generated by the 2-polygraph Σ ; the circuits are the *2-arrows* of $\langle \Sigma \rangle$. If f is one of them, then $s_1(f)$ and $t_1(f)$ respectively denote the source and target 1-arrows of f . \blacklozenge

One can find in (MacLane 1998) more information about 2-categories and (Chang and Lauda 2004) is a really comprehensive survey about the zoology of n -categories. Now, we sketch how SKS formulas are related to the 2-arrows of $\langle \Sigma \rangle$. First, let us build a map from 2-arrows to (families) of formulas.

Since both \mathcal{A} and \mathcal{V} , the sets of atoms and formulas variables, are countable, one is allowed to choose a strict linear order on each set, so that the elements of \mathcal{A} are denoted by a_1, a_2, a_3 , etc. and the elements of \mathcal{V} are denoted by x_1, x_2, x_3 , etc.

We define, for each circuit f , a map $\pi(f)$ acting on atoms and formulas. Let us start by giving its source and target:

- First, one defines $\pi(a) = \mathcal{A}$ and $\pi(f) = \mathcal{F}$.
- Then, one extends $\pi(x \star_0 y) = \pi(x) \times \pi(y)$, for any pair (x, y) of 1-arrows in $\langle \Sigma \rangle$.

- Finally, for any circuit f , $\pi(f)$ has source and target given by:

$$\pi(f) : \pi(s_1(f)) \rightarrow \pi(t_1(f)).$$

Then, we define $\pi(f)$ inductively:

- If f is the identity of \mathfrak{a} , then $\pi(f)$ is the identity of \mathcal{A} ; both identities are abusively denoted by \mathfrak{a} and \mathcal{A} .
- If f is (the identity of) \mathfrak{f} , then $\pi(f)$ is (the identity of) \mathcal{F} .
- $\pi(\wedge) : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{F}$ is defined by $\pi(\wedge)(x, y) = x \wedge y$, and similarly for \vee .
- $\pi(\top) : * \rightarrow \mathcal{F}$ is the constant map \top , and similarly for \perp .
- $\pi(\iota) : \mathcal{A} \rightarrow \mathcal{F}$ is the inclusion of \mathcal{A} into \mathcal{F} .
- $\pi(\nu) : \mathcal{A} \rightarrow \mathcal{A}$ sends each atom \mathfrak{a} to $\bar{\mathfrak{a}}$.
- $\pi(\delta) : \mathcal{A} \rightarrow \mathcal{A} \times \mathcal{A}$ sends \mathfrak{a} to $(\mathfrak{a}, \mathfrak{a})$.
- $\pi(\varepsilon) : \mathcal{A} \rightarrow *$ is the terminal map of \mathcal{A} .
- $\pi(\tau_{f,f}) : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{F} \times \mathcal{F}$ send (x, y) to (y, x) , and similarly for the other local permutators.
- If f has the shape $f = g \star_0 h$, then $\pi(f) = (\pi(g), \pi(h))$, the juxtaposition of both maps; this means that $\pi(g)$ acts on the first inputs and $\pi(h)$ on the last inputs.
- If f has the shape $f = g \star_1 h$, then $\pi(f) = \pi(h) \circ \pi(g)$, the composition of $\pi(g)$ and $\pi(h)$.

Then, one has to check that π is well-defined: it is compatible with the isotopy relation, since the cartesian product is a bifunctor. Finally, one associates a formula $\bar{\pi}(f)$ to each circuit f by application of $\pi(f)$ on some variables:

- One defines $v_n(\mathfrak{a}) = \mathfrak{a}_n$, the n^{th} atom variable, and $v_n(f) = x_n$, the n^{th} formula variable.
- One inductively extends v_n to $v_n(\mathfrak{x} \star_0 \mathfrak{u}) = (v_n(\mathfrak{x}), v_{n+1}(\mathfrak{u}))$, for every 1-cell \mathfrak{x} and 1-arrow \mathfrak{u} .
- Finally, one defines $\bar{\pi}(f) = \pi(f)(v_1(s_1(f)))$.

For example, the four aforegiven circuits are respectively sent to the formulas:

$$((\perp \vee x_1) \wedge \mathfrak{a}_1, \bar{\mathfrak{a}}_1), \quad \top, \quad \mathfrak{a}_1 \wedge \mathfrak{a}_1, \quad \mathfrak{a}_1 \wedge \bar{\mathfrak{a}}_1.$$

Hence, we have built a projection $\bar{\pi}$ from circuits to formulas. But, for the moment, we cannot define a canonical section φ , since the projection $\bar{\pi}$ is not compatible with the congruences $\equiv_{\mathcal{A}}$ nor $\equiv_{\mathcal{F}}$. Indeed, for example, both circuits $\nu \star_1 \nu$ and \mathfrak{a} are sent to the atom \mathfrak{a}_1 , since $\bar{\mathfrak{a}} \equiv_{\mathcal{A}} \mathfrak{a}$, yet they are two distinct circuits.

Furthermore, $\bar{\pi}^{-1}(\equiv_{\mathcal{A}})$ and $\bar{\pi}^{-1}(\equiv_{\mathcal{F}})$ do not constitute the whole kernel of $\bar{\pi}$. For example, both circuits $(\tau \star_1 \tau)$ and $\mathfrak{f} \star_0 \mathfrak{f}$ are sent to the pair (x_1, x_2) ; however, they are once again different circuits. These additional equalities are called *resource management equations* and the congruence they generate on circuits is denoted by \equiv_{Δ} .

There exist more than one possibility to solve this problem, the choice being left to the user. On a first approach, one can translate all the equalities defining $\equiv_{\mathcal{A}}$ and $\equiv_{\mathcal{F}}$ to equalities on circuits; then, add the equalities defining \equiv_{Δ} ; finally, consider circuits *modulo* the congruence generated by all of these equalities. Then, equivalence classes of circuits would correspond to families of formulas.

But equalities and computational objects rarely live together peacefully. Another idea consists in the replacement of equalities by computations proving these equalities; this point of view is more in adequation with the computational flavour of the considered objects. Then, both (Burrioni 1993) and (Baez and Dolan 1998) tell us that this calculus can be well-represented by objects in the dimension above.

2 The three dimensions of proofs

2.1 The proofs of system SKS

We give here a slightly different definition than the original one from (Brünnler 2004); non-closed formulas allow one to remove contexts in the rules, reducing them to a finite number.

Definition 3. The *inference rules* of system SKS are given by:

$$\begin{array}{l} \top \longrightarrow a \vee \bar{a} \quad (x \vee y) \wedge z \longrightarrow (x \wedge z) \vee y \quad \perp \longrightarrow a \quad a \vee a \longrightarrow a \\ a \wedge \bar{a} \longrightarrow \perp \quad (x \wedge y) \vee (z \wedge t) \longrightarrow (x \vee z) \wedge (y \vee t) \quad a \longrightarrow \top \quad a \longrightarrow a \wedge a \end{array}$$

From left to right, top to bottom, these rules are called *introduction* (or *axiom*), *switch*, *weakening*, *contraction*, *cointroduction* (or *cut*), *medial*, *coweakening* and *cocontraction*. \blacklozenge

From these elementary blocks, one builds *SKS proofs* exactly the same way one builds rewriting paths from rewriting rules: each inference rule α generates a proof step from $C[s(\alpha) \cdot \sigma]$ to $C[t(\alpha) \cdot \sigma]$, for every context C and every substitution σ - see (Baader and Nipkow 1998) or (Guiraud 2004a) for more information about these notions.

Then one can compose proof steps, provided the target of each one matches the source of the next one. Such a path going from a formula u to a formula v is called a *proof (from u to v)*; when $u = \top$, then the path is a *complete proof (of v)*.

Example 4. Here is a complete proof of $(a \wedge b) \vee (\bar{a} \vee \bar{b})$:

$$\begin{array}{l} \top \equiv_{\text{f}} \top \wedge \top \rightarrow (a \vee \bar{a}) \wedge \top \\ \rightarrow (a \vee \bar{a}) \wedge (b \vee \bar{b}) \\ \rightarrow (a \wedge (b \vee \bar{b})) \vee \bar{a} \equiv_{\text{f}} ((b \vee \bar{b}) \wedge a) \vee \bar{a} \\ \rightarrow ((b \wedge a) \vee \bar{b}) \vee \bar{a} \equiv_{\text{f}} (a \wedge b) \vee (\bar{a} \vee \bar{b}). \end{array}$$

The main problem coming from using term-flavoured notations for formulas is that it raises bureaucracy, that can hardly be controlled by equations.

A first example of bureaucracy is illustrated by the aforegiven SKS proof. Indeed, there are two possible proofs from $\top \wedge \top$ to $(a \vee \bar{a}) \wedge (b \vee \bar{b})$, generating two different complete proofs of $(a \wedge b) \vee (\bar{a} \vee \bar{b})$:

$$\begin{array}{ccc} \top \wedge \top & \longrightarrow & (a \vee \bar{a}) \wedge \top \\ \downarrow & & \downarrow \\ \top \wedge (b \vee \bar{b}) & \longrightarrow & (a \vee \bar{a}) \wedge (b \vee \bar{b}) \end{array}$$

But the user may desire to identify these two proofs: indeed, they only differ by the order of application of two introduction rules on different subformulas; so, in essence, they could be considered the same. Yet, they are distinct in the SKS formalism.

For this kind of bureaucracy, equations remain easy to craft. But there is another kind of bureaucracy that can hardly be defined equationally on formulas expressed as terms. Let us assume that F is a proof from u to v and that x and y are two formulas variables; then, one gets two different proofs from $(u \wedge x) \vee y$ to $(v \vee y) \wedge x$:

$$\begin{array}{ccc} (u \wedge x) \vee y & \longrightarrow & (v \wedge x) \vee y \\ \downarrow & & \downarrow \\ (u \vee y) \wedge x & \longrightarrow & (v \vee y) \wedge x \end{array}$$

Once again, these two proofs could be considered as equal. But here, equations are painful to define. This is because the term notation makes the two different compositions \star_0 and \star_1 appear different, yet they are similar in essence.

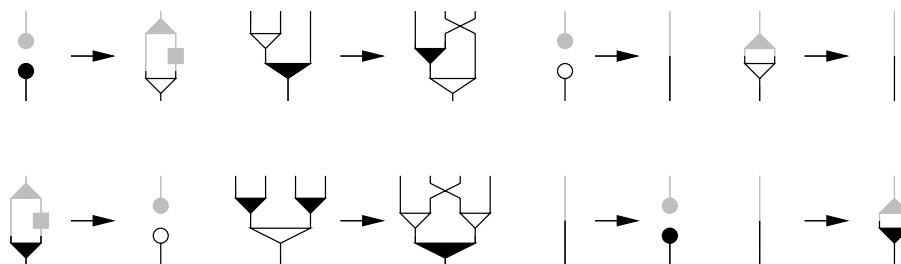
The two-dimensional interpretation of proofs makes it possible to produce automatically the equations corresponding to these two kinds of bureaucracy, provided the proofs are seen as three-dimensional objects.

2.2 The 3-polygraph of inference rules

Here, we associate a 3-polygraph to the inference rules of SKS. Let us give an informal definition of what is a 3-polygraph - the formal one can still be found in (Burroni 1993) or (Métayer 2003).

Definition 5. A 3-polygraph is a family $\Sigma = (\Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3)$ where $(\Sigma_0, \Sigma_1, \Sigma_2)$ is a 2-polygraph and Σ_3 is an additional set of 3-cells: each one is an arrow between two parallel circuits f and g , built from the 2-polygraph. \blacklozenge

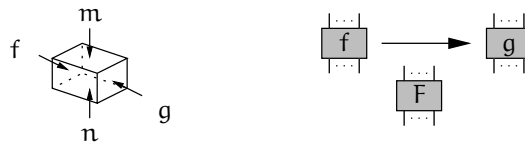
These 3-dimensional cells can be represented as computations from one circuit to another. The ones corresponding to the rules of SKS are the following eight 3-cells:



For the moment, we use this representation for proofs, but another one is used later in order to give a genuine three-dimensional vision of these objects.

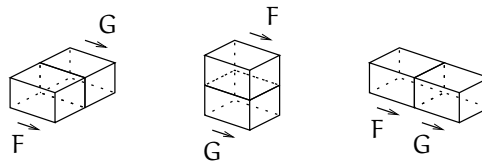
2.3 The 3-category of proofs

These 3-cells are seen as elementary components that are used to build bigger objects, representing (families of) proofs. To give an idea, let us imagine a proof as a block. Here are two different possible representations:

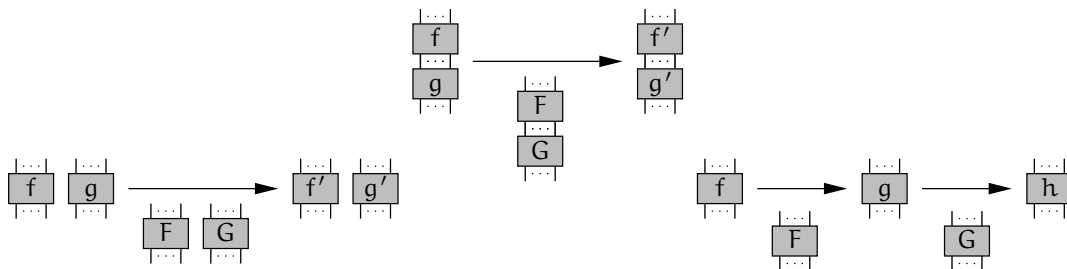


This is a proof F from a circuit f to a circuit g , both having m inputs and n outputs. To compare both representations, one can see the second one as composed of three vertical slices of the block: one before the block (f), one in the middle of the block (F) and one after the block (g).

These blocks can be pasted in three different ways, each one corresponding to one of its dimensions:



If one makes slices for each of these three constructions, one gets:



- The first composition produces a proof $F \star_0 G$ composed of two juxtaposed subproofs (one in each subformula); it is a new composition of proofs, linked to another one revealed by the two-dimensional interpretation of formulas (the juxtaposition of formulas).
- The second one yields a proof $F \star_1 G$ made of one subproof plugged into another one; it is also a new composition corresponding to the composition of formulas.
- The third composition is the usual composition $F \star_2 G$ of proofs, one after the other.

When these constructions are defined, one has to identify some of them along some relations in order to get the 3-arrows of the free 3-category. Among these relations are the *monoidal* ones:

- For any $i \in \{0, 1, 2\}$ and any 3-arrows F, G, H , the following *associativity relation* holds whenever one of its members is defined:

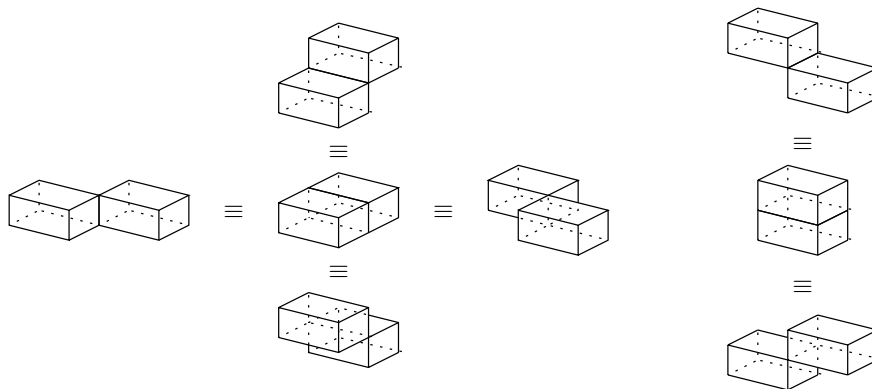
$$(F \star_i G) \star_i H = F \star_i (G \star_i H).$$

- For any 3-arrow F from f to g , with m inputs and n outputs, the following *unit relations* hold:

$$F \star_0 * = F = * \star_0 F, \quad F \star_1 \mathfrak{n} = F = \mathfrak{m} \star_1 F, \quad F \star_2 g = F = f \star_2 F.$$

To these families of equations, one adds the *isotopy relations*. Indeed, 3-dimensional constructions are one again seen as topological objects; this implies that they are identified *modulo* homeomorphic deformation, or isotopy.

The isotopy classes are given by the following equations, given graphically:



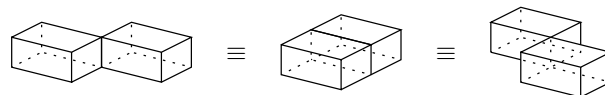
There are three types of isotopy, two of them being detailed thereafter.

3 Isotopy and bureaucracy

Isotopy relations on 3-dimensional arrows are divided into three families. Two of them are deeply linked with the two types of bureaucracy one encounters when dealing with SKS proofs or, more generally, with proofs expressed in the calculus of structures; each one is detailed thereafter. The third family comes from the only isotopy type one gets in dimension 2 (on circuits): it is a degenerated form of this one.

3.1 Isotopy and bureaucracy A

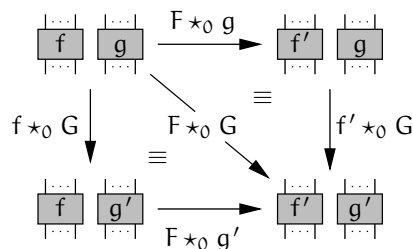
The first family of isotopy relations is given graphically by:



In equational form, it is expressed as:

$$(F \star_0 g) \star_2 (f' \star_0 G) \equiv F \star_0 G \equiv (f \star_0 G) \star_2 (F \star_0 g'),$$

for any $F : f \rightarrow f'$ and $G : g \rightarrow g'$. Thus, this isotopy identifies the three following proofs:



Now, let us see how this is linked to *type A* bureaucracy: in SKS, the applications of two proofs in two different subformulas must be done one after the other; and the two possibilities correspond to two different proofs.

For example, as stated earlier, one gets two different SKS proofs from $\top \wedge \top$ to $(a \vee \bar{a}) \wedge (b \vee \bar{b})$, depending on which of a or b is introduced first. In SKS this yields two different proofs; if one wants to identify them, then a canonical representative has to be chosen, which is not possible, for either choice would be arbitrary.

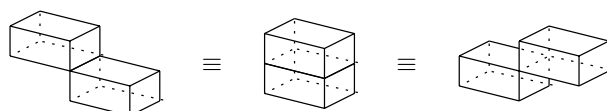
But, in the 3-category, a third proof exists, corresponding to the simultaneous introduction of a and b : this one can be chosen as a canonical representative.

Even more important, one may want to have equations describing this type of bureaucracy. This is not really hard to express in the SKS style, but it is definitely easier in the 3-dimensional framework.

Furthermore, it is only a SKS feature that the other type of bureaucracy appears to be totally different in essence than the type A one. Indeed, as described in the next paragraph, the two types are completely similar in the 3-dimensional language.

3.2 Isotopy and bureaucracy B

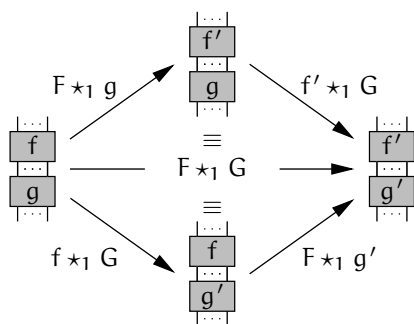
The second family of isotopy relations is:



In equational form, for any $F : f \rightarrow f'$ and $G : g \rightarrow g'$ such that the proof $F \star_1 G$ is defined:

$$(F \star_1 g) \star_2 (f' \star_1 G) \equiv F \star_1 G \equiv (f \star_1 G) \star_2 (F \star_1 g'),$$

Thus, this isotopy type identifies the three following paths:



This is deeply linked to *type B* bureaucracy: in SKS, one cannot apply at the same time one proof inside another one; one of them must be done before the other, yielding two different proofs.

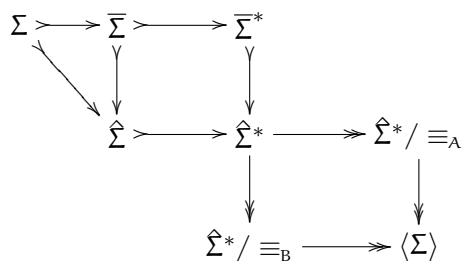
For example, let us consider a proof F from f to g , together with two variables x and y . Then, one gets two different SKS proofs from $(f \vee x) \wedge y$ to $(g \wedge y) \vee x$, depending on whether F is applied before or after the switch rule. Once again, there is no canonical choice between them, whereas in the 3-category there is one.

But, the main interest lies elsewhere: in the SKS style, it is quite hard to express equationally this bureaucracy type B, while in the 3-category, it is not painful at all, since both bureaucracy types are completely similar; indeed, each one corresponds to an isotopy relation, between \star_2 and \star_0 for type A, between \star_2 and \star_1 for type B.

3.3 Some geography

There is a point in which higher-dimensional rewriting excels: the classification of equational and deductive theories. And this is also the case for the calculus of structures and its offsprings, formalisms A and B - when concerned with classical propositional logic, these three are SKS, SKS *modulo* bureaucracy A and SKS *modulo* bureaucracies A and B. Formalism A is defined in (Guglielmi 2004a) and formalism B is in (Guglielmi 2004b).

Existing descriptions of the three formalisms look quite different, while they can be viewed as parts of a bigger scheme. Let us consider the 2-polygraph of formulas and build the following commutative diagram of 3-polygraphs over the free 2-category of formulas:



Objects in this diagram are encountered when constructing, step by step, the free 3-category generated by a 3-polygraph. Here is an informal description of them:

- One starts with a set Σ of 3-cells: these are rewriting rules on circuits.
- From these, one can consider all the rules, applied in any context, which yields $\bar{\Sigma}$, the set of one-step sequential reductions.
- Alternatively, one can consider all the rules applied in any existing context and possibly in parallel to build $\hat{\Sigma}$, the set of one-step parallel reductions.
- Then, one considers the reduction paths generated by $\bar{\Sigma}$: this produces the set $\bar{\Sigma}^*$ of sequential reductions. This is where the calculus of structures (here SKS) lives.
- Alternatively, the paths generated by $\hat{\Sigma}$ give the set $\hat{\Sigma}^*$ of parallel reductions. This is the biggest set of 3-arrows one can build from Σ : there, bureaucracy is at its highest level, since all the described proofs differing by the order of application of subproofs are distinguished; furthermore, there is at each time a third possible proof, consisting in the simultaneous application of both subproofs.
- Then one starts the quotients by isotopy relations. The first possibility is to quotient by the first family of isotopy relations, corresponding to bureaucracy type A. This yields $\hat{\Sigma}^* / \equiv_A$, where lives formalism A.
- As an alternative, one can instead quotient by the isotopy relations corresponding to bureaucracy type B, to get $\hat{\Sigma}^* / \equiv_B$ which has no equivalent in calculus of structures-derived formalisms.
- Finally, doing both quotients, one gets the free 3-category $\langle \Sigma \rangle$ generated by Σ , where all the bureaucracy is killed. This is where formalism B lives.

This diagram is indeed a map of where known formalisms are located. But it also encompasses still unknown formalisms that could prove to be useful, like $\hat{\Sigma}^*$ or $\hat{\Sigma}^*/\equiv_B$. This is an example of the freedom higher-dimensional rewriting lets to the user in the exact design of the proofs he wants to consider.

Another example of freedom is given later about the possibilities offered to the user for handling the equations between formulas.

4 A 3-dimensional representation for proofs

This section is a first attempt at representing proofs in 3 dimensions, so that one can view them as the genuine 3-dimensional objects they are.

4.1 The theoretical idea

In order to represent 2-arrows, Penrose diagrams are really convenient; they make 2-arrows appear as circuits, using the following scheme:

- Each 2-dimensional cell is pictured as a vertex in a graph (a 0-dimensional object).
- Each 1-dimensional cell is drawn as an edge in a graph (a 1-dimensional object).
- Each 0-dimensional cell is represented by a part of the plane which boundaries are the edges of the graph (2-dimensional objects).
- Then, the vertices and edges of the graph are thickened until they are 2-dimensional; note that in the circuit representation, wires are not thickened to make drawing easier, but they should be for sake of coherence.

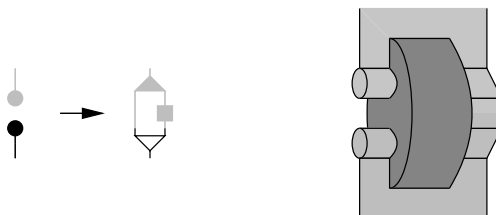
The application of a similar process to a 3-dimensional arrow gives:

- Each 3-dimensional cell is a point.
- Each 2-dimensional cell is a line (either open or between two points).
- Each 1-dimensional cell is a surface (either open or with a line as a boundary).
- Each 0-dimensional cell is a volume lying between surfaces.
- Finally, every object is thickened, if necessary, until it gets 3-dimensional.

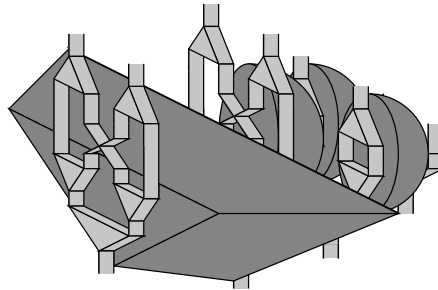
This associates *3-dimensional Penrose diagrams* to 3-arrows.

4.2 One glance at three-dimensional proofs

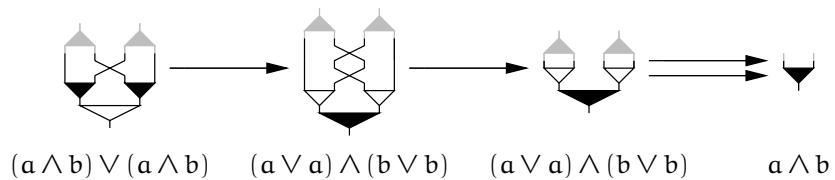
This is time to draw a 3-dimensional proof. First, let us make a Penrose diagram for a rule; the rewriting rule style is also given:



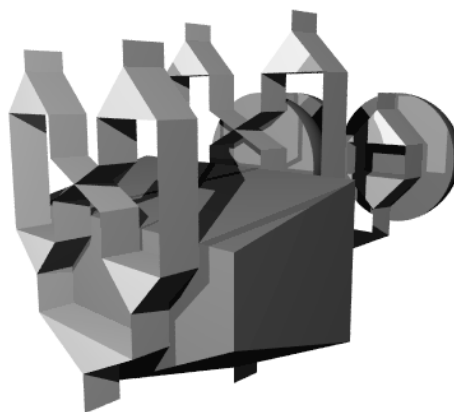
Then, Penrose diagrams for proofs are pastings of such 3-dimensional blocks:



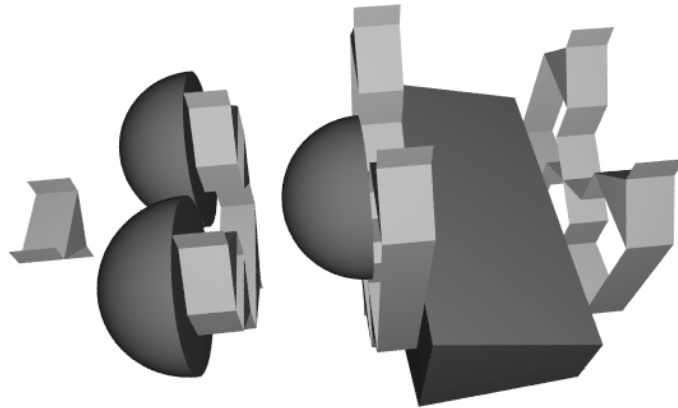
In order to understand how this object is built (and what lies behind some opaque volumes), one can make vertical slices of this object, to produce the following rewriting-style proof:



Since this representation uses only a fake third dimension, one could prefer to use a 3-dimensional modeler. This has many advantages, such as being able to turn around the object and make snapshots from different points of view. For example, two views of the same proof, presented in the next page, were generated using the freely available software POV-Ray¹.



¹ <http://www.povray.org>



Two remarks shall be made about the representations:

- On the 3-dimensional ones, less emphasis has been put on circuits, so that different types of wires or different operators appear the same while they should not.
- Some surfaces are not drawn, only for sake of clarity, but these objects should appear somewhat more closed.

This part is quite new and some work will be necessary to produce nice and more usable representations, so that the third dimension can provide more insight on what kind of objects proofs are.

5 The twilight zone

When the third dimension gets involved, one can ask whether this dimensional increase will stop or not. The answer is quite simple: no. Indeed there are, at least, two good reasons to proceed.

The first one is total abstract nonsense. In category theory, there is a proverb saying: *when one wants to study some objects, one should rather study their morphisms*. Let us add that morphisms between 3-arrows are 4-arrows in a 4-category.

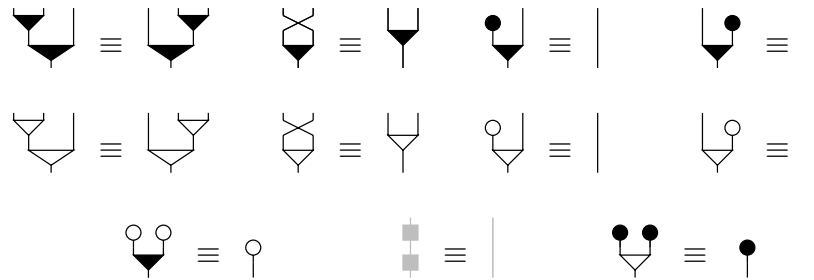
More concretely, there are two kinds of examples that give rise to 4-dimensional arrows: equations between formulas and local transformations on proofs. This section is about a short glance at these two issues.

5.1 Equations between formulas

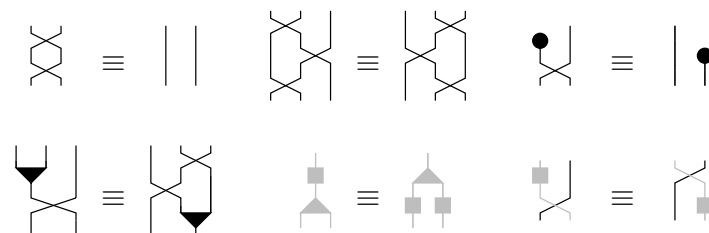
Previously, equations between formulas have been left aside. It has been said that they can be translated as equations between circuits. This is just one possibility, the higher-order rewriting framework allowing one to *choose* between many possible considerations. Here are three of them, but one can at least take any desired combination of them.

Equations are equations. The first possibility is, as stated before, to translate equations between formulas into equations between circuits. In that case, one considers circuits *modulo* two families of equations.

The first one is a faithful translation of the equations on formulas, so that, for example, one can recognize associativity of \wedge and \vee among them:



The second family purpose is to give the resource management operators their real meaning, so that, for example, δ really is a local duplicator; among others, one gets the following equations:



From equations to 3-dimensional isomorphisms. Rather than considering equations on formulas as equations on circuits, one can treat them as invertible computations. Indeed, equations are often clashing with computational considerations, so that, whenever possible, they are replaced by local computations.

Hence, one could replace the two aforementioned families of equations by two families of invertible 3-cells. For example, the equation enforcing the associativity of \wedge is split into two 3-cells:

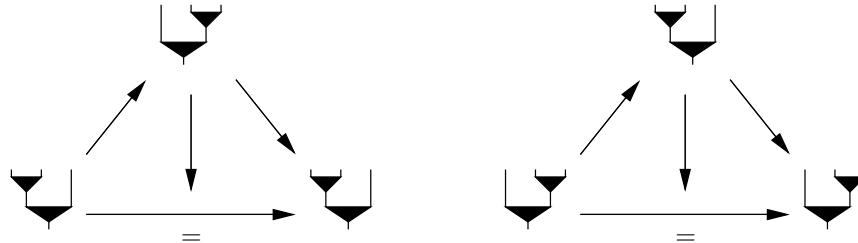


Then, in order to ensure that they are 3-dimensional isomorphisms, one adds equations between proofs: both possible composites are equal to the corresponding identity. Hence, this leaves no equation between objects of dimension 2, while two of them appear between objects of dimension 3 for each equation on formulas.

From equations to 4-dimensional computations. There is no reason to stop the process of lifting up equations. In order to achieve this, the pairs of 3-dimensional cells replacing equations are kept, but equations between 3-dimensional composites are lifted up.

Hence, instead of considering commutative diagrams between 3-dimensional arrows, one defines 4-dimensional cells: each one represents a *computation* from one composite to the identity 3-cell.

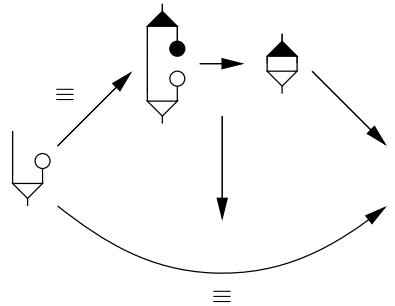
For example, the equation about the associativity of \wedge is finally replaced by two 3-cells, together with the following two 4-cells:



5.2 Local computations on proofs

The next example of 4-dimensional cells is in fact a generalization of the former one. Indeed, it arises whenever one wants to compute normal forms for proofs, *modulo* some user-specified equations. This encompasses the former example, since these equations can be the ones stating that two 3-cells are inverse one another.

As an example of generalized computation, the following 4-cell can be introduced in order to simplify proofs with a weakening followed by a contraction, both acting on the same atom:



In fact, any local computation on proofs can be replaced by a 4-cell. All the 4-cells being given, the computations they generate are the 4-dimensional arrows of a free 4-category.

5.3 Some temporary relief

This point of view immediately arises the following question: how can one use the fact that these computations are 4-dimensional objects? This comes with the subsidiary question: how can one represent 4-dimensional objects? In fact, this is not necessary at this point.

To explain this answer, let us step back by one dimension. Term rewriting is about some properties (termination and confluence) of computations on 2-dimensional objects. While considering the whole 2-dimensional structure of terms is really useful, the computations need not be seen as genuine 3-dimensional objects: the only purpose of doing so would be to identify reduction paths *modulo* bureaucracy. But term rewriting is not concerned with the classification of reduction paths (only their existence) and neither termination nor confluence are modified by bureaucracy.

Then comes proof theory which, with the higher-dimensional point of view, studies 3-dimensional objects, or rather computations between them. Hence, with the same arguments as above, considering the whole 3-dimensional structure of proofs shall prove to be useful. But the four dimensions of computations on proofs are not involved if one only wants to prove termination or confluence of proof normalization processes.

In conclusion, if it is only about (normalization of) proofs, then one can live with rewriting paths on 3-dimensional arrows. But when times will come when the classification of rewriting paths on proofs is concerned, then the fourth dimension will be useful.

For example to manage the six types of bureaucracy lurking in dimension 4.

6 Future directions

This paper presents a higher-dimensional rewriting point of view for the deep inference system SKS. Its main benefit is to provide a uniform setting for many possible systems, depending on what the user wants to emphasize; indeed, much freedom is left on how to consider bureaucracy or how to see equations.

Furthermore, higher-dimensional rewriting provides a common view on equations and computations between proofs: they are seen as 4-dimensional cells between proofs. So one just has to choose the local computations he wants to study, then the theory can be used to see if the generated calculus is terminating or not, confluent or not.

However, some work will be necessary here to provide the required tools, such as a recipe to craft termination orders like the one in (Guiraud 2004b) for 3-dimensional rewriting. Another tool will concern the study of 4-dimensional critical pairs; this will be an adaptation of one that is still under development for 3-dimensional critical pairs and will be described in a subsequent paper.

Aside from these computational issues, proofs seen as 3-dimensional objects are naturally equipped with a geometric representation, using 3-dimensional Penrose diagrams. The links between these pictures and proof nets still have to be explored. For the moment, we can at least say that the proposed 3-dimensional representations provide a completely different way to look at proofs.

Here, only system SKS has been mentioned, mainly because it is smaller than, for example, system SLLS for linear logic (Straßburger 2003). However, it is not very dangerous to conjecture that this translation shall work as well for SLLS.

A bit more risky is the assertion that the translation should also work with any deep inference-style formalism, provided there is no quantifier in the syntax of the formulas. Indeed, binders such as quantifiers or the abstraction in the λ -calculus are yet to be understood in the higher-dimensional point of view.

Finally, a bigger conjecture would be that such a translation can be made for sequent-style proofs (still with no binder allowed in the formulas). If this result holds, then one will be able to say that, in essence, proof theory studies 4-dimensional rewriting systems.

References

- Franz Baader and Tobias Nipkow, *Term rewriting and all that*, Cambridge University Press, 1998.
- John Carlos Baez and James Dolan, *Categorification*, ArXiv preprint, 1998.
- Kai Brännler, *Deep inference and symmetry in classical proofs*, Logos Verlag, 2004.
- Albert Burroni, *Higher-dimensional word problems with applications to equational logic*, Theoretical Computer Science 115(1), 1993.
- Eugenia Chang and Aaron Lauda, *Higher-dimensional categories: an illustrated guide book*, 2004.
- Alessio Guglielmi, *Formalism A*, note, 2004(a).
- Alessio Guglielmi, *Formalism B*, note, 2004(b).
- Alessio Guglielmi, *Deep inference and the calculus of structures*, project report, 2005.
- Yves Guiraud, *Présentations d'opérades et systèmes de réécriture [Operad presentations and rewriting systems]*, thèse de doctorat, Université Montpellier 2, 2004 (a).
- Yves Guiraud, *Termination orders for 3-dimensional rewriting*, submitted preprint, 2004 (b).
- Yves Lafont, *Towards an algebraic theory of boolean circuits*, Journal of Pure and Applied Algebra 184, 2003.
- Saunders MacLane, *Categories for the working mathematician*, Springer, 1998.
- François Métayer, *Resolutions by polygraphs*, Theory and Applications of Categories 11(7).
- Lutz Straßburger, *Linear logic and noncommutativity in the calculus of structures*, PhD thesis, Technischen Universität Dresden, 2003.

THE PROBLEM OF BUREAUCRACY AND IDENTITY OF PROOFS FROM THE PERSPECTIVE OF DEEP INFERENCE

Alessio Guglielmi (TU Dresden and University of Bath)

17.6.2005

Abstract

Deep inference offers possibilities for getting rid of much bureaucracy in deductive systems, and, correspondingly, to come up with interesting notions of proof identity. We face now the problem of *designing* formalisms which are intrinsically bureaucracy-free. Since we have a design problem, it is important to elaborate definitions that will remain useful for many years to come. I propose a discussion of several proposals. The discussion will hopefully be also a good way of introducing deep inference to those who don't know it.

In my talk I will explain in detail and with examples all the notions quickly sketched below. It is apparently extremely simple stuff, but there are subtle issues that only experienced proof theorists might appreciate; I will try to address them. The proposed solutions are currently discussed on the mailing list Frogs. By the time of the workshop, in addition to my proposed definitions, I will have also the opinions of the participants to the discussions.

Bureaucracy and Identity

Bureaucracy and identity of proofs are intimately related.

There is no formal notion of bureaucracy, but I guess the consensus is that, when two proofs are *morally the same*, but they differ in *inessential details*, then this is due to *bureaucracy*.

If this is so, we should conclude that eliminating bureaucracy should lead us to eliminate the inessential details that blur the 'sameness', i.e., identity, of proofs.

We should agree that, for any given logic, there are several possible notions of identity of proofs, and people can invent more and more of them.

Given a notion of identity and a formalism, either the formalism is able to express the identical proofs or it isn't: in the latter case, we have bureaucracy, and we have an enemy.

Our goal is to attack some specific, important kinds of bureaucracy, in order to improve the ability of proof theory to deal with bureaucracy. It is hopeless to try and define bureaucracy once and for all. However, it is now possible to define formalisms which get rid of the most brutal and medieval forms of bureaucracy.

Bureaucracy in the Formalism and in the Deductive System

I will use in the following the syntax of the calculus of structures (CoS) [WS].

There are several sources of bureaucracy, and I think it is convenient to address them separately. It should be possible to make a broad distinction between bureaucracy induced by the formalism and bureaucracy induced by the specific deductive system used (in the given formalism).

What I call formalism A [1] takes care of bureaucracy of the kind

$$\begin{array}{c}
 [R' T'] \\
 r' \text{-----} \\
 [R' T] \\
 r \text{-----} \\
 [R T]
 \end{array}
 \text{ vs. }
 \begin{array}{c}
 [R' T'] \\
 r \text{-----} \\
 [R T'] \\
 r' \text{-----} \\
 [R T]
 \end{array}
 ,$$

where the order of the application of two inference rules doesn't morally matter. In formalism A, one can write

$$\begin{array}{c}
 R' \quad T' \\
 [r \text{---} r' \text{---}] \\
 R \quad T
 \end{array}$$

and the problem is solved. This is an example of formalism-related bureaucracy: CoS only sees the two derivations above, and doesn't express the one below.

However, consider a deductive system where associativity is explicit (I mean, we are not working modulo associativity). Consider the following two derivations:

$$\begin{array}{c}
 [[a a] a] \\
 \text{ass} \text{-----} \\
 [a [a a]] \\
 \text{ac}_- \text{-----} \\
 [a a] \\
 \text{ac}_- \text{-----} \\
 a
 \end{array}
 \text{ vs. }
 \begin{array}{c}
 [[a a] a] \\
 \text{ac}_- \text{-----} \\
 [a a] \\
 \text{ac}_- \text{-----} \\
 a
 \end{array}
 .$$

They might be considered 'morally the same', but it is difficult to fix the problem in the formalism definition. Perhaps, a better idea is to fix the deductive system. For example, one can propose a deductive system with sort of a 'general atomic contraction', quotient by associativity, and go for the derivation

$$\begin{array}{c}
 [a a a] \\
 \text{gac}_- \text{-----} \\
 a
 \end{array}
 .$$

So, this could be an example of fixing the bureaucracy problems by fixing the deductive system (inside a given formalism).

A Problem with Commutativity and Associativity

In my opinion, the *very first* source of bureaucracy in *all deductive systems* in *all formalisms* is associativity and commutativity (when present) in formulae. I mean, in most cases, we do not want to distinguish formulae, and so proofs, just because of the order of associations, right?

The only practical way of dealing with commutativity is working under an equivalence relation that takes care of it. Associativity offers some more options. Anyway, working under associativity and commutativity, in a deductive system, is difficult. Actually, it is also dangerous.

Consider

$$\begin{array}{c}
 \begin{array}{cc}
 E & C \\
 [\mid \mid] \\
 [A B] A
 \end{array} \\
 * \frac{\quad}{\begin{array}{cc}
 A [B A] \\
 [\mid \mid] \\
 D & F
 \end{array}}
 \end{array}$$

This is a derivation (in formalism B [2]) in which two derivations are vertically composed by *, and we work under commutativity and associativity. The problem is that this is the only way we have in formalism B for representing (what I could graphically and imprecisely represent as)

$$\begin{array}{c}
 \begin{array}{cc}
 E & C \\
 [\mid \mid] \\
 [A B] A \\
 \mid \backslash \mid \\
 A [B A] \\
 [\mid \mid] \\
 D & F
 \end{array}
 \end{array}$$

However, the same derivation above could also stand for

$$\begin{array}{c}
 \begin{array}{cc}
 E & C \\
 \mid & \mid \\
 [[A B] A] \\
 \mid & \mid \\
 F & D
 \end{array}
 \end{array}$$

and this of course is *morally different!*

What can we do? Well, we could stop working under commutativity and associativity: this way we could easily distinguish between the two

cases. However, if we drop commutativity and associativity, we get back all the bureaucracy in formulae, with a vengeance, because now this bureaucracy scales up to proof composition.

Possible Solutions

Apart from developing the ideas in [2], there is now the possibility of designing a geometric formalism that solves the problems mentioned above, which I called *wired deduction*. I posted its possible definition(s) to the mailing list Frogs, and this generated a discussion [3]. For convenience, I reproduce in the appendix the email with the definition, but there is no room for reporting all the issues discussed on Frogs.

It is too early to tell whether this is the long-term solution we are looking for, however, the new formalism certainly works well for classical logic, and this is what I'd like to show at the workshop, since the ideas of wired deduction are all clearly exposed also in the case of classical logic.

References

[1] Alessio Guglielmi. Formalism A. URL: <http://iccl.tu-dresden.de/~guglielm/p/AG11.pdf>.

[2] Alessio Guglielmi. Formalism B. URL: <http://iccl.tu-dresden.de/~guglielm/p/AG13.pdf>.

[3] Alessio Guglielmi, Stéphane Lengrand and Lutz Straßburger. Emails at URLs: <http://thread.gmane.org/gmane.science.mathematics.frogs/219>, <http://thread.gmane.org/gmane.science.mathematics.frogs/220>.

Web Site

[WS] <http://alessio.guglielmi.name/res/cos>.

Appendix

Delivered-To: Frogs
Date: Tue, 15 Mar 2005 17:32:09 +0100
To: Frogs, Michel Parigot
From: Alessio Guglielmi
Subject: [Frogs] Wires and pipes
Cc: Dominic Hughes
List-Post: Frogs
List-Page: <<http://frogs.prooftheory.org>>

Hello,

in this message I propose a formalism and a deductive system for classical propositional logic.

The formalism, which I'd like to call 'wired deduction' (weird deduction!) should be the first example of deductive derivation net: it is an intrinsically

bureaucracy-free, deductive and geometric formalism. It naturally subsumes CoS and formalisms A and B.

As usual, I will define the formalism by way of a deductive system, and the natural choice is classical propositional logic. It is possible to define the formalism in isolation, in two ways: 1) geometrically, as a set of graph-forming rules; 2) deductively, by the definition for formalism B I showed at the workshop, *enriched by wires* (see below for what wires are).

These definitions are almost trivial after you see a deductive system. Before posting their details, I would like to receive some reactions about the deductive system, which you find below.

This email has three parts: Motivations, Intuition and Technicalities. Reading up to Intuition should be enough to get a good idea, if you know already about CoS and KS.

I would be very grateful if somebody checks the technicalities, though. They are nontrivial and unfortunately very combinatorial. Clearly, they might be wrong, but it should be possible to fix any mistake without changing the general picture.

Please, if you check, let me know, even if you find no mistakes (positive information is still very useful!).

Ciao,

-Alessio

MOTIVATION

=====

The general motivation is devising a formalism which is bureaucracy-free and *intrinsically* so. Moreover, we want the formalism to be geometric.

Bureaucracy-free means that it should be possible to express, inside the formalism, canonical representatives of derivations which are 'morally the same' according to some notion. See the message

<<http://article.gmane.org/gmane.science.mathematics.frogs/219>>

for an exposition of these ideas. See also the notes

<<http://iccl.tu-dresden.de/~guglielm/p/AG11.pdf>> ,
<<http://iccl.tu-dresden.de/~guglielm/p/AG13.pdf>> ,
<<http://www.iam.unibe.ch/~kai/Current/prty.pdf>> .

'Intrinsically' bureaucracy-free means that the formalism disallows the very formation of some redundant derivations. This notion is closely related, somehow, to the idea that the formalism should be geometric.

For a formalism, being 'geometric' means that derivations are some sort of graphs over which one operates locally and modulo some basic symmetries like those due to commutativity and associativity.

Much of the inspiration for wired deduction comes from subatomic logic, especially the idea of wires and the `ww_` rule. See

<<http://iccl.tu-dresden.de/~guglielm/p/AG8.pdf>> .

In the design of the deductive system for classical logic, I wanted to get rid once and for all of the unit equations. They have no strong justification in terms of 'war to bureaucracy' and they cause some technical problems. I think that the system below is particularly convincing in this respect. It might be possible to do better than I did, in the sense that some technicalities can perhaps be simplified.

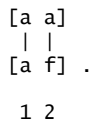
INTUITION
=====

I will attempt here a completely informal exposition of the system for classical logic, called KSw, which should be sufficient for people who know CoS and KS. The technical definitions are in the Technicalities part of this message (they are still subject to changes, of course).

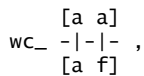
We need to operate under associativity and commutativity, in order to get rid of bureaucracy in formulae. However, in some cases we need to keep track of 'where the atoms come from'. The obvious solution would be to disambiguate these situations by resorting to occurrences. Wires go one step further: they allow following atoms and their transformations throughout a whole derivation.

The main idea goes as follows: there is a denumerable set of wires. Wires are neither created nor destroyed. To wires we associate atoms, and the association may vary in the course of the derivation.

Moreover, at any given time bunches of wires are organised into a tree of logical relations, which also can change over time. For example, the following is a derivation of a from (a V a) (so, it's a contraction):

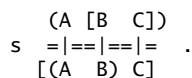


There are two wires, 1 and 2, vertically disposed, and we assume that time flows vertically going upwards. In the beginning, a is associated to 1 and f is associated to 2. In traditional logic, a would be a propositional variable and f would be the 'false' unit; both are atoms for us. Wires 1 and 2 are in a disjunction relation (indicated as usual in CoS). After some time wire 2 gets the value a, but the logical relation between wires does not change. We indicate this situation by the rule



which is of course supposed to apply in the middle of other wires. This is an example of atomic rule. A special feature of wired deduction is that atomic rules only work on wires, their values and their relations, by 'going through' a predetermined amount of them (in the case above, two). Atomic rules 'see' which atoms wires carry.

There is another kind of rule, the local rule. These are different than atomic rules, they only see *bunches* of wires, called *pipes*, and they reshuffle their logical relations. For example, take the switch rule



get disjunctive normal forms and then realising resolution and appeal to its completeness.

However, I also want to check that the complexity of proofs does not grow wrt KS, which is the place where we mostly study it. So, in the technicalities, you will find a complete proof of the admissibility of KS equations for KSw.

What is the secret of success? Part of the reason is in the fact that we can assume to have an unlimited supply of t's in conjunction and f's in disjunction. These atoms can be brought wherever they are needed by the switch rule. Doing this way does generate a small amount of bureaucracy of the deductive-system kind, for contraction and weakening rules. However, it is very easy to get rid of this bureaucracy by simple permutations. This is not very geometrical, but it is more geometrical than basically allowing $[R f] = R$ and $(R t) = t$ everywhere, so I went that way. In any case, there always is bureaucracy associated to the piling up of contraction and weakenings, as I showed in the previous message to Frogs, and this can be dealt with by using an appropriate deductive system with non-local rules (or by using a straightforward equivalence on proofs).

If you have more or less clear what I tried to explain above, you can jump directly to section 3 of the Technicalities and see KSw in action while getting rid of KS equations.

TECHNICALITIES =====

1 LANGUAGE

Definition We define the following:

- WW is a denumerable set of `_wires_`; we denote wires by natural numbers.
- PP is a set of `_pipes_`; we denote pipes by A, B, C, D and various decorations.
- SSF is the language of `_scheme skeleton formulae_`, produced by

$SSF ::= WW \mid PP \mid [SSF \ SSF] \mid (SSF \ SSF)$

and such that no wire and no pipe appears twice in any element of SSF; we denote scheme skeleton formulae by K; an `_instance_` of a scheme skeleton formula K is a scheme skeleton formula obtained by replacing in K any pipes by scheme skeleton formulae.

Example $K = [(1 \ 2) \ A]$ is a scheme skeleton formula, while $[(1 \ 1) \ A]$ is not. $K' = [(1 \ 2) \ (A \ B)]$ is an instance of K, while $[(1 \ 2) \ (A \ A)]$ is not. $[(1 \ 2) \ [(3 \ 4) \ (5 \ 6)]]$ is an instance of K'.

Definition AA is a denumerable set of `_atoms_`; we denote atoms by a, b and c; on atoms we have an involution $-: AA \rightarrow AA$ (i.e., $--a = a$); two special atoms f and t, called `_units_`, belong to AA, and $-f = t$. A `_scheme formula_` is a couple $(K, \text{wr } K \rightarrow AA)$, where `wr K` is the set of wires appearing in K; if no pipes appear in K, then the scheme formula is a `_formula_`; formulae are denoted by F.

Example If $K = [(1 \ 2) \ A]$ then $(K, \{1 \rightarrow a, 2 \rightarrow t\})$ is a scheme formula; if

$K' = [(1\ 2)\ 3]$ then $(K', \{1 \rightarrow a, 2 \rightarrow t, 3 \rightarrow a\})$ is a formula, corresponding to the classical propositional logic formula $((a \wedge t) \vee a)$.

Definition The equivalence \equiv on SSF is defined as the minimal equivalence relation such that

$$\begin{aligned} [K\ K'] &\equiv [K'\ K] , \\ (K\ K') &\equiv (K'\ K) , \\ [K\ [K'\ K'']] &\equiv [[K\ K']\ K''] , \\ (K\ (K'\ K'')) &\equiv ((K\ K')\ K'') , \\ \text{if } K &\equiv K' \text{ then } [K\ K''] &\equiv [K'\ K''] \text{ and } (K\ K'') &\equiv (K'\ K'') . \end{aligned}$$

The equivalence \equiv is applied naturally wherever scheme skeleton formulae appear. `_Structures_`, denoted by P, Q, R, T, U and V , are formulae modulo \equiv .

Examples and Notation We usually omit indicating wires, and we write, for example, $[(a\ t)\ a]$ in the place of $[(1\ 2)\ 3], \{1 \rightarrow a, 2 \rightarrow t, 3 \rightarrow a\}$. We have that $[(a\ b)\ [f\ a]] \equiv [a\ [f\ (b\ a)]]$. We drop unnecessary parentheses, so $[a\ [f\ (b\ a)]]$ can be written as $[a\ f\ (b\ a)]$.

Definition Two structures R and T are `_isomorphic_` if in their respective \equiv -equivalence classes there are two formulae which are equal modulo some permutation of wires.

Example Let

$$\begin{aligned} R &= ([[(1\ 2)\ 3]]_{\equiv}, \{1 \rightarrow a, 2 \rightarrow b, 3 \rightarrow c\}) , \\ T &= ([[(4\ 5\ 6)]]_{\equiv}, \{4 \rightarrow c, 5 \rightarrow a, 6 \rightarrow b\}) . \end{aligned}$$

Clearly,

$$T = ([[(5\ 6)\ 4]]_{\equiv}, \{5 \rightarrow a, 6 \rightarrow b, 4 \rightarrow c\}) ;$$

we can consider the permutation $\{1 \leftrightarrow 5, 2 \leftrightarrow 6, 3 \leftrightarrow 4\}$, and this shows that R and T are isomorphic.

Notation We usually do not indicate pipes, rather we use structure notation. So, for example, $([A\ B]\ C)$ is indicated as $([R\ T]\ U)$. This allows for an important shortcut: when we repeat letters, like in $(R\ R)$, we mean any structure

$$([([K\ K'])_{\equiv}, m), \text{ where } m: \text{wr } (K + K') \rightarrow AA,$$

such that $([K]_{\equiv}, m')$ and $([K']_{\equiv}, m'')$ are isomorphic, where m' and m'' are the restrictions of m to $\text{wr } K$ and $\text{wr } K'$, respectively.

Example $([R\ R]\ a)$ can be instantiated as $([(T\ T\ U)\ (T\ T\ U)]\ a)$ and $([(b\ f)\ (b\ f)]\ a)$, for example, but not as $([b\ c]\ a)$. $([R\ T]\ a)$ instead does not impose any restriction on R and T .

Definition An `_atomic inference rule_` is any expression of the kind

$$r \frac{F}{F'}$$

where F and F' are formulae such that the same wires appear in F and F' ; r is the `_name_` of the rule. We adopt a notation such that wires are not explicitly indicated, but they can be 'followed', for example

$$\text{ww}_- \frac{((1\ 2), \{1 \rightarrow f, 2 \rightarrow t\})}{([1\ 2], \{1 \rightarrow a, 2 \rightarrow f\})}$$

is denoted by

$$\text{ww}_- \frac{(f\ t)}{-|-|-} \quad \text{or} \quad \text{ww}_- \frac{(t\ f)}{-X-} .$$

Definition A *_local inference rule_* is any expression of the kind

$$r \frac{K}{K'} ,$$

where *K* and *K'* are scheme skeleton formulae where no wires appear and such that the same pipes appear in both; *r* is the *_name_* of the inference. We adopt a notation where we join vertically the pipes; for example

$$s \frac{(A\ [B\ C])}{\text{-----}} \\ [(A\ B)\ C]$$

is denoted by

$$s \frac{(A\ [B\ C])}{=|==|==|=} \quad \text{or} \quad s \frac{(A\ [C\ B])}{=|===X=}$$

Example System *KSw* for classical propositional logic is defined by the following rules

$$\text{wi}_- \frac{(t\ t)}{[a\ -a]} , \quad \text{wc}_- \frac{[a\ a]}{[a\ f]} , \quad \text{ww}_- \frac{(f\ t)}{[a\ f]} , \quad \text{for all } a \text{ in } AA,$$

$$s \frac{(A\ [B\ C])}{[(A\ B)\ C]} , \quad m \frac{[(A\ B)\ (C\ D)]}{([A\ C]\ [B\ D])} .$$

The first three rule (schemes) are atomic, the last two are local. They are called, respectively *_wired interaction_*, *_wired contraction_*, *_wired weakening_*, *_switch_* and *_medial_*.

2 COMPOSITION OF RULES

This part needs to be completed. For now, suffice to say that we compose rules like in the calculus of structures. Of course, it is possible to define more geometric notions of compositions, like for formalism B.

3 CLASSICAL PROPOSITIONAL LOGIC

Proposition The *_contraction_* rule

$$c_- \frac{[P\ P]}{\sim|\sim\wedge\sim} ,$$

is derivable for KSw.

Proof By structural induction on P. If P = a then consider

$$\begin{array}{c} S[a \ a] \\ \text{wc}_- \quad | \text{---} | \text{---} | \text{---} \\ | [a \ f] \\ S^* = | = | = | = \\ [S\{a\} \ f] \end{array} .$$

If P = [R T] then consider

$$\begin{array}{c} [R \ R \ T \ T] \\ \text{c}_- \quad | \sim | \sim \wedge \sim \\ [R \ R \ | \] \\ \text{c}_- \quad \sim | \sim \wedge \sim | \\ [R \ T \] \end{array} .$$

If P = (R T) then consider

$$\begin{array}{c} [(R \ T) \ (R \ T)] \\ \text{m} \quad = | = = X = = | = \\ ([\ | \ R \] \ [\ T \ T \]) \\ \text{c}_- \quad | \ | \ \sim | \sim \wedge \sim \\ ([R \ R] \ | \) \\ \text{c}_- \quad \sim | \sim \wedge \sim | \\ (R \ T \) \end{array}$$

<>

Proposition The following rules

$$\begin{array}{c} f \\ \text{aw}_- \quad \sim | \sim \quad (\text{_atomic weakening_}), \\ a \end{array}$$

$$\begin{array}{c} [a \ a] \\ \text{ac}_- \quad \sim | \sim \wedge \sim \quad (\text{_atomic contraction_}), \\ a \end{array}$$

$$\begin{array}{c} f \\ \text{r1} \quad \sim | \wedge | \sim, \quad \text{r2} \quad \frac{(f \ f)}{\sim | \sim \wedge \sim}, \quad \text{r3} \quad \frac{R}{\sim | \sim ! \sim}, \\ (f \ f) \quad f \quad (R \ t) \end{array}$$

$$\begin{array}{c} t \\ \text{r5} \quad \sim | \sim ! \sim, \quad \text{r6} \quad \frac{[t \ t]}{\sim | \sim \wedge \sim}, \quad \text{r8} \quad \frac{[R \ f]}{\sim | \sim \wedge \sim}, \\ [t \ t] \quad t \quad R \end{array}$$

are derivable for KSw.

Proof Consider, respectively:

$$\begin{array}{c} (S\{f\} \ t) \\ S^* = | = | = | = \\ | (f \ t) \\ \text{ww}_- \quad | \text{---} | \text{---} | \text{---} \\ | [a \ f] \\ S^* = | = | = | = , \\ [S\{a\} \ f] \end{array}$$

$$\begin{array}{c} [a \ a] \\ \text{c}_- \quad \sim | \sim \wedge \sim , \\ a \end{array}$$

$$\begin{array}{l}
(S(f) \ t \ t \ t \ t) \\
s^* = |=====| == | | | | = \\
| (| | | | t \ t) \\
wi_ | | | | -|-|- \\
| (| | | t \ t [t \ f]) \\
wi_ | | | | -|-|- | | \\
| (| | [t \ f] [t \]) \\
2.s | | | | =|====X====| = \\
| (| | | | [| t (f \ f)]) \\
s | =/ /==| == | =/ / = \\
| [(f \ f) (| [| |)] \\
s^* = |==| |==| | | = \\
| [(| |) (| [t \ t)]] \quad S(f \ f) \\
wc_ | | | | -|-|- \quad aw_ | | \sim | \sim \\
| [(| |) (| [t \ f)]] \quad | (f \ t) \\
s | | | | =|==| == | = \\
| [(| |) (f \ t) |] \quad ww_ | -|-|- \\
ww_ | | | | -|-|- | | \quad | [f \ f] \quad (S\{R\} \ t) \\
[S (f \ f) \ f \ f \ f] \quad , \quad s^* = |==| == | = \quad , \quad s^* = |==| == | = , \\
\quad \quad \quad [S\{f\} \ f] \quad \quad \quad S(R \ t)
\end{array}$$

$$\begin{array}{l}
(S\{t\} \ t \ t) \\
s^* = |==| == | | = \\
| (t \ t \ t) \\
wi_ | -|-|- | \\
| ([t \ f] |) \\
s | | | | \\
| [| (f \ t)] \\
ww_ | | | | -|-|- \\
s^* | =|==| == | = , \quad c_ \sim | \sim \wedge \sim , \quad s^* = |==| == | = . \\
[S [t \ t] \ f] \quad \quad \quad t \quad \quad \quad [S\{R\} \ f]
\end{array}$$

<>

Theorem <PP> In KSw, if S{P} is provable then S[P P] is provable.

Proof Induction on the length of the proof D of S{P}.

Base Case: If D = [(t t_t) f_f], we have to show that [[(t n.t) (t n.t)] t_t) f_f] is provable, for n ≥ 0. Take

$$\begin{array}{l}
[(\ t \ t \ n.t \ n.t \ t_t) \ f_f] \\
wi_ \quad -|-|- \quad | \quad | \quad | \quad | \\
| [([\ t \ f] \ | \ | \ | \) \ |] \\
aw_ \quad | \sim | \sim \quad | \quad | \quad | \quad | \\
| [([\ | \ t] \ n.t \ | \ | \) \ |] \\
2.s \quad =|====><====| = \quad | \quad | \\
| [((t \ n.t) \ (t \ n.t) \) \ t_t) \ f_f] .
\end{array}$$

Inductive Cases: If the bottommost rule instance in the proof of S{P} is like in

$$\begin{array}{l}
\overline{| |} \\
S\{P\} \\
r = |==| = \\
S\{P\}
\end{array}
\quad \text{or} \quad
\begin{array}{l}
\overline{| |} \\
S\{P'\} \\
r | = | = \\
S\{P\}
\end{array}
\quad \text{or} \quad
\begin{array}{l}
\overline{| |} \\
S\{P'\} \\
r | - | - \\
S\{P\}
\end{array}$$

then use the induction hypothesis on

$$\begin{array}{l}
\overline{| |} \\
S[P \ P] \\
r = |==| = \\
S [P \ P]
\end{array}
\quad \text{or} \quad
\begin{array}{l}
\overline{| |} \\
S[P' \ P'] \\
r | = | = \\
S [P \ P]
\end{array}
\quad \text{or} \quad
\begin{array}{l}
\overline{| |} \\
S[P' \ P'] \\
r | - | - \\
S [P \ P] .
\end{array}$$

On Two Forms of Bureaucracy in Derivations

Kai Brünnler¹ and Stéphane Lengrand²

¹ Institut für angewandte Mathematik und Informatik
Neubrückstr. 10, CH – 3012 Bern, Switzerland

² PPS, Université Paris 7, France

Abstract. We call irrelevant information in derivations *bureaucracy*. An example of such irrelevant information is the order between two consecutive inference rules that trivially permute. Building on ideas by Guglielmi, we identify two forms of bureaucracy that occur in the calculus of structures (and, in fact, in every non-trivial term rewriting derivation). We develop term calculi that provide derivations that do not contain this bureaucracy. We also give a normalisation procedure that removes bureaucracy from derivations and find that in a certain sense the normalisation process is a process of cut elimination.

1 Introduction

Consider the following two proofs in a sequent system for classical logic:

$$\frac{\frac{\frac{}{\vdash A, B, \bar{A}, C}}{\vdash A, B, \bar{A} \vee C}}{\vdash A \vee B, \bar{A} \vee C}}{\vdash A \vee B, \bar{A} \vee C} \quad \text{and} \quad \frac{\frac{\frac{}{\vdash A, B, \bar{A}, C}}{\vdash A \vee B, \bar{A}, C}}{\vdash A \vee B, \bar{A} \vee C}}{\vdash A \vee B, \bar{A} \vee C} .$$

Clearly, these two proofs are essentially the same, and we prefer not to distinguish them. More to the point, the sequent calculus forces us to choose an order of two rule applications that we do not want to choose because it is not relevant. Let us call *bureaucracy* this fact that a proof-theoretic formalism forces us to distinguish morally identical proofs. Proof nets, introduced by Girard [4] for linear logic, are a less bureaucratic formalism than the sequent calculus. They have also been developed for classical logic, for example by Robinson [13] and by Lamarche and Straßburger [11]. Proof nets have the merit that they do not distinguish between proofs such as the above. However, establishing the correctness of a proof net generally requires checking a global criterion which is more algorithmic than deductive. The notion of *deduction step* is lost when moving from the sequent calculus to proof nets. Let us informally call a formalism *deductive* if it has a notion of inference step. That is of course a vague notion that can be

made more precise in several ways, for example in asking for a locally checkable correctness criterion. The question then is whether there is a formalism that is both: bureaucracy-free and deductive. The quest for such *deductive proof nets* was initiated by Guglielmi. Starting from the calculus of structures [5,6] he designs two formalisms that reduce bureaucracy without losing deductiveness: they are called *Formalism A* [7] and *Formalism B* [8]. These formalisms are just steps towards deductive proof nets and not the final result. These formalisms still allow to form inessentially different proofs. However, they provide a third proof which is a canonical representative of the two. So, compared to the sequent calculus or the calculus of structures, the set of proofs grows larger. The ultimate aim in this quest is then a deductive formalism that does not allow the formation of non-canonical proofs in the first place.

Formalisms A and B address two kinds of bureaucracy that occur in every non-trivial term rewriting derivation. Formalism A addresses bureaucracy type A, where one has to choose an order for two consecutive applications of rewrite rules that apply to disjoint, i.e. non-overlapping, subterms of a term. Formalism B addresses bureaucracy type A and also bureaucracy type B, where one has to choose an order between two consecutive applications of rewrite rules where one rule applies to a term which is unchanged by the other rule because it is inside a variable. Clearly, equivalence in Formalism B then corresponds to standard notion of “equivalence modulo trivial permutation”.

The starting point of the work presented in this paper was our goal to provide a normalisation procedure for Formalisms A and B which, given a bureaucratic derivation, yields its bureaucracy-free representant. In [7,8] Guglielmi does not provide such a normalisation procedure, though he defines a set of proofs that seems rich enough to accommodate bureaucracy-free representatives. To achieve our goal, we found it natural and also necessary to depart a bit from the definitions in [7,8] in two ways. First, we use a term calculus to define derivations. This is just a notational difference, comparable to the difference between the λ -calculus and natural deduction in minimal logic. Second, we found it hard to work modulo the equational theory that is used not only in [7,8] but also generally in systems in the calculus of structures. It typically contains equations like associativity and commutativity of conjunction and disjunction, for example. We drop the equational theory and add those equations that are needed for completeness as rules.

The plan of the paper is as follows: we first present a linear term rewriting system that is a deductive system for classical propositional logic. We go on to define proof terms for derivations in Formalism A and give a rewriting system for these proof terms that removes bureaucracy, which, as we will see, turns out to be a process of cut elimination. The following section, where we do the same for Formalism B, is intended to be the heart of the paper, but is still a bit of a construction site. The central notion though is already there, it is that of a *tube* which is a placeholder which winds through a derivation and contains another

derivation. Tubes put an end to bureaucracy type B. Some discussion ends the paper.

2 Propositional Logic as Term Rewriting

A deductive system in the calculus of structures [6] is just a term rewriting system modulo an equational theory, cf. [10]. We should point out that the questions one asks about these systems are rather different. Typical properties of interest of a term rewriting system are termination and confluence, systems in the calculus of structures typically are neither. Typical questions to ask of systems in the calculus of structures are about the admissibility of rules and about the existence of certain normal forms for derivations. Nevertheless, term rewriting systems is what they are. In this section we present a linear term rewriting system on formulas in propositional logic such that a formula A rewrites to a formula B iff A implies B . This system is essentially obtained from system SKS from [2,1] by removing all equations and adding some of them as rewrite rules. The system is rather idiosyncratic and is not really central to the ideas developed here. We present it just in order to show that linear term rewriting indeed can serve as a proof-theoretic formalism and also to have some rules as running examples. Formalisms A and B as we present them easily generalise to any linear term rewriting system.

Formulas. There are propositional variables, denoted by v . Propositional variables v and their negations \bar{v} are *atoms*, they are denoted by a, b , and so on. The letters A, B, C denote formulas, which are defined as follows:

$$A ::= f \mid t \mid a \mid (A \vee A) \mid (A \wedge A)$$

where f and t are the units *false* and *true*. We define \bar{A} , the *negation* of the formula A , in the usual way:

$$\begin{array}{l} \bar{f} = t \quad \overline{A \vee B} = \bar{A} \wedge \bar{B} \\ \bar{t} = f \quad \overline{A \wedge B} = \bar{A} \vee \bar{B} \end{array} \quad \bar{v} = v \quad .$$

Term rewriting rules. A system of rewrite rules for classical propositional logic is given in Figure 1. The subsystem on the left is called KSf where K means classical, S means calculus of structures and f is for (equation-)free. The entire system is called SKSf, where the first S is for symmetric. On top of the arrow is the *label* of the rewrite rule. The labels of the rules on the left are short for duplication, unit, commutativity, identity, switch, weakening and contraction. Their dual rules on the right have the same name but with the prefix “co-”. When viewing formulas as terms, we view atoms as constants. A rewrite rule containing an atom, like $f \xrightarrow{aw\downarrow} a$ is shorthand for the set of rewrite rules one obtains by replacing a by any atom.

We have soundness and completeness for classical propositional logic.

$ \begin{array}{l} t \xrightarrow{\text{du}\downarrow} t \wedge t \quad A \xrightarrow{\text{un}\downarrow} A \vee f \\ A \vee B \xrightarrow{\text{co}\downarrow} B \vee A \\ t \xrightarrow{\text{i}\downarrow} \bar{A} \vee A \\ (A \vee B) \wedge (C \vee D) \xrightarrow{\text{s}\downarrow} (A \vee C) \vee (B \wedge D) \\ f \xrightarrow{\text{w}\downarrow} A \quad A \vee A \xrightarrow{\text{c}\downarrow} A \end{array} $	$ \begin{array}{l} A \wedge t \xrightarrow{\text{un}\uparrow} A \quad f \vee f \xrightarrow{\text{du}\uparrow} f \\ A \wedge B \xrightarrow{\text{co}\uparrow} B \wedge A \\ A \wedge \bar{A} \xrightarrow{\text{i}\uparrow} f \\ (A \wedge C) \wedge (B \vee D) \xrightarrow{\text{s}\uparrow} (A \wedge B) \vee (C \wedge D) \\ A \xrightarrow{\text{c}\uparrow} A \wedge A \quad A \xrightarrow{\text{w}\uparrow} t \end{array} $
--	--

Fig. 1. Rewrite rules for propositional logic

- Theorem 1**
1. $t \rightarrow_{\text{KSf}}^* A$ iff A is valid.
 2. $A \rightarrow_{\text{SKSf}}^* B$ iff A implies B .

Proof. Soundness in both cases follows from a simple induction on the length of the derivation and the observation that implication is closed under conjunction and disjunction. System KSf is complete: a formula can be derived from its conjunctive normal form via the rules $\text{c}\downarrow, \text{co}\downarrow, \text{s}\downarrow$. If the formula is valid, then each of the nested disjunctions in the conjunctive normal form contains two dual atoms. By $\text{w}\downarrow, \text{un}\downarrow$ this formula can be derived from a formula where all atoms except for the two dual atoms are removed. By $\text{i}\downarrow$ we derive this from a conjunction of lots of occurrences of t , which is derived from t by $\text{du}\downarrow$. The completeness direction of 2) is then a matter of constructing a derivation from A to B in SKSf for each derivation from t to $\bar{A} \vee B$ in KSf, see [1] for details.

Linear rewrite rules. From SKSf we obtain a linear rewriting system SKSfl, where l is for linear, which is shown in Figure 2. Derivability in this system is the same as in the nonlinear system, for details see [1].

- Theorem 2**
1. $A \rightarrow_{\text{SKSf}}^* B$ iff $A \rightarrow_{\text{SKSfl}}^* B$
 2. $A \rightarrow_{\text{KSf}}^* B$ iff $A \rightarrow_{\text{KSfl}}^* B$

3 Formalism A

Consider the following two rewrite paths (or derivations in the calculus of structures):

$$\text{ac}\downarrow \frac{(a \vee a) \wedge (b \vee b)}{\text{ac}\downarrow \frac{a \wedge (b \vee b)}{a \wedge b}} \quad \text{and} \quad \text{ac}\downarrow \frac{(a \vee a) \wedge (b \vee b)}{\text{ac}\downarrow \frac{(a \vee a) \wedge b}{a \wedge b}} .$$

$\begin{aligned} \mathbf{t} &\xrightarrow{\text{du}\downarrow} \mathbf{t} \wedge \mathbf{t} & A &\xrightarrow{\text{un}\downarrow} A \vee \mathbf{f} \\ A \vee B &\xrightarrow{\text{co}\downarrow} B \vee A \\ \mathbf{t} &\xrightarrow{\text{ai}\downarrow} \bar{a} \vee a \\ (A \vee B) \wedge (C \vee D) &\xrightarrow{\text{s}\downarrow} (A \vee C) \vee (B \wedge D) \\ (A \wedge B) \vee (C \wedge D) &\xrightarrow{\text{m}} (A \vee C) \wedge (B \vee D) \\ (A \vee B) \vee (C \vee D) &\xrightarrow{\text{m}_0\downarrow} (A \vee C) \vee (B \vee D) \\ \mathbf{f} &\xrightarrow{\text{w}_0\downarrow} \mathbf{f} \wedge \mathbf{f} & \mathbf{f} &\xrightarrow{\text{aw}\downarrow} a & a \vee a &\xrightarrow{\text{ac}\downarrow} a \end{aligned}$	$\begin{aligned} A \wedge \mathbf{t} &\xrightarrow{\text{un}\uparrow} A & \mathbf{f} \vee \mathbf{f} &\xrightarrow{\text{du}\uparrow} \mathbf{f} \\ A \wedge B &\xrightarrow{\text{co}\uparrow} B \wedge A \\ a \wedge \bar{a} &\xrightarrow{\text{ai}\uparrow} \mathbf{f} \\ (A \wedge C) \wedge (B \vee D) &\xrightarrow{\text{s}\uparrow} (A \wedge B) \vee (C \wedge D) \\ (A \wedge B) \vee (C \wedge D) &\xrightarrow{\text{m}} (A \vee C) \wedge (B \vee D) \\ (A \wedge B) \wedge (C \wedge D) &\xrightarrow{\text{m}_0\uparrow} (A \wedge C) \wedge (B \wedge D) \\ a &\xrightarrow{\text{ac}\uparrow} a \wedge a & a &\xrightarrow{\text{aw}\uparrow} \mathbf{t} & \mathbf{t} \wedge \mathbf{t} &\xrightarrow{\text{w}_0\uparrow} \mathbf{t} \end{aligned}$
---	---

Fig. 2. Linear rewrite rules for propositional logic

They essentially differ in the order in which the two rules are applied. No matter which of the two derivations we choose, it contains irrelevant information. We now define Formalism A which provides a third derivation which stores no information about the order between the two applications of $\text{ac}\downarrow$. The solution is of course very simple: we introduce a parallel composition of derivations.

Proof terms. Proof terms (or just terms) of Formalism A, denoted by R, S, T, U are defined as follows:

$$R ::= \text{id} \mid \rho \mid (R \mid R) \mid (R . R)$$

where id is *identity*, ρ is the label of a rewrite rule from Figure 2, $(R_1 \mid R_2)$ is *parallel composition* and $(R_1 . R_2)$ is *sequential composition*.

Typing rules. A judgement $A \xrightarrow{R} B$ which can be derived by the typing rules given in Figure 3 from the rewrite rules (aka typing axioms) in Figure 2 says that the proof term R allows to derive A implies B . Not each term is typeable, for example $\text{s}\downarrow . \text{s}\downarrow$ is not. In general, terms are typeable in different ways. For example we have $a \xrightarrow{\text{id}} a$, just like $b \xrightarrow{\text{id}} b$. We have soundness and completeness for classical propositional logic since we have the following theorem:

Theorem 3 *There is a proof term R of Formalism A with $A \xrightarrow{R} B$ iff $A \rightarrow_{\text{SKSfl}}^* B$.*

Proof. The direction from left to right is an easy induction on the typing derivation. The converse is easy to see since a rewrite rule can be applied at an arbitrary depth with a proof term build from the label of the rewrite rule, identity and parallel composition, and consecutive rule applications are represented using sequential composition.

$$\boxed{
\begin{array}{c}
A \xrightarrow{\text{id}} A \quad \frac{A \xrightarrow{R} B \quad B \xrightarrow{S} C}{A \xrightarrow{R.S} C} \\
\\
\frac{A \xrightarrow{R} C \quad B \xrightarrow{S} D}{A \wedge B \xrightarrow{R|S} C \wedge D} \quad \frac{A \xrightarrow{R} C \quad B \xrightarrow{S} D}{A \vee B \xrightarrow{R|S} C \vee D}
\end{array}
}$$

Fig. 3. Typing rules for Formalism A

Reduction rules. The reduction relation \rightarrow_A is given by the following rewrite rules:

$$\begin{array}{l}
R . \text{id} \rightarrow R \\
\text{id} . R \rightarrow R \\
\text{id} | \text{id} \rightarrow \text{id} \\
(R | S) . (T | U) \rightarrow (R . T) | (S . U)
\end{array}$$

Theorem 4 *The reduction relation \rightarrow_A is convergent.*

Proof. Each rule decreases the sum of the number of occurrences of id and the number of occurrences of parallel composition. Local confluence is easily checked.

We call normal forms of \rightarrow_A *canonical*. The reduction rules preserve types. If we call the typing rule for sequential composition “cut”, then they actually correspond to cut elimination steps in the typing derivation, as we will see in the proof of the following theorem. Note however that the cut rule for a typing derivation has nothing to do with a cut rule that may or may not be part of the logical system we represent using rewrite rules. In our case, all the rules with an up-arrow are in some sense cuts. Their admissibility follows from the previous section and is unrelated to the following theorem.

Theorem 5 (Subject reduction) *If $A \xrightarrow{R} B$ and $R \rightarrow_A^* S$ then $A \xrightarrow{S} B$.*

Proof.

$$\begin{array}{c}
\frac{A \xrightarrow{\text{id}} A \quad A \xrightarrow{R} B}{A \xrightarrow{\text{id}.R} B} \quad \sim \quad A \xrightarrow{R} B \\
\\
\frac{A \xrightarrow{\text{id}} A \quad B \xrightarrow{\text{id}} B}{A \wedge B \xrightarrow{\text{id}| \text{id}} A \wedge B} \quad \sim \quad A \wedge B \xrightarrow{\text{id}} A \wedge B
\end{array}$$

$$\begin{array}{c}
\frac{A \xrightarrow{R} E \quad B \xrightarrow{S} F}{A \wedge B \xrightarrow{R|S} E \wedge F} \quad \frac{E \xrightarrow{T} C \quad F \xrightarrow{U} D}{E \wedge F \xrightarrow{T|U} C \wedge D} \\
\hline
A \wedge B \xrightarrow{(R|S).(T|U)} C \wedge D \\
\sim \\
\frac{\frac{A \xrightarrow{R} E \quad E \xrightarrow{T} C}{A \xrightarrow{R.T} C} \quad \frac{B \xrightarrow{S} F \quad F \xrightarrow{U} D}{B \xrightarrow{S.U} D}}{A \wedge B \xrightarrow{(R.T)|(S.U)} C \wedge D}
\end{array}$$

Example. The two derivations from the beginning of the section are the following terms: $(\text{ac}\downarrow | \text{id}) . (\text{id} | \text{ac}\downarrow)$ and $(\text{id} | \text{ac}\downarrow) . (\text{ac}\downarrow | \text{id})$. Both normalise to $(\text{ac}\downarrow | \text{ac}\downarrow)$.

However, there still is bureaucracy remaining in the canonical derivations of Formalism A. Consider the following two derivations:

$$\text{ac}\downarrow \frac{(b \vee b) \wedge a}{b \wedge a} \quad \text{and} \quad \text{co}\downarrow \frac{(b \vee b) \wedge a}{a \wedge (b \vee b)} \quad , \\
\text{co}\downarrow \frac{b \wedge a}{a \wedge b} \quad \text{and} \quad \text{ac}\downarrow \frac{a \wedge (b \vee b)}{a \wedge b} \quad ,$$

which have the following proof terms: $(\text{ac}\downarrow | \text{id}) . \text{co}\downarrow$ and $\text{co}\downarrow . (\text{id} | \text{ac}\downarrow)$. There is no proof term in Formalism A that composes the two rules in such a way that no order between them is fixed. The next section will provide such a bureaucracy-free proof term.

4 Formalism B

Given an occurrence of an inference rule in a term, in general this rule can be permuted a certain distance to the left and a certain distance to the right (possibly both zero) until it hits another occurrence of an inference rule such that the two collide (do not permute). The actual position of the inference rule within these two points is irrelevant. To capture this free space between the two collision points we introduce *tubes*. Tubes have names, they have a start and an end, and they can be filled with derivations.

Types and proof terms. Starting from Formalism A, we extend the definition of formulas, which we now call types, and that of terms as follows:

$$A ::= \mathbf{f} | \mathbf{t} | a | (A \vee A) | (A \wedge A) | x_A^A$$

and

$$R ::= \text{id} | \rho | (R | R) | (R . R) | x \triangleright | \triangleleft x \quad .$$

where x is a name for a tube, in x_B^A the types A and B respectively are premise and conclusion of the tube, $x \triangleright$ marks the start of the tube x and $\triangleleft x$ marks the

end of the tube x . For each term we require that each tube name occurs at most once as a tube start and at most once as a tube end. Now our typing rules need to keep track of an environment ε , which is a finite partial mapping from tube names to terms. We write R, ε to denote a pair of a term and an environment. Given an environment ε which is undefined for x we write $\varepsilon, x : R$ to denote the environment which only differs from ε by mapping x to R .

Typing rules. The typing rules for Formalism B are shown in Figure 4.

$$\boxed{
\begin{array}{c}
A \xrightarrow{\text{id}, \varepsilon} A \qquad \frac{A \xrightarrow{R, \varepsilon} B \quad B \xrightarrow{S, \varepsilon} C}{A \xrightarrow{R.S, \varepsilon} C} \\
\\
\frac{A \xrightarrow{R, \varepsilon} C \quad B \xrightarrow{S, \varepsilon} D}{A \wedge B \xrightarrow{R|S, \varepsilon} C \wedge D} \qquad \frac{A \xrightarrow{R, \varepsilon} C \quad B \xrightarrow{S, \varepsilon} D}{A \vee B \xrightarrow{R|S, \varepsilon} C \vee D} \\
\\
\frac{A \xrightarrow{R, \varepsilon} B}{A \xrightarrow{x \triangleright, \varepsilon, x : R} x_B^A} \qquad \frac{A \xrightarrow{R, \varepsilon} B}{x_B^A \xrightarrow{\triangleleft x, \varepsilon, x : R} B}
\end{array}
}$$

Fig. 4. Typing rules for Formalism B

Just like in Formalism A, we have soundness and completeness for classical propositional logic since we have the following theorem:

Theorem 6 *For all formulas A, B there is a proof term R of Formalism A with $A \xrightarrow{R} B$ iff there is a proof term T of Formalism B with $A \xrightarrow{T} B$.*

Proof. The direction from left to right is obvious, just take an empty environment. For the converse, we first inductively define the premise $p(A)$ of a type A by pulling it over the propositional connectives and letting $p(x_B^A) = p(A)$. We define the conclusion $c(A)$ likewise, letting $c(x_B^A) = c(B)$. Now, by an easy induction on the typing derivation we establish that for all types A, B if we have $A \xrightarrow{T} B$ in Formalism B then there is a term R in Formalism A such that $p(A) \xrightarrow{R} c(B)$.

Normalisation process. To obtain a bureaucracy-free representant of a proof, we start from a proof term in Formalism A. The normalisation process has three steps.

The first step is an initialisation, in which for every rule we add its inner tubes, e.g. $\text{co}\downarrow$ is replaced by $(x \triangleright | y \triangleright) \cdot \text{co}\downarrow \cdot (\triangleleft y | \triangleleft x)$. We define the corresponding environment to map all occurring tubes to id .

The second step extends tubes as much as possible. It is a normalisation using the rewrite rules of Formalism A and the following rewrite rule which work both on the term and on the environment. The term R is either a parallel composition or an inference rule. The expression $S\{R\}\dots\{T\}$ denotes a term with (fixed occurrences of) subterms $R\dots T$. We refer to the first two rules as *tube extension* and to the third rule as *tube fusion*.

$$\frac{\langle x . R \quad \varepsilon, x : T}{\longrightarrow} \quad \langle x \quad \varepsilon, x : T.R$$

$$\frac{R . x \triangleright \quad \varepsilon, x : T}{\longrightarrow} \quad x \triangleright \quad \varepsilon, x : R.T$$

$$\frac{S\{x \triangleright\}\{\langle x . y \triangleright\}\{\langle y \rangle\} \quad \varepsilon, x : T, y : U}{\longrightarrow} \quad S\{x \triangleright\}\{\text{id}\}\{\langle x \rangle\} \quad \varepsilon, x : T.U, y : \text{id}$$

The third step is a cleanup phase, when all empty tubes are discarded:

$$\frac{S\{x \triangleright\}\{\langle x \rangle\} \quad \varepsilon, x : \text{id}}{\longrightarrow} \quad S\{\text{id}\}\{\text{id}\} \quad \varepsilon$$

Examples. The minimal example are the terms $(\text{id} \mid \text{ac}\downarrow) . \text{co}\downarrow$ and $\text{co}\downarrow . (\text{ac}\downarrow \mid \text{id})$ that both rewrite to:

$$(\text{id} \mid x \triangleright) . \text{co}\downarrow . (\langle x \mid \text{id} \rangle) \quad , \quad x : \text{ac}\downarrow \quad .$$

More than one rule can be inside a tube. $(\text{id} \mid \text{ac}\downarrow) . \text{co}\downarrow . (\text{ac}\uparrow \mid \text{id})$ rewrites to:

$$(\text{id} \mid x \triangleright) . \text{co}\downarrow . (\langle x \mid \text{id} \rangle) \quad , \quad x : \text{ac}\downarrow . \text{ac}\uparrow \quad .$$

Tubes can be nested. $((\text{ac}\downarrow \mid \text{id}) \mid \text{id}) . \text{co}\downarrow . (\text{id} \mid \text{co}\downarrow)$ rewrites to:

$$(x \triangleright \mid \text{id}) . \text{co}\downarrow . (\text{id} \mid \langle x \rangle) \quad , \quad \begin{array}{l} x : (y \triangleright \mid \text{id}) . \text{co}\downarrow . (\text{id} \mid \langle y \rangle) \\ y : \text{ac}\downarrow \end{array} \quad .$$

The reduction relation preserves types:

Theorem 7 (Subject reduction) *If $A \xrightarrow{R} B$ and $R \rightarrow_{\mathbb{B}}^* S$ then $A \xrightarrow{S} B$.*

Proof. It is easy to check that the first and third step preserve typing, we give the necessary transformation of the typing derivation for tube extension in the

second step. Tube fusion works similarly. The derivation

$$\begin{array}{c}
\frac{A \xrightarrow{T,\varepsilon} B}{A \xrightarrow{x \triangleright, \varepsilon, x:T} x_B^A} \quad \frac{\frac{A \xrightarrow{T,\varepsilon} B}{x_B^A \xrightarrow{x \triangleright, \varepsilon, x:T} B} \quad B \xrightarrow{R,\varepsilon, x:T} C}{x_B^A \xrightarrow{\triangleleft x.R, \varepsilon, x:T} C} \\
\hline
\Delta \\
D \xrightarrow{S\{x \triangleright\}\{\triangleleft x.R\}, \varepsilon, x:T} E
\end{array}$$

transforms into

$$\begin{array}{c}
\frac{A \xrightarrow{T,\varepsilon} B \quad B \xrightarrow{R,\varepsilon} C}{A \xrightarrow{T.R,\varepsilon} C} \quad \frac{A \xrightarrow{T,\varepsilon} B \quad B \xrightarrow{R,\varepsilon} C}{A \xrightarrow{T.R,\varepsilon} C} \\
\hline
\frac{A \xrightarrow{x \triangleright, \varepsilon, x:T.R} x_C^A} \quad \frac{x_C^A \xrightarrow{\triangleleft x, \varepsilon, x:T.R} C} \\
\hline
\Delta[x_B^A/x_C^A] \\
D \xrightarrow{S\{x \triangleright\}\{\triangleleft x\}, \varepsilon, x:T.R} E
\end{array}$$

where a typing derivation for $B \xrightarrow{R,\varepsilon} C$ can be obtained from the one for $B \xrightarrow{R,\varepsilon, x:T} C$ since neither end of tube x can occur in R .

Conjecture 8 *The normalisation process is convergent modulo naming of tubes.*

5 Discussion

Is all bureaucracy gone now? Unfortunately, no. This work is only at the beginning and even the basic notions of types and terms in Formalism A are not stable yet. There still is bureaucracy due to associativity of sequential and parallel composition, such as $(R.T).U$ versus $R.(T.U)$. For sequential composition this is easy to get rid of by using multiary function symbols and by writing $(R.T.U)$. We have to suitably generalise the typing rule as follows:

$$\frac{A_1 \xrightarrow{R_1} A_2 \quad A_2 \xrightarrow{R_2} A_3 \quad \dots \quad A_{n-1} \xrightarrow{R_{n-1}} A_n}{A_1 \xrightarrow{R_1.R_2 \dots R_{n-1}} A_n}$$

For parallel composition, however, the case is more complicated. In general, we cannot be sure that our canonical terms are bureaucracy-free until we have shown them to be in one-to-one correspondence with equivalence classes of rewriting derivations modulo trivial permutation. We are not there yet.

3-categories. Obtaining associativity of parallel composition will also be necessary in order to achieve our goal of making terms in Formalism B form a *3-category* à la Albert Burroni [3]. It seems that arrows in a 3-category capture exactly the bureaucracy we have in mind. See also Yves Guiraud’s work [9] on the relationship between deep inference and 3-categories.

Type checking redundancy. The system of typing rules given for formalism B has the disadvantage that a derivation in a tube has to be type checked twice: once for the start of the tube and once for the end of the tube. It would be interesting to develop a type system where it has to be type checked only once, maybe by defining proof terms and types as

$$A ::= f \mid \mathbf{t} \mid a \mid (A \vee A) \mid (A \wedge A) \mid x$$

and

$$R ::= \text{id} \mid \rho \mid (R \mid R) \mid (R . R) \mid x \triangleright \mid \triangleleft x \mid (x, A) \triangleright \mid \triangleleft (x, A) \quad .$$

and replacing the rules for the tubes by the following ones:

$$A \xrightarrow{(x,A)\triangleright,\emptyset} x \qquad x \xrightarrow{\triangleleft(x,A),\emptyset} A$$

$$\frac{A \xrightarrow{R\{(x,C)\triangleright\}\{(x,D)\},\varepsilon} B \quad C \xrightarrow{S,\varepsilon} D}{A \xrightarrow{R\{x\triangleright\}\{\triangleleft x\},\varepsilon,x:S} B} \quad .$$

Church vs. Curry We chose Curry-style typing for brevity, but it could be done in Church style. Then we need two parallel constructors, one for conjunction and one for disjunction. All inference rules are then parametrised by their types and instead of $A \xrightarrow{\text{id}} A$ for every A , we have for each A an $\text{id}(A)$ such that $A \xrightarrow{\text{id}(A)} A$. Church style could be more convenient for enforcing associativity of parallel composition or for type checking.

Rewriting Logic. There is a close connection with rewriting logic that needs to be made explicit. In the language of rewriting logic [12], our goal with Formalism B is to give canonical representants for arrows in the initial model of a rewrite theory if the rewrite theory is linear and without equations. Speaking of rewriting logic: the deductive system for rewriting logic as given in [12] already provides proof terms that are free of bureaucracy type A. Its congruence rule corresponds to our rule for parallel composition. However, this deductive system does not provide derivations that are free of bureaucracy of type B. To see that one needs to consider an example with two rules that do not permute and a third rule that permutes through both.

Bureaucracy in the formalism vs. bureaucracy in the logic. The axioms (or rewrite rules) of SKSf just served as an example here, we really are addressing bureaucracy in the formalism, which is independent of the particular logic that we are formalising. For the attack on logic-independent bureaucracy two obvious directions for further work are the extension of our approach 1) to term rewriting systems in general, not only those with linear rules, and 2) to term rewriting systems modulo equations. But there is also logic-dependent bureaucracy that needs to be taken care of such as in classical logic a weakening followed by a contraction, to name a simple example.

References

1. Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. PhD thesis, Technische Universität Dresden, September 2003.
2. Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 347–361. Springer-Verlag, 2001.
3. Albert Burroni. Higher-dimensional word problems with applications to equational logic. *Theoretical Computer Science*, 115(1):43–62, 1993.
4. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
5. Alessio Guglielmi. The calculus of structures website. Available from <http://www.ki.inf.tu-dresden.de/~guglielm/Research/>.
6. Alessio Guglielmi. A system of interaction and structure. Technical Report WV-02-10, Technische Universität Dresden, 2002. To appear in *ACM Transactions on Computational Logic*.
7. Alessio Guglielmi. Formalism A. Manuscript. <http://iccl.tu-dresden.de/~guglielm/p/AG11.pdf>, 2004.
8. Alessio Guglielmi. Formalism B. Manuscript. <http://iccl.tu-dresden.de/~guglielm/p/AG13.pdf>, 2004.
9. Yves Guiraud. The three dimensions of proofs. Manuscript. <http://iml.univ-mrs.fr/~guiraud/recherche/cos.pdf>, 2005.
10. Ozan Kahramanoğulları. Implementing system BV of the calculus of structures in Maude. In Laura Alonso i Alemany and Paul Égré, editors, *Proceedings of the ESSLLI-2004 Student Session*, pages 117–127, Université Henri Poincaré, Nancy, France, 2004.
11. François Lamarche and Lutz Straßburger. Naming proofs in classical propositional logic. In Paweł Urzyczyn, editor, *Typed Lambda Calculi and Applications, TLCA 2005*, volume 3461 of *Lecture Notes in Computer Science*, pages 246–261. Springer-Verlag, 2005.
12. Narciso Martí-Oliet and José Meseguer. Rewriting logic: roadmap and bibliography. *Theoretical Computer Science*, 285(2):121–154, 2002.
13. Edmund P. Robinson. Proof nets for classical logic. *Journal of Logic and Computation*, 13(5):777–797, 2003.

Completeness of MLL proof-nets w.r.t. weak distributivity

Jean-Baptiste Joinet

Equipe *Preuves-Programmes-Systèmes*
CNRS - Université Paris 7 (UMR 7126)

Case 7014, 2 place Jussieu

F-75251 Paris cedex 05, France

joinet@pps.jussieu.fr,

WWW home page: www-philosophie.univ-paris1.fr/Joinet

Abstract. We examine ‘weak-distributivity’ as a rewriting rule $\mathbb{W}\mathbb{D}$ defined on multiplicative proof-structures (so, in particular, on multiplicative proof-nets: MLL). This rewriting does not preserve the type of proof-nets, but does nevertheless preserve their correctness. The specific contribution of this paper, is to give a direct proof of completeness for $\mathbb{W}\mathbb{D}$: starting from a set of simple generators (proof-nets which are a n -ary \otimes of \wp -ized axioms), any mono-conclusion MLL proof-net can be reached by $\mathbb{W}\mathbb{D}$ rewriting (up to \otimes and \wp associativity and commutativity).

1 Preliminaries

1.1 Multiplicative Linear Logic: sequent calculus and proof-nets

The formulas of Multiplicative Linear Logic [1] are defined from the following grammar:

$$A = \underbrace{X, X^\perp, Y, Y^\perp, \dots}_{\text{Atoms}} \mid \underbrace{A \otimes A}_{\text{Tensor}} \mid \underbrace{A \wp A}_{\text{Par}}$$

Negation (“orthogonal”) $(.)^\perp$ is not a connective, but a defined unary operation over formulas, inductively defined by:

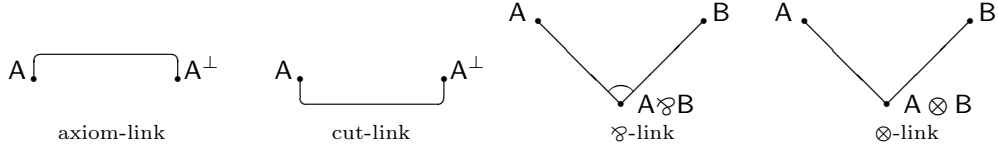
$$(X)^\perp = X^\perp, (X^\perp)^\perp = X, (A \otimes B)^\perp = A^\perp \wp B^\perp, (A \wp B)^\perp = A^\perp \otimes B^\perp$$

A sequent is a multiset Γ of formulas, written $\vdash \Gamma$. The rules of sequent calculus MLL (from which MLL sequent calculus derivations is inductively defined as usual) are :

$$\frac{}{\vdash A, A^\perp} \quad \text{cut} \frac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta} \quad \wp \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \wp B} \quad \otimes \frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}$$

(Identity axiom)

Definition 1 Let A, B be any formulas. The following four minigraphs are called the multiplicative ‘links’.



In those minigraphs, edges are intended to be oriented: in both directions for axiom-links and cut-links, and following the natural downward orientation of space for the others (the curve edge in the \wp -link is just an indication used later). A formula occurrence (vertex) which is a target (resp. a source) of an (oriented) edge in a link is a ‘conclusion’ (resp. a ‘premise’) of that link if it is reached downward (resp. if it is the source of a downward beginning edge).

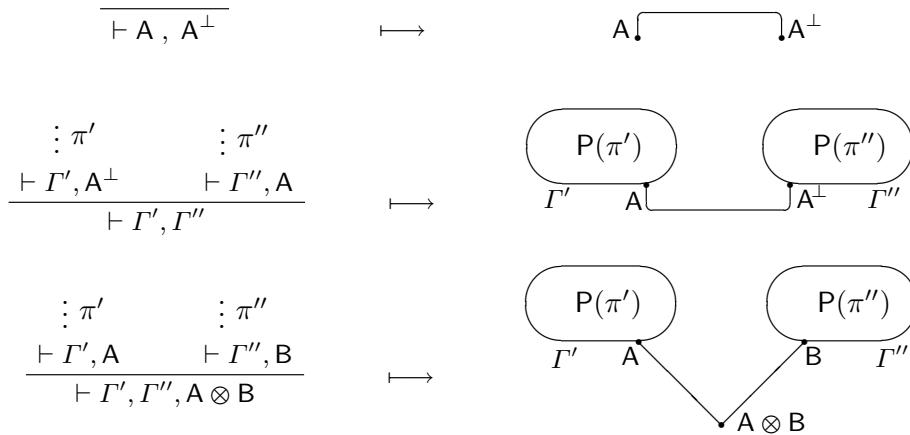
Definition 2 A *structure* is a graph whose vertices are (labeled with) formulas inductively built by using the rules below:

1. links are structures,
2. structures are closed by multiset-union,
3. structures are closed by identification (in a structure) of occurrences of (vertices labeled with) a same formula, if this identification preserves the fact that any formula occurrence is conclusion of at most one link and premise of at most one link.

The top (resp. bottom) formulas of a structure are called its hypothesis (resp. conclusions).

In the sequel, we will mainly work with the cut-free fragment of the set of proof-structures. Also, we will only represent, in structures, the formulas being conclusion of axiom-links. Indeed, taking account of the mark put on \wp -links (the curve edge), this information is sufficient for recovering all missing formulas (in the cut free fragment, the ambiguity occasioned, when some formulas are so missing, by commutativity, is harmless).

Let π a sequent calculus proof in MLL. Let $P(\pi)$ be the structure (with same conclusions as π) obviously defined by recurrence on π construction as presented below:



$$\frac{\vdots \pi^-}{\vdash \Gamma, A, B} \quad \longmapsto \quad \begin{array}{c} \text{P}(\pi^-) \\ \Gamma \quad A \quad B \\ \swarrow \quad \searrow \\ \text{A} \wp \text{B} \end{array}$$

Definition 3 The set of ‘multiplicative proof-nets’ (still noted MLL) is the range of $P(\cdot)$.

Let us recall that $P(\cdot) : \text{MLL} \longrightarrow \{\text{structures}\}$ is neither injective (reason why it is interesting), nor surjective (reason why it is a little bit difficult to deal with proof-nets). And indeed, in the prehistory of proof-nets, to find an intrinsic characterization of proof-nets among structures (standard old “sequentialization” problem) was the first question to solve :

Definition 4 A structure S is sequentializable, if $P(\pi) = S$ for some sequent derivation $\pi \in \text{MLL}$.

In this paper, we will deeply use sequentialization tools worked out by Girard [1] and Danos and Regnier [2], and in particular:

Definition 5 Let S be a structure.

- A *switching* in S is any sub-graph of S one gets by erasing one edge in any \wp -link in S .
- A structure S satisfies the Danos-Regnier criterion ($S \Vdash DR$) if any of its switchings is acyclic and connected.
- A structure S (possibly) with hypothesis, such that $S \Vdash DR$, is called a *module*.

Proposition 6 (Danos-Regnier) Let S an hypothesis free structure.

$$S \text{ is sequentializable} \quad \text{iff} \quad S \Vdash DR$$

1.2 Pretypes of modules and orthogonality

In this subsection a few tools and results used later are recalled. They all come from Danos thesis (see [3]).

Definition 7 The “border” of a structure S is the multiset of its hypothesis and conclusions (notation: $\text{Border}(S)$).

For sake of simplicity, every time no attention is needed to the specific formulas being conclusion or hypothesis of the structures under consideration, and in particular in the present subsection, we will forget them, then just using indices (integers) to describe their border.

Definition 8 Let S a structure and σ a switching of S . The partition of $\text{Border}(S)$ induced by σ , is the quotient of $\text{Border}(S)$ by the relation defined by: “ n is (in σ) in the same connected component as m ”.

To note a given partition of, say, $\{1, 2, 3, 4, 5\}$, for instance $\{\{1\}, \{2, 4\}, \{3, 5\}\}$, we will use the simplified notation: $\underline{1} \underline{24} \underline{35}$.

Definition 9 The pretype¹ of S is the set P_S of all partitions induced over $\text{Border}(S)$ by all switchings of S .

Definition 10 If $\text{Border}(S) = \{1, \dots, n\}$ and $\text{Border}(S') = \{1', \dots, n'\}$, we note $S :: S'$ the graph resulting of the plugging of S and S' together via the border (identifying vertex i with i').

Definition 11 The *meeting graph* $\mathcal{G}(p, q)$ of partitions p, q over $\{1, \dots, n\}$, is the graph whose set of vertices is $p \uplus q$, and such that one puts one edge between a class in p and a class in q for each point they share.

Examples Meeting graphs of partitions over $\{1, 2, 3, 4, 5\}$

$$\begin{array}{ccc}
 p = \underline{1} \underline{23} \underline{45} & & r = \underline{123} \underline{45} \\
 \quad | \diagup | \diagdown | & & \quad () \diagdown \diagup | \\
 q = \underline{12} \underline{34} \underline{5} & & q = \underline{12} \underline{34} \underline{5} \\
 \mathcal{G}(p, q) & & \mathcal{G}(r, q)
 \end{array}$$

Definition 12 p is orthogonal to q (notation: $p \perp q$), if $\mathcal{G}(p, q)$ is acyclic and connected (so in the example above: $p \perp q$, but $r \not\perp q$).

Definition 13 Two sets P and Q of partitions over $\{1, \dots, n\}$ are orthogonal, if they are pointwise orthogonal (notation: $P \perp Q$)

Definition 14 $P^\perp = \{q ; \forall p \in P, q \perp p\}$

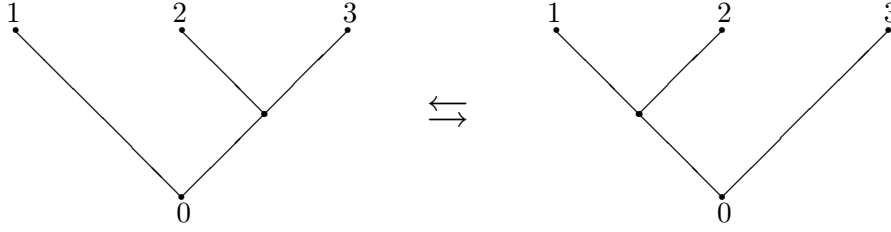
Remark 1 $P \perp Q \Rightarrow P \subseteq Q^\perp$

Proposition 15 (Danos) $S :: S'$ is a proof-net $\Leftrightarrow P_S \perp P_{S'}$

Examples

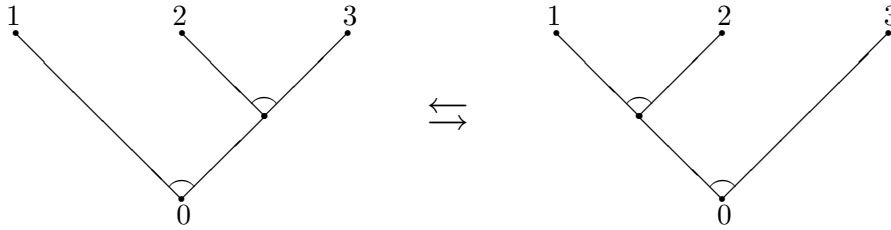
1. \otimes -associativity:

¹ I follow Maieli and Puite who, in [4], call ‘pretype’ of a module what Danos calls in [3] its ‘type’ (a terminology that the former reserve to the double orthogonal of the ‘type’ in the sense of Danos)



The two modules above have same pretype: $\{0\underline{123}\}$. We thus can replace in a given proof-net any sub-graph like one of those above by the other one: although such a replacement changes the type of the proof-net, by proposition 15, the satisfaction of the *DR*-correctness criterion is preserved.

2. \wp -associativity:



The two modules above have same pretype: $\{0\underline{1} \underline{2} \underline{3}, \underline{1} 0\underline{2} \underline{3}, \underline{1} \underline{2} 0\underline{3}\}$. Same remark as above.

(To put in in one slogan: “Pretype equality² = free associativity”).

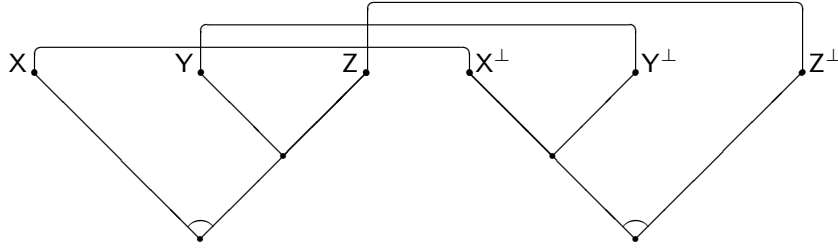
Remark As far as *commutative* MLL is in concern, structures differing only up to commutativity are not distinguished. This means however, that when one plugs S and S' together to build $S::S'$, one has to pay attention to continue to identify same indices of the border.

2 Weak-distributivity: a proof-net, a computational morphism

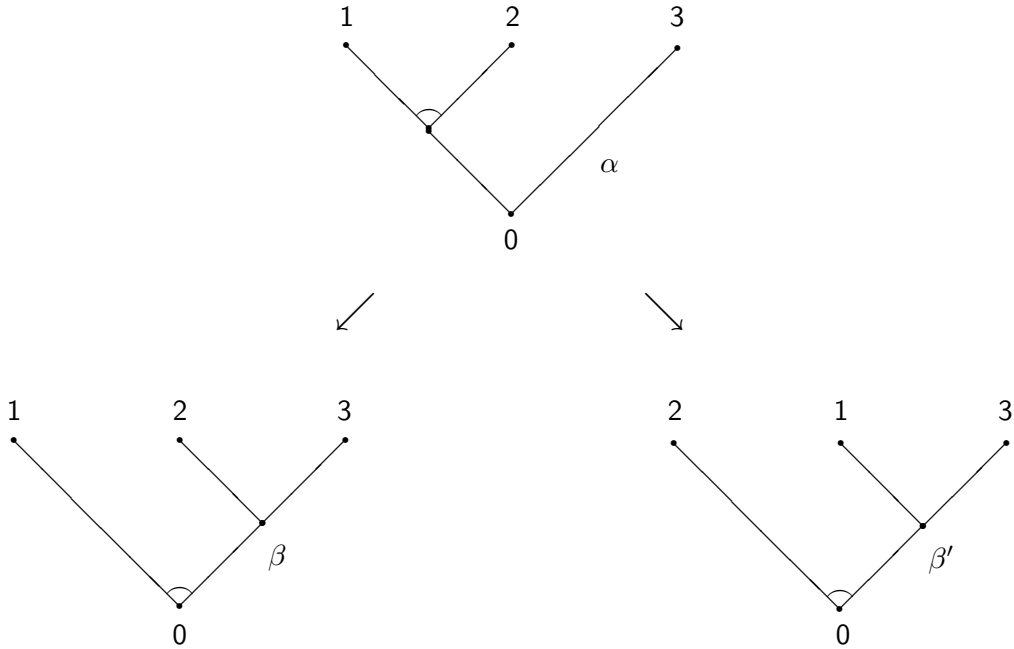
The formula $(A \vee B) \wedge C \rightarrow A \vee (B \wedge C)$ usually called the *weak-distributivity* formula (WD), is a theorem of LK (sequent calculus for classical logic) and even actually of its fragment just presented: multiplicative linear logic. Indeed, the multiplicative version of WD, the sequent $(A \wp B) \otimes C \vdash A \wp (B \otimes C)$ or its one-sided avatar $\vdash (A^\perp \otimes B^\perp) \wp C^\perp, A \wp (B \otimes C)$ are derivable in the corresponding sequent calculi.

It is easy to see that in MLL proof-net syntax, there is a unique cut free net N_{WD} with conclusions $X \wp (Y \otimes Z), (X^\perp \otimes Y^\perp) \wp Z^\perp$ (X, Y, Z distinct atoms). Here it is:

² Notice that the equality of the pretypes, in both case above, is not affected by the commutativity of the connective respectively involved



Lemma 16 Under cut-elimination, N_{WD} dynamically acts as a ‘surgical morphism’ thanks to which one is able to replace in any cut-free proof-net any sub-graph of shape α by either the module β or the module β' (all pictured below), anything else remaining unchanged.



Proof One has to observe the effect of cut elimination, when N_{WD} (or, to be precise, $N_{WD}[A/X, B/Y, C/Z]$) is cut against a proof-net with a terminal α , thus corresponding to a conclusion $(A \wp B) \otimes C$. Naturally, the module α being in general ‘deep’ (i.e. not terminal) in the proof-net, one previously needs to complete N_{WD} downwards, in the obvious way, by identities [5]. \square

Nota: in MLL, cut-elimination is deterministic (for a given instance of a cut-link, a unique reduction step applies). So the apparent non determinism comes here from the fact that the spatial representation of binary links (\otimes and \wp -links) does not uniquely determine their conclusion (proof-nets are defined up to commutativity).

We now internalize as a rewriting rule the transformation realized by N_{WD} under cut-elimination.

3 Weak-distributivity as a sound and complete rewriting

3.1 The completeness problem

Definition 17 Let $\overset{\text{wd}}{\rightsquigarrow}_1$ be the smallest binary relation over the set of structures including $\{(\alpha, \beta), (\alpha, \beta')\}$ and compatible with structures construction. Let $\overset{\text{wd}}{\rightsquigarrow}$ denotes the transitive and reflexive closure of $\overset{\text{wd}}{\rightsquigarrow}_1$, a.k.a. the rewriting rule over structures generated by $\overset{\text{wd}}{\rightsquigarrow}_1$.

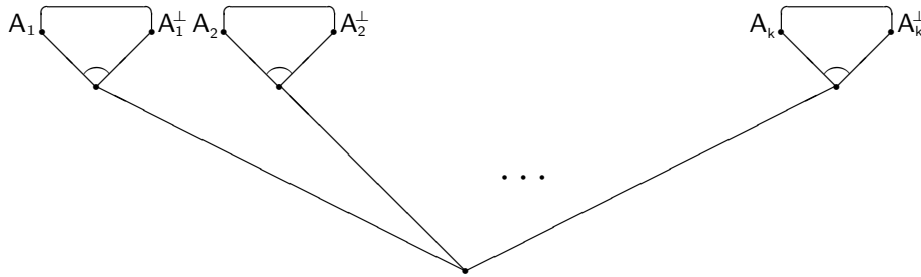
Notice that $\overset{\text{wd}}{\rightsquigarrow}_1$ rewriting does not preserved the type (the proof-net conclusions), nevertheless:

Proposition 18 The set of MLL proof-nets is closed under $\overset{\text{wd}}{\rightsquigarrow}$ rewriting.

Proof By lemma 16 and closure of MLL proof-nets under cut-elimination. This could as well be proved by means of the ‘Pretypes’ technics presented in subsection 1.2 (as Maieli and Puite did in [4]). \square

A natural question then is whether any proof-net can be obtained that way from a relevant basic set of proof-nets (*completeness problem*). Taking account of the kind of \otimes/\wp permutations involved in $\overset{\text{wd}}{\rightsquigarrow}$ rewriting, a rather natural candidate for this set of generators is the set of proof-nets in which the \wp -links are ‘as high as possible’, the \otimes -links ‘as low as possible’. To give a more compact representation of those proof-nets, it is useful to consider proof-nets up to \otimes associativity, an harmless quotientation as far as conservation of proof-nets type is no more in concern (see subsection 1.2), and to use an n -ary \otimes -link for the representation:

Definition 19 A proof-net is in canonical form if it is (a representative) of the following form :



We will refer to *axiom links whose conclusions are both premises of a same \wp -link* (as above) as \wp -ized axioms. So a proof-net in canonical form is “a k -ary \otimes of \wp -ized axioms”.

So, to sum up within the ‘canonical form’ terminology, one knows from proposition 18, that:

Proposition 20 (Soundness) Any proof-structure generated from a proof-net in canonical form by $\overset{\text{wd}}{\rightsquigarrow}$ rewriting is a proof-net.

And the problem which remains to solve is the following:

Problem (Completeness) Can any proof-net be generated from some canonical proof-net by \rightsquigarrow rewriting?

That question (or at least a similar one) has already been solved by categorical means in [6]. The aim of the present section, however, is to give an original, direct, combinatorial proof.

Notice that the naive idea which likely comes first in mind to prove completeness (namely by just reversing the rewriting) is wrong: the converse \rightsquigarrow_1^- of \rightsquigarrow_1 rewriting does *not* generally preserve the correctness criterion: proof-nets are *not* closed under \rightsquigarrow^- (the transitive and reflexive closure of \rightsquigarrow_1^-). Worst, there exists proof-nets for which any \rightsquigarrow^- rewriting leads out of the set of proof-nets (examples listed at the beginning of subsection 3.3).

However, we are going to show that inasmuch one considers only proof-nets with a unique conclusion ('mono-conclusion' proof nets) and one works up to associativity and commutativity, then:

1. in a proof-net (not being in canonical form), *there exists* a sub-graph of shape β for which one of the corresponding \rightsquigarrow^- replacements is correct;
2. some ordinal attached to proof-structures decreases when one performs such a replacement.

3.2 Doubly splitting \otimes_{\wp}

Definition 21

1. Following Girard's terminology in [1], an instance of a \otimes -link in a (say, connected) structure S is a *splitting- \otimes* , if the erasure in S of that link (conclusion vertex and edges) produces two unconnected structures.
2. Following Danos terminology in [3], an instance of a \wp -link in a (say, connected) structure S is a *splitting- \wp* (*un \wp scindant*), if the erasure of the two edges of that link in S produces two unconnected graphs (we will then note the upper one - a structure - by S_+).
3. In a proof-structure S , a splitting \wp -link surmounted by a \otimes -link which itself splits the corresponding S_+ , will be called a *doubly splitting \otimes_{\wp}* .

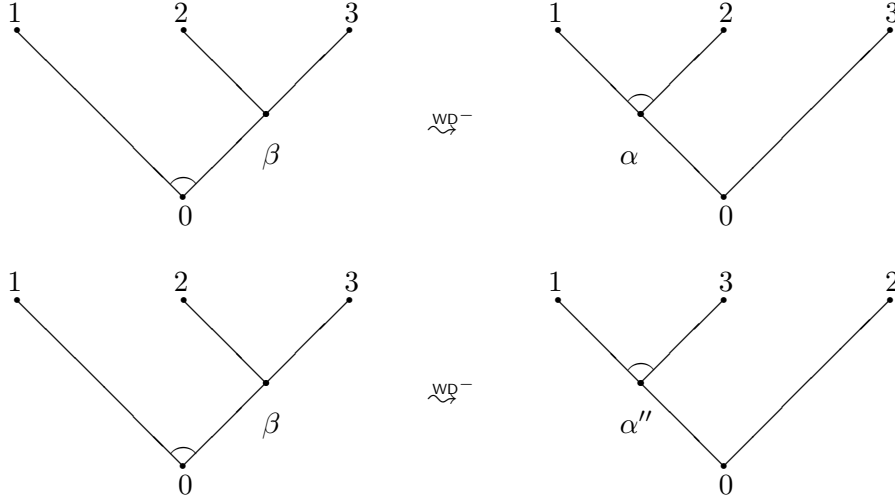
Concerning proof-nets splittings, we will use soon the two following lemmas both picked up from the saga of proof-nets 'sequentialization' tools.

Lemma 22 (Girard, [1]) Let N a proof-net with no terminal \wp -link. If the set of \otimes -link in N is not empty, then (the set of terminal \otimes -link in N is not empty and) one of them is a splitting- \otimes .

Lemma 23 (Danos, [3]) Let N a proof-net. If the set of \wp -link in N is not empty, one of them is a splitting- \wp .

Proposition 24 The $\overset{\text{WD}^-}{\rightsquigarrow}$ rewriting rule preserves correctness when applied to a doubly splitting $\overset{\otimes}{\circlearrowleft}$.

Proof Let us consider again the module β met in lemma 16 (the treatment of β' is similar). Up to \circlearrowleft and \otimes commutativity, β has two images (α and α'') by $\overset{\text{WD}^-}{\rightsquigarrow}$ as pictured below:



Let us calculate, for each module above, its pretype (the various partitions induced by switchings over its border $\{0, 1, 2, 3\}$) and the orthogonal of this pretype. We have:

$$\begin{aligned}
 P_\beta &= \{\underline{023 \ 1}, \underline{10 \ 23}\} & P_\alpha &= \{\underline{023 \ 1}, \underline{013 \ 2}\} \\
 P_\beta^\perp &= \{\underbrace{\underline{0 \ 12 \ 3}}_{p_1}, \underbrace{\underline{0 \ 13 \ 2}}_{p_2}\} & P_\alpha^\perp &= \{p_1\} \\
 & & P_{\alpha''} &= \{\underline{023 \ 1}, \underline{012 \ 3}\} \\
 & & P_{\alpha''}^\perp &= \{p_2\}
 \end{aligned}$$

Let $\mathbf{N}[\beta]$ be a proof-net containing β as a sub-module. We note $\mathbf{N}[\]$ the complementary module to β in $\mathbf{N}[\beta]$ (a.k.a. the context). Because $\mathbf{N}[\] :: \beta$ is a proof-net, we know that $P_{\mathbf{N}[\]} \perp P_\beta$ (by proposition 15), and thus that $P_{\mathbf{N}[\]} \subseteq P_\beta^\perp$ (by remark 1). From our computation just above, we thus have $P_{\mathbf{N}[\]} \subseteq \{p_1, p_2\}$.

However, by invoking now our computations for α (resp. α'') and the results just cited again, one sees that $\mathbf{N}[\alpha]$ (resp. $\mathbf{N}[\alpha'']$) is a proof-net only if $P_{\mathbf{N}[\]} \subseteq \{p_1\}$ (resp. $P_{\mathbf{N}[\]} \subseteq \{p_2\}$). In other words, applying $\overset{\text{WD}^-}{\rightsquigarrow}$ rewriting to $\mathbf{N}[\beta]$ is correct if and only if either $P_{\mathbf{N}[\]} = \{p_1\}$ or $P_{\mathbf{N}[\]} = \{p_2\}$.

But this is precisely what happens when the \circlearrowleft -link and the \otimes -link of β form together a doubly splitting $\overset{\otimes}{\circlearrowleft}$ in $\mathbf{N}[\beta]$. \square

3.3 Existence of a correct \wp^- strategy

We will now show that it is always possible to find such a doubly splitting \otimes situation. Such a statement happens to be false while one keeps the usual syntax unquotiented, but true when one considers only mono-conclusion nets (in terms of expressive power, nothing is lost: $\vdash_{\text{MLL}} \Gamma$ iff $\vdash_{\text{MLL}} \wp \Gamma$) and up to associativity and commutativity of \otimes and \wp .

All of the three conditions: (1) *mono-conclusion* proof-nets, (2) *associativity*, and (at least some form of) (3) *commutativity* appear compulsory. Indeed, If we drop condition (n) (where $n \in \{1, 2, 3\}$), keeping the two others, the unique cut-free proof-net having for conclusions the formula(s) indicated in the corresponding item [n] below is a counter-example (it does not include any sub-graph of shape β for which the corresponding \wp^- replacement is correct):

$$\begin{aligned} & (X \wp Y) \otimes Z, Z^\perp \wp (Y^\perp \otimes X^\perp) \quad (\text{mono-conclusion}) [1] \\ & ((X \wp Y) \otimes Z) \wp (Z^\perp \wp (Y^\perp \otimes X^\perp)) \quad (\text{associativity}) [2] \\ & X \wp ((Y \wp Y^\perp) \otimes X^\perp) \quad (\text{commutativity}) [3] \end{aligned}$$

We now state the series of lemmas, combined hereafter to prove the existence of a correct \wp^- strategy.

Lemma 25 Let N be a mono-conclusion cut-free proof-net. If the terminal link of N is a \wp -link whose premises (up to associativity) all are conclusions of identity-axioms, then N does not contain any \otimes -link.

Proof Else, let us consider a \otimes -link in N and p a maximal downward path in N beginning with one of the edges of that \otimes -link. N being mono-conclusion, the last edge of p must be one of the edges of the terminal \wp -link of N . Hence in p (sequences of) \otimes -link edges alternates at least once with (sequences of) \wp -link edges. Up to associativity, the terminal \wp -link of N thus should be surmounted by a \otimes -link. \square

To save time (by avoiding drawings), the proofs we give now (of lemmas 26 and 28) release upon sequentialization. This is of course just a matter of convenience.

Notation: if π is a sequent calculus derivation, we note $|\pi|$ the number of non 0-ary rules in π .

Lemma 26 Let N be a cut-free proof-net. If N do not contain any \otimes -link, then N is either an identity-axiom or a \wp -ized identity-axiom.

Proof By sequentialization theorem, it suffices to prove the lemma for cut-free sequent calculus. Let π an MLL derivation of $\vdash \Gamma$ and let r be the last rule of π . If $|\pi| = 0$, then r is an identity-axiom and we are done. Else, as there is no \otimes -rule, r has to be a \wp rule. Let π^- the immediate sub-proof of π . By induction hypothesis (and the case as of a \wp -ized identity-axiom being impossible, π^- 's terminal sequent having more than one formula, namely the two sub-formulas

of the formula having the introduced \wp as main connective) it has to be an identity-axiom, as expected. \boxtimes

Lemma 27 Let \mathbf{N} be a mono-conclusion cut-free proof-net. If the terminal link of \mathbf{N} is a \wp -link whose premises (up to associativity) all are conclusions of axioms, then \mathbf{N} is a \wp -ized identity-axiom.

Proof By lemmas 25 and 26. \boxtimes

Lemma 28 In any mono-conclusion proof-net, there exists a \wp -link.

Proof (induction on sequentialization) Let $\mathbf{N} : A$ a proof-net. Let π a sequentialization of \mathbf{N} and r its last rule:

$$\pi \left\{ \begin{array}{c} \vdots \\ \hline \vdash A \end{array} \right. r$$

1. The case $r = \text{axm}$ is impossible.
2. If $r = \wp$ -rule, we have it.

3. If $r = \otimes$ -rule, then $A := A' \otimes A''$ and $\pi := \left\{ \begin{array}{cc} \vdots \pi_1 & \vdots \pi_2 \\ \hline \vdash A' & \vdash A'' \\ \hline \vdash A' \otimes A'' \end{array} \right. r$. By induction hypothesis on π_i , one concludes. \boxtimes

Lemma 29 Let \mathbf{N} a mono-conclusion proof-net. If no splitting \wp has a \otimes among its premises (up to associativity and commutativity), then \mathbf{N} is in canonical form (tensorization of \wp -ized axioms).

Proof Let us consider the terminal vertex of \mathbf{N} .

1. If it is a \wp -link, being terminal, it is a splitting one. Thus by our main hypothesis each of its premises (up to associativity) is conclusion of an axiom. So, by lemma 27, \mathbf{N} is a \wp -ized axiom.
2. If it is a \otimes node, the mono-conclusion proof-net \mathbf{N} has no terminal \wp . So, lemma 22 applies, and deleting that \otimes , one gets proof-nets which themselves are mono-conclusion (else, \mathbf{N} itself would not be mono-conclusion). While one of the mono-conclusion proof-nets so produced ends with a \otimes -link, we are allowed to use lemma 22 again. Obstinate performing the corresponding \otimes deletion process until it ends, we eventually get a set of mono-conclusion proof-nets \mathbf{M}_i , each of them ending with a \wp -link (indeed, any such \mathbf{M}_i being mono-conclusion, the case where its conclusion vertex is conclusion of an identity axiom link is excluded). Those \wp -links are themselves splitting (in \mathbf{N}) and, by our main hypothesis, they are thus surmounted by identity-axioms only. Hence by lemma 27, they are the \wp -links of \wp -ised axioms. So that \mathbf{N} actually is a k -ary tensor ($k \geq 1$) of \wp -ized axioms. \boxtimes

Notation

- N^- is the proof-net one gets from N by deleting iteratively and obstinately terminal \wp -links (any conclusion vertex in N^- thus is conclusion of either a \otimes -link or an identity-axiom link).
- let v be the conclusion vertex of a given ‘terminal’ (up to associativity) \wp -link in N . Then $N \uparrow^{v^-}$ the proof-net obtained from N by deleting iteratively terminal (up to associativity) \wp -links only up to v (i.e. we delete only those terminal \wp -links that one can reach without deleting v).

Lemma 30 Let N be a mono-conclusion cut-free proof-net whose terminal link is a \wp -link. If N includes a \otimes link, then N contains a \otimes_{\wp} for which \wp_{\wp}^- rewriting is correct.

Proof Every terminal vertex of N^- is conclusion of an axiom or a \otimes -link and, by the \otimes existence hypothesis and lemma 27 applied to N , at least one of those vertices is conclusion of a \otimes -link. Hence, by lemma 22 applied to N^- , at least one of them is a splitting \otimes . Let v_1 be the conclusion vertex of that \otimes . Because N is mono-conclusion and terminates with a \wp -link, in N , v_1 is premiss of a \wp -link. Let v_2 the conclusion vertex of that \wp -link. Up to \wp associativity and commutativity, the other premiss v_3 of that \wp -link can be chosen terminal in N^- . So that in $N \uparrow^{v_2^-}$, the superposition v_1 (or more precisely the superposition of the corresponding links) forms a double splitting \otimes_{\wp} . By proposition 24, performing in $N \uparrow^{v_2^-}$ the corresponding \wp_{\wp}^- reduction step preserves correctness. If we finally reintroduce as terminal \wp -links as previously dropped from N (an operation which always preserves correctness), one finally gets a correct \wp_{\wp}^- reduct of N . \square

Remark 2 Let N be a cut-free proof-net. If N is mono-conclusion, then any proof-net N_+ (see def. 21) produced by a splitting of N due to a splitting \wp in N , has no other conclusions than the two premises of that splitting \wp .

Theorem 31 Let N a mono-conclusion cut-free proof-net. Then either N is in canonical form (n -ary \otimes of \wp -ized axioms) or there exists in N a \otimes_{\wp} whose \wp_{\wp}^- rewriting is correct.

Proof N is mono-conclusion. So by lemma 28, it contains a \wp -link. Hence, by lemma 23, there exists a splitting \wp . Thus:

1. If none of those splitting \wp is surmounted (up to \wp -associativity) by a \otimes -link: then, by lemma 29, N is a n -ary tensor of “ \wp -ized axioms”.
2. Else, there exists a splitting \wp surmounted (up to \wp -associativity) by a \otimes -link: let us complete N^+ (the “upper” proof-net produced by the corresponding splitting) by applying a \wp -link to both its conclusions (see remark 2). Applying lemma 30 to that proof-net, one can find above its terminal \wp -link, a \otimes s.t. the corresponding \otimes_{\wp} is doubly splitting (in that proof net).

The $\mathfrak{W}_1^{\mathfrak{D}^-}$ step thus preserves correctness (and remains so when performed in the original proof-net: indeed, proof-nets are closed by the replacement of a mono-conclusion sub-proof-net by a mono-conclusion proof-net). \square

Remains now to prove that the $\mathfrak{W}_1^{\mathfrak{D}^-}$ rewriting is noetherian. For this, we will use the definition below, inspired by the one given in [4] for $\mathfrak{W}_1^{\mathfrak{D}}$. Let us first present two notations. Omitting in a given cut-free structure the identity-axioms, one gets a multi-set of trees which, following ordinary conventions for spatial representation of partial orders, defines a partial strict order $<$ over the vertices of the structure (so $<$ means “strictly below”). Also, in what follows, \wedge stands for the ‘infimum’ relative to $<$.

Definition 32 The complexity $C(v)$ of a vertex v in N is defined by

$$C(v) = \#\{p, \wp\text{-link in } N, \text{ s.t. } p \wedge v < v\}$$

Remark If N is a mono-conclusion proof-net, then for any v in N , $C(v)$ is the number of \wp -links in N which are not above v .

Definition 33 The complexity $C(N)$ of a proof-net N is defined as:

$$\sum_{t \otimes\text{-link in } N} C(t)$$

Proposition 34 The $\mathfrak{W}_1^{\mathfrak{D}}$ rewriting rule makes the complexity decrease. Associativity and commutativity, however, left it unchanged.

Theorem 35 (Completeness) Any mono-conclusion proof-net can be obtained from a proof-net in canonical form (a big \otimes of \wp -ized axioms links) by $\mathfrak{W}_1^{\mathfrak{D}}$ rewriting.

Proof By theorem 31 and proposition 34.

4 Epilogue

This result was first proved to build a bridge between MLL proof-nets and ‘deep inference’ formalisms (Calculus of Structure, CoS) elaborated by the proof theory group in Dresden (A. Guglielmi [7] and others).

Roughly speaking, a given derivation in the multiplicative fragment of the calculus of structures, corresponds to a given $\mathfrak{W}_1^{\mathfrak{D}}$ rewriting strategy. Roughly only, because, in Calculus of structures, ‘generators’ are only \wp -ized axioms (not n -ary \otimes of them), a specific construction being separately added to introduce new \wp -ized axioms *at any time of the rewriting*.

Even if this simple remarks could seem self evident for the Calculus of structures community, I nevertheless hope it could be a useful mean for improving the dialogue between deep-inference and proof-nets workers.

References

- [1] GIRARD J.-Y. (1987), Linear logic. *Theoretical Computer Science*, 50:1–102.
- [2] DANOS V. AND REGNIER L. (1995), The structure of multiplicatives. *Archives for Mathematical Logic*, 28, 181-203, 1989.
- [3] DANOS V. (1990), La logique linéaire appliquée à l'étude de divers processus de normalisation (principalement du λ -calcul). *Thèse de doctorat*, Université Paris 7, juin 1990.
- [4] MAIELI R. AND PUITE Q. (2005), Modularity of proof nets: generating the type of a module. *Archive for Mathematical Logic*, Volume 44, Number 2, 167-193, February 2005.
- [5] DANOS V., JOINET J-B., SCHELLINX H. (2003), Computational isomorphisms in classical logic. *Theoretical Computer Science*, Vol. 294, issue 3, pp.353-378", 2003
- [6] DEVARAJAN H., HUGHES D., PLOTKIN G., PRATT V. (1999), Full completeness of the multiplicative linear logic of Chu spaces. *Logic in Computer Science*, 234-242, 1999
- [7] GUGLIELMI A. (2004), A System of Interaction and Structure. *Technical Report WV-02-10* (8 November 2004), International Center for Computational Logic Technische Universität Dresden, 01062 Dresden, Germany to appear on *ACM Transactions on Computational Logic*
- [8] BECHET D., P. DE GROOTE, RETORÉ C. (1997), A complete axiomatisation for the inclusion of series-parallel orders. *RTA 97 LNCS* volume 1232, 1997
- [9] RETORÉ C. (1999), Handsome proof-nets: R&B graphs, perfect matchings and Series-Parallel graphs. *INRIA Research report RR-36-52*, 1999

Rewritings in Polarized (Partial) Proof Structures*

Christophe Fouqueré and Virgile Mogbil

LIPN – UMR7030

CNRS – Université Paris 13

99 av. J-B Clément, F-93430 Villetaneuse, France

{christophe.fouquere, virgile.mogbil}@lipn.univ-paris13.fr

Abstract. This paper is a first step towards a study for a concurrent construction of proof-nets in the framework of linear logic after Andreoli's works, by taking care of the properties of the structures. We limit here to multiplicative linear logic. We first give a criterion for closed modules (i.e. validity of polarized proof structures), then extend it to open modules (i.e. validity of partial proof structures) distinguishing criteria for acyclicity and connectability. The keypoint is an extensive use of the fundamental structural properties of the logics. We consider proof structures as built from n -ary bipolar objects and we show that strongly confluent (local) reductions on such objects are an elegant answer to the correctness problem. This has natural applications in (concurrent) logic programming.

1 Introduction

Girard in his seminal paper [8] gave a parallel syntax for multiplicative linear logic as oriented graphs called *proof-nets*. A *correctness criterion* enables one to distinguish sequentializable proof-structures (the so called proof-nets) from "bad" structures. After Girard's long trip correctness criterion, numerous equivalent properties were found. In particular, Danos and Regnier [7] proved that *switched* proof-structures should be trees. Furthermore, Danos implemented the criterion by means of a contraction relation on proof structures: binary connectives are the main elementary objects of the structures and are reduced by the relation. While a lot of research has been done on such correctness criteria, it still remains to study sequentialization of *polarized* as well as *partial* proof-structures. We generalize in this paper Danos and Regnier results to these two cases and show that the framework of proof-net rewritings leads to elegant results. In our case, structures are built from n -ary bipolar objects and we show that a strongly confluent (local) reduction may be defined as these elementary objects really take care of fundamental properties.

Such structures arise naturally after Andreoli's works [2–4] in logic programming: after showing in [1] that linear logic, a resource-conscious logic, may be used as a programming language¹ using a standard, sequential approach, he switches to a proof-net presentation as this syntax affords a desequentialized presentation of proofs, hence a

* Partially supported by ACI NIM project Géométrie du Calcul (GEOCAL), France.

¹ Full first-order linear logic can be used as a programming language. However, we restrict in this paper to propositional multiplicative linear logic.

concurrent way to compute them at the expense of a correctness criterion that guarantees to recover sequentialization, i.e. validity of proofs.

In this paper, we search for a generalization of Andreoli's results in order to have full expressivity. For that purpose, we depart from his approach by adopting a graph point of view. *Modules*, as graph elements, arise naturally from proof nets. In a few words, associativity, commutativity and focalization lead to polarize formulae, hence to stratify proofnets.² It turns out that polarization may enhance proof search, hence is central to prove that full linear logic could be a logic programming language. This fundamental notion was later considered in Girard's works [9], and also in Laurent's works about Polarized Linear Logic (LLP). Consequently our basic objects are proof structures with two strata we call *bipolar structures*: bipolarity is a key tool to get a rewriting system for checking correctness (2). Bipolar structures become computational structures as composition of such structures corresponds to some kind of progression rule in logic programming. As we shall show in the next sections, applying such a rule is nothing more than a composition of partial proof structures whose correctness is stated locally.

Andreoli set up this desequentialized framework for middleware infrastructures. In such applications, software agents must satisfy requests or goals by executing concurrently actions on a shared environment: actions transform the environment by deleting resources and creating new sets of results. Andreoli focused on *transitory* proof-structures, i.e. actions always create new resources. Moreover, he imposes prerequisites of actions to be satisfied in order to execute them: the proof construction is done bottom-up. As we shall see, these two hypotheses greatly simplify the problem of defining formally conditions under which actions may be undertaken. On the contrary, we constrain neither the structure of modules, nor the application order. It is then possible to define actions that kill resources or to anticipate consequences of resources still to be acquired. Furthermore, we depart from Andreoli's approach for defining a correctness criterion. His method is based on a computation of domination forests in the spirit of Murawski and Ong's approach [14]. We adopt here a completely different strategy. We define reduction relations in order to get the correctness property.

The following section gives basic definitions. We formally present modules from elementary ones, graphically and in terms of formulae. We specify in which sense a module is correct, i.e. computation is allowed. Section 3 is devoted to closed modules. A closed module is equivalent to a proof structure. Although closed modules are an extreme special case of modules, the methodology we use introduces naturally the way we consider open modules. In a first attempt, in the spirit of the resolution rule in logic programming we define a rewriting rule on modules: a transformation of a module may be viewed as a deconstruction of the proof structure. Correct normal forms are easily characterized. Extending the Danos-Regnier criterion,³ we deduce a correctness criterion for closed modules as our rewriting rule and its inverse are stable wrt connectedness and acyclicity. We define next a modified version of the previous rewriting system: using polarization and focalization, the reduction becomes fully local: each step reduces one elementary object of our system without any global condition. Open

² Distributivity contributes to it when dealing with the additive part of linear logic.

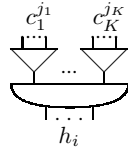
³ The Danos-Regnier criterion is based on graph properties of proof nets: correct proof structures, i.e. proof nets, are in some sense the connected and acyclic ones.

modules, i.e. modules without constraints, are studied in section 4. We prove that the Danos-Regnier criterion may be extended to open modules replacing the connectedness by a *connectability* property. We give two rewriting systems as acyclicity and connectability differ fundamentally. These two systems may be viewed as variations over the one we give for closed modules. We end with a study on incrementality wrt composition of modules. In terms of computation, elementary modules compete to modify some current open module (the environment): actions are concurrent when two such elementary modules are composed in disjoint parts of the environment. It is then crucial to be able to define rewriting systems that commute with composition. We show that we have to restrict previous rewriting systems for that purpose. However, the rewriting systems have to be split into two parts: one commutes with composition, the other is a post-treatment necessary to test correctness of composition.⁴

2 Basic definitions

Elementary bipolar modules are our basic blocks. They are interpreted as elementary actions that can take place during an execution. In terms of graph, applying an action is represented as a wire, i.e. composition, of the corresponding (elementary) module onto the current graph.

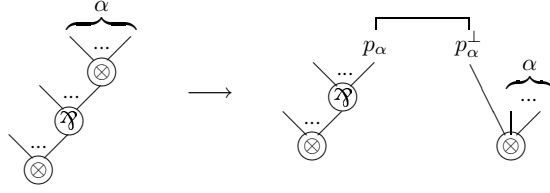
Definition 1. An elementary bipolar module (EBM) M is given by a finite set $\mathcal{H}(M)$ of propositional variables (called hypotheses) h_i and a non empty finite set $\mathcal{C}(M)$ varying over k of finite sets of propositional variables (called conclusions) c_k^j . Variables are supposed pairwise distinct.⁵ The set of propositional variables appearing in M is noted $v(M)$. Equivalently, one can define it as an oriented graph with labelled pending links and one positive pole under a finite set of negative poles. Its type $t(M)$ and drawing are given in the following way:

$$t(M) = (\otimes_i h_i) \multimap (\wp_k (\otimes_{j_k} c_k^{j_k}))$$


Informally, the EBM has the following operational bottom-up reading: being given in some context a multiset of hypotheses (i.e. their tensor), this one is replaced by (\multimap) each of the multisets of conclusion, these last have to be used in separate contexts (\wp is the logical dual of \otimes). This specification of modules comes from the fact that connectives are naturally split into two sets: e.g. \otimes is said positive, while \wp is negative. Propositional variables are declared positive, and their negation negative. Formulae alternate positive and negative levels up to propositional variables. Note that we use conveniently a two-sided style for formula and sequent presentations, even if our basic objects are proofnets. It is in fact possible to flatten proofnets to get bipolar structures related by links on fresh variables:

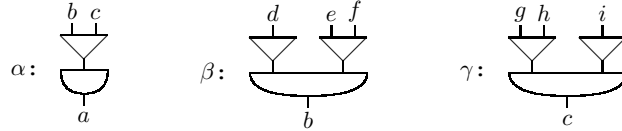
⁴ Complements and some technical proofs are available in <http://xxx.lanl.gov/cs/0411029>.

⁵ This restriction is taken for simplicity. The framework can be generalized if we consider multisets (of hypotheses and conclusions) instead of sets, and add as required a renaming mechanism: the results in this paper are still true.



If we notice that a variable and its negation cannot be together linked to negative nodes (it would contradict the correctness criterion), we can always suppose that, say, positive variables are linked to negative nodes. Finally, it may be the case that some bipolar structure (thus beginning with a positive node at bottom) has no negative variable: add then the constant $\mathbf{1}$, neutral for \otimes . Allowing abusively unary \otimes and \mathfrak{N} connectives, these (elementary) bipolar structures are the clauses of our programming language. We thus conveniently suppose that $\mathfrak{N}_k F_k = \otimes_k F_k = F_1$ when the domain of k is of cardinal 1. Moreover, if the domain of i is empty, $(\otimes_i h_i) \multimap C = \mathbf{1} \multimap C$ and if the domain of j_k for some k is empty $(\otimes_{j_k} c_k^{j_k}) = \perp$.

Example 1. The EBMs α , β and γ of respective types $t(\alpha) = a \multimap (b \otimes c)$, $t(\beta) = b \multimap (d \mathfrak{N} (e \otimes f))$ and $t(\gamma) = c \multimap ((g \otimes h) \mathfrak{N} i)$ are drawn in the following way:



Three kinds of EBMs are of special interest: An EBM is *initial* (resp. *final*) iff its set of hypotheses is empty (resp. its set of conclusions is empty). An EBM is *transitory* iff it is neither initial nor final. Initial EBMs allow to declare available resources, though final EBMs stop part of a computation by withdrawing a whole set of resources. Transitory EBMs can be seen as definite clauses in standard logic programming. Roughly speaking, a (bipolar) module (BM) is a set of EBMs such that a label appears at most once as a conclusion and at most once as a hypothesis. A label appears as a conclusion and as a hypothesis when two EBMs are linked by this label. As we search for correctness criteria wrt composition of modules (i.e. execution of the program), we give below an inductive definition of bipolar modules.

Definition 2 (BM). A bipolar module (BM) is defined inductively in the following way:

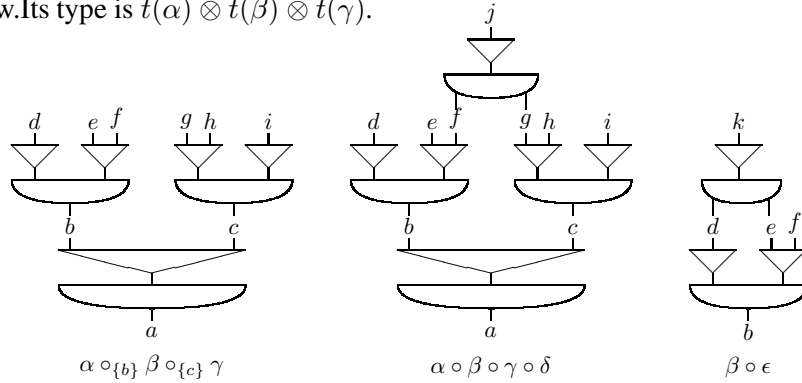
- An EBM is a BM.
- Let M and N be a BM, let $I = (\mathcal{C}(M) \cap \mathcal{H}(N)) \cup (\mathcal{H}(M) \cap \mathcal{C}(N))$, their composition wrt the interface I , $M \circ_I N$ is a BM with :
 - the set of hypothesis (resp. conclusions) $\mathcal{H}(M \circ_I N)$ is the hypothesis (resp. conclusions) of M and N which are not in I
 - $t(M \circ_I N) = t(M) \otimes t(N)$ and $v(M \circ_I N) = v(M) \cup v(N)$.

The border $b(M)$ of a BM M is the union of the hypotheses and the conclusions.

The informal explanation given before is more general than this definition because we define BM incrementally. However, we abusively do not consider these differences in the following as properties will be proven in the general case. The interface will

be omitted when it is clear from the context. Note that the interface may be empty: it only means that two computations are concurrently undertaken, currently without any shared resources. A BM may not correspond to a valid computation: e.g. we do not want to accept that some action uses two resources in disjunctive situation! Correctness has obviously to be defined wrt the underlying Linear Logic as we do below. Finally, note that when a BM is correct, it represents the history of the computation whereas its conclusion is the current available environment.

Example 2. The composition of the EBM's α , β and γ is the BM $\alpha \circ_{\{b\}} \beta \circ_{\{c\}} \gamma$ drawn below. Its type is $t(\alpha) \otimes t(\beta) \otimes t(\gamma)$.



Definition 3 (Correctness (wrt sequentialization)). Let M be a BM, M is correct iff there exists a formula C built with the connectives \otimes and \wp , and the variables $\mathcal{C}(M)$ such that the sequent $\mathcal{H}(M), t(M) \vdash C$ is provable in Linear Logic.

Example 3. Let us give two more BM's δ and ϵ of respective types $(f \otimes g) \multimap j$ and $(d \otimes e) \multimap k$.

- The following sequent is provable in LL: $a, t(\alpha \circ \beta \circ \gamma \circ \delta) \vdash d \wp (e \otimes j \otimes h) \wp i$.
The (correct) BM $\alpha \circ \beta \circ \gamma \circ \delta$ is drawn in the previous figure.
- The BM $\beta \circ \epsilon$ is not correct: there is a cycle through d and e .

As we shall focus first on characterizing correctness on closed modules, and then generalize our results to open modules, we adjoin to the term *correct* the kind of modules we speak of, e.g. c-correct when the module is closed, o-correct when it is open.

3 Closed modules

A closed module is a BM where the sets of hypotheses and conclusions are empty. Correctness of closed modules may be tested either in sequent calculus or by means of (simple oriented) graphs called *proof-nets*. We use this latest representation in this section. A *correctness criterion* enables one to distinguish sequentializable proof-structures (say such oriented graphs) from "bad" structures. The reader may find in [7] the definitions of proof structures and switchings. One generalizes this definition to n -ary connectives in the obvious way (taking care of associativity and commutativity of \otimes and \wp) in place of standard binary ones. One modifies in the same way the definitions of switching introducing generalized switches. In particular a n -ary \wp connective has n switched

positions. One still can define switched proof-structures and a criterion generalizing Danos-Regnier correctness criterion: A closed module M is DR-correct iff for all generalized switches s on M^o , $s(M^o)$ is acyclic and connected, where M^o is the proof structure associated to $t(M)^\perp$.⁶ We immediately have the following proposition as a corollary of the DR-criterion theorem (remember that a c-correct module is a correct closed module):

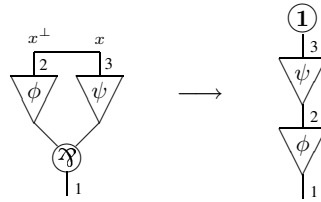
Proposition 1 (c-correction). *Let M be a closed module,
 M is c-correct iff $t(M) \vdash$ is provable in Linear Logic, iff M is DR-correct.*

Remember that the equivalent (binary) Danos correctness criterion may be implemented by means of a contraction relation on proof structures. However, intermediate reduced structures may not be describable in terms of (bipolar) modules. Moreover such a contraction relation does not take advantage of the incremental definition of modules as a composition of elementary bipolar modules. A first idea consists of representing the resolution step (implicit in EBM's composition) in terms of modules. We first give below such a (small step) reduction rule that is stable wrt correctness with ∇_{\square} as the correct normal form, where ∇_{\square} denotes the terminal EBM (i.e. smallest final and initial). We give then a second proposal that takes care of the focalization property. Though a resolution step reduces one variable, this second formulation uses as a whole the structure of a module thanks to focalization. The focalization property states that a sequent is provable iff there exists a proof s.t. decomposition of the positive stratum of formulae is done in one step. Considering bipolar modules, it means that one may define a reduction relation s.t. each step reduces one positive-negative pair of nodes.

Let \sim_{Θ}^* be the transitive closure of the following relation defined on literals of a proof-structure Θ : let u and v be two literals of Θ , $u \sim_{\Theta} v$ iff u^\perp and v are in the same subtree with root \otimes of the formula corresponding to Θ . We note $u \sim^* v$ when there is no ambiguity. In the following, we consider proof-structures modulo neutrality of the constant 1 and associativity of connective \wp .

Definition 4 (Small step reduction rule).

Let \rightarrow be the reduction relation given by:
 if $\forall v$ a literal of ψ , $v \not\sim^* x^\perp$ then



Theorem 1 ((small steps) Correctness criterion). *Let M be a closed BM, M is correct iff $M^o \rightarrow^* 1$.*

Briefly speaking, one can prove that the relation \rightarrow and the inverse relation are stable wrt DR-correctness by induction over the height of ψ . One may want to get rid of the (global) condition in favor of a local condition. This is possible thanks to the structure of modules. Suppose M is a correct closed module, then one may define an equivalent proof-net by sufficiently adding fresh variables as described in the introduction. It is easy to prove that the constraint is satisfied by x or x^\perp for each variable x .

⁶ The type is sufficient to build a proof structure as by construction of modules axioms are uniquely defined. We abusively note $s(M)$ in place of $s(M^o)$ in the following.

However, the reduction system being not strongly confluent, a reduction on a variable may lead to a proof structure on which the condition is not always satisfied. There are two cases where this does not happen: either all variables on a tensor have their negation on the same \mathfrak{N} , or the converse interchanging \mathfrak{N} and \otimes . The (big step) reduction relation \rightarrow in Fig. 1 uses this fact. Note that this system is confluent and terminates.

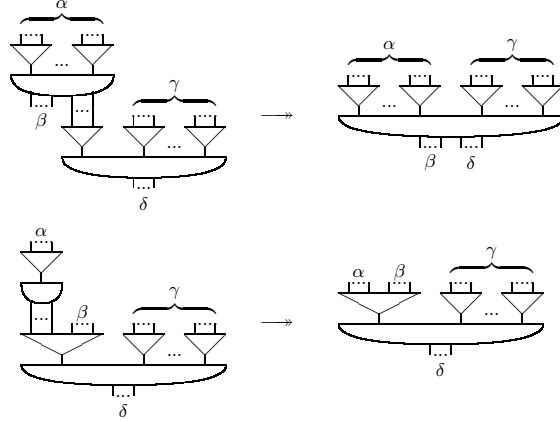


Fig. 1. Big step reduction relation.

Proposition 2 (Stability). *Let M and N be two closed modules and $M \rightarrow N$, M is c -correct iff N is c -correct.*

Proof. One can define a function from left switched module onto right switched module stable wrt acyclicity, connectedness, and the inverse properties. \square

Theorem 2. *A closed module M is c -correct iff $M \rightarrow^* \bigvee \bigwedge$.*

Proof. As the reduction rules are stable wrt correctness, it remains to prove that a correct non-terminal closed module M can always be reduced. We define a partial relation on negative poles: a negative pole is smaller than another one if there exists a positive pole s.t. the first negative pole is linked to the bottom of the positive pole and the second negative pole is linked to the top of the positive pole. We consider the transitive closure of this relation.

If maximal negative poles do not exist then there exists at least one cycle in the module alternating positive and negative poles. We can then define a switching function on the module (choosing the correct links for negative poles) s.t. the switched module has a cycle. Hence contradiction.

So let us consider one of the maximal negative pole, and the corresponding positive pole. We remark that such a negative pole has no outgoing links (the module is closed and the negative pole is maximal). If the positive pole has other negative poles, we can omit the maximal negative pole by neutrality. Otherwise, let us study the incoming negative poles.

If there is no such incoming link, then M is the terminal module. If each incoming negative pole has at least one link going to another positive pole, then one can define a switching function using for each of these negative poles one of the link that does not go to the positive pole we considered first. Hence the switched module is not connected

(there are no outgoing links). Hence contradiction. So there exists at least one incoming negative pole with the whole set of links associated to the positive pole: the first rule applies and we are finished. \square

Note that this proof extensively uses the bipolar nature of modules. Moreover, the proof may have been given considering minimal poles in place of maximal poles, and for each proof only one of the two reduction rules is sufficient and necessary! Finally, the same technique as Guerrini [10] used for Danos criterion may be applied here to get a linear algorithm. The technique we present here is quite close to the one used by Bechet [5]. However his definitions of modules were more restrictive, and the application concerne mainly non commutative logic.

4 Open modules

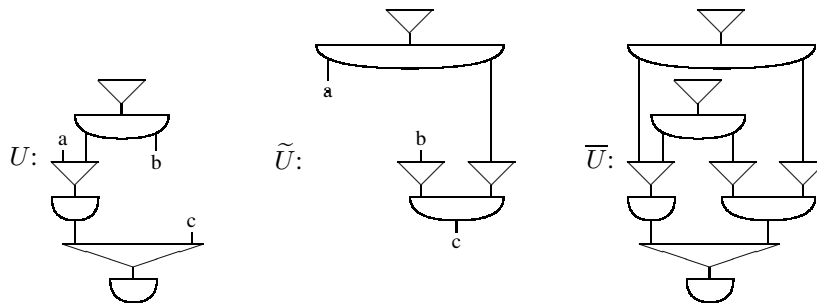
We focus in this section on open modules, i.e. partial polarized proof-structures. An *open module* is a possibly non closed BM. Studying correctness of open modules is a necessary step towards the specification of a logic programming language based on bipolar modules. We search for correctness criteria valid in the general case, hence extending Andreoli's works based on Murawski and Ong criterion. The criterion we give for closed modules is a good basis as it is well suited for bipolar modules and takes care simultaneously of acyclicity and connectedness.

4.1 O-correction

The bigstep reduction relation presented in the previous section is not sufficient to characterize again correctness of open module. Let U be the first module of the next example. Bigstep reductions on U leave the negative pole with variable a unchanged, hence the normal form is not the one required by the c -correctness theorem, though U has to be considered correct. The c -correctness theorem 2 cannot be straightfully extended to open modules.

Correctness of open modules is defined wrt correctness of closed extensions. We define a closed module N to be a *closure* of an open module M iff M is a submodule of N . Such closures are abusively noted \overline{M} without referring to N when there is no ambiguity. As a BM is a graph with pending edges, one defines submodules and induced modules as expected. We use the notation \widetilde{M} for the module \overline{M} without M but with border $b(M)$ (cf def.2).

Example 4. In the next figure, \overline{U} is a closure of U .



Note that the composition of U with a set of only initial/final EBMs is a closure too.

An open module M is *o-correct* iff there exists a c-correct closure of M . The open module U of the example is o-correct because the given closure is c-correct. Note that there is no other c-correct closure. Hence it is not possible in general to split the problem of finding a closure into finding a completion by initial modules and final modules. In the previous section, we defined a rewriting system able to test the correctness of a closed module. As this system is stable wrt connectedness and acyclicity, it is invariant wrt the Danos-Regnier criterion. In order to take care of open modules, we extend connectedness to connectability (acyclicity is treated easily) and prove that connectability and acyclicity are necessary and sufficient for o-correctness. However, we are not able to define a single rewriting system that commutes with composition. An open module M is *acyclic* if for all generalized switches s on M , $s(M)$ is acyclic. Note that a submodule of an acyclic module is obviously acyclic.

An open module M is *connectable* iff there exists a connected closure \overline{M} s.t. \widetilde{M} is acyclic. As a connected closed module is already connectable (just take itself as closure), the connectability is an extension of the connectedness property. We give an equivalent definition: an open module M is connectable iff the closed module $M \circ F$ is connected where F is a *full connector* EBM for M , i.e. F has as hypotheses the set of conclusions of M , is final if M has no hypothesis or has a negative pole with one conclusion for each of its hypotheses. In fact if there exists a connected closure \overline{M} then $M \circ \widetilde{M}$ is connected. So *a fortiori*, $M \circ F$ is connected. The converse comes from the definition.

Theorem 3 (o-correctness). *An open module M is o-correct iff M is acyclic and connectable.*

Proof. By definition o-correctness implies acyclicity and connectability. If M is acyclic and there exists a connected closure \overline{M} st \widetilde{M} is acyclic then by induction on the number of cycles of \overline{M} , one can construct an acyclic and connected closure of M .

If there is a cycle σ in \overline{M} then by hypothesis $\sigma \cap b(M) \neq \emptyset$. Suppose there exists a hypothesis of M $h \in \sigma \cap b(M)$, one defines N to be \widetilde{M} where we substitute a fresh label h' to h . Let N' be the composition of the initial EBM of border $\{h\}$, the final EBM of border $\{h'\}$ and N . $M \circ N'$ has one cycle less than \overline{M} and is a connected closure.

Otherwise the elements of $\sigma \cap b(M)$ are conclusions of M . Let c be such a conclusion. We consider the following cases:

- if c in $\sigma \cap b(M)$ is the only conclusion of a negative pole n , then one can do the same thing as in the previous case.
- else let d be a conclusion in $\sigma \cap b(M)$ distinct from c of n . One renames c (resp. d) in \widetilde{M} in c' (resp. d') to get N . One defines also an EBM D with one conclusion d' and two hypotheses c and d , and an initial EBM E with conclusion c' . Then $X = M \circ D \circ E \circ N$ is a connected closure of M and $D \circ E \circ N$ is acyclic. Hence X is a connected closure of $M \circ D$ and $E \circ N$ is acyclic. We suppressed the cycle σ . However, it may be the case that there were a cycle through d and D doubles it ! For that purpose, we transform M to get rid of this extra cycle. Let M' be M where

we identify the two edges labelled c and d in one labelled d' . Then $M' \circ E \circ N$ is a connected closure of M' and $E \circ N$ is acyclic. Moreover the number of cycles in $M' \circ E \circ N$ is one less than in \overline{M} . Thus there exists N' acyclic such that $M' \circ N'$ is c-correct. Hence $M \circ D \circ N'$ is c-correct. \square

4.2 Acyclicity criterion: a contraction relation \rightarrow

An open module M restricted to the subset I of $b(M)$ is the subgraph of M where we omit pending edges not in I . We denote it $M \downarrow_I$. Informally an open module M restricted to I is a submodule of border I . The restriction of an open module to the empty set is a closed module. Restriction gives naturally an equivalent definition of acyclicity for open modules: an open module M is *acyclic* iff the closed module $M \downarrow_\emptyset$ is acyclic. Hence the proposition given in the previous section applies:

Proposition 3 (acyclicity). *An open module M is acyclic if $M \downarrow_\emptyset \rightarrow^* \bigcup \nabla$.*

Proof. $M \downarrow_\emptyset$ is a closed module and $M \downarrow_\emptyset \rightarrow^* \bigcup \nabla$ then by stability of acyclicity (of the inverse relation) $M \downarrow_\emptyset$ is acyclic. M is then acyclic. \square

Note that the converse is not true, otherwise acyclic closed modules would be correct! A way to characterize acyclicity by means of a reduction relation is to enlarge the reduction \rightarrow (quotienting the set of normal forms). Splitting the negative poles suffices to continue reduction until we get a non-empty set of $\bigcup \nabla$: closing modules may link disjoint connected components. It is then obvious to deduce a necessary and sufficient condition for acyclicity. Andreoli considered in [4] only transitory proof-structures. A *transitory proof-structure* is equivalent to a BM without hypothesis⁷ such that negative poles have always conclusions and obtained by a bottom-up composition of EBMs. As negative poles have pending edges, there is always a way to connect it to other parts of the module: if a transitory module M is acyclic then M is connectable. Hence a transitory module M is o-correct iff M is acyclic. The reduction relation we give to test acyclicity can be considered as an alternative to Andreoli's method.

4.3 Connectability criterion: a contraction relation \rightarrow_c

The proof of the correctness of the big step reduction relation for closed modules gives the keys for finding a connectability property that relies on the structure of an open module (and not on the modules candidate to close it !). Proof of theorem 2 is based on reducing first maximal negative poles. In the case of open modules, maximal elements may have pending edges that should be connected in the closure. But we notice that we keep connectability if we replace the whole set of pending edges for such an element by just one pending edge. With this in mind, we consider the (non directed) contraction relation of Fig. 2 on (contracted) modules. The first three rules are a n -ary formulation of Danos contraction relation. Danos [6] proved correctness of the relation for (closed) proof-structures only, though we extend the results to (open) bipolar modules.

⁷ In fact, there may be hypotheses in built modules but these are unused.

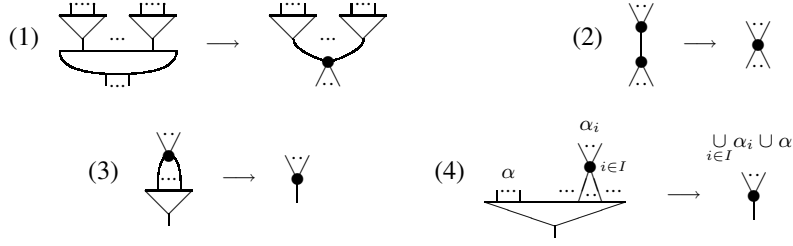
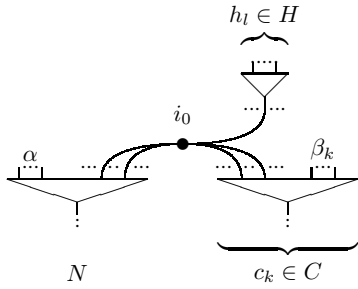


Fig. 2. Contraction relation.

Rule (4) is restricted to cases where the negative pole is such that for all $i \in I$, $\alpha_i \cap b(M) \neq \emptyset$ and $\alpha \subseteq b(M)$ where $b(M)$ is the set of pending edges of M , i.e. the border set. The sets I and α may be empty. We denote by \rightarrow_c one rewriting step and by \rightarrow_c^* the reflexive and transitive closure of \rightarrow_c . We call *contracted node* a black node. Note that rule (4) is simply the rewriting of a negative pole in a contracted node if the condition is satisfied. Thus acyclicity is not preserved but connectability is.

Proposition 4. *The relation \rightarrow_c^* is strongly confluent and terminates.*

Proof. The first rule acts just as a mark. We can forget it: it is just for convenience. Each rule applies locally and strictly decreases the number of negative poles and contracted nodes. The rules are disjoint except for a pair of negative poles linked by the same contracted node i_0 for which rule (4) can be applied (it is a trivial case), and except in the particular case where the left hand side of rule 4 is reduced to the one of rule 3: in this case the results are identical. \square

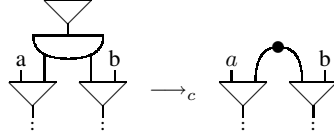


We extend the notions of switching to modules with contracted nodes: contracted nodes are treated as positive poles. Acyclicity, connectedness, closure and connectability are extended in the same way. As in section 3, our strategy consists of characterizing amongst normal forms of this relation the correct ones, and prove stability of, say, connectability.

Let M be an open module and f the corresponding normal form. By definition if f does not contain a negative pole then f is a set of contracted nodes $\{n_j\}_{j \in J}$ s.t. all pending edges are in $b(M)$. We use the notation \mathbf{cc} for a set of contracted nodes $\{n_j\}_{j \in J}$ s.t. for all $j \in J$ n_j has at least one edge in the border $b(M)$ except if $|J| = 1$. If f contains a negative pole N then, f being a normal form of relation \rightarrow_c , rule (4) does not apply on N . Hence the set I as defined by rule (4) is st there exists $i_0 \in I$, $\alpha_{i_0} \cap b(M) = \emptyset$. Moreover this contracted node i_0 is linked to hypotheses of negative poles $\{h_l\}_{l \in L}$ and to conclusions of only negative poles $\{c_k\}_{k \in K}$ st each of them has other conclusions $\beta_k \neq \emptyset$ not linked to i_0 (otherwise rule (2) applies for such nodes): see figure just above.

If we suppose the negative pole N is a maximal one (i.e. $H = \emptyset$), there is a switching (on α or on some $i \neq i_0$ and on one of each β_k) s.t. f (as closures of f) is not connected. Thus f is not connectable.

Example 5. The following subform implies not connectability:



Proposition 5 (stability). *Connectability is stable wrt (resp. inverse) contraction rules.*

Proof. The three first rules satisfy obviously stability as does the reverse relation. Let M be an open module s.t. $M \rightarrow_c M'$ by the contraction rule (4) and there exists \overline{M} connected and \widetilde{M} acyclic. Obviously $M' \circ \widetilde{M}$ is connected. Concerning stability of the inverse relation, let M be an open module s.t. $M \rightarrow_c M'$ by the contraction rule (4) and let F be a full connector EBM for M' . Note that $b(M') = b(M)$. The connectability of M' implies that $M' \circ F$ is connected. Wrt rule (4), because for all $i \in I$, $\alpha_i \cap b(M) \neq \emptyset$ and $\alpha \subseteq b(M)$, for every switches s , $s(M \circ F)$ is connected too. \square

By stability of connectability of the relation and its inverse we have:

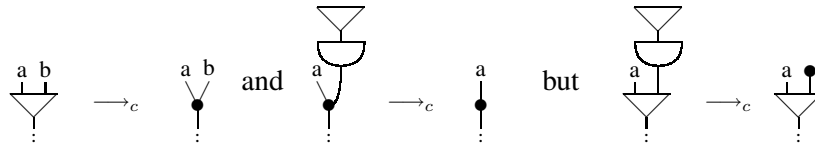
Theorem 4. *Let M be an open module, M is connectable iff $M \rightarrow_c^* \mathbf{cc}$. Hence an open module M is o-correct iff M is acyclic and $M \rightarrow_c^* \mathbf{cc}$.*

5 Composition of modules

In the sequel we discuss an incremental criterion to test the composition of an open module with an EBM. Let M be an o-correct open module and E an EBM s.t. $b(M) \cap b(E) \neq \emptyset$ (otherwise the test is easy). As seen above, acyclicity and connectability, hence o-correctness, of M may be decided by computing normal forms. Our aim is to decide the o-correctness of the composition $M \circ E$ 'incrementally' i.e. not directly but o-correctness of M being given. This leads us to define a specific contraction relation \rightarrow_w to replace \rightarrow_c . From the previous section we have:

$$M \text{ is o-correct iff } M|_{\emptyset} \rightarrow_w^* \bigcup \text{ and } M \rightarrow_c \mathbf{cc}$$

Because of the restriction of M to the empty border, the acyclicity condition given above does not commute with composition. It is the same for connectability: even if there is preservation of the border with \rightarrow_c , a choice is made for the completion of M which may be different from the way composition with E is done. For example:



In the sequel we show that if we release the restriction operation we can incrementally manage acyclicity. The relax of the (implicit) completion in the rewriting rules dealing with connectability gives also an incremental criterion for connectability.

5.1 Incremental acyclicity: \rightarrow

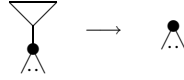
The restriction to empty set is stable wrt the reduction \rightarrow i.e. if M is an open module s.t. $M \rightarrow N$ then $M|_{\emptyset} \rightarrow N|_{\emptyset}$. Hence an incremental test for acyclicity follows:

Proposition 6. *Let M be an open module s.t. $M \rightarrow^* f$ and E an EBM. $M \circ E$ is acyclic if $(f \circ E)|_{\emptyset} \rightarrow^* \bigvee$.*

Proof. If $M \rightarrow^* f$ then $(M \circ E) \rightarrow^* (f \circ E)$. Following previous remark, $(M \circ E)|_{\emptyset} \rightarrow^* (f \circ E)|_{\emptyset}$. Thus if $(f \circ E)|_{\emptyset} \rightarrow^* \bigvee$ then $(M \circ E)|_{\emptyset} \rightarrow^* \bigvee$. \square

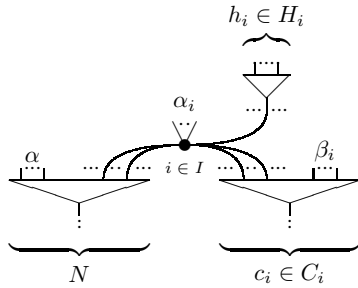
5.2 Contraction relation (without completion): \rightarrow_w

We consider the rewriting system given to test connectability where rule (4) is restricted to the following degenerated case ($\alpha = I = \emptyset$ and application of rule (2)):



We denote by \rightarrow_w one rewriting step and by \rightarrow_w^* the reflexive and transitive closure of \rightarrow_w . As it is a subsystem of the previous one, the relation \rightarrow_w^* terminates and is still strongly confluent (there is only trivial independant pairs).

We study the normal forms. By definition an open module contracts in a normal form composed with only contracted nodes or contracted modules where each negative pole N is of the following form:



- I is a (possibly empty) set of contracted nodes,
- each $i \in I$ is linked to a set C_i of other negative poles by conclusions and to a set H_i of other negative poles by hypothesis (the sets C_i and H_i may be empty). Moreover for all $c_i \in C_i$ $\beta_i \neq \emptyset$,
- α and α_i are (possibly empty) subsets of $b(M)$ for all $i \in I$.

We focus on the two possible forms of negative pole:

- there exists $i_0 \in I$ s.t. $\alpha_{i_0} = H_{i_0} = \emptyset$. We denote such forms by **notcc**.
- for all $i \in I$, $\alpha_i \neq \emptyset$ or $H_i \neq \emptyset$. These negative poles may be considered in the previous system \rightarrow_c^* .

If a normal form has no negative poles then it is a set of contracted nodes. We add to the **notcc** forms the case where there is at least one contracted node without pending edges and other nodes.

In order to compare these normal forms with the normal forms of \rightarrow_c observe that: (i) by definition of normal forms, if $I = \emptyset$ then $\alpha \neq \emptyset$, and if $I \neq \emptyset$ then $|I| \geq 2$ or $\alpha \neq \emptyset$, (ii) for all $i \in I$ for all $c_i \in C_i$ we have $\beta_i \neq \emptyset$. It follows that if a normal form

g wrt \rightarrow_w^* of an open module M contains a **notcc** subform then there is a generalized switch s.t. g is not connected. The stability of connectedness wrt \rightarrow_w^* being given, M is not connected (neither its closures), thus not connectable.

Remark that the **notcc** forms are already in the previous system: they are normal forms which are not the **cc** forms! In fact the **notcc** subforms are invariant wrt the previous system \rightarrow_c^* . Moreover as stability of connectability of the inverse relation is easily proven, we have:

Theorem 5. *Let M be an open module, M is connectable iff $M \rightarrow_w^* g$ s.t. **notcc** $\notin g$.*

Proof. Let M be s.t. $M \rightarrow_w^* g$. If **notcc** $\in g$ then g is not connected (neither its closures) and by stability of connectedness M is not connectable. Conversely, if **notcc** $\notin g$ then $g \rightarrow_c^* \mathbf{cc}$ by invariance of **notcc** wrt \rightarrow_c^* . By theorem 4, g is connectable. The result is obtained by stability of connectability of the inverse relation wrt \rightarrow_w^* . \square

Hence, an open module M is o-correct iff M is acyclic and $M \rightarrow_w^* g$ s.t. **notcc** $\notin g$. By confluence property and theorem 5 we have an incremental test: Let M be a connectable open module s.t. $M \rightarrow_w^* g$ and E an EBM s.t. $b(M) \cap b(E) \neq \emptyset$. We have:

$$M \circ E \text{ is connectable iff } f \circ E \rightarrow_w^* g \text{ s.t. } \mathbf{notcc} \notin g.$$

5.3 A test for composition

Testing the composition of an EBM E on a correct module M may be done in the following way. We associate to such a module M a pair (f, g) such that $M \rightarrow^* f$ and $M \rightarrow_w^* g$. We compute the pair (f', g') associated to $M \circ E$: $f \circ E \rightarrow^* f'$ and $g \circ E \rightarrow_w^* g'$. Then E may be plugged onto M , i.e. the composition is correct, iff $f' \downarrow_{\emptyset} \rightarrow^* \bigcup$ and **notcc** $\notin g'$. This test may be implemented in such a way that pre-computations are done in M in order to optimize the test. Moreover this allows for a concurrent treatment for testing composition by only locking a reduced part of the module M .

6 Conclusion

Studying the correctness of open modules is a necessary condition towards incremental composition of partial proof-nets. Furthermore their concurrent construction allows for a new approach in designing logic programming languages besides standard ones [1, 11, 13]. In the Horn fragment as well as with linear logic, 'classical' logic programming is based on a step by step reduction of goals to be proven by means of a resolution or a progression rule, i.e. the correctness of a computation is reduced to a pattern recognition between some part of the current goal and the head of a chosen clause. More complex than the propositional Horn fragment, pattern recognition is done wrt the whole current environment when considering, e.g. the full linear logic [12]. In all these cases, the operational model is unable to reveal possible concurrent computations. A contrario, the proof net approach is a natural framework as each proof net represents a whole bunch of sequentialized computations: commuting rules lead to the same proof net.

For that purpose, we first extend the classical rewriting criterion of Danos to the n-ary bipolar case for testing the correctness of closed modules. We show in particular that polarization greatly simplifies the rewriting procedure. We finally modify the criterion to take care of open modules proving that correctness of open modules reduces to testing linearly acyclicity and connectability. This includes Danos results in a more general framework. It also extends Andreoli's works by removing constraints on objects we consider.

An interesting remaining question is to take care of exponential modalities in polarized and partial proof-structures. Even if Andreoli proves the focalization property for the whole linear logic, management of exponentials with proof nets requires extra structure such as boxes, i.e. bounded regions, as their behaviour is context dependent. In our opinion, this could yield a local management of the boxes, just considering transformation on part of the region border thanks to polarization and focalisation. Our characterization of proofs as a composition of complex objects can then be extended to multiplicative exponential polarized proof-structures in the same spirit, i.e. by (concurrently) reducing such structures.

References

1. Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.
2. Jean-Marc Andreoli. Focussing and proof construction. *Annals of Pure and Applied Logic*, 107(1–3):131–163, 2001.
3. Jean-Marc Andreoli. Focussing proof-net construction as a middleware paradigm. In A. Voronkov, editor, *CADE*, volume 2392 of *Lecture Notes in Computer Science*, pages 501–516. Springer, 2002.
4. Jean-Marc Andreoli and Laurent Mazaré. Concurrent construction of proof-nets. In M. Baaz and J. A. Makowsky, editors, *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 29–42. Springer, 2003.
5. Denis Béchet. Incremental parsing of lambek calculus using proof-net interfaces. In *ACL/SIGPARSE*, editor, *Eighth International Workshop on Parsing Technologies*, Nancy, France, April. citeseer.ist.psu.edu/668958.html.
6. Vincent Danos. *Une application de la logique linéaire à l'étude des processus de normalisation (principalement de λ -calcul)*. PhD thesis, Université Denis Diderot, Paris 7, 1990.
7. Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28(3):181–203, 1989.
8. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
9. Jean-Yves Girard. On the unity of logic. *Ann. Pure Appl. Logic*, 59(3):201–217, 1993.
10. Stefano Guerrini. Correctness of multiplicative proof nets is linear. In *LICS*, 1999.
11. Dale Miller. Forum: A multiple-conclusion specification logic. *Theor. Comput. Sci.*, 165(1):201–232, 1996.
12. Dale Miller. Overview of linear logic programming. In T. Ehrhard, J.-Y. Girard, P. Ruet, and P. Scot, editors, *Linear Logic in Computer Science*, volume 316 of *London Mathematical Society Lecture Note*. Cambridge University Press, 2004.
13. Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Ann. Pure Appl. Logic*, 51(1-2):125–157, 1991.
14. Andrzej Murawski and Luke Ong. Dominator trees and fast verification of proof nets. In *LICS'00*, pages 181–191. IEEE Computer Society Press, 2000.

Labelled Structures and Provability in Resource Logics - extended abstract -

E. Dumoulin and D. Galmiche

LORIA - Université Henri Poincaré
54506 Vandœuvre-lès-Nancy Cedex, France
dumoulin{galmiche}@loria.fr

Abstract We aim to emphasize the interest of labelled structures for analyzing provability in some resource logics. Labels and constraints allow to capture the semantic consequence relation in some resource logics, like BI logic that combines intuitionistic and linear connectives. They provide new methods in proof theory which are based on specific structures, namely dependency graphs or labelled proof nets. Such semantic structures are central for the analysis of provability and the generation of proofs or countermodels. Knowing that BI is conservative w.r.t. Multiplicative Intuitionistic Linear Logic (MILL), we consider MILL from the BI perspective and show how labelled proof structures can provide a new based-on connection characterization of MILL provability. We also provide an algorithm that builds MILL proof nets and its related connection method based on labelled structures and constraints. The generation of proofs and countermodels is analyzed in this context.

1 Introduction

Since last years there is an increasing amount of interest for logical systems that are resource sensitive. Among so-called resource logics, we can mention Linear Logic (LL)[11] with its resource consumption interpretation, Bunched Implications logic (BI) [17] with its resource sharing interpretation but also order-aware non-commutative logic (NL) [1]. As specification logics, they can represent features as interaction, resource distribution and mobility, non-determinism, sequentiality or coordination of entities. For instance, BI has been recently used as an assertion language for mutable data structures [12]. In this context, it is important to verify pre- or post-conditions expressed in this logic but also to discover non-theorems and, if possible, to provide explanation about non-validity by generating readable and usable countermodels.

In this paper we aim to discuss about models and labelled structures that capture the interactions and the semantics of resources. We present structures with labels and constraints, mainly labelled proof structures or nets, that can be seen as a kind of abstraction of formulae derivations in which we mainly manipulate labels and associated constraints instead of formulae. These structures give a geometrical representation of possible interactions in a formula [3]. Our perspective is to define methods in proof theory with a focus on semantic structures that provide a bridge between semantics and syntax and then could capture the essence of provability.

It is not trivial to find adequate methods in proof theory for the above mentioned resource logics. It is due to the specific management of resources or sets of resources (like bunches in BI or NL) and also to the difficulty to capture the specific interactions between connectives (additive and multiplicative connectives in BI, commutative and non-commutative connectives in NL), namely the particular semantics of these logics. Based-on semantics methods like tableaux or connection-based methods cannot be obtained for BI from standard methods with prefixes as defined for classical, intuitionistic or linear logics [14]. The notion of prefix is not strong enough to capture the semantics of BI and thus it is necessary to introduce another structure to deal with semantic information, namely a dependency graph (or resource graph) from which provability can be studied.

The correspondence between (resource) semantics and syntactic labels and constraints, used to defined new labelled calculi for BI, can be seen in both directions. For instance, in the case of BI without \perp , the labels and constraints directly reflect the elementary Kripke semantics of the logic [6] and then the relationships between semantics and dependency (or resource graphs) are clearly identified. In the case of BI (with \perp), the specific labels and constraints of the new proof theory do not reflect the initial Grothendieck topological semantics for which BI has been proved complete.

But, as the dependency graph associated to a BI formula contains the necessary information for analyzing provability, we can also deduce a new simple resource semantics that is complete for BI [8]. Even if our approach can be illustrated for a given logic, namely BI, we have to discuss and study if it can be developed for other logics in which we consider the formulae as resources and if it leads to new proof-theoretical foundations that are appropriate for verification tools, for instance in separation logics [12,20]. Moreover, it is important to notice that the labelled calculi with constraints can be used with different proof search methods: a tableaux method that deals with dependency graphs [9] but also a connection method that deals with particular labelled trees (including constraints) [5]. It is known that connection methods drastically reduce the search space compared to calculi analyzing the outer structure of formulae such as sequent or tableau calculi [2,22]. By introducing labels and constraints in a connection-based characterization of provability, we propose a simple and natural way to capture the essence of the provability and thus we generalize the based-on prefixes methods that are defined for IL or MLL [14]. The connection-based characterization of provability in BI with constraints has been defined in [5] and refined in [15] and provides an alternative method based on formula trees with constraints.

Starting from our previous results, we aim to emphasize the use and the interest of labelled structures with constraints by defining a connection-based characterization of provability for MILL. In fact, it is derived from the one proposed for BI by taking into account that BI is conservative w.r.t. MILL. This point was only mentioned in [5] and then we develop it in this paper in order to illustrate the underlying concepts. There exists a connection-based characterization of provability of Multiplicative Linear Logic (MLL) [14] based on prefixes but not for MILL because prefixes are not adapted to capture the initial semantics of MILL. Our new method is based on constraints that allow to capture the Urquhart’s semantics of MILL at a syntactic level and generalize prefixes. Another reason to consider MILL is the possibility to relate our results with the notion of proof net that is a particular geometric representation of proofs defined in Linear Logic. As it was previously done for MLL [4], we can show that such a connection-based characterization and its related method for MILL can lead to an algorithm for the construction of MILL proof nets. Conversely, this algorithm can be seen as a new connection method, that also builds in parallel sequent proofs. This method starts with the formula (or decomposition) tree and builds, step by step and automatically, axiom-links (or connections) and partial proof nets, being guided by strategies and resolution of constraints. Taking into account labels and constraints attached to different positions in the formula tree, some steps of proof nets construction are only possible if some constraints are satisfied. Another interesting point to study is the generation of proofs and mainly of countermodels in case of non-provability in MILL.

In section 2, we summarize what is BI logic, mainly in a semantic perspective and we remind our results about provability based on dependency (or resource) graphs. BI being conservative w.r.t. MILL, we derive new results for MILL from the ones obtained for BI. In section 3 we develop the main concepts of a new characterization of provability in MILL. In section 4 we propose an algorithm for MILL proof nets construction, based on labels and constraints, that can be considered as a connection method that implements our new characterization. In section 5 we focus on the generation of proofs and countermodels from this connection method. This paper is focused on a particular approach of the quest of the essence of provability and of proofs and we expect to relate it to other studies based on other kinds of structures and semantic objects and also with works on games semantics for proof-search.

2 From BI to MILL

The logic of Bunched Implications (BI) provides a logical analysis of the basic notion of resource, that is central in computer science, with well-defined proof-theoretic and semantic foundations [18,19]. Its propositional fragment freely combines multiplicative (or linear) $*$ and \multimap connectives and additive (or intuitionistic) \wedge , \rightarrow and \vee connectives [17] and can be seen as a merging of intuitionistic logic (IL) and multiplicative intuitionistic linear logic (MILL). BI has a Kripke-style semantics (interpretation of formulae) [17] which combines the Kripke semantics of IL and Urquhart’s semantics of MILL. The latter uses possible worlds, arranged as a commutative monoid and justified in terms of “pieces of information” [21]. The key property of the semantics is the

sharing interpretation. The (elementary) semantics of the multiplicative conjunction, $m \Vdash A * B$ iff there are n_1 and n_2 such that $n_1 \bullet n_2 \sqsubseteq m$, $n_1 \Vdash A$ and $n_2 \Vdash B$, is interpreted as follows: the resource m is sufficient to support $A * B$ just in case it can be divided into resources n_1 and n_2 such that n_1 is sufficient to support A and n_2 is sufficient to support B . Thus, A and B do not share resources. Similarly, the semantics of the multiplicative implication, $m \Vdash A \multimap B$ iff for all n such that $n \Vdash A$, $m \bullet n \Vdash B$, is interpreted as follows: the resource m is sufficient to support $A \multimap B$ just in case for any resource n which is sufficient to support A the combination $m \bullet n$ is sufficient to support B . Thus, the function and its argument *do not share* resources. In contrast, if we consider the standard Kripke semantics of the additives \wedge and \rightarrow the resources are shared. Because of the interaction of intuitionistic and linear connectives and its sharing interpretation, BI is different from Linear Logic (LL) and does not admit the usual number-of-uses reading [17]. BI logic has a sequent calculus with bunches with good properties but not well adapted to proof-search following backward reasoning (from the goal to the axioms).

In order to capture the specific interactions between connectives, for instance additive and multiplicative connectives in BI, and finally the semantics of connectives, we have defined labelled calculi including specific labels, constraints that allow to build semantic structures, called dependency (or resource graphs) [7]. Such structures contain the necessary semantical information from which provability can be studied. We can generate proofs or countermodels [6] from such labelled structures that can be also defined for other mixed logics like Non-commutative logic (NL) for which one has no simple resource semantics but only a bunched calculus not adapted to proof-search. In fact, a resource graph corresponds to a particular set of constraints and in order to propose a based-on connection method for BI we have integrated similar constraints in this formalism and thus proposed a new characterization of provability [5].

An important result is that BI logic is conservative w.r.t. Multiplicative Intuitionistic Linear Logic (MILL) and thus we can try to adapt the previous approach based on labelled semantic structures to MILL and analyze its impact on provability. It corresponds to generate semantic structures from MILL's Urquhart's semantics [21] and to develop a characterization of provability with labels and constraints that capture this semantics. It can be done with labelled tableaux calculi [6] but here we prefer to focus on connection-based calculi with constraints. A connection-based characterization of provability with constraints for BI has been defined in [5] and refined in [15]. Here we present its adaptation for MILL that was mentioned in previous works on BI but not explicitly developed. Our focus on MILL is also motivated by the possible relationships between our works and methods for proof nets construction or verification [4,16].

3 A Characterization of Provability in MILL

In this section, we focus on the notions of *labels* and *constraints* that are adequate for capturing the semantics of the connectives and their interactions. As said before, a connection-based characterization of provability with constraints for BI has been defined in [5] and refined in [15]. Thus, we present here its specialization (or refinement) for MILL and thus provide a new characterization based on labels and constraints for this logic. There exist methods based on prefixes defined for instance for IL or MLL [13,14,22] cannot be applied to MILL, the notion of prefix being not enough strong for capturing MILL semantics constraints. In fact, this development corresponds to start from MILL's Urquhart's semantics and to build specific semantic structures.

3.1 Labels and constraints

Given an alphabet C (for instance $a, b, c \dots$), C^0 , the set of *atomic labels on C* is defined as the set C extended with the unit symbol e . Then we define C^* , the set of *labels on C* , as the smallest set including C^0 closed by composition ($x, y \in C^*$ implies $xy \in C^*$).

Let us note that $aabcc$, $cbaca$ and $cbcaal$ are equivalent by definition (associativity and identity 1. A *constraint* is an expression $x \leq y$ in which x and y are labels. A constraint $x \leq x$ is an *axiom* and we write $x = y$ to denote $x \leq y$ and $y \leq x$. The inference rules used for reasoning on constraints are:

$$\frac{x \leq y}{xz \leq yz} \text{func} \qquad \frac{x \leq z \quad z \leq y}{x \leq y} \text{trans}$$

The rule *trans* formalizes the transitivity of \leq and the rule *func* corresponds to the compatibility of the label composition for \leq . In this system, given a constraint k and a set of constraints H , we denote $H \vdash k$ the deduction of k from H . The notation $H \approx K$, where K is a non-empty set of constraints, means that for all $k \in K$, $H \vdash k$.

3.2 Indexed formula trees with labels

Here we recall the standard notions coming from previous characterizations of provability by matrix [14,22]. A *decomposition tree* of a formula A is its representation as syntactic tree with nodes called *positions*. A position u exactly identifies a subformula of A denoted $f(u)$. An *atomic position* is a position for an atomic formula. The decomposition tree induces a partial order \ll on the positions such that the root is the least element and if $u \ll v$ then u dominates v in the tree. In fact, we do not distinguish a formula A from its decomposition tree. For each position, we assign a polarity $pol(u)$, a principal type $ptyp(u)$ and a secondary type $styp(u)$. Therefore, we have different principal types depending on the connective and the associated polarity. For instance in BI we have four principal types named $\alpha, \beta, \pi\alpha, \pi\beta$.

Depending of the principal type, we associate a label $slab(u)$ and sometimes a constraint $kon(u)$ to a position u . Such a label is either a position or a position with a tilde in order to identify the formula that introduces resources. We define constraints in order to capture the composition and distribution of formulae that are considered as resources. The *labelled signed formula* $lsf(u)$ of a position u is a triple $(slab(u), f(u), pol(u))$ and is denoted $slab(u) : f(u)^{pol(u)}$.

The construction of the indexed formula tree is inductively defined in Figure 1.

$lsf(u)$	$ptyp(u)$	$kon(u)$	$lsf(u_1)$	$lsf(u_2)$
$x : (A \multimap B)^0$	$\pi\alpha$	$xu = \tilde{u}$	$u : A^1$	$\tilde{u} : B^0$
$x : (A * B)^1$	$\pi\alpha$	$u\tilde{u} \leq x$	$u : A^1$	$\tilde{u} : B^1$
$x : (A \multimap B)^1$	$\pi\beta$	$xu = \tilde{u}$	$u : A^0$	$\tilde{u} : B^1$
$x : (A * B)^0$	$\pi\beta$	$u\tilde{u} \leq x$	$u : A^0$	$\tilde{u} : B^0$

Figure 1. Signed formulae for MILL

For a given formula A the root position a_0 has a polarity $pol(a_0) = 0$, a label $slab(a_0) = 1$ and the signed formula $1 : (A)^0$ where 1 is the identity of the label composition. u_1 and u_2 are respectively the first and second subpositions. The subpositions inherit the formula and the polarity of the position. The principal type of a position u depends on its principal type and its polarity and the associated constraint is built from its principal connector and its label.

Like in BI, the constraints associated to $\pi\alpha$ formulae are called *assertions* and those associated to $\pi\beta$ formulae are called *requirements* and they must be satisfied from the set of assertions.

In fact, the rules $*_R, \multimap_L$ and \rightarrow_L of BI's sequent calculus, that deal with $\pi\beta$ formulae, divide contexts and distribute resources. Because of weakening and contraction, we can have several occurrences of formulae and then we introduce a notion of *multiplicity* μ attached to $\pi\beta$ formulae. In this presentation, we keep this notion for MILL but we expect to show that a multiplicity of 1 is enough in MILL. Therefore, like in [22], we consider a formula A associated to a multiplicity μ and call this couple an *indexed formula* A^μ . Then the indexed formula tree for A^μ ($indt(A^\mu)$) is inductively defined as follows

Definition 1. u^κ is an indexed position of $indt(A^\mu)$ iff

- 1) u is a position in the decomposition tree of A .
- 2) Let $u_1 \ll \dots \ll u_n$ be all the $\pi\beta$ -positions that dominate u in the decomposition tree of A , then
 - a) $\mu(u_i) \neq 0$, $1 \leq i \leq n$ and
 - b) $\kappa = m_1 \dots m_n$, $1 \leq m_i \leq \mu(u_i)$, $1 \leq i \leq n$.

The order relation \ll^μ between two indexed positions u^κ and v^τ is defined in the following way: $u^\kappa \ll^\mu v^\tau$ iff $u \ll v$ and κ is an initial sequence of τ . We denote $Occ(u^\kappa)$ the set of indexed positions $u^{\kappa\tau}$ that are in A^μ .

3.3 Paths, connections and covers

In this paragraph, we consider the adaptations for MILL of the notions of paths, connections and covers as defined in [15] for BI.

Definition 2 (Path). Let A^μ be an indexed formula, u^κ an indexed position of $\text{indt}(A^\mu)$ and u_1, u_2 the immediate successors of u . The set of paths of A^μ is inductively defined as the smallest set such that:

1. $\{a_0\}$ is a path where a_0 is the root;
2. If s is a path that includes u^κ then
 - a) if $\text{ptyp}(u^\kappa) \in \{\alpha, \pi\alpha\}$ then, $(s \setminus u^\kappa) \cup u_1 \cup u_2$ is a path,
 - b) if $\text{ptyp}(u^\kappa) \in \{\beta, \pi\beta\}$ then, $(s \setminus u^\kappa) \cup u_1$ and $(s \setminus u^\kappa) \cup u_2$ are paths.

An *atomic path* is a path that only contains atomic positions. A *configuration* of A^μ is a finite set of paths of A^μ .

Definition 3 (Reduction). A reduction of an indexed formula A^μ is a finite sequence $(S_i)_{1 \leq i \leq n}$ of configurations in A^μ such that S_{i+1} is obtained from S_i by reduction of a position u in a path s of S_i following Definition 2. We say that S_{i+1} is obtained by reduction of S_i of u in s .

Definition 4 (Connection). A connection is a couple $\langle u, v \rangle$ of atomic positions such that $f(u) = f(v)$, $\text{pol}(u) = 1$ and $\text{pol}(v) = 0$. We denote Con the set of connections of A^μ .¹

Definition 5 (Cover). Let A^μ be an indexed formula. A connection $\langle u, v \rangle$ in A^μ covers a path s in A^μ if $u, v \in s$. Let S be a set of paths in A^μ , a cover of S is the set C defined as $C = \{ (s, \langle u, v \rangle) / s \in S \text{ and } \langle u, v \rangle \in \text{Con} \text{ and } \langle u, v \rangle \text{ cover } s \}$ such that

$$(s, \langle u, v \rangle) \in C \text{ and } (s, \langle u', v' \rangle) \in C \text{ imply that } u = u' \text{ et } v = v'.$$

A cover of A^μ is a cover of the set of atomic (consistent) paths in A^μ .

Given a formula A de BI, we use the following notations. The set of positions of A is denoted Pos . Given a set of positions $p \subseteq \text{Pos}$ we introduce:

- the set of positions of type $\pi\alpha$: $P_\alpha(p) = \{ u \mid u \in p \text{ et } \text{ptyp}(u) = \pi\alpha \}$
- the set of positions of type $\pi\beta$: $P_\beta(p) = \{ u \mid u \in p \text{ et } \text{ptyp}(u) = \pi\beta \}$
- the set of positions of secondary type $\pi\alpha_i$: $S_{\alpha_i}(p) = \{ u \mid u \in p \text{ et } \text{styp}(u) = \pi\alpha_i \} (i \in \{1, 2\})$
- the set of positions of secondary type $\pi\beta_i$: $S_{\beta_i}(p) = \{ u \mid u \in p \text{ et } \text{styp}(u) = \pi\beta_i \} (i \in \{1, 2\})$
- the set of positions of secondary type $\pi\alpha$: $S_\alpha(p) = S_{\alpha_1}(p) \cup S_{\alpha_2}(p)$
- the set of positions of secondary type $\pi\beta$: $S_\beta(p) = S_{\beta_1}(p) \cup S_{\beta_2}(p)$
- the set of constants: $\Sigma_\alpha(p) = \{ \text{slab}(u) \mid (\exists v \in P_\alpha(p))(u \text{ in the set of subpositions of } v) \}$
- the set of variables: $\Sigma_\beta(p) = \{ \text{slab}(u) \mid (\exists v \in P_\beta(p))(u \text{ in the set of subpositions of } v) \}$
- the set of assertions: $\mathcal{K}_\alpha(p) = \{ \text{kon}(u) \mid u \in P_\alpha(p) \}$
- the set of obligations: $\mathcal{K}_\beta(p) = \{ \text{kon}(u) \mid u \in P_\beta(p) \}$

When $p = \text{Pos}$, we simply write $P_\alpha, P_\beta, S_{\alpha_i}, S_{\beta_i}, S_\alpha, S_\beta, \Sigma_\alpha$ et Σ_β .

An example. Let us consider the formula $A^\mu \equiv (p * ((q \multimap r) * s)) \multimap ((p * (q \multimap r)) * s)$ of MILL. Figure 2 presents its indexed formula tree.

The two sets of positions $\pi\alpha$ and $\pi\beta$ are $P_\alpha = \{ a_0, a_1 a_3, a_{11} \}$ and $P_\beta = \{ a_4, a_8, a_9 \}$. Moreover, the set of assertions is $\mathcal{K}_\alpha = \{ a_0 = \tilde{a}_0, a_1 \tilde{a}_1 = a_0, a_3 \tilde{a}_3 \leq \tilde{a}_1, \tilde{a}_9 a_{11} \leq \tilde{a}_{11} \}$ and the set of obligations is $\mathcal{K}_\beta = \{ a_3 a_4 = \tilde{a}_4, a_8 \tilde{a}_8 \leq \tilde{a}_0, a_9 \tilde{a}_9 \leq a_8 \}$.

The reduction path process from $\{a_0\}$ provides the following atomic paths:

$$s_1 = \{ a_2, a_5, a_7, a_{10} \}, s_2 = \{ a_2, a_5, a_7, a_{12}, a_{13} \}, s_3 = \{ a_2, a_5, a_7, a_{14} \}, s_4 = \{ a_2, a_6, a_7, a_{10} \}, \\ s_5 = \{ a_2, a_6, a_7, a_{12}, a_{13} \} \text{ et } s_6 = \{ a_2, a_6, a_7, a_{14} \}.$$

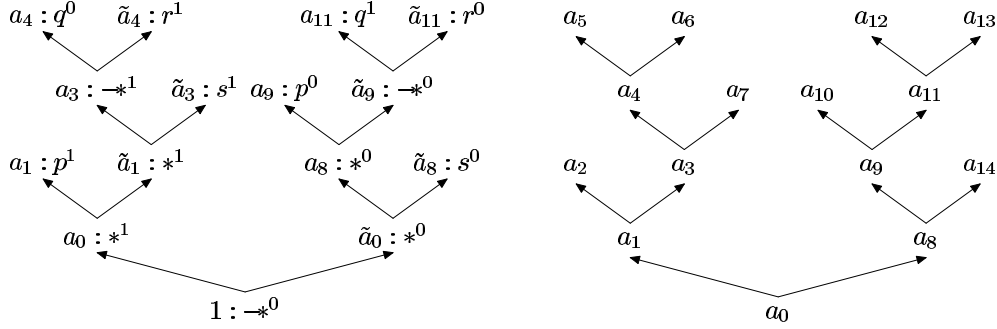
Their respective connections are:

$$\langle a_2, a_{10} \rangle, \langle a_{12}, a_5 \rangle, \langle a_7, a_{14} \rangle, \langle a_2, a_{10} \rangle, \langle a_6, a_{13} \rangle \text{ and } \langle a_7, a_{14} \rangle \text{ and we have a cover } C \text{ of } A^\mu : \\ C = \{ (s_1, \langle a_2, a_{10} \rangle), (s_2, \langle a_{12}, a_5 \rangle), (s_3, \langle a_7, a_{14} \rangle), (s_4, \langle a_2, a_{10} \rangle), (s_5, \langle a_6, a_{13} \rangle), (s_6, \langle a_7, a_{14} \rangle) \}.$$

The set of constraints from connections generated by C is then

$$\mathcal{K}_C = \{ (s_1, a_1 \leq a_9), (s_2, a_{11} \leq a_4), (s_3, \tilde{a}_3 \leq \tilde{a}_8), (s_4, \tilde{a}_1 \leq \tilde{a}_9), (s_5, \tilde{a}_4 \leq \tilde{a}_{11}), (s_6, \tilde{a}_3 \leq \tilde{a}_8) \}.$$

¹ Remark: the first position in a connection is the one of polarity 1.



u	$pol(u)$	$f(u)$	$ptyp(u)$	$styp(u)$	$slab(u)$	$kon(u)$
a_0	0	$(p * ((q \multimap r) * s)) \multimap ((p * (q \multimap r)) * s)$	$\pi\alpha$	—	1	$a_0 = \tilde{a}_0$
a_1	1	$p * ((q \multimap r) * s)$	$\pi\alpha$	$\pi\alpha_1$	a_0	$a_1 \tilde{a}_1 \leq a_0$
a_2	1	p	—	$\pi\alpha_1$	a_1	—
a_3	1	$(q \multimap r) * s$	$\pi\alpha$	$\pi\alpha_2$	\tilde{a}_1	$a_3 \tilde{a}_3 \leq a_1$
a_4	1	$q \multimap r$	$\pi\beta$	$\pi\alpha_1$	a_3	$a_3 a_4 = \tilde{a}_4$
a_5	0	q	—	$\pi\beta_1$	a_4	—
a_6	1	r	—	$\pi\beta_2$	\tilde{a}_4	—
a_7	1	s	—	$\pi\alpha_2$	\tilde{a}_3	—
a_8	0	$(p * (q \multimap r)) * s$	$\pi\beta$	$\pi\alpha_2$	\tilde{a}_0	$a_8 \tilde{a}_8 \leq \tilde{a}_0$
a_9	0	$p * (q \multimap r)$	$\pi\beta$	$\pi\beta_1$	a_8	$a_9 \tilde{a}_9 \leq a_8$
a_{10}	0	p	—	$\pi\beta_1$	a_9	—
a_{11}	0	$q \multimap r$	$\pi\alpha$	$\pi\beta_2$	\tilde{a}_9	$\tilde{a}_9 a_{11} = \tilde{a}_{11}$
a_{12}	1	q	—	$\pi\alpha_1$	a_{11}	—
a_{13}	0	r	—	$\pi\alpha_2$	\tilde{a}_{11}	—
a_{14}	0	s	—	$\pi\beta_2$	\tilde{a}_8	—

Figure 2. Indexed formula tree of $(p * ((q \multimap r) * s)) \multimap ((p * (q \multimap r)) * s)$

3.4 A new characterization of MILL provability

We now derive a new based-on characterization for MILL based on constraints that generalize the standard notion of prefix, already adapted to the MILL fragment [14]. It is derived from previous works in BI [15].

Definition 6 (MILL-substitution). Let A^μ an indexed formula. A MILL-substitution is an application $\sigma : \Sigma_\beta \rightarrow \Sigma_\alpha^*$, that can be extended to labels and constraints as follows:

- $\sigma(x) = x$ if x is a constant or if $x = 1$,
- $\sigma(x \bullet y) = \sigma(x) \bullet \sigma(y)$,
- $\sigma(x \leq y) = \sigma(x) \leq \sigma(y)$.

Definition 7 (MILL-certification). Let A^μ an indexed formula. A MILL-certification for A^μ is an application $\gamma : P_\beta \rightarrow \wp(P_\alpha)$ that associates, to any indexed position of principal type $\pi\beta$, a subset of the set of positions with $\pi\alpha$ as principal type.

Definition 8 (Complementarity). Let A^μ be an indexed formula and σ a MILL-substitution, a path s of A^μ is complementary under σ , or σ -complementary, if it is covered by a connection $\langle u, v \rangle$ such that $\sigma(\mathcal{K}_\alpha) \approx \sigma(\text{slab}(v)) \leq \sigma(\text{slab}(u))$. A cover C is complementary under σ if all paths s are complementary under σ .

Definition 9 (CMILL-Provability). A formula A of MILL is CMILL-provable if there exist a multiplicity μ , a cover C of the set of atomic paths of A^μ , a MILL-substitution σ and a MILL-certification γ for A^μ such that:

- (CBI1) the reduction relation \triangleleft is irreflexive,

- (CBI2) $\forall (s, \langle u, v \rangle) \in C, \forall w \in P_\beta(\{u, v\}), \gamma(w) \subseteq P_\alpha(s)$,
(CBI3) $\forall (s, \langle u, v \rangle) \in C, \forall w \in P_\beta(\{u, v\}), \sigma(k(\gamma(w))) \approx \sigma(k(w))$,
(CBI4) $\forall (s, \langle u, v \rangle) \in C, \forall x \in \Sigma_\beta(\{u, v\}), \sigma(x) \in \Sigma_\alpha(s)$,
(CBI5) $\forall (s, \langle u, v \rangle) \in C, \sigma(\mathcal{K}_\alpha(s)) \approx (\sigma(\text{slab}(v)) \leq \sigma(\text{slab}(u)))$.

Let us come back to our example. In order to find a MILL-substitution σ from \mathcal{K}_C , we consider:
 $\sigma(a_9) = a_1, \sigma(a_4) = a_{11}, \sigma(\tilde{a}_8) = \tilde{a}_3, \sigma(\tilde{a}_4) = \tilde{a}_{11}, \sigma(a_8) = X, \sigma(\tilde{a}_9) = Y$.

Then we have to compute $\sigma(\mathcal{K}_\alpha) \approx \sigma(\mathcal{K}_\beta)$ and then 1. $a_0 = \tilde{a}_0, a_1 \tilde{a}_1 \leq a_0, a_3 \tilde{a}_3 \leq \tilde{a}_1, Y a_{11} = \tilde{a}_{11} \approx a_3 a_{11} = \tilde{a}_{11}$

2. $a_0 = \tilde{a}_0, a_1 \tilde{a}_1 \leq a_0, a_3 \tilde{a}_3 \leq \tilde{a}_1, Y a_{11} = \tilde{a}_{11} \approx X \tilde{a}_3 \leq \tilde{a}_0$

3. $a_0 = \tilde{a}_0, a_1 \tilde{a}_1 \leq a_0, a_3 \tilde{a}_3 \leq \tilde{a}_1, Y a_{11} = \tilde{a}_{11} \approx a_1 Y \leq X$

From 1. we directly deduce $Y = a_3$ and also $\gamma(a_4) = \{a_{11}\}$. The obligation of 1. is the one of the position a_4 and in order to verify it we use the assertion $a_3 a_{11} = \tilde{a}_{11}$ of position a_{11} . From 3. we deduce a trivial solution for X that is $X = a_1 a_3$ and also that $\gamma(a_9) = \emptyset$. The condition 2. is verified because we have:

$$\frac{\frac{a_1 \leq a_1 \quad a_3 \tilde{a}_3 \leq \tilde{a}_1}{a_1 a_3 \tilde{a}_3 \leq a_1 \tilde{a}_1} \text{func} \quad a_1 \tilde{a}_1 \leq a_0}{a_1 a_3 \tilde{a}_3 \leq a_0} \text{trans} \quad \frac{a_0 = \tilde{a}_0}{a_1 a_3 \tilde{a}_3 \leq \tilde{a}_0} \text{trans}$$

and then we deduce $\gamma(a_8) = \{a_0, a_1, a_3\}$ since $a_0 = \tilde{a}_0, a_1 \tilde{a}_1 \leq a_0, a_3 \tilde{a}_3 \leq \tilde{a}_1$ are the respective assertions of a_0, a_1, a_3 .

In order to verify the conditions (CBI2) and (CBI5), let us consider the following table

$(s, \langle u, v \rangle)$	$P_\beta(\{u, v\})$	$P_\alpha(s)$	$\Sigma_\beta(\{u, v\})$	$\Sigma_\alpha(s)$
$(s_1, \langle a_2, a_{10} \rangle)$	a_8, a_9	a_0, a_1, a_3	$a_8, \tilde{a}_8, a_9, \tilde{a}_9$	$a_0, \tilde{a}_0, a_1, \tilde{a}_1, a_3, \tilde{a}_3$
$(s_2, \langle a_{12}, a_5 \rangle)$	a_4, a_8, a_9	a_0, a_1, a_3, a_{11}	$a_4, \tilde{a}_4, a_8, \tilde{a}_8, a_9, \tilde{a}_9$	$\tilde{a}_0, a_1, \tilde{a}_1, a_3, \tilde{a}_3, a_{11}, \tilde{a}_{11}$
$(s_3, \langle a_7, a_{14} \rangle)$	a_8, a_9	a_0, a_1, a_3	a_8, \tilde{a}_8	$a_0, \tilde{a}_0, a_1, \tilde{a}_1, a_3, \tilde{a}_3$
$(s_4, \langle a_2, a_{10} \rangle)$	a_8, a_9	a_0, a_1, a_3	$a_8, \tilde{a}_8, a_9, \tilde{a}_9$	$a_0, \tilde{a}_0, a_1, \tilde{a}_1, a_3, \tilde{a}_3$
$(s_5, \langle a_6, a_{13} \rangle)$	a_4, a_8, a_9	a_0, a_1, a_3, a_{11}	$a_4, \tilde{a}_4, a_8, \tilde{a}_8, a_9, \tilde{a}_9$	$\tilde{a}_0, a_1, \tilde{a}_1, a_3, \tilde{a}_3, a_{11}, \tilde{a}_{11}$
$(s_6, \langle a_7, a_{14} \rangle)$	a_8, a_9	a_0, a_1, a_3	a_8, \tilde{a}_8	$a_0, \tilde{a}_0, a_1, \tilde{a}_1, a_3, \tilde{a}_3$

Moreover, $\gamma(a_9) = \emptyset \subseteq P_\alpha(s)$, for any path $s \in \{s_1, s_2, s_4, s_5, s_6\}$ and $\gamma(a_8) = \{a_0, a_1, a_3\} \subseteq P_\alpha(s)$ for any path $s \in \{s_1 \dots s_6\}$. Moreover, for any path, $s \in \{s_2, s_5\}$, $\gamma(a_4) = \{a_{11}\} \subseteq P_\alpha(s)$. Then the condition (CBI2) is verified. In addition, for any path $s \in \{s_1, s_2, s_4, s_5\}$ we have $\sigma(a_9) = a_1 \in \Sigma_\alpha(s)^*$ and $\sigma(\tilde{a}_9) = a_3 \in \Sigma_\alpha(s)^*$ for any path $s \in \{s_1, s_2, s_3, s_4, s_5, s_6\}$ we have $\sigma(a_8) = a_1 a_3 \in \Sigma_\alpha(s)^*$ and $\sigma(\tilde{a}_8) = \tilde{a}_3 \in \Sigma_\alpha(s)^*$, and for any path $s \in \{s_2, s_5\}$ we have $\sigma(a_4) = a_{11} \in \Sigma_\alpha(s)^*$ and $\sigma(\tilde{a}_4) = \tilde{a}_{11} \in \Sigma_\alpha(s)^*$.

It remains to compute the reduction relation \triangleleft that is obtained by the transitive closure of the domination relation \ll , the instantiation relation \sqsubset and the deduction relation \sqsubset' . The instantiation relation induced by σ is

$$a_1 \sqsubset a_9, a_3 \sqsubset a_9, a_1 \sqsubset a_8, a_{11} \sqsubset a_4$$

and the deduction relation induced by γ is

$$a_0 \sqsubset' a_8, a_1 \sqsubset' a_8, a_3 \sqsubset' a_8, a_{11} \sqsubset' a_4.$$

The reduction relation \triangleleft is represented in Figure 3. As the graph is acyclic, A^μ is valid in MILL.

As illustrated by the example, the constraints have composed labels on the lefthand side. Moreover, the constraints for the implication deal with equality.

3.5 Soundness and Completeness

These properties are proved like the corresponding ones for BI [15] with adequate restrictions to MILL.

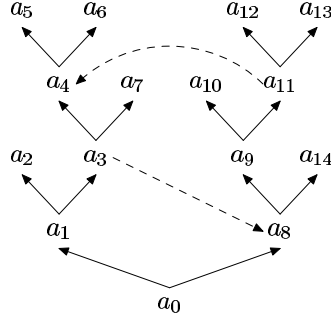


Figure 3. Reduction order for $(p * ((q \multimap r) * s)) \multimap ((p * (q \multimap r)) * s)$

Definition 10 (Complete reduction). Let A^μ be an indexed formula and C be a cover of A^μ , a reduction $(S_i)_{1 \leq i \leq n}$ in A^μ is complete for C if C is a cover of S_n .

Definition 11 (Proper reduction). Let A^μ be an indexed formula and σ be a MILL-substitution pour A^μ , a reduction $(S_i)_{1 \leq i \leq n}$ is σ -proper if

- (i) $\forall S \in (S_i)_{1 \leq i \leq n}, \forall s \in S, \sigma(\mathcal{K}_\alpha(s)) \approx \sigma(\mathcal{K}_\beta(s))$
- (ii) $\forall S \in (S_i)_{0 \leq i \leq n}, \forall s \in S, \forall x \in \Sigma_\beta(s), \sigma(x) \in \Sigma_\alpha(s)^*$.

Definition 12 (Realization). Let A^μ be an indexed formula and s be a path of A^μ . A CMILL-interpretation of s in a resource model $K = \langle (M, \sqsubseteq, \bullet, e), \models, \llbracket - \rrbracket \rangle$ is a function $\| - \|: \Sigma_\alpha(s) \rightarrow M$ that can be extended to labels $\Sigma_\alpha(s)^*$ with $\| 1 \| = e$ and $\| xy \| = \| x \| \bullet \| y \|$.

Given a MILL-substitution σ , a realization of s is a couple $(\| - \|, \sigma)$ such that:

1. For any assertion $x \leq y \in \mathcal{K}_\alpha(s), \| \sigma(x) \| \sqsubseteq \| \sigma(y) \|$.
2. For any position $u \in s$ such that $lsf(u) = x : A^1, \| \sigma(x) \| \models A$.
3. For any position $u \in s$ such that $lsf(u) = x : A^0, \| \sigma(x) \| \not\models A$.

A path is said realizable if there exists a realization of s in a model K . A configuration is realizable if there is at least one of its paths that is realizable.

Lemma 1. Let A^μ be an indexed formula, σ be a MILL-substitution for A^μ and $(S_i)_{1 \leq i \leq n}$ be a σ -proper reduction for A^μ , if S_i is σ -realizable then S_{i+1} is σ -realizable.

Lemma 2. Let A^μ be an indexed formula and σ be a MILL-substitution for A^μ . If a path s is complementary under σ then it is not realizable under σ .

Proof. Let us suppose s σ -complementary and realizable for an interpretation ι in a resource model M . s is σ -complementary because it contains a connection that is σ -complementary. In fact, s contains a connection $\langle u, v \rangle$ that is complementary and such that

$$f(u) = f(v), pol(u) = 1, pol(v) = 0 \text{ and } \sigma(\mathcal{K}_\alpha(s)) \approx \sigma(\text{slab}(v)) \leq \sigma(\text{slab}(u)).$$

As s is realizable, we have $\| \sigma(\text{slab}(u)) \| \models f(u), \| \sigma(\text{slab}(v)) \| \not\models f(u)$ et $\text{slab}(v) \sqsubseteq \text{slab}(u)$ that is contradictory because, by monotonicity, $\text{slab}(v) \sqsubseteq \text{slab}(u)$ and $\| \sigma(\text{slab}(u)) \| \models f(u)$ imply $\| \sigma(\text{slab}(v)) \| \models f(u)$.

In order to study the properties of this characterization, we consider the Urquhart's resource semantics for MILL [21].

Theorem 1 (Soundness of the characterization). If a formula A is CMILL-provable then it is valid in the resource semantics of MILL.

Proof. The conditions (CBI1) to (CBI5) of Definition 9 are verified because A^μ is provable. Let us assume that A is not valid in the semantics, then there exists a resource model M such that $e \not\models A$. Then, the initial set of paths $S_1 = \{ \{ a_0 \} \}$ is realizable under σ for the interpretation $\| - \|$ with an empty domain. The above mentioned conditions imply that there exists a reduction $(S_i)_{1 \leq i \leq n}$ from S_1 , that is complete for C , σ -proper and such that each path of S_n contains at least a connection of C . As S_1 is realizable under σ , by Lemma 1, S_n is also realizable under σ . But S_n cannot be complementary from Lemma 2. That is contradictory and then A is valid.

Theorem 2 (Completeness of the characterization). *If a formula A of MILL is valid in the resource semantics, then it is CMILL-provable.*

Proof. It is sufficient to show that if A is provable then it is CMILL-provable. The proof is by induction on the derivation in the MILL calculus deduced from LBI. Let us remind that a sequent $\Gamma \vdash A$ is provable in LBI iff $\Phi_\Gamma \multimap A$ is provable in LBI. We only consider the case \multimap_R , the others being similar.

By induction hypothesis, we assume that $\Gamma, A \vdash B$ is CMILL-provable and show that $\Gamma \vdash A \multimap B$ is also CMILL-provable. As $\Gamma, A \vdash B$ is CMILL-provable, there exist a multiplicity μ , an atomic reduction $R_1 = (\mathcal{S}_i)_{1 \leq i \leq n}$ of $((\Phi_\Gamma * A) \multimap B)^\mu$, a cover C of S_n , a MILL-substitution σ and a MILL-certification γ for A^μ that satisfies the conditions of Definition 9.

From the atomic reduction R_1 for $((\Phi_\Gamma * A) \multimap B)^\mu$, we can build an atomic reduction R_2 for $(\Phi_\Gamma \multimap (A \multimap B))^\mu$. In the following figure, we give the first steps of R_1 on the lefthand side and those of R_2 and the righthand side.

$$\left. \begin{array}{c} \{ 1 : ((\Phi_\Gamma * A) \multimap B)^0 \} \\ \{ a_0 : (\Phi_\Gamma * A)^1, \tilde{a}_0 : B^0 \} \\ \{ a_1 : \Phi_\Gamma^1, \tilde{a}_1 : A^1, \tilde{a}_0 : B^0 \} \\ \vdots \end{array} \right| \begin{array}{c} \{ 1 : (\Phi_\Gamma \multimap (A \multimap B))^0 \} \\ \{ a_0 : \Phi_\Gamma^1, \tilde{a}_0 : A \multimap B^0 \} \\ \{ a_1 : \Phi_\Gamma^1, a_i : A^1, \tilde{a}_i : B^0 \} \\ \vdots \end{array}$$

After the two first reduction steps, we observe that the two paths of R_1 and R_2 contain the same signed formulae modulo a label renaming a_1 in a_i and \tilde{a}_1 in \tilde{a}_i . Consequently, the next steps of R_2 can be the same as for R_1 and both reductions introduce same signed formulae and label constraints modulo renaming. The assertions of the two first steps of R_1 , $\{ a_0 = \tilde{a}_0, a_1 \tilde{a}_1 \leq a_0 \}$, are weaker than those of R_2 , $\{ a_0 = \tilde{a}_0, a_0 a_i = \tilde{a}_i \}$. Since R_1 provides a set of atomic paths satisfying CMILL-provability (see Definition 9), by induction hypothesis, the set of atomic paths for R_2 also satisfies these conditions.

4 Connections and Proof Nets construction

The notion of proof nets has been introduced by Girard [11] in order to deal with the intrinsic parallelism of the sequent calculus. It has been defined for various fragments of linear logic and studied from both construction and verification perspectives [4,16]. It is known that there are strong relationships between connection methods and proof nets construction for MLL [4] and our aim is to analyze if the previous results can be related to MILL proof nets construction.

4.1 Proof nets with constraints

Let us first present the main ideas that are derived from the principles used in MLL and adapted to MILL with an emphasis on constraints. This approach is based on the construction of the decomposition tree with semantical information included in the tree: formulae, polarities, labels, constraints associated to subformulae.

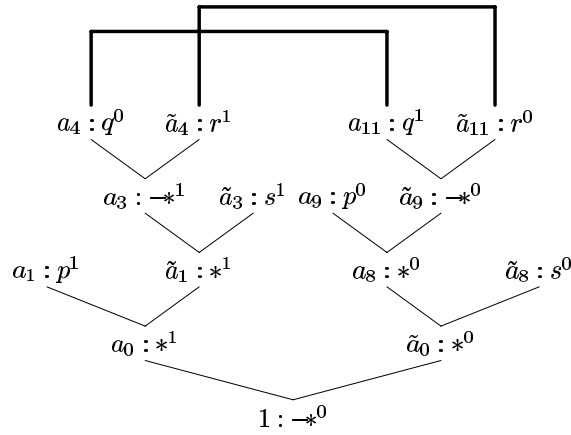
Then, from basic results about permutabilities and proof-search strategies in Linear Logic [10], we know that we have to treat the connectives $\multimap^1, *^0, *^1$ and \multimap^0 following this order. It means that the $\pi\beta$ -formulae have to be dealt before the $\pi\alpha$ formulae.

Therefore, we start from leaves (belonging to Σ_β) of a subformula of type $\pi\beta$ that is the highest in the labelled tree and try to connect them to leaves of the set Σ_α in order to build axiom-links (or connections). Each time the subformulae of a $\pi\beta$ -formula belong to a net under construction, we generate a MILL-substitution σ and then apply it to the obligation associated to the formula. For instance, for a connection $\langle u, v \rangle$ in which $slab(u)$ is a variable and $slab(v)$ is a constant, we have $\sigma(slab(u)) = slab(v)$. When two premisses of a subformula of type $\pi\beta$ are conclusions of two partial proof nets, we merge them and extend the resulting net with a $\pi\beta$ -link and provide a new proof net. We also add the corresponding constraint to the set of obligations after the application

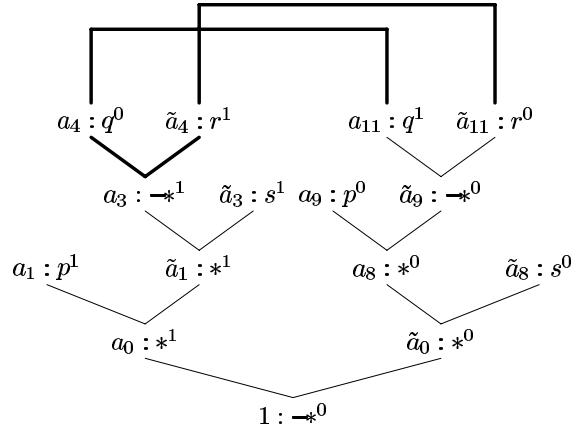
of σ . When two premisses of a subformula of type $\pi\alpha$ belongs to the same net, we extend it with a $\pi\alpha$ -link and add the associated constraint to the set of assertions.

During this construction, several choices of connections are possible and then if, after a first choice, the resolution of constraints leads to a failure we must backtrack and going on with another choice. It is necessary to test all possibilities until the net covers all the initial decomposition tree. Then we can deduce if it is provable or not.

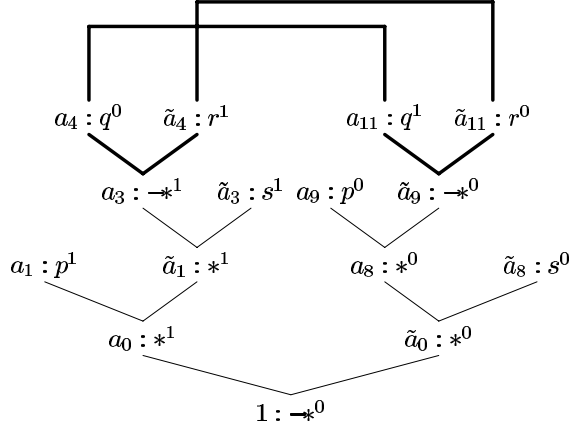
Let us consider the example of Section 3. First, we build the initial labelled tree that is the one of Figure 2. The highest $\pi\beta$ formula in this tree is a_4 and thus we start by trying to connect its subformulae (or subpositions) a_5 and a_6 . Let us start with a_5 that is connected to a_{12} because $f(a_5) = f(a_{12}) = q, pol(a_5) = 0$ and $pol(a_{12}) = 1$. We obtain an elementary net \mathcal{R}_1 and $\sigma(a_4) = a_{11}$. Then, we connect a_6 to the only possible position a_{13} because $f(a_6) = f(a_{13}) = r, pol(a_6) = 0$ et $pol(a_{13}) = 1$. We create a new elementary net \mathcal{R}_2 and generate $\sigma(\tilde{a}_4) = \tilde{a}_{11}$.



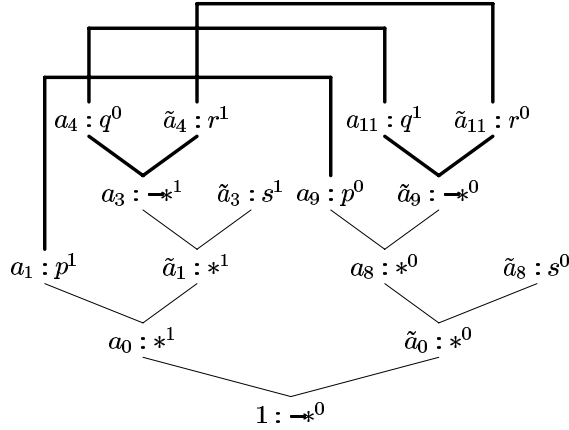
Thus, we merge \mathcal{R}_1 and \mathcal{R}_2 into a new net \mathcal{R}_1 with a $\pi\beta$ -link and we apply σ to the obligation $a_3a_4 = \tilde{a}_4$. Then, we deduce $a_3a_{11} = \tilde{a}_{11}$ (*Req1*).



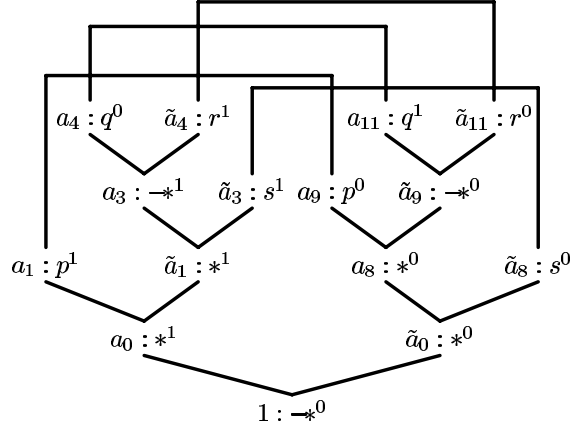
The leaves a_{12} et a_{13} are now conclusions of the net \mathcal{R}_1 and also the premisses of the position a_{11} that is a $\pi\alpha$ -position. Then, we extend \mathcal{R}_1 with a $\pi\alpha$ -link and obtain $\sigma(\tilde{a}_9) = a_3$ because the positions a_4 and a_{11} are linked and the assertion becomes $a_3a_4 = \tilde{a}_4$. Thus, it satisfies the obligation of a_4 . Let us remark that, in order to satisfy *Req1*, we have used the position a_{11} .



Position a_9 corresponds to a $\pi\beta$ formula and its subposition a_{11} has been already treated. Then we need to consider the position a_{10} . We connect it with a_2 (that is the only possibility) and generate a new net \mathcal{R}_2 and deduce $\sigma(a_9) = a_1$. As the two subpositions of a_9 are the conclusions of \mathcal{R}_1 and \mathcal{R}_2 , we merge them with extension by a $\pi\beta$ -link in order to provide a new net \mathcal{R}_1 . The application of σ to the obligation provides $a_1 a_3 \leq \sigma(a_8)$. Then we have a solution for a_8 that is $\sigma(a_8) = a_1 a_3$. In order to satisfy this obligation, we have not to use assertions.



It remains to consider the $\pi\beta$ -position a_8 . Its subposition a_{14} does not belong to a net and then we connect it to a_2 and thus we have $\sigma(\tilde{a}_8) = \tilde{a}_3$. It provides a new elementary net \mathcal{R}_2 . The positions a_9 and a_{14} respectively belong to \mathcal{R}_1 and \mathcal{R}_2 and are the premisses of a_8 . We can then merge \mathcal{R}_1 and \mathcal{R}_2 into a new \mathcal{R}_1 with a $\pi\beta$ -link and apply σ to the obligation associated to a_8 : $a_1 a_3 \tilde{a}_3 \leq \tilde{a}_0$ (*Req2*). At position a_3 , the two premisses are in fact conclusions of \mathcal{R}_1 and then we extend it with a $\pi\alpha$ -link and we add the assertion $a_3 \tilde{a}_3 \leq \tilde{a}_1$ to our set of constraints. By compatibility, we have $a_1 a_3 \tilde{a}_3 \leq a_1 \tilde{a}_1$ (*Ass*). As before, we extend the net at the position a_1 with a $\pi\alpha$ -link and a new assertion $a_1 \tilde{a}_1 \leq a_0$ and then (*Ass*) becomes, by transitivity, $a_1 a_3 \tilde{a}_3 \leq a_0$. Finally, at position a_0 , we extend the net \mathcal{R}_1 with a $\pi\alpha$ -link and by transitivity (*Ass*) becomes $a_1 a_3 \tilde{a}_3 \leq \tilde{a}_0$ and consequently (*Req2*) is verified. As for the obligation (*Req1*), we are in position to claim that the assertions necessary to satisfy (*Req2*) come from positions a_0, a_1 and a_3 . We then conclude that all connections are σ -complementary and then A^μ is provable and then valid in MILL.



This method for proof nets construction corresponds to a particular algorithm that implements the based-on connection characterization of provability and then can be viewed as a new connection method for MILL. We observe in our example that the connections we find are the same than the ones obtained in the example developed in Section 3. We have generated connections that could be σ -complementary and then verified, step by step, the admissibility of the MILL-substitution. Moreover, for the resolution of constraints, we can say which assertions is used for the satisfaction of an obligation without using the process of path reduction. In addition, the computation of requirements from the assertions and of the certification are automatically made step by step.

4.2 An algorithm for proof nets construction

Let us describe more formally this algorithm that is based on the algorithm dedicated to the construction of MILL proof nets [4] in which, in addition to the search of connections, one must verify that the requirements are satisfied from the assertions. Here, we only describe the main steps of the algorithm.

input: Formula A

output: Proof net of A or failure.

step 0: Construction of the labelled decomposition tree.

step 1: Choice of a $\pi\beta$ -position pm , not being already treated² and highest in the labelled tree.

step 2: If the left subposition pbg of pm is not already treated, connect pbg as follows:

If pbg is a leaf then

choose a position pa_1 such that $slab(pa_1)$ constant, $pol(pa_1) \neq pol(pbg)$ and $f(pa_1) = f(pbg)$;
 build an elementary net R_1 ;
 generate $\sigma(slab(pbg)) = slab(pa_1)$

Else return failure.

step 3: If the right subposition pbd of pm is not already treated, connect pbd as follows:

If pbd is a leaf then

choose a position pa_2 such that $slab(pa_2)$ constant, $pol(pa_2) \neq pol(pbd)$ and $f(pa_2) = f(pbd)$;
 build an elementary net R_2 ;
 generate $\sigma(slab(pbd)) = slab(pa_2)$

Else return failure.

step 4: Merge R_1 and R_2 into a net R at position pbm and apply σ to its obligation (Ob) that becomes the current obligation;

Verify if (Ob) is an axiom: if yes return at step 1;

step 5: For all $\pi\alpha$ -positions,

if the two premisses of a $\pi\alpha$ -position, pam , are conclusions of R then

extend the net at this position; apply σ to the assertion of pam ; add it to the resolution system;

verify (Ob) from the set of assertions.

² A position is treated if it belongs to a net

step 6: If the initial labelled tree is not completely covered by the net R then
 if there is no $\pi\beta$ -position to treat and at least a $\pi\beta$ -position u has one subposition for which connections are possible then
 if there are nets then
 break the net from the current position to the position u and return to step 1
 else if it remains $\pi\beta$ positions to treat then return to step 1.
 else return failure
 else return the net.

To verify an obligation means here that each time an assertion is added to the resolution set, one does the transitive and compatible closure of the assertion added from the last generated constraint and compare the new constraint with the last obligation.

This algorithm can be proved correct and complete from similar proofs of the algorithm for MILL proof nets [4] with addition of a specific treatment of the constraints.

Theorem 3 (Correctness). *If the algorithm returns a proof net for A then the formula A is provable in MILL.*

Theorem 4 (Completeness). *If a formula A is provable in MILL then the algorithm returns a proof net for A .*

Moreover this algorithm provides a connection method for MILL because it builds, step by step, a set of connections, a cover, a substitution and a certification such that A is CMILL-provable.

If we aim to relate the connection method associated to our new characterization and the construction of a proof net, we again consider our example with the formula $(p*((q \multimap r)*s)) \multimap ((p*(q \multimap r))*s)$. We can observe that they both generate the same set of connections, namely $\langle a_2, a_{10} \rangle$, $\langle a_{12}, a_5 \rangle$, $\langle a_7, a_{14} \rangle$, $\langle a_2, a_{10} \rangle$, $\langle a_6, a_{13} \rangle$ and $\langle a_7, a_{14} \rangle$. Moreover, the MILL-substitutions and the MILL-certifications are the same in both cases. We aim, in future works, to study such an algorithm also in the context of verification [16] by focusing on the constraint resolution.

5 Generation of proofs and countermodels

The previous algorithm builds a set of connections and a proof net in case of provability in MILL. It corresponds to a proof-search procedure with forward reasoning (from axioms to the goal formula) that can, step by step, build a proof in the MILL sequent calculus. In parallel with the proof nets construction, the algorithm builds a proof in a top-down way, from axioms (corresponding to axiom-links) by application of inference rules each time the corresponding partial nets are extended by a new link.

Let us come back to our example and show how the algorithm builds a sequent proof. The first connection $\langle a_5, a_{12} \rangle$ corresponds to the sequent $q \vdash q$ and the second one to $r \vdash r$. Then we build the sequent $q, q \multimap r \vdash r$ by the $\pi\beta$ -link of the position a_4 , and then, at position a_{11} , the $\pi\alpha$ -link provides the sequent $q \multimap r \vdash q \multimap r$.

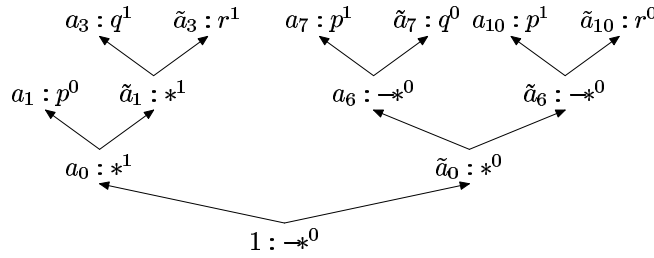
With the connection $\langle a_4, a_{11} \rangle$, we build the sequent $p \vdash p$ and the $\pi\beta$ -link at position a_9 generates the sequent $p, q \multimap r \vdash p*(q \multimap r)$. Then, the connection $\langle a_7, a_{14} \rangle$ provides the sequent $s \vdash s$ and the $\pi\beta$ -link applied to position a_8 generates $p, q \multimap r, s \vdash (p*(q \multimap r))*s$. Finally, successive applications of $\pi\alpha$ -links to the positions a_3, a_1, a_0 (following this order) provide, step by step, the sequent $p, (q \multimap r)*s \vdash (p*(q \multimap r))*s$, the sequent $p*((q \multimap r)*s) \vdash (p*(q \multimap r))*s$ and finally the sequent $\vdash (p*((q \multimap r)*s)) \multimap ((p*(q \multimap r))*s)$.

The final sequent proof built in parallel of the proof net construction is the following:

$$\begin{array}{c}
\frac{q \vdash q \quad r \vdash r}{q, q \multimap r \vdash r} \multimap_L \\
\frac{\frac{p \vdash p \quad q \multimap r \vdash q \multimap r}{p, q \multimap r \vdash p * (q \multimap r)} \multimap_R}{p, q \multimap r, s \vdash p * (q \multimap r) * s} *R \\
\frac{\frac{\frac{p, q \multimap r, s \vdash (p * (q \multimap r)) * s}{p, (q \multimap r) * s \vdash (p * (q \multimap r)) * s} *L}{p * ((q \multimap r) * s) \vdash (p * (q \multimap r)) * s} *L}{\vdash (p * ((q \multimap r) * s)) \multimap ((p * (q \multimap r)) * s)} \multimap_R
\end{array}$$

The most interesting point is the case of non-provability of a formula and then the possible generation of countermodels in a given semantics. Recent results on proof-search in BI, and consequently in MILL and IL [6], are based on a specific semantic structures, called resource graphs, that graphical representation of the set of assertions generated through the proof-search process. In case of non-provability, we can extract countermodels from such structures that can be also considered directly as geometric representations of countermodels. With our approach based on connections and on proof nets construction the questions of generation and representation of countermodels also arise. It appears that we can extract a countermodel from the labelled formula tree, an incomplete set of connections and the partial proof structure built before the failure in the construction process.

Let us illustrate this point with an example of a non-provable formula, namely the formula $(p \multimap (q * r)) \multimap ((p \multimap q) * (p \multimap r))$. Its indexed formula tree is given in Figure 4.

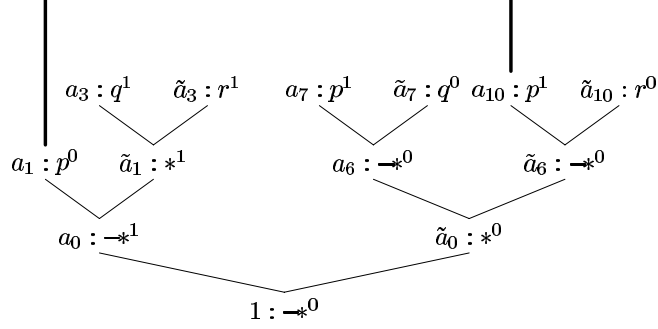


u	$pol(u)$	$f(u)$	$ptyp(u)$	$styp(u)$	$slab(u)$	$kon(u)$
a_0	0	$(p \multimap (q * r)) \multimap ((p \multimap q) * (p \multimap r))$	$\pi\alpha$	—	1	$a_0 = \tilde{a}_0$
a_1	1	$p \multimap (q * r)$	$\pi\beta$	$\pi\alpha_1$	a_0	$a_0 a_1 = \tilde{a}_1$
a_2	0	p	—	$\pi\beta_1$	a_1	—
a_3	1	$q * r$	$\pi\alpha$	$\pi\beta_2$	\tilde{a}_1	$a_3 \tilde{a}_3 \leq \tilde{a}_1$
a_4	1	q	—	$\pi\alpha_1$	a_3	—
a_5	1	r	—	$\pi\alpha_2$	\tilde{a}_3	—
a_6	0	$(p \multimap q) * (p \multimap r)$	$\pi\beta$	$\pi\alpha_2$	\tilde{a}_0	$a_6 \tilde{a}_6 \leq \tilde{a}_0$
a_7	0	$p \multimap q$	$\pi\alpha$	$\pi\beta_1$	a_6	$a_6 a_7 = \tilde{a}_7$
a_8	1	p	—	$\pi\alpha_1$	a_7	—
a_9	0	q	—	$\pi\alpha_2$	\tilde{a}_7	—
a_{10}	0	$p \multimap r$	$\pi\alpha$	$\pi\beta_2$	\tilde{a}_6	$\tilde{a}_6 a_{10} = \tilde{a}_{10}$
a_{11}	1	p	—	$\pi\alpha_1$	a_{10}	—
a_{12}	0	r	—	$\pi\alpha_2$	\tilde{a}_{10}	—

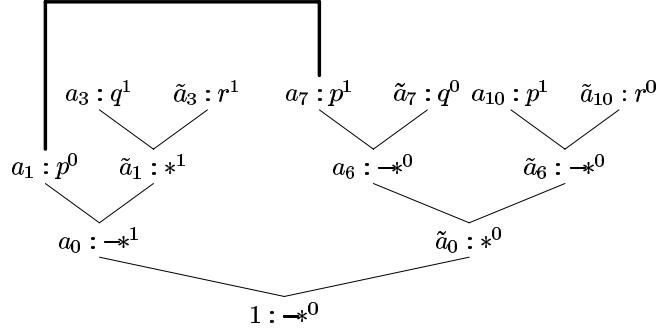
Figure 4. Indexed formula tree for $(p \multimap (q * r)) \multimap ((p \multimap q) * (p \multimap r))$

Let us build a proof net following our algorithm. The only variable on a leaf is a_1 the position of which is a_2 . There are two possibilities: either we connect it with the position a_{11} or with the

position a_8 . Let us try to connect with the the position a_{11} (with label a_{10}). We obtain the MILL-substitution $\sigma(a_1) = a_{10}$. Then, we consider the position a_4 where $lsf(a_4) = a_3 : q^1$ and try to connect it. The only possibility is a_9 with the label \tilde{a}_7 that does not belong to the set of variables Σ_β but to the set of constants Σ_α . As we have $\sigma : \Sigma_\beta \rightarrow \Sigma_\alpha^*$, we obtain a failure.



Let us erase the previous connections and try now to connect a_2 with a_8 , having the label a_7 . We deduce $\sigma(a_1) = a_7$.



As in the previous case, no connection is possible because the set of non-treated leaves is included in Σ_α and then it is not possible to build connections. Therefore, we conclude that the formula is not valid in MILL.

From the indexed labelled tree and the connection in the second case (and consequently from the MILL-substitution), we can build a countermodel. The forcing relation is defined as follows: for $p : a_7 \Vdash p$ and $a_{10} \Vdash p$, for $q : a_3 \Vdash q$, for $r : \tilde{a}_3 \Vdash r$.

In order to check, we can show that $a_0 \Vdash p \multimap (q * r)$ and $\tilde{a}_0 \not\Vdash (p \multimap q) * (p \multimap r)$. We have $a_3 \Vdash q$ and $\tilde{a}_3 \Vdash r$ and then $a_3 \tilde{a}_3 \Vdash q * r$. As $a_3 \tilde{a}_3 = \tilde{a}_1$ we deduce $\tilde{a}_1 \Vdash q * r$. Moreover, we have $a_7 \Vdash p$ and, from $a_0 a_1 = \tilde{a}_1$ and $\sigma(a_1) = a_7$, we deduce that $a_0 a_7 = \tilde{a}_1$ and consequently $a_0 a_7 \Vdash q * r$ and $a_0 \Vdash p \multimap (q * r)$. To show that $\tilde{a}_0 \not\Vdash (p \multimap q) * (p \multimap r)$, we must show either $a_6 \not\Vdash p \multimap q$ or $\tilde{a}_6 \not\Vdash p \multimap r$ because $a_6 \tilde{a}_6 = \tilde{a}_0$. We have $a_{10} \Vdash p$ and thus $\tilde{a}_6 a_{10} \not\Vdash r$ since $\tilde{a}_6 a_{10} = \tilde{a}_{10}$ and $\tilde{a}_{10} \not\Vdash r$.

We have generated a countermodel of $(p \multimap (q * r)) \multimap ((p \multimap q) * (p \multimap r))$ from the labelled formula tree with existing connections before the failure of the construction attempts. Such a structure can be seen as a graphical representation of the countermodel like it is the case with a resource graph with the related tableau method in BI [7].

Our new connection method for MILL, that corresponds to a method of proof nets construction, is well-adapted to avoid several kinds of redundancy one has to deal with in more standard backward reasoning methods. In addition, it allows to efficiently detect the non-provability of formulae because the initial labelled formula tree contains the necessary semantic information. It also generate a countermodel in case of non-provability. It is a key point of the approach to say that such labelled proof structures or nets eliminate some bureaucracy from deductive systems but also are central structures with enough semantics in order to generate either proofs or countermodels.

References

1. M. Abrusci and P. Ruet. Non-commutative logic I : the multiplicative fragment. *Annals of Pure and Applied Logic*, 101:29–64, 2000.
2. W. Bibel. On matrices with connections. *Journal of ACM*, 28(4):633–645, 1981.
3. C. Faggian and M. Hyland. Designs, disputes and strategies. In *16th Int. Workshop on Computer Science Logic, CSL 2002, LNCS 2471*, pages 442–457, September 2002. Edinburgh, Scotland.
4. D. Galmiche. Connection Methods in Linear Logic and Proof nets Construction. *Theoretical Computer Science*, 232(1-2):231–272, 2000.
5. D. Galmiche and D. Méry. Connection-based proof search in propositional BI logic. In *18th Int. Conference on Automated Deduction, CADE-18, LNAI 2392*, pages 111–128, 2002. Copenhagen, Denmark.
6. D. Galmiche and D. Méry. Semantic labelled tableaux for propositional BI without bottom. *Journal of Logic and Computation*, 13(5):707–753, 2003.
7. D. Galmiche and D. Méry. Resource graphs and countermodels in resource logics. In *Int Workshop on Disproving: Non-theorems, Non-validity, Non-Provability*, pages 59–75, Cork, Ireland, July 2004.
8. D. Galmiche, D. Méry, and D. Pym. Resource Tableaux (extended abstract). In *16th Int. Workshop on Computer Science Logic, CSL 2002, LNCS 2471*, pages 183–199, September 2002. Edinburgh, Scotland.
9. D. Galmiche and J.M. Notin. Connection-based Proof Construction in Non-commutative Logic. In *10th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR'03, LNCS 2850*, pages 422–436, September 2003. Almaty, Kazakhstan.
10. D. Galmiche and G. Perrier. Foundations of proof search strategies design in linear logic. In *Logic at St Petersburg '94, Symposium on Logical Foundations of Computer Science, LNCS 813*, pages 101–113, St Petersburg, Russia, July 1994.
11. J.Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.
12. S. Ishtiaq and P. O’Hearn. BI as an assertion language for mutable data structures. In *28th ACM Symposium on Principles of Programming Languages, POPL 2001*, pages 14–26, London, UK, 2001.
13. C. Kreitz and H. Mantel. A matrix characterization for multiplicative exponential linear logic. *Journal of Automated Reasoning*, 32(2):121–166, 2004.
14. C. Kreitz and J. Otten. Connection-based theorem proving in classical and non-classical logics. *Journal of Universal Computer Science*, 5(3):88–112, 1999.
15. D. Méry. *Preuves et Sémantiques dans des Logiques de Ressources*. PhD thesis, Université Henri Poincaré, Nancy 1, 2004.
16. A. S. Murawski and C.-H. L. Ong. Fast verification of MLL proof nets via IMLL. *ACM Transactions on Computational Logic*, 2004. To appear.
17. P.W. O’Hearn and D. Pym. The Logic of Bunched Implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.
18. D.J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002.
19. D.J. Pym, P.W. O’Hearn, and H. Yang. Possible worlds and resources: The semantics of BI. *Theoretical Computer Science*, 315(1):257–305, 2004.
20. J. Reynolds. Separation logic: A logic for shared mutable data structures. In *IEEE Symposium on Logic in Computer Science*, pages 55–74, Copenhagen, Denmark, July 2002.
21. A. Urquhart. Semantics for Relevant Logic. *Journal of Symbolic Logic*, 37:159–169, 1972.
22. L.A. Wallen. *Automated Proof search in Non-Classical Logics*. MIT Press, 1990.

Purity through Unravelling

Robert Hein and Charles Stewart

Technische Universität Dresden, Germany

Abstract. We divide attempts to give the structural proof theory of modal logics into two kinds, those *pure formulations* whose inference rules characterise modality completely by means of manipulations of boxes and diamonds, and those *labelled formulations* that leverage the use of labels in giving inference rules. The widespread adoption of labelled formulations is driven by their ability to model features of the model theory of modal logic in its proof theory.

We describe here an approach to the structural proof theory of modal logic that aims to bring under one roof the benefits of both the pure and the labelled formulations. We introduce two proof calculi, one labelled sequent formulation and one pure formulation in the calculus of structures that are shown to be in a systematic correlation, where the latter calculus uses deep inference with *shaped modal rules* to capture in a pure manner the manipulations that the former calculations mediates through the use of labels.

We situate this work within a larger investigation into the proof theory of modal logic that solves problems that existed with the earlier investigation based on *prefix modal rules*. We hold this development provides yet stronger evidence justifying the claim that good, pure proof theory for modal logic needs deep inference.

1 Introduction

Modal logic is an essential part of both computational logic and philosophical logic, for reasons among which we bring attention to:

1. Modalities allow parts of propositions to be marked as having different semantics to other parts, for example in the way that various flavours of linear logic use the "!" and "?" modalities to indicate where structural identities are valid;
2. Modal propositional logic allows increased expressivity over classical propositional logic, but in a controlled manner allowing many useful, decidable languages to be formulated, by contrast to the situation in predicate logic;
3. Modal logic captures a notion of locality that has proven especially useful in the theory of concurrency, due to the close relationship of model invariance and bisimulation, a closeness that has resulted in the fundamental characterisation of bisimulation in Hennessy-Milner logic;
4. Classical modal logic is well suited to algebraic semantics, since if we regard classical propositional logic as receiving its most fundamental semantics in

terms of Boolean algebras, modalities are then modelled by operators on these algebras. Hence if we are interested in the relationship between logic and algebra, modal logic provides good soil.

However, proof theory has been the least successful part of the investigation into modality, by comparison with the success the parallel investigations into the model theory by means of Kripkean frame semantics, Jónsson-Tarski algebraic semantics, and characterisations of decision procedures using tableaux algorithms.

Phiniki Stouppa's work on the proof theory of S5 [Sto04], provides a survey of calculi used to express theories of modal logic and classifies them into two types: those that attempt to give direct formulations of inference rules using box and diamond, which we here call *pure formulations* and those *labelled formulations* that leverage the use of labels in giving inference rules, which allow them to model features of frame semantics in proof theory. She observes that all of the pure formulations that are capable of giving cut-free characterisations of S5 need to use forms of inference less shallow than those of Gentzen's sequent calculus. Her survey concentrated on describing the pure formulations, which were, with the exception of display logic, observed to achieve a fairly low level of generality: that is to say, most of the calculi proposed could give cut-free characterisations of only a few modal logics. The joint paper of Stewart and Stouppa advanced the hypothesis that non-shallow inference is necessary for cut-free pure systems that express S5 [SS04].

Deep inference is a property of inference systems that is sharper than merely being non-shallow, representing the possibility of performing inferences at any depth. It is not immediate to observe that display logic, like the calculus of structures, is a system of deep inference, however it follows from regarding the reversible structural rules as defining an equivalence class of inferential judgements that allow the logical inferences to be applied to formulae occurring in structures of arbitrarily complex relationships to one another. This view is consistent with the algebraic semantics of display logic as structads of Lamarche [Lam01].

Labelled formulations have captured more modal logic systems, and have described them in a way that has a stronger intuitive connection with the model theory and algebraic semantics of modal logic than pure formulations. However, pure formulations retain their interest, for a number of reasons:

- Following Avron, we may say that one of reasons proof theory is valuable is that it expresses logical concepts in a manner independent from a given model theory [Avr01]. This issue is essential to the idea of proof-theoretic semantics;
- Pure characterisations tend to wear the locality of modal logic on their sleeves, as it were, and so we may hope that they are a better source of intuitions for applications in concurrency and in the war against bureaucracy;
- Labels provide a very strong structural glue, similar to the predicate calculus in expressivity, and so we may be concerned about an issue dual to that

which led to Guglielmi’s characterisation of mismatch [Gug03]. Mismatch, as Guglielmi described it, occurs when the structural glue of a certain proof calculus is not strong enough to characterise cut-free characterisations of certain logics; here, we have that the glue is extremely strong relative to the modelled logics, and the resulting concern is that the expressions of the logic are not constrained to be natural, in the way that Gentzen’s characterisations of classical propositional logic are widely viewed to be natural.

We will outline here an approach to the structural proof theory of modal logic that aims to combine the benefits of both the pure and the labelled formulations. We introduce two classes of proof systems, one labelled and one pure for what we call the *3/4-Scott-Lemmon modal logics*, which are those normal modal logics characterised by axioms of the shape $\diamond^h \Box^i A \supset \Box^j A$ (or equivalently, of shape $\diamond^j A \supset \Box^h \diamond^i A$):

1. A reformulation of Alex Simpson’s graph-structured labelled sequent calculus where the labels are restricted to be tree-structured, a change which can be regarded an *unravelling* of the graph-structured calculus, by analogy to the standard technique of unravelling used in the model theory of modal logic.
2. A proof calculus with deep inference rules, given as an extension of Brünnler’s system **SKS** of classical logic in the calculus of structures. We call this the *shaped modal rule calculus*, or just *shaped calculus* by contrast to the earlier calculus of Stewart and Stouppa based on *prefixed modal rules*.

The significance of these calculi is established by a number of results and one conjecture:

1. It is shown that, with cut, both of these calculi are complete with respect to provability of the various systems they are intended to capture.
2. Cut-elimination for the tree-structured calculus is conjectured; some evidence for this conjecture is discussed in [Hei05] and in an extended version of this paper [HS05]. The problems are named and possible solutions suggested.
3. A structurally recursive transformation from proofs in the tree-structured sequent calculus to proofs in the shaped calculus is given, where the tree structure of labels maps directly onto the hierarchical shape of structures in the calculus of structures. This map does not introduce cuts, hence we obtain cut-elimination for the shaped calculus as a corollary.

More broadly, we can say that the formulation of the tree-structured calculus shows that the methodology of unravelling, used as a part of the model-theoretic proof of the Goldblatt–Thomason theorem [BdRV01], has a proof-theoretic application, while the translation shows that tree-structured use of labels can be captured in a calculus of deep inference; the combination of these claims we can summarise by means of the slogan *deep inference can express labels purely*.

The next section introduces the calculus of structures for classical logic and the basic normal modal logic K. Section 3 describes the prefixed modal characterisation of the so-called systems of the cube. Section 4 gives a graph-structured

calculus characterisation essentially the same as that given in Alex Simpson's PhD thesis. Section 5 describes the unravelling intuition, the tree-structured calculus and the shaped modal calculus.

2 The Basic Modal Logic in the Calculus of Structure

The *calculus of structures* (CoS) is a proof formalism that can be seen as a generalisation of Gentzen's sequent calculus. In sequent calculus, rules translate formula syntax into proof structure, when seen bottom-up. For instance, conjunction is translated into proof tree branching and disjunction into the multiset comma. In CoS this distinction between formulae and proof structure disappears, both receive the same syntactic treatment, resulting in what we simply call a *structure*. Further, CoS enjoys *deep inference*, i.e. proof rules in CoS do not apply just on the top connective, but anywhere inside a structure. Many rules known from the sequent calculus, like the identity axiom and the cut-rule, as well as the structural rules for weakening and contraction have closely corresponding rules in CoS. Brünnler [Brü04] gives a detailed account of the CoS in particular for classical propositional and predicate logic.

We will first present the system for the normal modal logic **K**. It is a conservative extensions of Brünnler's propositional system. The design of the additional pair of rules, $k \downarrow$ and $k \uparrow$, follows Guglielmi's recipe [Gug02].

Definition 1. *The language of pre-structures the generated by the following syntax:*

$$S ::= a \mid \bar{S} \mid \Box S \mid \Diamond S \mid [S, \dots, S] \mid (S, \dots, S)$$

where a is a propositional variable and \Box and \Diamond are the usual modal operators. (\dots) and $[\dots]$ are conjunction and disjunction. Empty conjunction and disjunction are denoted as \mathbf{t} and \mathbf{f} respectively. The set of structures is the set of equivalence classes as defined by the smallest congruence relation that respects the equations of figure 1. A structure context $S\{ \ }$ is a structure with exactly one occurrence of the hole $\{ \ }$. That hole may not be in the scope of a negation. The empty context is just the hole. The structure $S\{R\}$ is a context $S\{ \ }$, where the hole has been replaced by the structure R . We drop the curly brackets if the hole is filled by a disjunction, i.e. $S[R, T] = S\{[R, T]\}$ and similar for conjunction.

Definition 2. *A derivation is a sequence of structures, such that a single structure S is a derivation, and, if $\frac{R}{S\{U\}}$ is a derivation and $\rho \frac{S\{U\}}{S\{V\}}$ is an instance of*

some given rule, then $\frac{R}{S\{U\}}$ is a derivation. Let \mathcal{D} be a derivation and $S\{ \ }$ a

context. Then the derivaton $S\{\mathcal{D}\}$, called \mathcal{D} in context $S\{ \ }$, is the derivation derived from \mathcal{D} by putting each structure of \mathcal{D} into the context $S\{ \ }$, leaving

Associativity	Commutativity	Identity
$(R, (T, U)) = (R, T, U)$	$(R, T) = (T, R)$	$(R, \mathbf{tt}) = R$
$[R, [T, U]] = [R, T, U]$	$[R, T] = [T, R]$	$[R, \mathbf{ff}] = R$
DeMorgan duality		
$\overline{\mathbf{tt}} = \mathbf{ff}$	$\overline{(R, T)} = (\overline{R}, \overline{T})$	$\overline{\forall R} = \exists \overline{R}$
$\overline{\mathbf{ff}} = \mathbf{tt}$	$\overline{[R, T]} = (\overline{R}, \overline{T})$	$\overline{\exists R} = \forall \overline{R}$
$\overline{\square R} = \diamond \overline{R}$	$\overline{\diamond R} = \square \overline{R}$	
$[\mathbf{tt}, \mathbf{tt}] = \mathbf{tt}$	$(\mathbf{ff}, \mathbf{ff}) = \mathbf{ff}$	$\square \mathbf{tt} = \mathbf{tt}$
$\diamond \mathbf{ff} = \mathbf{ff}$		
Double negation		Congruence
$\overline{\overline{R}} = R$	if $R = T$, then $S\{R\} = S\{T\}$	
Variables (y is not free in R)		
$\forall xR = \forall yR[x/y]$ $\exists xR = \exists yR[x/y]$ $\forall yR = \exists yR = R$		

Fig. 1. Equations for structures.

$i \downarrow \frac{S\{\mathbf{tt}\}}{S[R, \overline{R}]}$	$i \uparrow \frac{S(R, \overline{R})}{S\{\mathbf{ff}\}}$
$k \downarrow \frac{S\{\square[R, T]\}}{S[\square R, \diamond T]}$	$s \frac{S([R, U], T)}{S[(R, T), U]}$
$w \downarrow \frac{S\{\mathbf{ff}\}}{S\{R\}}$	$w \uparrow \frac{S\{R\}}{S\{\mathbf{tt}\}}$
$c \downarrow \frac{S[R, R]}{S\{R\}}$	$c \uparrow \frac{S\{R\}}{S(R, R)}$
$k \uparrow \frac{S(\diamond R, \square T)}{S\{\diamond(R, T)\}}$	

Fig. 2. System **SKS – K**.

the order and the applied rules in place, i.e for some $\mathcal{D} = \frac{R}{T}$ and $S\{ \}$ we get $S\{\mathcal{D}\} = \frac{S\{R\}}{S\{T\}}$. A proof is a derivation that starts with \mathbf{tt} .

Figure 2 depicts the rules for the CoS system for modal logic **K**. We call the system **SKS – K**. The system is *symmetric*, i.e. it is divided into up and down fragment. Two rules which share the name but have opposing arrows form a *dual* pair. They are in this simple syntactic relation:

$$\rho \downarrow \frac{S\{U\}}{S\{V\}} \text{ is dual to } \rho \uparrow \frac{S\{\overline{V}\}}{S\{\overline{U}\}}$$

which is of course symmetric. So, the *identity axiom* $i \downarrow$ is dual to the *cut rule* $i \uparrow$. Rules $w \downarrow$ and $w \uparrow$ are called *weakening* and *co-weakening*, while $c \downarrow$ and $c \uparrow$ are *contraction* and *co-contraction* respectively. The *switch-rule* s is dual to itself and is therefore written without arrows. The rule $k \downarrow$ resembles the **axiom K**:

$$\square(A \supset B) \supset (\square A \supset \square B)$$

Its dual is $k \uparrow$. Having this symmetry, all derivations can be flipped upside down, and with all structures negated, this results in the *dual derivation* where all instances of rules are replaced by their dual in reverse order.

In CoS, showing the admissibility of the up fragment, corresponds to cut-elimination. Due to the duality, it is a straightforward exercise to remove any up-rule as long as the cut-rule $i \uparrow$ itself remains available.

This characterisation of the basic normal modal logic \mathbf{K} provides the common basis point for how both the earlier prefix modal and the shaped modal approach introduced in this paper axiomatise modal theories in the calculus of structures, and we regard the treatment here as a canonical expression of this modal logic in the calculus of structures, because of the naturality of the three design elements of the logic:

1. Modalities are regarded as unary operators on propositions that may freely be regarded as determining the contexts in which inference rules may be applied. This is the most free, and thus most natural way, in which we may interpret the abolition of the distinction between formula and structure that characterises the calculus of structures;
2. The necessitation rule, from $\vdash \phi$ infer $\vdash \Box\phi$, can be captured by the equation $\mathbf{tt} = \Box\mathbf{tt}$ in the presence of the above characterisation of deep inference. This characterisation is preferred, since we attempt to put propositional biequivalences that can be captured by linear equations in the equational theory.
3. The $k \downarrow$ rule is the inference rule that was determined as we said, by the so-called recipe, which determines how pairs of structure-forming connectives in duality are related. Together with the specification of dual structures and the rules for switch and cut, it is the recipe that characterises how duality is handled in the calculus of structures.

Hence, we consider this characterisation to be essentially the right characterisation of \mathbf{K} in the calculus of structures.

3 Prefix Modal Rules

By contrast to this, the methodology of the calculus of structures does not obviously determine how modalities other than \mathbf{K} are to be given. The prefix modal characterisation was the first approach to be studied, according to the dictum that one should begin by tackling the simplest approach that could possibly work. It follows from the observation that many of the most studied axioms for modal logic can be presented in the form:

$$\mu\phi \Longrightarrow \nu\phi$$

where μ and ν are possibly empty sequences of modalities, for example the modal logic $\mathbf{K4}$ is characterised in the Hilbert system by adding the 4 axiom $\Box\phi \Longrightarrow \Box\Box\phi$ to the system characterising \mathbf{K} . More examples of axioms of this form are listed in figure 5, together with two important axioms that do not take this form, namely the axiom $\Box M$ and the Gödel-Löb axiom.

Given an axiom of this form, one can realise it as a rule in one of two possible ways, either by adding the down rule:

$$\frac{S\{\mu R\}}{S\{\nu R\}}$$

or its converse as the up rule:

$$\frac{S\{\bar{\nu}R\}}{S\{\bar{\mu}R\}}$$

The choice of which rule to adopt is not free, since we want the set of down rules to be cut-free. Stewart and Stouppa [SS04] studied the ‘cube’ of modal logics obtained from the 32 axiomatisations of modal logic which correspond to the possible subsets of the set of axioms $\{D, T, 4, B, 5\}$, including the six most intensively studied modal logics:

1. **K** itself;
2. **D**, obtained by extending system **K** with rule **D**;
3. **M**, obtained by extending system **K** with rule **T**: note this system is often named **T**, a nomenclature we avoid to prevent confusion with Gödel’s system **T**.
4. **S4**, obtained by extending system **M** by rule **4**;
5. **B**, obtained by extending system **M** by rule **B**;
6. **S5**, obtained by extending system **M** by rule **5**, or equivalently by extending system **M** by rules **B** and **4**.

The system there gives the orientations of the rules as follows:

$$d \frac{S\{\Box R\}}{S\{\Diamond R\}} \quad t \downarrow \frac{S\{\Box R\}}{S\{R\}} \quad 4 \downarrow \frac{S\{\Diamond \Diamond R\}}{S\{\Diamond R\}}$$

$$b \downarrow \frac{S\{\Diamond \Box R\}}{S\{R\}} \quad 5 \downarrow \frac{S\{\Diamond \Box R\}}{S\{\Box R\}}$$

Note that the **d** rule is self-dual, and so needs no choice. The choice for the rule **5** in fact changed from the earliest presented system to the choice above, and Hein observed that the converse of the rule **4** is more robust [BS04]. The methodology for choosing a orientation is rather ad-hoc, based on testing possible combinations of rival orientations, and seeing if one can either show cut-elimination (in that paper, by model-theory or by cut-freeness preserving translation) or find a counter-example. Due to the huge range of possible combinations, and the difficulty of finding general cut-elimination arguments, this approach ran out of steam.

4 Labelled Modal Logic

We now present a labelled sequent calculus that is based on Alex Simpson’s PhD thesis [Sim93]. In fact, we only changed the formalism from intuitionistic to classical logic, a move whose correctness was a design goal of Simpson. The idea

is that, after putting label to formulae, we can use a directed graph (on labels) to represent the modal (or intensional) relationship between all the formulae in a sequent. Of course, a directed edge between two labels is nothing but a binary first-order predicate (the accessibility relation!). If we now think of labels as possible worlds and interpret labelled formulae as some sort of valuation we shall recognise that Simpson took Kripke semantic verbatim and mixed it up with proof theory.

Note that there are plenty of other labelled systems in the literature whose labelling algebras try to implement Kripke semantics. None of them is as simple and as useful to proof theory as Simpson's.

It is a good practise in sequent calculus to have a designated rule for each logical connective. This is troublesome for the modalities. Take the following candidate of a \Box -rule in a one-sided system, ignoring any context for now:

$$\Box \frac{\vdash A}{\vdash \Box A}$$

This rule is too strong, as it does not take the intensionality of \Box into account. The system designer has to *somehow* drag along the context in the right way, depending on the particular modal logic. Since we employ Kripke semantics, we can have a rule that exactly characterises the meaning of $\Box A$, namely that "A holds at all accessible worlds" regardless of context

$$\Box \frac{x \triangleright y \vdash \Delta, y:A}{x \vdash \Delta, x:\Box A}$$

Read $x \triangleright y$ as *y is accessible from x* and $y:A$ as *A holds at y*. Sequents are one-sided, however the directed graph that occupies the left side can be seen a premise or guard when reading the sequent. Consequently, y is not meant to be a particular "possible world" but any accessible. For instance, read the premise of the above \Box -rule instance as: *Given an arbitrary Kripke model, if $x \triangleright y$ maps onto the model, then A is true at y under the same mapping.*

Here, we shall not further deal with the details of labelled sequent semantics. Consequently we omit a soundness proof for the rules below. We will present syntactic side more formally. There is some notation and other changes compared to Simpson [Sim93], in particular the treatment of graphs, however they are not essential to the spirit:

Definition 3. A labelled sequent is a structure $G \vdash \Delta$, where G is a (directed) graph and Δ is a multiset of labelled modal formulae. Here, a graph is a finite set of binary relations on variable labels $x \triangleright y$. We call a graph empty, if it does not contain any relations. However, an empty graph must still contain a singleton node or label. We usually write \emptyset for the empty graph instead of the singleton label. Which label that is, can always be inferred from context. Consider a sequent $G \vdash \Delta$. For any $x:A \in \Delta$, x must occur in G , or if G is empty, x must be its singleton node. The graph must be rooted, i.e. there must be a label from which every other label can be reached by the transitive-reflexive closure of the graph. Such a label is called the root. Roots do not have to be unique.

Given a set of labelled sequent rules $\frac{G_1 \vdash \Delta_1 \quad \dots \quad G_n \vdash \Delta_n}{G \vdash \Delta}$ for $n \geq 0$, a derivation is a tree of labelled sequents, where each node is an instance of a rule.

$$\begin{array}{c}
\text{Ax} \frac{}{G \vdash x:A, x:\neg A} \qquad \qquad \qquad \text{T} \frac{}{G \vdash x:\top} \\
\text{weak} \frac{G \vdash \Delta}{G \vdash \Delta, x:A} \qquad \qquad \qquad \text{contr} \frac{G \vdash \Delta, x:A, x:A}{G \vdash \Delta, x:A} \\
\wedge \frac{G \vdash \Delta, x:A \quad G \vdash \Delta, x:B}{G \vdash \Delta, x:A \wedge B} \\
\text{V}_L \frac{G \vdash \Delta, x:A}{G \vdash \Delta, x:A \vee B} \qquad \qquad \qquad \text{V}_R \frac{G \vdash \Delta, x:B}{G \vdash \Delta, x:A \vee B} \\
\text{□} \frac{G, x \triangleright y \vdash \Delta, y:A}{G \vdash \Delta, x:\text{□}A} \qquad \qquad \qquad \text{◇} \frac{G, x \triangleright y \vdash \Delta, y:A}{G, x \triangleright y \vdash \Delta, x:\text{◇}A} \\
\text{cut} \frac{G \vdash \Delta, x:A \quad G \vdash \Delta', x:\neg A}{G \vdash \Delta, \Delta'} \\
\text{geom} \frac{G, x_{11} \triangleright x'_{11}, \dots, x_{1k_1} \triangleright x'_{1k_1} \vdash \Delta \quad \dots \quad G, x_{n1} \triangleright x'_{n1}, \dots, x_{nk_n} \triangleright x'_{nk_n} \vdash \Delta}{G, x_1 \triangleright x'_1, \dots, x_m \triangleright x'_m \vdash \Delta}
\end{array}$$

For □, y must not occur in the conclusion.
For geom, none of the variables in $\mathbf{y}_i, 1 \geq i \geq n$ may occur in the conclusion.
(\mathbf{y}_i are the ones that are existentially quantified in the corresponding first-order condition.)

Fig. 3. System **GS1g – K** with cut and geom rule scheme.

A proof of a modal formula A , is a derivation ending in $\emptyset \vdash x:A$, whose leafs are instances of zero-premise rules.

Now, have a look at the top half of figure 3. Note, that once one ignores graphs and labels, it shows the Gentzen-Schütte system with explicit structural rules **GS1** for classical propositional logic. Note, how adding graphs and labels does not affect the propositional rules. Remember how the □-rule precisely encodes the meaning of □ in Kripke semantics. The ◇-rule does the same for ◇. Here, the relation $x \triangleright y$ remains in the conclusion because ◇ A means that only at some accessible world y formula A holds and other y might exist. Adding the two modal rules yields a system for modal logic **K**, that we consequently call **GS1 – K**.

How can we characterise modal logics stronger than **K**, say **S4** or **S5**? We systematically extend the system with certain instances of the geom rule scheme. Read on, on how it works, the motivation will follow shortly. Observe, how geom regards individual relations of the graph, i.e. works on the graphs "atomic" level and leaves the formula context intact. As such, geom is able to encode first-order conditions on directed graphs, or better now, on Kripke frames. Formally, given a first-order condition ρ :

$\text{T} \frac{G, x \triangleright x \vdash \Delta}{G \vdash \Delta}$	$\text{B} \frac{G, x \triangleright y, y \triangleright x \vdash \Delta}{G, x \triangleright y \vdash \Delta}$
$4 \frac{G, x \triangleright y, y \triangleright z, x \triangleright z \vdash \Delta}{G, x \triangleright y, y \triangleright z \vdash \Delta}$	$5 \frac{G, x \triangleright y, x \triangleright z, y \triangleright z \vdash \Delta}{G, x \triangleright y, x \triangleright z \vdash \Delta}$
$.3 \frac{G, x \triangleright y \vdash \Delta[y/z] \quad G, x \triangleright y, x \triangleright z, y \triangleright z \vdash \Delta \quad G, x \triangleright y, x \triangleright z, z \triangleright y \vdash \Delta}{G, x \triangleright y, x \triangleright z \vdash \Delta}$	

Fig. 4. Some instances of the **geom** rule scheme. See figure 5 for the corresponding axioms.

$$\forall \mathbf{x} P_1 \wedge \dots \wedge P_m \supset \bigvee_j \exists \mathbf{y}_j Q_{j1} \wedge \dots \wedge Q_{jk_j}$$

there is a corresponding instance of the **geom** rule scheme:

$$\rho \frac{G, Q_{11}, \dots, Q_{1k_1} \vdash \Delta \quad \dots \quad G, Q_{n1}, \dots, Q_{nk_n} \vdash \Delta}{G, P_1, \dots, P_m \vdash \Delta}$$

with the proviso that none of the variables in \mathbf{y}_j may occur in the conclusion. The Q_{ij} and P_l are some binary relations $u \triangleright v$. To understand the correspondence, view **geom** as using two-sided sequents. The placement of the graph on the left, or negative side, was not accidental. Ignore the context Δ . Now, flip the rule upside down, i.e. make the contrapositive and get the first-order condition. The existential quantifiers come from the rules proviso.

We obtain a particular class of first-order formulae, called *geometric sequents* that play an important role in geometric logic, cf. Vickers [Vic93]. The proof theoretical interest in geometric sequents lies in the fact that any theory whose axioms are geometric sequents have a cut-free characterisation in sequent calculus. See Negri [Neg01] how in the first-order case, ordinary cut-elimination is easily extended to geometric theory. Simpson [Sim93] shows cut-elimination indirectly through normalisation as he is more focused on the natural deduction systems, while Hein [Hei05] shows that cut-eliminations extends easily for the labelled sequent calculus presented above.

Theorem 4. *The cut-rule is admissible for system **GS1** – **K** + **geom**.*

How does adding **geom**-rules give us logics other than **K**? Recall that in Hilbert systems we systematically get stronger systems by adding more axioms. The branch of modal logic known as correspondence theory tells us that many of these axioms correspond to first-order frame conditions, i.e. the resulting logic is semantically characterised by Kripke frames under the respective conditions. For instance, axiom **T** corresponds to

$$\text{reflexivity:} \quad \forall x. x \triangleright x$$

Figure 5 gives the frame conditions associated with other logics. As it happens, many of the interesting frame conditions are geometric, including not only the aforementioned "cube" of modal logics, but all Scott-Lemmon axioms and more, and all of these can be captured by instances of **geom**; figure 4 gives examples.

To characterise for instance the logic **S5**, we add the rules **T** and **5** to system **GS1 – K**. For logic **M** we just add rule **T**.

As an example, see the this proof in **GS1 – K + T** of $\Box(\Diamond\neg A \vee A)$, which is a theorem of **M**:

$$\begin{array}{c}
\text{Ax} \frac{}{x \triangleright y, y \triangleright y \vdash y: \neg A, y: A} \\
\Diamond \frac{}{x \triangleright y, y \triangleright y \vdash y: \Diamond \neg A, y: A} \\
\text{T} \frac{}{x \triangleright y \vdash y: \Diamond \neg A, y: A} \\
\vee_R \frac{}{x \triangleright y \vdash y: \Diamond \neg A, y: \Diamond \neg A \vee A} \\
\vee_L \frac{}{x \triangleright y \vdash y: \Diamond \neg A \vee A, y: \Diamond \neg A \vee A} \\
\text{contr} \frac{}{\Box \frac{}{x \triangleright y \vdash y: \Diamond \neg A \vee A} \\ \emptyset \vdash x: \Box(\Diamond \neg A \vee A)}
\end{array}$$

Of course, not all normal modal logics can be characterise by geometric conditions, some even require second-order conditions. Also, not every geometric condition characterises a normal modal logic. While reflexivity corresponds to logic **M**, its negation irreflexivity: $\forall x(x \triangleright x \supset \perp)$, which is also a geometric condition, is not modally definable.

Unfortunately, with the inclusion of first-order machinery, labelled sequent calculus becomes a not conceptually pure formalism. As hinted in the previous paragraph, it allows some weird systems that correspond to non-modal logics, but even within derivations for modal systems, sequents may occur that can not be described using a modal formula. A well known tool from modal logic, the *standard translation* $ST_x(_)$, defines a satisfiability preserving map from modal formulae into first-order formulae. It so characterises which first-order formulae can be represented modally:

$$\begin{aligned}
ST_x(\Box A) &= \forall y(x \triangleright y \supset ST_y(A)) \\
ST_x(\Diamond A) &= \exists y(x \triangleright y \wedge ST_y(A))
\end{aligned}$$

The propositional connectives are mapped in the obvious way. Atomic formulae are mapped to unary predicates. The translation of \Box is the important bit. Note, how the case for \Box resembles a labelled sequent: $x \triangleright y \vdash x: A$. Unfortunately, standard translation is not surjective, i.e. there are first-order formulae that do not correspond to a modal formula. For instance, a formula $z \triangleright z \supset \phi$ can never fit the scheme, since in $x \triangleright y$ the variables x and y must be different. Let us consider an example to see what works. Take the formula $\Box(\Box A \vee \Box B)$ and apply the standard translation recursively. We get:

$$ST(\Box(\Box A \vee \Box B))_x = \forall y(x \triangleright y \supset \forall z(y \triangleright z \supset Bz) \vee \forall u(y \triangleright u \supset Au))$$

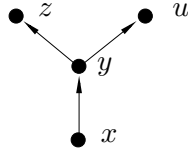
Since we did not translate any \Diamond we did not generate existential quantifiers. consequently all the variables are essentially free, i.e. only in the scope of universal quantifiers. Dropping the quantifiers and applying some logical manipulations, we get an equivalent formula:

$$x \triangleright y \wedge y \triangleright z \wedge y \triangleright u \supset ST_z(A) \vee ST_u(B)$$

Observe that the relations $x \triangleright y, y \triangleright z, y \triangleright u$ in the premise, viewed as a directed graph, form a tree with root x :

3/4-Scott-Lemmon axioms		
T	$\Box A \supset A$	reflexive $\forall x.x \triangleright x$
B	$\Diamond \Box A \supset A$	symmetric $\forall xy.x \triangleright y \supset y \triangleright x$
4	$\Box A \supset \Box \Box A$	transitive $\forall xyz.x \triangleright y \wedge y \triangleright z \supset x \triangleright z$
5	$\Diamond \Box A \supset \Box A$	euclidian $\forall xyz.x \triangleright y \wedge x \triangleright z \supset y \triangleright z$
CD	$\Diamond A \supset \Box A$	unique $\forall xyz.x \triangleright y \wedge x \triangleright z \supset y = z$
C4	$\Box \Box A \supset \Box A$	dense $\forall xy.x \triangleright y \supset \exists z(x \triangleright z \wedge z \triangleright y)$
other Scott-Lemmon axioms		
D	$\Box A \supset \Diamond A$	serial $\forall x \exists y.x \triangleright y$
CR	$\Diamond \Box A \supset \Box \Diamond A$	confluency $\forall xyz.x \triangleright y \wedge x \triangleright z \supset \exists u(x \triangleright u \wedge y \triangleright u)$
non-Scott-Lemmon		
□M	$\Box(\Box A \supset A)$	shift-reflexive $\forall xy(x \triangleright y \supset y \triangleright y)$
GL	$\Box(\Box A \supset A) \supset \Box A$	Gödel-Löb

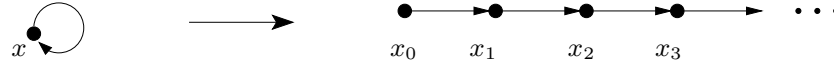
Fig. 5. Some modal axioms and their corresponding frame conditions.



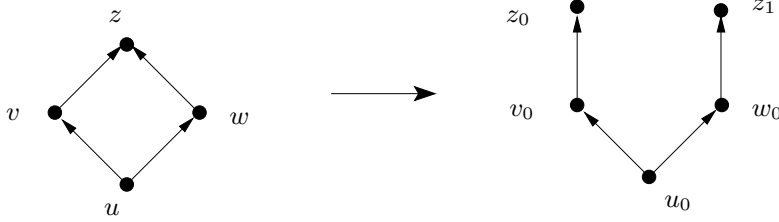
And in fact, it should be easy to see that the binary predicates, or accessibility relations, that appear during recursive application of the \Box case of $ST_x(-)$ must always form a tree. Consequently, if we want a conceptually pure labelled calculus we need to restrict the graphs to trees. Unfortunately, looking just at the examples in figure 4, they all introduce, when looking bottom up, relations to the graph that make it less tree-like. And in fact, through a naive restriction we would lose almost all interesting systems. We need to unravel!

5 Unravelling

We need to *unravel* the geometric rules. All other rules are blind with respect to a restriction to trees. What do we mean by unravelling? Unravelling or *unfolding* is a well know technique from modal model theory. It exploits a property of modal logic known as the *tree-model-property*, cf. e.g. Blackburn, et al. [BdRV01]. It says, that any satisfiable formula is also satisfiable in a finite model, whose underlying frame relation forms a tree. Unravelling is the procedure to build the tree-model out of the original. As hinted above, we can pretend our labelled sequents to be Kripke models. To unravel a graph, we need to pick a label from which every other label can be reach via some path. We call that label the *root*. In practice there is always such a root, since the rules only ever introduce new labels that succeed old ones. Unravelling generates for each possible finite path that starts at the root a label in the unfolded graph. Of course, cycles



a)



b)

Fig. 6. a) unfolding a single reflexive relation. b) unfolding a converging graph.

allow arbitrary long paths, hence unfolding will give us trees with infinite long branches. Remember, that graphs store a record of the course of the proof. Proofs are, of course, always finite objects, hence we will not need infinite branches. As unravelling may produce infinite trees, our new rules will unravel only a single step at once. A consequence is, that we may have to apply an unravelled rule many times compared to a single application for a "folded" rule, in what otherwise may essentially be the same proof.

Figure 6 only give some very simple examples of how directed graphs unravel. However, it should be obvious, that unravelling can produce highly complex results. System **GS1g** – **K** + **geom** covers logics corresponding to arbitrary geometric frame conditions. With what we call *3/4-Scott-Lemmon*, we have identified a class of geometric frame conditions and corresponding rules, that unravel simply enough for us to handle during this early stage of research. Yet this small fragment will allow us to formulate many interesting modal logics, such as the "cube": M, B, S4 and S5. Correspondence theory is the branch of modal logic that studies the relationship between modal logic and classical (first- and second-order) logic. An early correspondence result by Lemmon and Scott [LS77] states that for $h, i, j, k \leq 0$, modal formulae of type

$$\diamond^h \Box^i A \supset \Box^j \diamond^k A \quad (1)$$

correspond to the first-order frame condition

$$\forall w, v, u (w \triangleright^h v \wedge w \triangleright^j u \supset \exists x (v \triangleright^i x \wedge u \triangleright^k x)) \quad (2)$$

Notation: We write \Box^n for $\overbrace{\Box \cdots \Box}^{n\text{-times}}$, and similarly for \diamond^n . We write $x \triangleright^n y$ as abbreviation of $\exists x_1, \dots, x_{n-1} (x \triangleright x_0 \wedge x_0 \triangleright x_1 \wedge \cdots \wedge x_{n-1} \triangleright y)$ if $n \geq 1$, and $x = y$ if $n = 0$. Instead of actually using equations like $x = y$, we will commonly substitute y for x throughout the respective formula or sequent. Note also, that we can drop the existential quantifiers in the antecedent of (2) that are hidden

by the \triangleright^n notation in favour of universal quantification. For now, Scott-Lemmon axioms are still a bit too broad, we further restrict the class of axioms of the form (1) and (2) to those with $k = 0$.

$$\diamond^h \Box^i A \supset \Box^j A \quad (3)$$

corresponds to the first-order frame condition

$$\forall w, v, u (w \triangleright^h v \wedge w \triangleright^j u \supset v \triangleright^i u) \quad (4)$$

Still, most of the interesting modal logic axioms are captured. We will call this class of axioms *3/4-Scott-Lemmon*.

As an exception, we will additionally consider axiom **D**, which is of form (1) with $i = k = 1$ and $h = j = 0$. It simplifies in a convenient way to make it work with our technique. In fact, it is so simple that it does not actually need to unravel. We shall call the resulting labelled calculus **GS1 – K(t)** for logic **K**,

$3/4SL \frac{G, w \triangleright^h v, w \triangleright^j u, v \triangleright^i u' \vdash \Delta, u':\Delta'}{G, w \triangleright^h v, w \triangleright^j u \vdash \Delta, u:\Delta'}$	$D \frac{G, x \triangleright y \vdash \Delta}{G \vdash \Delta}$
<p>For D, y must not occur in the conclusion. For 3/4SL, u' and $v_k, 1 \leq k < i$ in $v \triangleright^i u'$ must not occur in the conclusion.</p>	

Fig. 7. System **GS1 – K(t)**: unraveled rule schemes 3/4SL and D.

where the (t) indicates the restriction to trees. Except for the geometric ones, its rules are identical to the unrestricted calculus, presented earlier. To generate systems for the stronger logics, we will add instances of the unravelled geometric rules instead. Figure 9 illustrates the unfolding in the case of 3/4-Scott-Lemmon. For each geometric condition of the form (4)

$$\rho : \forall wvu (w \triangleright^h v \wedge w \triangleright^j u \supset v \triangleright^i u)$$

that characterises the desired modal logic, add an instance of the scheme 3/4SL:

$$\rho \frac{G, w \triangleright^h v, w \triangleright^j u, v \triangleright^i u' \vdash \Delta, u':\Delta'}{G, w \triangleright^h v, w \triangleright^j u \vdash \Delta, u:\Delta'}$$

where, $u:\Delta'$ denotes a multiset of modal formulae which are all labelled with u . If the logic characterised by seriality: $\forall x \exists y. x \triangleright y$, then add the rule D to the system. Cf. figure 7. We write **GS1 – K(t)**⁺ for an arbitrary extension of **GS1 – K(t)**. We append the names of the respective rules if we need to name a particular system.

$T \frac{G, x \triangleright x' \vdash \Delta, x':\Delta'}{G \vdash \Delta, x:\Delta'}$	$B \frac{G, x \triangleright y, y \triangleright x' \vdash \Delta, x':\Delta'}{G, x \triangleright y \vdash \Delta, x:\Delta'}$
$4 \frac{G, x \triangleright y \triangleright z, x \triangleright z' \vdash \Delta, z':\Delta'}{G, x \triangleright y, y \triangleright z \vdash \Delta, z:\Delta'}$	$5 \frac{G, x \triangleright y, x \triangleright z, y \triangleright z' \vdash \Delta, z':\Delta'}{G, x \triangleright y, x \triangleright z \vdash \Delta, z:\Delta'}$

Fig. 8. Some instances of the 3/4SL rule scheme. See figure 4 for their "folded" cousins.

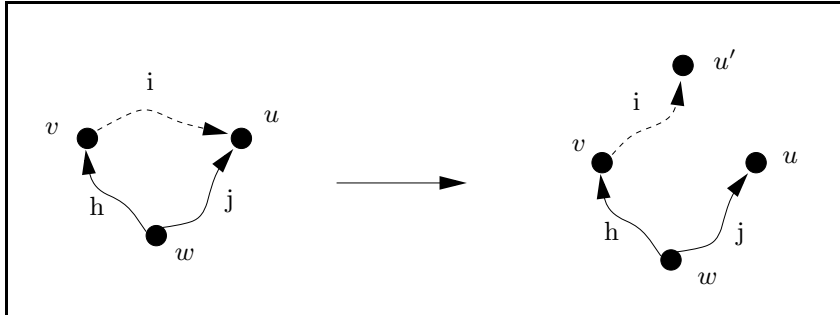


Fig. 9. Unravelling 3/4-Scott-Lemmon frame conditions.

Of course, unravelling comes at a price. We lose the easy cut-elimination argument we had due to the geometric origin of our rules. The reason is that unravelling duplicates labels, with new labels being semantically identical to old ones. Accordingly, in the 3/4SL-rule scheme, cf. figure 7, it is label u that gets replicated as u' . Because the graphs are trees now, we have no direct machinery available to refer from one to the other. To nevertheless characterise that they are meant to be the same "possible world," we may change labels of some formulae from u' to u , when viewed top down. Unravelled geometric rules can not leave the formulae context untouched, which was crucial for easy cut-elimination.

6 Forget about Labels, Use Deep Inference!

Thanks to unravelling, we get this:

Proposition 5. For any $\mathbf{GS1t} - \mathbf{K(t)}^+$ derivation $\begin{matrix} G_1 \vdash \Delta_1 \dots G_n \vdash \Delta_n \\ \vdots \mathcal{D} \\ G \vdash \Delta \end{matrix}$, where $n \geq 0$, if G is a tree, then every graph occurring in \mathcal{D} is a tree with the same root.

Proof. By inspection of the rules of $\mathbf{GS1t} - \mathbf{K(t)}^+$. No relations ever get added to the graph going top down in a derivation. Any new labels added going up are fresh, so the graph remains a tree. \square

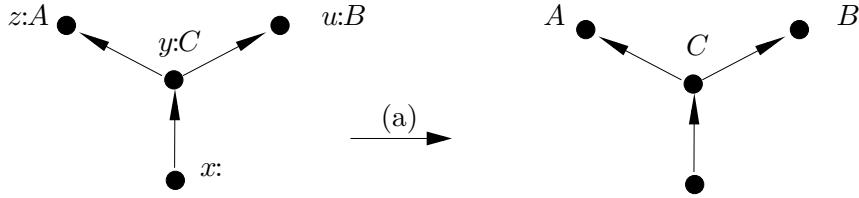
We claimed that $\mathbf{GS1} - \mathbf{K(t)}^+$, the unravelled labelled calculus was conceptually pure, i.e. every step of a derivation could be represented using a modal formula. This can in fact be done by applying the inverse standard translation to each sequent. The technicalities are available at [Hei05]. We will make an informal argument considering as example the sequent

$$x \triangleright y, y \triangleright z, y \triangleright u \vdash z:A, u:B, y:C.$$

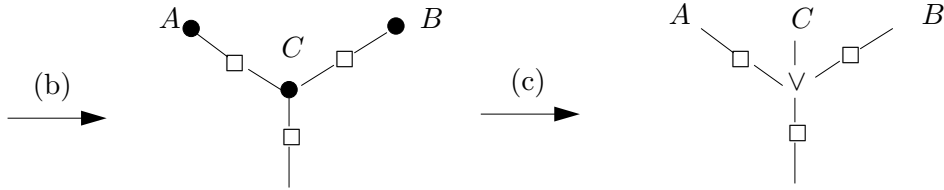
We will change this representations stepwise. Let us draw the graph, which the proposition assures to always be a tree. Then, we attach each labelled formula to the node that carries its label. As we can see, labels have become unnecessary, so we remove them (a):

$$\begin{array}{c}
\text{h} \downarrow \frac{S(\Box R, \Box T)}{S\{\Box(R, T)\}} \qquad \text{h} \uparrow \frac{S\{\Diamond[R, T]\}}{S\{\Diamond R, \Diamond T\}} \\
3/4\text{SL} \downarrow \frac{S\{\Box[R_h \Box[R_{h-1}, \Box \dots \Box[R_1, \Box^i T] \dots]]\}}{S[\Box[R_h, \Box[R_{h-1}, \Box \dots \Box[R_1] \dots]], \Box^j T]} \quad \text{D} \downarrow \frac{S\{\Box \mathbf{ff}\}}{S\{\mathbf{ff}\}} \\
3/4\text{SL} \uparrow \frac{S\{\Diamond(R_h, \Diamond(R_{h-1}, \Diamond \dots \Diamond(R_1, \Diamond^j T) \dots))\}}{S(\Diamond(R_h, \Diamond(R_{h-1}, \Diamond \dots \Diamond(R_1) \dots)), \Diamond^i T)} \quad \text{D} \uparrow \frac{S\{\mathbf{tt}\}}{S\{\Diamond \mathbf{tt}\}}
\end{array}$$

Fig. 10. Additional rules for system **SKS** – **K**⁺



According to the standard translation, relations, i.e. the arrows, correspond to boxes, so let us replace them (b):



In the original sequent, the formulae A , B and C were in a disjunctive relationship. A and B are now behind boxes, but the propositional relationship remains. If we put them in disjunction accordingly (c), we get a modal syntax tree of the following formula, which is equivalent to the original sequent:

$$\Box(\Box A \vee \Box B \vee C)$$

Observe this: Rules of the sequent calculus that were able to apply to the original sequent can not apply anymore to the result of this transformation. The formulae A , B and C that were exposed in the sequent are now hidden deep inside a larger formula. What we now need is *deep inference*!

We now return to the calculus of structures. All we need to do to transform whole derivations is to map rules of the labelled sequent calculus **GS1** – **K(t)**⁺ to derivations in CoS. The needed CoS system will be based on **SKS** – **K** that was introduced in section 2.

In [Brü04] Brünnler translates from the propositional sequent calculus to CoS. Since both **GS1** – **K(t)**⁺ and **SKS** – **K** are conservative extensions of propositional systems, we only need to extend Brünnler's method to the modalities and the geometric rules. To get a system equivalent to **GS1** – **K(t)**⁺ we

$\top \downarrow \frac{S\{\Box T\}}{S\{T\}}$	$\mathbf{B} \downarrow \frac{S\{\Box[R, \Box T]\}}{S[\Box R, T]}$
$4 \downarrow \frac{S\{\Box T\}}{S\{\Box\Box T\}}$	$5 \downarrow \frac{S\{\Box[R, \Box T]\}}{S[\Box R, \Box T]}$

Fig. 11. Instances of scheme 3/4SL \downarrow for **T,B,4,5**.

need to add the rules of figure 10 to **SKS** – **K**. We name the system **SKS** – **K**⁺. The translation to CoS does not introduce any new cuts. A detailed technical account can be found at [Hei05]. The following presentation will be less formal. Translation in the case of the \Box -rule is trivial. Both, premise and conclusion map to the same structure (or formulae), so the rule maps to identity and disappears in CoS.

$$\Box \frac{x \triangleright y \vdash y:A}{x \vdash x:\Box A} \quad \Longrightarrow \quad = \frac{\Box A}{\Box A}$$

The \Diamond -rule maps to the **k** \downarrow -rule:

$$\Diamond \frac{x \triangleright y \vdash y: _ , y:A}{x \triangleright y \vdash y: _ , x:\Box A} \quad \Longrightarrow \quad \mathbf{k} \downarrow \frac{\Box[_ , A]}{[\Box _ , \Diamond A]}$$

A notable anomaly occurs in the translation of the conjunction rule with essentially corresponds to the switch. Here, and only here, we need a new rule **h** \downarrow in order to distribute boxes over conjunctions. No similar rule is needed for first-order predicate logic, nor do Stewart and Stouppa [SS04] need it. Finally, the 3/4SL rule scheme maps to 3/4SL \downarrow . See figure 11 for some instances of the 3/4SL \downarrow scheme. They remain simple, despite the complexity of the scheme, as the parameter *h* is 0 or 1 for the interesting cases.

Theorem 6. *For any derivation in **GS1** – **K**⁺ with cut, there exists a derivation in **SKS** – **K**⁺ with the same number of cuts and equivalent premise and conclusion.*

7 Conclusion

This paper has:

1. Argued for the desirability of approaches to modal logic that can move between pure and labelled formalisation;
2. Provided a labelled sequent calculus formalism that can be mapped in a straightforward, cut-freeness preserving manner onto a pure formalism in the calculus of structures;
3. Situated this research within an ongoing research investigation into the characterisation of modal logic in calculi with deep inference, in particular it is observed that the proof-theoretic embedding depends in a strong manner on arbitrary deepness of inference.
4. An extended version of this paper [HS05] provides grounds for the conjecture that the two formalisms are cut-free.

Naturally, this achievements would be strengthened by settling properly the status of cut-elimination for the systems. Furthermore, the value of proof theories of modal logic that span labelled and pure formalisations would be increased if we were to have a deeper connection between them, such as a reverse mapping from the pure formalisation to the labelled whose compositions with the given mapping conserved useful structural properties.

Lastly, the prefix modal programme need not be considered a dead end. The properly displayed rules characterising many axioms of modal logic introduced by Kracht for display logic [Kra96] can be regarded a prefix modal rules, and so examining the relationship between the calculus of structures and display logic has particular value.

References

- [Avr01] A. Avron. The method of hypersequents in the proof theory of propositional non-classical logics. In W. Hodges et al. (eds.), *Logic: From Foundations to Applications*, pages 1–32. Oxford University Press, 1996.
- [BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*, 2001.
- [Brü04] Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. Logos Verlag, Berlin, 2004.
- [BS04] Robert Hein. *Counterexample*.
URL <http://alessio.guglielmi.name/res/cos/cex.html>, 2004.
- [Gug02] Alessio Guglielmi. *Recipe*.
URL <http://iccl.tu-dresden.de/~guglielm/p/AG2.pdf>, 2002.
- [Gug03] Alessio Guglielmi. *Mismatch*.
URL <http://iccl.tu-dresden.de/~guglielm/p/AG9.pdf>, 2003.
- [Hei05] Robert Hein. *Geometric theories and proof theory of modal logic*. Master’s thesis, Technische Universität Dresden, 2005.
URL http://bitschnitzer.de/robert_thesis.ps.gz.
- [HS05] Robert Hein and Charles Stewart. *Purity through Unravalling with a Discussion of Cut-Elimination*. URL <http://bitschnitzer.de/ptu+ce.ps>, 2005.
- [Kra96] M. Kracht. Power and weakness of the modal display calculus. In, H. Wansing (ed.), *Proof Theory of Modal Logic*, pages 93–121. Kluwer, Dordrecht, 1996.
- [Lam01] Francois Lamarche. On the algebra of structural contexts. Manuscript, 2001.
- [LS77] E. J. Lemmon and Dana Scott. *An Introduction to Modal Logic*, 1977.
- [Neg01] Sara Negri. Contraction-free sequent calculi for geometric theories, with an application to Barr’s theorem. Report No.22, Institut Mittag-Leffler, The Royal Swedish Academy of Science, 2001.
- [Sim93] Alex Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, 1993.
- [SS04] Charles Stewart and Phiniki Stouppa. A systematic proof theory for several modal logics. In *Proceedings of the 5th International Conference on Advances in Modal Logic (AiML-2004)*. King’s College Publications, 2004.
- [Sto04] Phiniki Stouppa. *The design of modal proof theories: the case of S5*. MSc thesis, Technische Universität Dresden, 2004.
- [Vic93] Steven Vickers. Geometric logic in computer science. In *Theory and Formal Methods 1993*, pages 37–54. Springer Workshops in Computer Science, 1993.

Hierarchical Proof Structures

Ewen Denney¹, John Power^{2*}, and Konstantinos Tourlas²

¹ USRA/RIACS, NASA Ames Research Center, CA 94035, USA

² Laboratory for the Foundations of Computer Science, King's Buildings,
University of Edinburgh, EH9 3JZ, SCOTLAND

Abstract. Motivated by structure arising in tactic-based theorem proving, we develop the concept of hierarchical proof tree or *hiproof* by characterising a geometrically natural definition in terms of its family of proof views. We first recall a definition of hierarchical proof tree, explaining its axioms and illustrating by example. We then describe notions involved with proof views. Then we characterise hierarchical proof trees in terms of dags of proof views. Our ultimate goal is to axiomatise the structure required for constructing and navigating tactic-based proofs. This is work in progress towards that end.

1 Introduction

Consider a proof by induction as represented by Figure 1(a): the nodes are labelled by tactic identifiers, inclusion of one node in another indicates a subtactic relationship, and the arrows represent sequential composition. The diagram is read as follows: the proof consists of invoking an induction tactic, **Induction**. That consists of applying an induction rule, **Ind-Rule**, which then generates two subgoals. The first subgoal is handled by the **Base** tactic, the second by the **Step** tactic. In turn, **Step** is defined as first applying the **Rewrite** tactic, and then the **Use-Hyp** tactic, with **Base**, **Rewrite** and **Use-Hyp** treated as primitive. In contrast to the usual presentations of a proof by induction, the emphasis is on tactics rather than on goals and proof steps.

For a structurally somewhat more complex proof, consider Figure 1(b). At the most abstract level, the proof consists of applying **T1**, and then **DP**. The tactic **T1** first applies **T2**, generating two subgoals, the first of which is handled by **WF**. The second is handled by **DP**, which applies **Normalise** and then **Taut**.

These examples reflect, albeit very abstractly, the hierarchical structure of tactics as appear in proof assistants such as [1–3]. In [4], we took a first abstract step towards developing a definition and mathematical theory of such hierarchy, ultimately aimed towards the development of interfaces, both graphical interfaces for individual theorem provers and interfaces between theorem provers. Our central definition, abstracting from Figures 1(a) and 1(b), was that of a *hierarchical proof tree* or *hiproof*. We analysed an appropriate choice of axioms for hiproofs, repeated here in Section 2, then we studied the relationship between a hiproof and its underlying ordinary proof, gave a notion of refinement of

* John Power has been supported by EPSRC grant no. GR/586372/01.

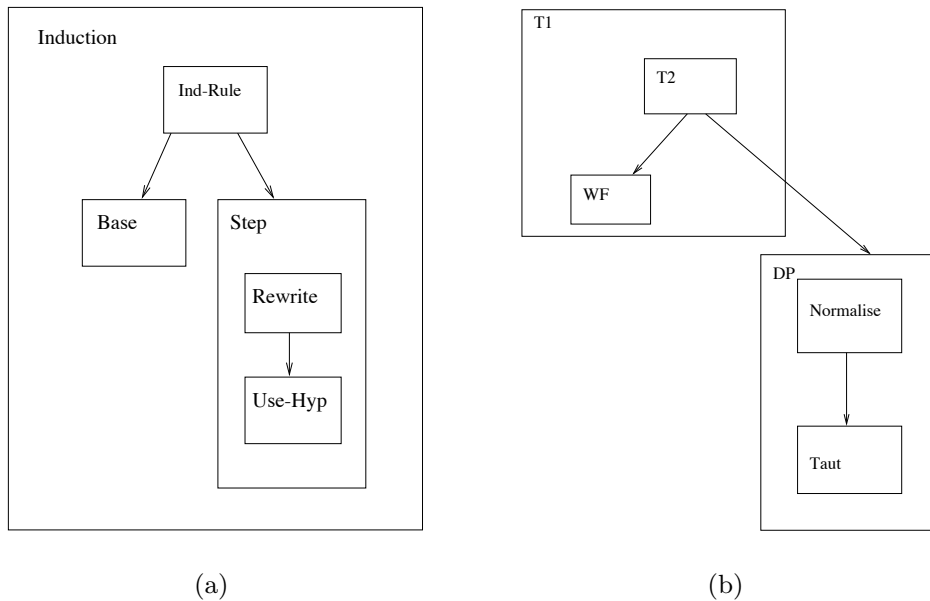


Fig. 1. Two hierarchical proofs

hiproofs, and characterised hiproofs in terms more amenable to implementation. In particular, using a subtle notion of map, we showed that the obvious inclusion of a category of ordinary proof trees into one of hierarchical proof trees has a left adjoint, with that left adjoint yielding a natural notion of the *skeleton* of a hierarchical proof.

For the purposes of this paper, we shall refer to the notion of hierarchical proof we developed in [4] as a *hiproof of type 1*. In [4] we also introduced *type 2 hiproofs*, which are an equivalent formulation of type 1 hiproofs designed to facilitate implementation. We do not discuss type 2 hiproofs in this paper, but we want to give a third equivalent formulation of the notion. So, for overall consistency, we shall refer to the central new construct of this paper as a *type 3 hiproof*. One of the overall goals of this work is to study a diversity of possible formulations of hierarchical proof structure in order to understand the common structure behind hierarchy in tactic-based theorem proving.

Now consider a hiproof of type 1. A user is unlikely to view all the information it contains at once: the main point of structuring it hierarchically is that the proof can be viewed at different levels of abstraction in a sense we shall make precise. In particular, consider the hiproof in Figure 1(b). At the highest level of abstraction it can be thought of as the two step proof, T1 followed by DP. We can think of this as an abstract proof, where T1 and DP have no internal structure, and so are regarded as atomic steps. At the lowest level of abstraction, on the other hand, the proof tree is formed from the atomic steps T2, WF, Normalise, and Taut, in the obvious way. This is the *skeleton* of the hiproof, as defined in

[4], where it was characterised as a naturally arising left adjoint. Rather than consider the whole skeleton directly, we could have unfolded just one of the top-level tactics, T1 or DP, thus yielding four possible “views” of this hiproof (see Figure 2) in total. More generally, we could unfold any abstract node (which may itself be a tree of abstract nodes), replacing the node with its immediate contents, yielding another tree.

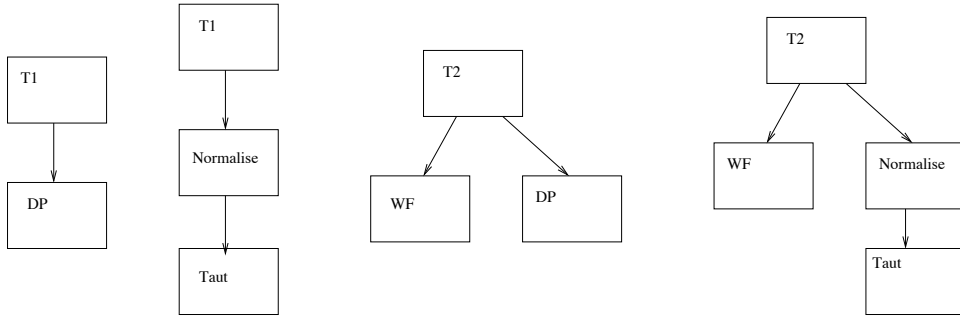


Fig. 2. Four views of a single hiproof

We call the proof trees that result from such sequences of unfolding *proof views*. The skeleton of a hiproof is the special case where no nodes are abstract. A proof view is a tree of inferences where some of the nodes may be abstract steps, i.e., tactics. Sets of proof trees can be seen as a dynamic interpretation of a tactic-based proof, where the possible traces of the proof’s unfolding embody its hierarchy. This raises the question: can we take such views as primitive? In other words, can we reformulate the notion of hiproof in terms of a set of proof trees which are self-consistent in some sense? In this paper, we formalise the various relevant concepts and constructions, and formulate the theorem.

In Section 2, we recall the definition of type 1 hiproof and explain its axioms, illustrated by examples. In Section 3, we define the notion of type 3 hiproof. In Section 4, we show how to construct a type 1 hiproof from a type 3 hiproof. And in Section 5, we give the converse, constructing a type 3 hiproof from a type 1 hiproof.

2 Hierarchical proof trees

In this section, we recall the notion of a hierarchical proof tree or type 1 hiproof from [4]. To motivate the definition, we first analyse, by means of an example, the relationship between tactics and standard notions of formal proof such as proofs in natural deduction style.

Example 1. Consider a natural deduction proof of $A \Rightarrow A \wedge (x = x)$, as in Figure 3. The obvious (backwards) proof is implication introduction, followed

by conjunction introduction, and then applying axiom and reflexivity to the two subgoals. The essential information of the proof is the sequence of inference

$$\frac{\frac{\frac{}{A \vdash A} \text{Ax} \quad \frac{}{A \vdash x = x} \text{Refl}}{A \vdash A \wedge (x = x)} \text{And-I}}{\vdash A \Rightarrow A \wedge (x = x)} \text{Imp-I}$$

Fig. 3. A simple natural deduction proof

rules, with the order of those rules represented by a proof tree as in Figure 4(a). Typically, however, theorem provers allow the use of higher-level tactics that

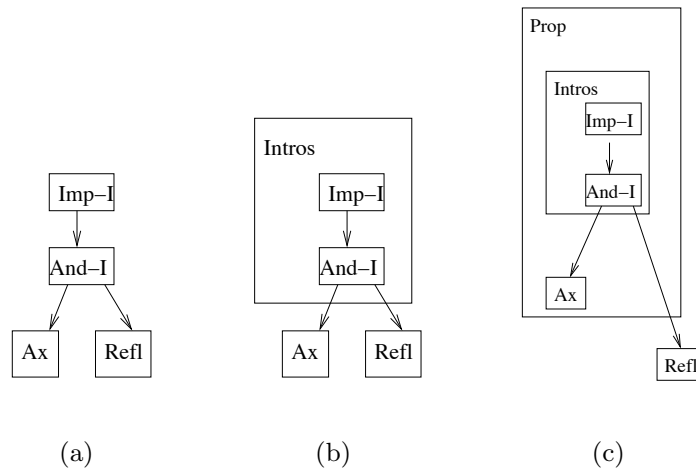


Fig. 4. Introducing hierarchy in proof diagrams by grouping

group together the application of a number of low-level inferences. For example, it is common to have an **Intros** command, which performs all possible introduction rules. We can indicate this on the proof diagram by grouping **Imp-I** and **And-I** together, as in Figure 4(b). We could go further and define a tactic, **Prop**, which first calls **Intros**, and then tries to use axioms wherever possible. This gives the hierarchical structure of Figure 4(c). \square

Example 1 shows that proofs can be represented as tactic- (or axiom and inference)-labelled trees with hierarchical structure on the set of nodes. The tree structure is straightforward, but the hierarchical structure and its interaction with the tree structure are more complex. We formalise the hierarchical structure by a partial order, with $v \leq_i w$ represented visually by depicting the node v as

sitting inside the node w . The partial order satisfies axioms to the effect that it is generated by a (finite) forest, and it is sometimes convenient to regard it as such. Our hierarchical trees are labelled by tactics, so we henceforth assume that Λ is a fixed non-empty set of *tactic identifiers* or *method identifiers*. We write $isroot_F(v)$ (or $isroot_{\rightarrow}$) for the assertion that there is a tree in forest F with root v , and we write $siblings_F(v, v')$ (or $siblings_{\rightarrow}(v, v')$) if v and v' have the same parent or are both roots.

Definition 1. A hierarchical proof tree, or (*type 1*) hiproof for short, consists of a tuple $\langle V, \leq_i, \rightarrow_s, tac \rangle$, comprising a (necessarily finite) forest qua poset $i = \langle V, \leq_i \rangle$ and a forest $s = \langle V, \rightarrow_s \rangle$, together with a function $tac : V \rightarrow \Lambda$ which labels the nodes in V with tactic identifiers in Λ , subject to the following conditions:

1. arrows always target outer nodes: whenever $v \rightarrow_s w_1$ and $w_1 <_i w_2$, then $v <_i w_2$
2. arrows always emanate from inner nodes: whenever $w_1 \leq_i v$ and $v \rightarrow_s w_2$ then $v = w_1$
3. inclusion and sequence are mutually exclusive: whenever $v \leq_i w$ and $v \rightarrow_s^* w$, then $v = w$
4. given any two nodes v and v' which both lie at the top inclusion level, or are both immediately included in the same node, then at most one of v, v' has no incoming \rightarrow_s edge:

$$\forall v, v' \in V. siblings_i(v, v') \wedge isroot_s(v) \wedge isroot_s(v') \implies v = v'.$$

□

Note the subtlety in the first condition, especially in combination with the third: an arrow from a node v can only go to an outer node *relative* to the inclusion level of v . So, for instance, Example 1 satisfies the condition. Observe that the fourth condition together with finiteness imply that there is a unique node that is maximal with respect to \leq_i and has no incoming \rightarrow_s edge, acting as a kind of hierarchical root.

The main theorem justifying the axioms in [4] shows that every hiproof unfolds to give an ordinary proof (its skeleton). But here we analyse the axioms by looking at some non-examples. The axioms are designed to ensure that none of the diagrams in Figure 5 forms a hiproof.

In tactical theorem proving, one tactic is followed by another, which unfolds to give another tactic, and so on. So tactics are invoked ‘at the most abstract level.’ But Figure 5(a) contradicts that because if T1 is followed by T3 and T2 unfolds to T3, the more abstract T2 should follow T1. Equivalently, it would be permissible for T3 to follow T1, but then the fact that T2 is an abstraction of T3 would be irrelevant to the proof and should not be added after the composition of T1 and T3. Conversely, when a tactic finishes executing, control flows from the most recently executed tactic, i.e. the innermost, outwards, but Figure 5(b) contradicts that. We want to exclude Figure 5(c) too in order to avoid circularity

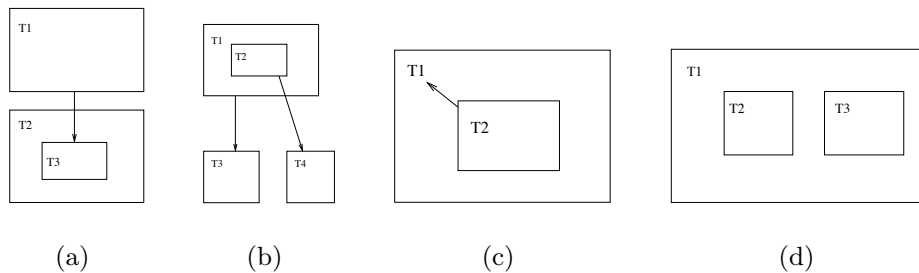


Fig. 5. Four non-examples of hiproofs

of unfolding and sequencing. Finally, Figure 5(d) fails because tactic T1 should unfold to give a unique subsequent tactic to execute, not two.

The first condition in the definition of hiproof prohibits the inclusion hierarchy from being ‘downwards’ transcended by composition, e.g. as in Figure 5(a). The second condition precludes Figure 5(b). The third condition precludes Figure 5(c): the similar structure with the arrow pointing the other direction is already precluded by the second condition. And the fourth condition precludes Figure 5(d) as well as the similar non-proof example obtained from Figure 5(d) by removing the node labelled T1. For a positive example of a hiproof, consider Figure 1(b).

The main ideas behind the definition of hiproof can be understood in terms of Figures 1(a) and 1(b). Although motivated by diagrams, we have abstracted away from geometry to discrete mathematical structure. The central features are as follows:

- we do not require tactic identifiers to be unique as a tactic may be applied repeatedly in a proof. But we informally refer to proof nodes by their tactic identifiers where there is no ambiguity.
- there are only two relationships that can hold between nodes: inclusion, representing the unfolding of a tactic into its definition, with arrows representing sequential composition. For example, in Figure 1(b), the decision procedure DP unfolds to give the composition of `Normalise` with `Taut`.
- hiproofs are essentially tree-like in that subgoals are independent: a tactic acts on a single subgoal. That is not generally the case in tactical theorem proving, and we intend to extend the definition accordingly in future work. Tactics usually return a list of subgoals, but we abstract away from the order on child tactics.

A hiproof, therefore, consists of a finite collection of tactic-labelled nodes, related by inclusion and composition. Although the diagrams represent abstract versions of full proofs, we are interested in how such proofs are constructed, and so we consider partial proofs as well-formed.

3 Hiproofs as families of proof views

A hierarchical proof yields and can be characterised by a collection of non-hierarchical proofs, i.e., by simple inference trees, that are self-consistent in a sense that we make precise in this section. These ordinary proofs generated by the hiproof can be regarded as views of the hiproof at various levels of abstraction given by all possible “unfoldings”. Such views may, for instance, appear on a computer screen when one clicks on a particular node of a hiproof.

A first attempt to characterise hiproofs in terms of such views is to try to characterise a hiproof by the set of its partial underlying proofs. But that is not subtle enough as it does not distinguish between the two hiproofs in Figure 6, both of which would be interpreted as the set of two trees, $\{T1, T2 \rightarrow T3\}$. So we need to consider the total underlying proofs of a hiproof. The second hiproof in Figure 6 now has interpretation $\{T1 \rightarrow T3, T2 \rightarrow T3\}$, and the first is as before.

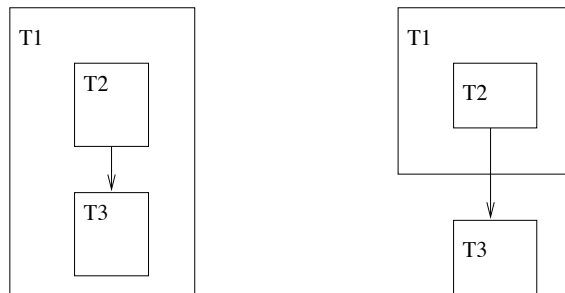


Fig. 6. Two distinct hiproofs with the same underlying proof

But that is still not delicate enough: taking the sets of the underlying proofs of a hiproof does not distinguish between hiproofs with the structures $T1 \leq_i T2$ and $T2 \leq_i T1$. So we replace sets by lists that represent the sequence of unfoldings of a hiproof. But there are several ways in which a hiproof can be unfolded. If we take all possible unfoldings, we obtain the structure of a dag, which, finally, has sufficient structure to provide a characterisation. The main technical part of our work involves characterising those dags that thus arise.

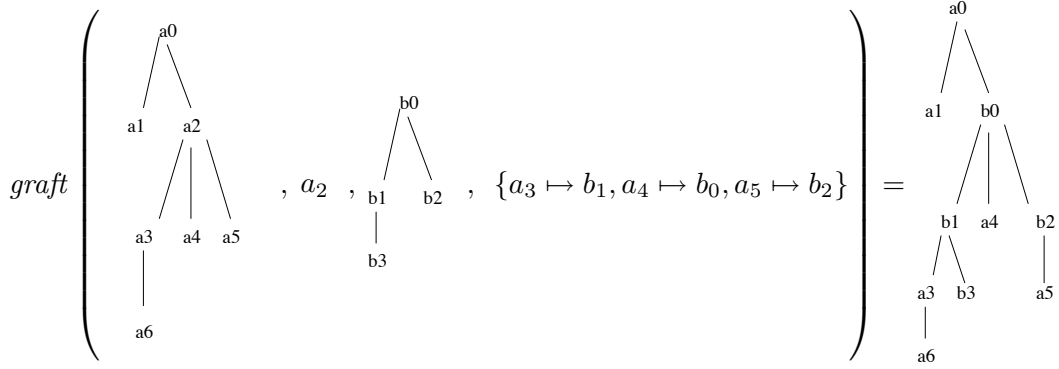
The characterisation requires a delicate operation that grafts a tree t' at a vertex v of a given tree t , with an embedding map, f , from the children of v into the nodes of t , thus generalising the idea of substituting a tree for a leaf vertex in another tree. Formally, the definition is as follows:

Definition 2. Let $t_A = \langle V_A, \rightarrow_A, r_A \rangle$ and $t_B = \langle V_B, \rightarrow_B, r_B \rangle$ be (rooted) trees, $v_0 \in V_A$, and f a map from the children of v_0 to V_B . Then $\text{graft}(t_A, v_0, t_B, f) \stackrel{\text{def}}{=} \langle V, \rightarrow, r \rangle$ is the tree where

- $V = V_A \setminus \{v_0\} + V_B$,
- $v \rightarrow v'$ if and only if either of the following hold
 1. $v, v' \in V_A \setminus \{v_0\}$ and $v \rightarrow_A v'$
 2. $v, v' \in V_B$ and $v \rightarrow_B v'$
 3. $v \in V_A$, $v' = r_B$ and $v \rightarrow_a v_0$
 4. v' is a child of v_0 and $v = f(v')$
- $r = r_B$ if $v_0 = r_A$ or otherwise $r = r_A$. □

Note that we discard the vertex v_0 instead of the root of the tree being grafted.

Example 2.



It is routine to extend the definition of grafting to incorporate labelling. Labels are taken in the set Λ of tactic identifiers. We do not insist that the labelling functions on the two trees be consistent with each other on v_0 and r_B .

Definition 3. Let $\langle t_A, l_A \rangle$ and $\langle t_B, l_B \rangle$ be trees labelled over Λ and $v_0 \in V_A$. Then

$$\text{graft}(\langle t_A, l_A \rangle, v_0, \langle t_B, l_B \rangle, f)$$

is the labelled tree $\langle \text{graft}(t_A, v_0, t_B, f), l \rangle$ where $l : V_A \setminus \{v_0\} + V_B \rightarrow \Lambda$ is defined as follows:

1. whenever $v \in V_B$, $l(v) = l_B(v)$
2. whenever $v \in V_A$ and $v \neq v_0$, $l(v) = l_A(v)$. □

For simplicity of presentation we shall tacitly assume that, whenever we write $\text{graft}(t_A, v_0, t_B, f)$, the sets V_A and V_B of vertices in t_A and t_B are disjoint, i.e., $V_A \cap V_B = \emptyset$, whereby also $v_0 \notin V_B$. So the set of vertices underlying $\text{graft}(t_A, v_0, t_B, f)$ is $V_A \setminus \{v_0\} \cup V_B$.

In order to give a coherent definition of a type 3 hiproof, we first need to observe a few facts about grafting. They are as follows:

Lemma 1. (*Injectivity of grafting*) $\text{graft}(t, v_0, t_1, f) = \text{graft}(t, v_0, t'_1, f')$ implies $t_1 = t'_1$ and $f = f'$. \square

Lemma 2. (*Commutativity of grafting*)
 $\text{graft}(\text{graft}(t_0, v_1, t_1, f_1), v_2, t_2, f_2) = \text{graft}(\text{graft}(t_0, v_2, t_2, f_2), v_1, t_1, f_1)$. \square

Lemma 3. (*Independence of grafts*) Let $t_0 = \langle V_0, \rightarrow_0, r_0 \rangle$ be a tree and v_1, v_2 be distinct vertices in V_0 . Then

$$\text{graft}(\text{graft}(t_0, v_1, t_1, f_1), v_2, t_2, f_2) = \text{graft}(\text{graft}(t_0, v_2, t'_2, f'_2), v_1, t'_1, f'_1)$$

implies $t_i = t'_i$ and $f_i = f'_i$, for $i = 1, 2$. \square

We can now formally define type 3 hiproofs:

Definition 4. A hiproof of type 3 is a tuple $\langle U, \rightsquigarrow, \tau, \beta \rangle$, where

- $\langle U, \rightsquigarrow \rangle$ is a dag,
- $\tau : U \rightarrow \Lambda\text{-Tree}$ is a function, assigning to each vertex $u \in U$ a Λ -labelled tree $\tau(u)$, and
- β assigns to each pair $\langle u, u' \rangle \in \rightsquigarrow$ a vertex in $\tau(u)$.

Writing $u \overset{v}{\rightsquigarrow} u'$ to mean the existence of $u, u' \in U$ such that $\langle u, u' \rangle \in \rightsquigarrow$ and $\beta(\langle u, u' \rangle) = v$, the above data is subject to the following conditions:

1. $\langle U, \rightsquigarrow \rangle$ has a source, which we denote by u_\top
2. if $u \overset{v}{\rightsquigarrow} u'$, there exists a labelled tree γ and a map f such that $\tau(u') = \text{graft}(\tau(u), v, \gamma, f)$
3. if $u_0 \overset{v}{\rightsquigarrow} u_1$ and $u_0 \overset{v}{\rightsquigarrow} u_2$, then $u_1 = u_2$, and
4. if $u_0 \overset{v_1}{\rightsquigarrow} u_1$ and $u_0 \overset{v_2}{\rightsquigarrow} u_2$, there exists u_3 such that $u_1 \overset{v_2}{\rightsquigarrow} u_3$ and $u_2 \overset{v_1}{\rightsquigarrow} u_3$. \square

This definition formulates the discussion at the beginning of the section, that a hiproof can be represented as a dag of proof trees. The top is the most abstract proof, the bottom (as we will see) is the most concrete proof, i.e. the underlying skeleton. The second and third conditions give the meaning of the dag in terms of the grafting of proof trees, while the fourth is a completeness condition: if you can unfold nodes separately, then you can unfold them together.

Proposition 1. A dag $\langle U, \rightsquigarrow \rangle$ satisfying the conditions on a dag in Definition 4 has a (necessarily unique) sink u_\perp .

Proof. (Sketch) This follows from general theorems on abstract rewriting systems satisfying the diamond property, as the rewriting system in question is strongly and uniquely normalising. \square

4 From hiproofs of type 3 to hiproofs of type 1

In this section, we construct a type 1 hiproof from a type 3 hiproof. This first requires some notation, then a subtle construction combining a type 1 hiproof with a tree: we build our final type 1 hiproof by an inductive process, requiring several intermediary type 1 hiproofs as inductive steps, hence the need to combine a type 1 hiproof inductively with a tree.

First observe that by Lemma 1, whenever $u \overset{v}{\rightsquigarrow} u'$ holds in a type 3 hiproof, the labelled tree γ and map f given by the third condition are unique up to isomorphism such that $\tau(u') = \text{graft}(\tau(u), v, \gamma, f)$. We shall therefore write $u \overset{v, \gamma, f}{\rightsquigarrow} u'$, instead of $u \overset{v}{\rightsquigarrow} u'$, when knowledge of the unique γ and f is required. By Lemma 3, we have the following:

Proposition 2. *For $v_1 \neq v_2$, whenever $u_0 \overset{v_1, \gamma_1, f_1}{\rightsquigarrow} u_1 \overset{v_2, \gamma_2, f_2}{\rightsquigarrow} u_3$ and $u_0 \overset{v_2, \gamma'_2, f'_1}{\rightsquigarrow} u_2 \overset{v_1, \gamma'_1, f'_2}{\rightsquigarrow} u_3$ in a type 3 hiproof, one has $\gamma_1 = \gamma'_1$, $\gamma_2 = \gamma'_2$, $f_1 = f'_1$ and $f_2 = f'_2$. \square*

We extend the notation to finite lists of $\langle v, \gamma \rangle$ pairs. Then for every such list $\sigma = \langle v_0, \gamma_0 \rangle, \dots, \langle v_{n-1}, \gamma_{n-1} \rangle$, we write

$$u \overset{\sigma}{\rightsquigarrow} u'$$

to mean

$$u \overset{v_0, \gamma_0}{\rightsquigarrow} u_0 \overset{v_1, \gamma_1}{\rightsquigarrow} \dots \overset{v_{n-1}, \gamma_{n-1}}{\rightsquigarrow} u_n \text{ and } u_n = u' .$$

As usual, we write $\langle v, \gamma \rangle :: \sigma$ for the list with head $\langle v, \gamma \rangle$ and tail σ .

Now we inductively present a family F of operations on hiproofs of type 1, indexed by paths $u \overset{\sigma}{\rightsquigarrow} u'$ in a hiproof of type 3. The idea is to gradually build up a (type 1) hiproof by combining all the trees along a path in a hiproof of type 3. We need an operation $h \triangleleft^{v_0} \gamma$ extending a type 1 hiproof h using the labelled tree γ at the node v_0 . This is defined as follows:

Definition 5. *Let $h = \langle V_h, \leq_i, \rightarrow_s, \text{tac} \rangle$ be a hiproof of type 1, and let $v_0 \in V_h$ be \leq_i -minimal. Then given a Λ -labelled tree $\gamma = \langle V_\gamma, \rightarrow, r, l \rangle$, and a map f from the children of v_0 (with respect to \rightarrow_s) into V_γ , define $h \triangleleft^{v_0, f} \gamma$ to be $\langle V', \leq'_i, \rightarrow'_s, \text{tac}' \rangle$ where*

- $V = V_h + V_\gamma$
- $v \leq'_i u$ if and only if either of the following hold
 1. $v, u \in V_h$ and $v \leq_i u$
 2. $v \in V_\gamma$, $u \in V_h$ and $v_0 \leq_i u$
- $v \rightarrow'_s u$ if and only if either of the following hold
 1. $v, u \in V_\gamma$ and $v \rightarrow u$
 2. $u \in V_h$, $v \in V_\gamma$, u is a child of v_0 , and $f(u) = v$
 3. $v, u \in V_h$, $v \neq v_0$ and $v \rightarrow_s u$
- $\text{tac}' = \text{tac} + l$. \square

The difference between this construction and that of $\text{graft}(h, v_0, \gamma, f)$ is that here h can be an arbitrary hiproof (whereas for grafting h must be a tree) and we keep the node v_0 and put γ inside it (since v_0 is \leq_i -minimal it has no ‘contents’) rather than replacing v_0 by γ . Again, when we write $h \triangleleft^{v_0, f} \gamma$, we shall tacitly assume that the sets V_h and V_γ underlying h and γ respectively, are disjoint. This allows us to regard the set of vertices underlying $h \triangleleft^{v_0} \gamma$ as being simply $V_h \cup V_\gamma$. And by “suitable” labelled trees, we mean γ_1 and γ_2 for which $V_h \cap V_{\gamma_1} = \emptyset$, $V_h \cap V_{\gamma_2} = \emptyset$ and $V_{\gamma_1} \cap V_{\gamma_2} = \emptyset$.

Lemma 4. (*Commutativity of \triangleleft*) *If v_1 and v_2 are distinct vertices in a type 1 hiproof h , then $h \triangleleft^{v_1, f_1} \gamma_1 \triangleleft^{v_2, f_2} \gamma_2 = h \triangleleft^{v_2, f_2} \gamma_2 \triangleleft^{v_1, f_1} \gamma_1$ for any suitable Λ -labelled trees.* \square

Proposition 3. (*Well-definedness of \triangleleft*) *Let $h = \langle V, \leq_i, \rightarrow_s, \text{tac} \rangle$ be a hiproof of type 1 and γ any suitable Λ -labelled tree. $h \triangleleft^{v, f} \gamma$ is a hiproof of type 1 whenever $v \in V$ is \leq_i -minimal and f is a suitable map.* \square

Now let $u \xrightarrow{\sigma} u'$ be a path in a hiproof of type 3 and h any hiproof of type 1. We proceed to define F_σ by induction on the length of the list σ :

- $F_u(h) = h$
- $F_{u \xrightarrow{v, f} u'}(h) = F_{u'}(h \triangleleft^{v, f} \gamma)$.

Any labelled tree may be trivially regarded as a “flat” hiproof of type 1. Formally:

Definition 6. *We define the embedding, $\mathcal{E} : \mathbf{Tree} \rightarrow \mathbf{Hiproof}_1$, as the map which takes $\gamma = \langle V, \rightarrow, r, l \rangle$ to $\langle V, \text{id}_V, \rightarrow, l \rangle$.* \square

However, we will often blur the distinction and regard trees as type 1 hiproofs when convenient.

We showed in [4] that \mathcal{E} extends to a functor from a category whose objects are proof trees to one whose objects are type 1 hiproofs, and proved that it has a left adjoint, characterising a natural construction of the skeleton, or underlying ordinary proof, of a type 1 hiproof.

Corollary 1. *$F_{u \xrightarrow{\sigma} u'}(\tau(u))$ is always a well-formed hiproof of type 1.* \square

Writing $\sigma \sim \sigma'$ to mean that (v, γ, f) -lists σ and σ' are permutations of one another, we finally have:

Theorem 1. *Whenever $u \xrightarrow{\sigma} u_\perp$ and $u \xrightarrow{\sigma'} u_\perp$ are paths in a type 3 hiproof, one has*

1. $\sigma \sim \sigma'$; and
2. for all h , $F_{u \xrightarrow{\sigma} u_\perp}(h) = F_{u \xrightarrow{\sigma'} u_\perp}(h)$.

\square

Theorem 1 shows that no matter which path is used from a type 3 hiproof, F combines the trees along that path into the same type 1 hiproof. We can now define μ_{31} .

Definition 7. $\mu_{31}(h_3) = F_\sigma(\mathcal{E}(u_\top))$, where $u_\top \xrightarrow{\sigma} u_\perp$ is any path in h_3 . \square

5 From type 1 to type 3 hiproofs

In this section, we give a converse to our construction of a type 1 hiproof from a type 3 hiproof. We first recall the definition of the skeleton of a type 1 hiproof from [4].

Definition 8. Let $h = \langle V, \leq_i, \rightarrow_s, t \rangle$ be a type 1 hiproof. We define the skeleton of h , written $\mathbf{sk}_1(h)$, to be the Λ -labelled tree $\langle V_T, \rightarrow_T, r \rangle$, corresponding to the finite poset $T = \langle V_T, \leq_T \rangle$, where V_T are the leaves of \leq_i , and $v_1 \leq v_2$ if and only if there exists a $v \in V$ such that $v_2 \leq_i v$ and $v_1 \rightarrow_s v$. \square

For example, the skeleton of Figure 1(b) is the rightmost tree in Figure 2.

Proposition 4. The definition above gives a well-formed tree.

Proof. We must show that for all $v \in V_T$, there exists a unique path from r to v . There must exist a $(>_i^1 \cup \rightarrow_s)$ -path from the root of h_1 to v , by induction. This path must be unique since each vertex has a unique predecessor. \square

Definition 9. Let $h_3 = \langle U, \rightsquigarrow, \tau, \beta \rangle$ be a type 3 hiproof. We define the skeleton of h_3 , written $\mathbf{sk}_3(h_3)$, to be the tree, $\tau(u_\perp)$, where u_\perp is the sink guaranteed by Proposition 1. \square

The next result shows that this is a reasonable definition of skeleton. In other words, it corresponds to skeletons of type 1.

Proposition 5. For all hiproofs h_3 of type 3, $\mathbf{sk}_3(h_3) = \mathbf{sk}_1(\mu_{31}(h_3))$ and for all hiproofs h_1 of type 1, $\mathbf{sk}_1(h_1) = \mathbf{sk}_3(\mu_{13}(h_1))$. \square

In order to define the translation from type 1 to type 3, we first need to define a notion of abstraction of hiproofs at a node.

Definition 10. Let $h_1 = \langle V, \leq_i, \rightarrow_s, t \rangle$ be a type 1 hiproof, and let $z \in V$. We define the abstraction of h_1 at z , written $\mathbf{abs}_1(z, h_1)$, to be the type 1 hiproof $\langle V', \leq'_i, \rightarrow'_s, t' \rangle$, where $V' = \{v \in V \mid v \not\leq_i z\}$, \leq'_i and t' are the restrictions of \leq_i and t to V' , and $v_1 \rightarrow'_s v_2$ if and only if $v_1, v_2 \in V'$, and either $v_1 \neq z$ and $v_1 \rightarrow_s v_2$, or $v_1 = z$ and there exists a $v \leq_i z$ such that $v \rightarrow_s v_2$. \square

This is, in some sense, dual to the skeleton since it throws away the contents of a node. We say that h' is an abstraction of h if h' is the abstraction of h at v for some v .

To define a map from type 1 to type 3, we must extend the definition of abstraction for type 1 to sets of nodes. First note that if h_1 nodes z_1 and z_2 are \leq_i -incomparable, then $\mathbf{abs}_1(z_1, \mathbf{abs}_1(z_2, h_1)) = \mathbf{abs}_1(z_2, \mathbf{abs}_1(z_1, h_1))$.

Let I be a \leq_i -incomparable set of nodes in h_1 . Then, $\mathbf{abs}_1(I, h_1)$ is well-defined via

$$\begin{aligned} \mathbf{abs}_1(\{z\}, h_1) &= h_1 \\ \mathbf{abs}_1(I \cup \{z\}, h_1) &= \mathbf{abs}_1(z, \mathbf{abs}_1(I, h_1)), \text{ for } z \notin I. \end{aligned}$$

Abstraction can also be defined for type 3 hiproofs but this is not needed here to define the translation.

Definition 11. Define $\mu_{13} : \mathbf{Hiproof}_1 \rightarrow \mathbf{Hiproof}_3$ as the function sending each hiproof $h_1 = \langle V, \leq_i, \rightarrow_s, tac \rangle$ of type 1 to the hiproof $\langle U, \rightsquigarrow, \tau, \beta \rangle$ of type 3 given by the following data:

- The component trees are the skeletons of the abstractions of h_1 for all \leq_i -incomparable subsets
- $t_1 \rightsquigarrow^v t_2$ if and only if $t_1 = \mathbf{sk}_1(\mathbf{abs}_1(I_1, h_1))$, $t_2 = \mathbf{sk}_1(\mathbf{abs}_1(I_2, h_1))$, where $I_1 = I \cup \{v\}$, $v \notin I$, $I_2 = I \cup \{v' \mid v' <_i^1 v\}$.

□

We can show that $u_\top = \mathbf{sk}_1(\mathbf{abs}_1(I_{max}, h_1))$, where I_{max} is the set of \leq_i -maximal nodes, and u_\perp is the skeleton of h_1 , i.e. $\mathbf{sk}_1(\mathbf{abs}_1(\{\}, h_1))$. When $t = \mathbf{sk}_1(\mathbf{abs}_1(I, h))$, we will write t as t_I . Note that I is not unique, in general, since any node which is \leq_i -minimal, for example, can be added with no effect.

Proposition 6. μ_{13} is well-defined.

Proof. To show that μ_{13} is well-defined, we must show that $\mu_{13}(h_1)$ is a valid type 3 hiproof.

First we characterise \rightsquigarrow^* . Define $I <_i^1 I'$ if and only if $I' = I_0 \cup \{v\}$, $v \notin I_0$, and $I = I_0 \cup \{v' \mid v' <_i^1 v\}$. Then define $I \leq_i I'$ if and only if $I (<_i^1)^* I'$. Now, we have that $I \leq_i I'$ implies $\forall i \in I. \exists i' \in I'. i \leq_i i'$ (though not in the other direction). Clearly, $t_I \rightsquigarrow^* t_{I'}$ if and only if $I' \leq_i I$.

Next, we show that \leq_i is a partial order. It is reflexive and transitive by definition. To see that it is antisymmetric, suppose $I \leq_i I'$ and $I' \leq_i I$. Let $v \in I$. Then there exists a $v' \in I'$ such that $v \leq_i v'$, and a $v_2 \in I$ such that $v \leq_i v_2$. Hence $v = v'$ (by antisymmetry of \leq_i), and so $I \subseteq I'$. Likewise, $I' \subseteq I$, so $I = I'$, and we have shown antisymmetry.

This shows that $\mu_{13}(h_1)$ is a dag. We must also check the type 3 conditions:

1. u_\top is the top, since each $I \leq_i \{i \mid i \text{ is } \leq_i\text{-maximal}\}$.
2. Define the shell of a hiproof, $\mathbf{shell}(h)$, as $\mathbf{abs}_1(I_{max}, h)$, and $\mathbf{sub}(h, v)$ to be the hiproof ‘inside the node v in h ’, i.e. the hiproof rooted at the top node included in v , and with no nodes outwith v .
Then, if $t_1 = \mathbf{sk}_1(\mathbf{abs}_1(I \cup \{v\}, h)) \rightsquigarrow^v t_2 = \mathbf{sk}_1(\mathbf{abs}_1(I \cup \{v' \mid v' <_i^1 v\}, h))$, then it can be shown that $t_2 = \mathbf{graft}(t_1, v, \gamma, f)$, where $\gamma = \mathbf{tree}(h, v)$, is defined as $\mathbf{shell}(\mathbf{sub}(h, v))$, and f , mapping the children of v in t_1 to nodes in t_2 , is given by $v_1 \mapsto v_2$ if and only if v_2 is the parent of v_1 in t_2 , and so $v_2 \in \gamma$.
3. By the definition of \rightsquigarrow^v , $u_0 = \mathbf{sk}_1(\mathbf{abs}_1(I \cup \{v\}, h)) \rightsquigarrow^v u_1, u_2$ implies $u_1, u_2 = \mathbf{sk}_1(\mathbf{abs}_1(I \cup V', h))$, where $V' = \{v' \mid v' <_i^1 v\}$.
4. Suppose $t \rightsquigarrow^{v_1} t_1$ and $t \rightsquigarrow^{v_2} t_2$, then we have $t = \mathbf{sk}_1(\mathbf{abs}_1(I \cup \{v_1, v_2\}, h)) \rightsquigarrow^{v_1} \mathbf{sk}_1(\mathbf{abs}_1(I \cup V'_1 \cup \{v_2\}, h))$, and $t \rightsquigarrow^{v_2} \mathbf{sk}_1(\mathbf{abs}_1(I \cup \{v_1\} \cup V'_2, h))$. Then these come together confluent as $\mathbf{sk}_1(\mathbf{abs}_1(I \cup V'_1 \cup V'_2, h))$.

□

One needs to do a little work to show that the constructions yielding type 1 and type 3 hiproofs from each other are mutually inverse up to coherent isomorphism: the latter point requires some straightforward category theory to make precise.

6 Conclusions

We have presented definitions of two structures arising in tactic-based theorem proving. The primary definition encapsulates our graphical intuition for this form of hierarchical proofs, and the other corresponds more closely to dynamic traces of unfoldings or proof views.

In work that we have not described in this paper, we have also looked at ordered hiproofs, where the underlying proof structures are ordered trees (in contrast to the unordered trees considered here), and hierarchical structures induced from proofs forests, which we call *hitacs*. Indeed, there is a wide range of hi-structures which can be induced from a correspondingly wide range of tree-based proof structures. We aim to develop the necessary categorical machinery for placing them in a common framework. Finally, we have also developed a term language and equational calculus, as well as the natural connections to the semantic structures presented here.

References

1. Cheikhrouhou, L., Sorge, V.: PDS — A Three-Dimensional Data Structure for Proof Plans. In: Proceedings of the International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications (ACIDCA'2000), Monastir, Tunisia (2000)
2. Kapur, D., Nie, X., Musser, D.R.: An overview of the Tecton proof system. Theoretical Computer Science **133** (1994) 307–340
3. Richardson, J.D.C., Smaill, A., Green, I.: System description: proof planning in higher-order logic with Lambda-Clam. In: 15th International Conference on Automated Deduction. (1998) 129–133
4. Denney, E., Power, J., Turlas, K.: Hiproofs: A hierarchical notion of proof tree. In: Proceedings of Mathematical Foundations of Programming Semantics (MFPS). Electronic Notes in Theoretical Computer Science (ENTCS), Elsevier (2005)

Implementing Deep Inference in TOM

Ozan Kahramanoğulları¹, Pierre-Etienne Moreau², Antoine Reilles³

¹ Computer Science Institute, University of Leipzig
International Center for Computational Logic, TU Dresden
ozan@informatik.uni-leipzig.de

² LORIA & INRIA, Nancy, France

³ LORIA & CNRS, Nancy, France
{Pierre-Etienne.Moreau,Antoine.Reilles}@loria.fr

Abstract. The calculus of structures is a proof theoretical formalism which generalizes sequent calculus with the feature of deep inference: in contrast to sequent calculus, the calculus of structures does not rely on the notion of main connective and, like in term rewriting, it permits the application of the inference rules at any depth inside a formula. TOM is a pattern matching processor that integrates term rewriting facilities into imperative languages. In this paper, relying on the correspondence between the systems in the calculus of structures and term rewriting systems, we present an implementation of system BV of the calculus of structures in Java by exploiting the term rewriting features of TOM. This way, by means of the expressive power due to Java, it becomes possible to implement different search strategies. Since the systems in the calculus of structures follow a common scheme, we argue that our implementation can be generalized to other systems in the calculus of structures for classical logic, modal logics, and different fragments of linear logic.

1 Introduction

Developing new representations of logics, which address properties that are central to computer science applications, has been one of the challenging goals of proof theory. One of the crucial needs of such a line of research is the appropriate set of implementation tools, which are in harmony with the underlying proof theoretical formalism. Such tools then make it possible to test conjectures on the logic, proof theory of which is studied. This way, they do not only allow researchers to save time by producing counter examples for false conjectures, but also shed light to potential applications of the logic being studied.

The calculus of structures [5] is a proof theoretical formalism, like natural deduction, sequent calculus, and proof nets. The calculus of structures generalizes the sequent calculus while keeping properties, such as locality and modularity (see, e.g. [3, 15]), in focus that are important for computer science applications. Structures are expressions intermediate between formulae and sequents which unify these two latter entities. This way, they provide a greater control over the mutual dependencies between logical relations. The main feature that distinguishes this formalism is *deep inference*: in contrast to the sequent calculus,

the calculus of structures does not rely on the notion of main connective, and permits the application of the inference rules at any depth inside a structure. Applicability of the inference rules at any depth results in a richer combinatorial analysis of proofs than in the sequent calculus. Because proofs are constructed by manipulating and annihilating substructures, this formalism brings shorter proofs than all other formalisms supporting analytical proofs.

The calculus of structures was conceived, in [5], for introducing a logical system, called system **BV**, which extends multiplicative linear logic with the rules mix, nullary mix, and a self-dual, non-commutative logical operator. Due to the self-dual, noncommutative operator, system **BV** is of interest for applications where sequentiality plays an important role. In particular, as Bruscoli showed in [4], the non-commutative operator of **BV** captures precisely the sequential composition of process algebra, e.g. **CCS**. In fact, system **BV** can not be designed in the sequent calculus, as it was shown by Tiu in [17], since deep inference is crucial for deriving the provable structures of system **BV**. Kahramanoğulları showed, in [11], that system **BV** is NP-complete.

The calculus of structures also provides systems which bring insights to proof theory of different logics: in [2], Brünnler presents systems in the calculus of structures for classical logic; in [16], Straßburger presents systems for different fragments of linear logic; in [14], Stewart and Stouppa give systems for a collection of modal logics; in [18], Tiu presents a local system for intuitionistic logic. All the above mentioned systems follow a common scheme due to deep inference, which we exploit in this paper.

In the sequent calculus, because of the meta-level which causes branching while going up in the proofs, proofs are trees. However, in the calculus of structures, because meta-level of the sequent calculus is represented at the object level of the logical system [6], proofs are chains of inferences rather than trees. This observation and the applicability of the inference rules at any depth draws attention to a correspondence between the term rewriting systems [1] and systems of the calculus of structures. However, structures in a logical system are considered equivalent modulo an equational theory which makes it possible to observe the structures as equational classes of formulae with respect to the underlying equational theory of the system. Exploiting these observations, in [7], Hölldobler and Kahramanoğulları showed that systems in the calculus of structures can be expressed as term rewriting systems modulo equational theories.

TOM [13, 12] is a pattern matching preprocessor that integrates term rewriting and pattern matching facilities into imperative and functional languages such as C, Java, and Caml. In this paper, by resorting to the term rewriting features of TOM, we present a proof search implementation of system **BV** in Java. For this purpose, in several steps, we simulate the role played by the equational theory during proof search in the inference rules. We show that, instead of expressing commutativity and units as equalities in the underlying equational theory, role played by the equalities for unit and commutativity can be embedded into the inference rules of the system. This way, we express associativity in a list repre-

sentation of the structures, and implement the inference rules as term rewriting rules which apply to terms that represent structures.

Because, of the expressive power of Java, it becomes possible to easily implement any search strategy for proof search. In our implementation, we resort to a *global search* strategy: we stack the structures which are premises of the all bottom-up instances of the inference rules with respect to a heuristic function, and proceed with applying this procedure to the topmost structure in the stack till the top-most structure is the unit. Since proofs are constructed by annihilating dual atoms, the heuristic function is chosen in a way which respects the mutual relations between dual atoms.

Because systems in the calculus of structures follow a common scheme, our implementation provides a recipe for implementing systems for other logics in the calculus of structures.

The rest of the paper is organized as follows: in Section 2, we re-collect the notions and notations of the calculus of structures and system BV. Then, in Sections 3 and 4, we remove the equalities for unit, and commutativity from the equational theory, respectively, by simulating their roles in the inference rules. After presenting some methods for reducing the nondeterminism in proof search in Section 5, we describe our implementation in Sections 6 and 7. We conclude with summary and discussions in Section 8.

2 The Calculus of Structures and System BV

In this section, we re-collect some notions and definitions of the calculus of structures and system BV, following [5].

In the language of BV atoms are denoted by a, b, c, \dots . Structures are denoted by R, S, T, \dots and generated by

$$S ::= \circ \mid a \mid \underbrace{\langle S; \dots; S \rangle}_{>0} \mid \underbrace{[S, \dots, S]}_{>0} \mid \underbrace{(S, \dots, S)}_{>0} \mid \bar{S} \quad ,$$

where \circ , the *unit*, is not an atom. $\langle S; \dots; S \rangle$ is called a *seq structure*, $[S, \dots, S]$ is called a *par structure*, and (S, \dots, S) is called a *copar structure*, \bar{S} is the *negation* of the structure S . A structure R is called a *proper par structure* if $R = [R_1, R_2]$ where $R_1 \neq \circ$ and $R_2 \neq \circ$. Structures are considered equivalent modulo the relation \approx , which is the smallest congruence relation induced by the equations shown in Figure 1. There \mathbf{R} , \mathbf{T} and \mathbf{U} stand for finite, non-empty sequence of structures. A *structure context*, denoted as in $S\{ \ }$, is a structure with a hole that does not appear in the scope of negation. The structure R is a *substructure* of $S\{R\}$ and $S\{ \ }$ is its *context*. Context braces are omitted if no ambiguity is possible: for instance $S[R, T]$ stands for $S\{[R, T]\}$. A structure, or a structure context, is in *normal form* when the only negated structures appearing in it are atoms and no unit \circ appears in it.

There is a straightforward correspondence between structures which do not involve seq structures and formulae of multiplicative linear logic (MLL) which

Associativity	Commutativity	Negation
$\langle \mathbf{R}; \langle \mathbf{T}; \mathbf{U} \rangle \approx \langle \mathbf{R}; \mathbf{T}; \mathbf{U} \rangle$ $[\mathbf{R}, [\mathbf{T}]] \approx [\mathbf{R}, \mathbf{T}]$ $(\mathbf{R}, (\mathbf{T})) \approx (\mathbf{R}, \mathbf{T})$	$[\mathbf{R}, \mathbf{T}] \approx [\mathbf{T}, \mathbf{R}]$ $(\mathbf{R}, \mathbf{T}) \approx (\mathbf{T}, \mathbf{R})$	$\bar{\circ} \approx \circ$ $\overline{\langle \mathbf{R}; \mathbf{T} \rangle} \approx \langle \bar{\mathbf{R}}; \bar{\mathbf{T}} \rangle$ $\overline{[\mathbf{R}, \mathbf{T}]} \approx [\bar{\mathbf{R}}, \bar{\mathbf{T}}]$ $\overline{(\mathbf{R}, \mathbf{T})} \approx [\bar{\mathbf{R}}, \bar{\mathbf{T}}]$
Context Closure	Units	$\bar{\bar{\mathbf{R}}} \approx \mathbf{R}$
if $R = T$ then $S\{R\} = S\{T\}$ and $\bar{\bar{R}} = \bar{T}$	$\langle \circ; \mathbf{R} \rangle \approx \langle \mathbf{R}; \circ \rangle \approx \langle \mathbf{R} \rangle$ $[\circ, \mathbf{R}] \approx [\mathbf{R}]$ $(\circ, \mathbf{R}) \approx (\mathbf{R})$	Singleton
		$\langle R \rangle \approx [R] \approx (R) \approx R$

Fig. 1. Equivalence relations underlying BV.

do not contain the units 1 and \perp . For example $[(a, b), \bar{c}, \bar{d}]$ corresponds to $((a \otimes b) \wp c^\perp \wp d^\perp)$, and vice versa. Units 1 and \perp are mapped into \circ , since $1 \equiv \perp$, when the rules mix and mix0 are added to MLL.

$$\text{mix} \frac{\vdash \Phi \quad \vdash \Psi}{\vdash \Phi, \Psi} \qquad \text{mix0} \frac{}{\vdash}$$

For a more detailed discussion on the proof theory of BV and the precise relation between BV and MLL, the reader is referred to [5].

In the calculus of structures, an *inference rule* is a scheme of the kind $\rho \frac{T}{R}$, where ρ is the *name* of the rule, T is its *premise* and R is its *conclusion*. A typical (deep) inference rule has the shape $\rho \frac{S\{T\}}{S\{R\}}$ and specifies the implication $T \Rightarrow R$ inside a generic context $S\{ \}$, which is the implication being modeled in the system⁴. When premise and conclusion in an instance of an inference rule are equivalent, that instance is *trivial*, otherwise it is *non-trivial*. An inference rule is called an *axiom* if its premise is empty. Rules with empty contexts correspond to the case of the sequent calculus.

A (formal) *system* \mathcal{S} is a set of inference rules. A derivation Δ in a certain formal system is a finite chain of instances of inference rules in the system. A derivation can consist of just one structure. The topmost structure in a derivation, if present, is called the *premise* of the derivation, and the bottommost structure is called its *conclusion*. A derivation Δ whose premise is T , conclusion is R , and inference rules are in \mathcal{S} will be written as $\Delta \left\| \begin{array}{c} T \\ \mathcal{S} \\ R \end{array} \right.$. Similarly, $\Pi \left\| \begin{array}{c} T \\ \mathcal{S} \\ R \end{array} \right.$

will denote a *proof* Π which is a finite derivation whose topmost inference rule is an axiom. The *length* of a derivation (proof) is the number of instances of inference rules appearing in it.

⁴ Due to duality between $T \Rightarrow R$ and $\bar{R} \Rightarrow \bar{T}$, rules come in pairs of dual rules: a down-version and an up-version. For instance, the dual of the $\mathbf{a}\downarrow$ rule in Figure 2 is the cut rule. However, in the calculus of structures, the down rules provide sound and complete systems.

$$\begin{array}{c}
\circ\downarrow \frac{}{\circ} \quad \text{ai}\downarrow \frac{S\{\circ\}}{S[a, \bar{a}]} \quad \text{s} \frac{S([R, T], U)}{S[(R, U), T]} \quad \text{q}\downarrow \frac{S\langle [R, U]; [T, V] \rangle}{S\langle [R; T], \langle U; V \rangle \rangle}
\end{array}$$

Fig. 2. System BV

Two systems \mathcal{S} and \mathcal{S}' are *equivalent* if for every proof of a structure T in system \mathcal{S} , there exists a proof of T in system \mathcal{S}' , and vice versa.

The system $\{\circ\downarrow, \text{ai}\downarrow, \text{s}, \text{q}\downarrow\}$, shown in Figure 2, is denoted by BV, and called *basic system V*. The rules of the system are called *unit* ($\circ\downarrow$), *atomic interaction* ($\text{ai}\downarrow$), *switch* (s) and *seq* ($\text{q}\downarrow$).

3 Removing the Equalities for Unit

In this section, we present a system equivalent to system BV where the application of inference rules is explicit with respect to equalities for unit. We assume that these rules are applied to structures which are in normal form. However, this is not restrictive since a normal form of a structure can be equivalently obtained by applying the terminating and confluent term rewriting system resulting from orienting the equalities for negation and unit in Figure 1 from left to right [7]. Hence, the equalities for unit and negation can be equivalently removed from the underlying equational theory by considering only those structures that are in normal form.

$$\begin{array}{c}
\text{ax} \frac{}{[a, \bar{a}]} \quad \text{s}_1 \frac{S([R, W], T)}{S[(R, T), W]} \\
\text{ai}_1\downarrow \frac{S\{R\}}{S[R, [a, \bar{a}]]} \quad \text{ai}_2\downarrow \frac{S\{R\}}{S(R, [a, \bar{a}])} \quad \text{ai}_3\downarrow \frac{S\{R\}}{S\langle R; [a, \bar{a}] \rangle} \quad \text{ai}_4\downarrow \frac{S\{R\}}{S\langle [a, \bar{a}]; R \rangle} \\
\text{q}_1\downarrow \frac{S\langle [R, T]; [U, V] \rangle}{S\langle [R; U], \langle T; V \rangle \rangle} \quad \text{q}_2\downarrow \frac{S\langle R; T \rangle}{S[R, T]} \quad \text{q}_3\downarrow \frac{S\langle [W, T]; U \rangle}{S[W, \langle T; U \rangle]} \quad \text{q}_4\downarrow \frac{S\langle T; [W, U] \rangle}{S[W, \langle T; U \rangle]}
\end{array}$$

Fig. 3. System BVu

Definition 1. *The system in Figure 3 is called unit-free BV or BVu. The equalities for unit do not apply to system BVu.*

Theorem 1. [9] *System BV and system BVu are equivalent.*

The inference rules of system BVu allow the unit to be completely removed from the language of the BV structures. Furthermore, trivial application of inference rules are not possible in system BVu.

4 Removing the Equalities for Commutativity

In this section, we will remove the equalities for commutativity from the equational theory underlying system BVu by making the role played by these equalities explicit in the inference rules. We first need some modifications on the inference rules:

Definition 2. *We put the following restriction on system BVu : The structures W in the rules are restricted to atoms, copar structures and seq structures. In other words, structure W is not a proper par structure. We will call this system unit-free lazy BV or BVul .*

Proposition 1. *System BV and system BVul are equivalent.*

Proof: Observe that the rules in BVul are instances of the rules in BVu with restrictions on switch and seq rules. The case where W is a proper par structure is derivable in BVul : the case of the rule $\text{q}_4\downarrow$ being analogous to the case for the rule $\text{q}_3\downarrow$, for the rules s , and $\text{q}_3\downarrow$ take the following derivations:

$$\begin{array}{c} \frac{S([R, T, V], U)}{S([([R, U], T), V])} \\ \text{s}_1 \\ \frac{S([([R, U], T), V])}{S([(R, T), U], V]} \\ \text{s}_1 \\ \frac{S([(R, T), U], V]}{S[(R, T), [U, V]]} \\ = \end{array} \qquad \begin{array}{c} \frac{S\langle [R, V, T]; U \rangle}{S[R, \langle [V, T]; U \rangle]} \\ \text{q}_3\downarrow \\ \frac{S[R, \langle [V, T]; U \rangle]}{S[R, [V, \langle T; U \rangle]]} \\ \text{q}_3\downarrow \\ \frac{S[R, [V, \langle T; U \rangle]]}{S[[R, V], \langle T; U \rangle]} \\ = \end{array}$$

Definition 3. *The system in Figure 4 is called commutativity-free BV or BVc , where W is either an atom or a copar structure or a seq structure, and the equalities for unit and commutativity do not apply to BVc .*

Proposition 2. *System BV and system BVc are equivalent.*

Proof: Inference rules of BVc are instances of the inference rules of BV . The proof of the other direction is by inductive case analysis on the commutative application of the inference rules of BVul : let Π be the proof of R in BVul . By induction on Π , we construct a proof Π' of R in BVc .

- If Π is $ax \frac{}{[a, \bar{a}]}$, take the same rule in BVc . (observe that $ax \frac{}{[\bar{a}, a]}$ is an instance of this rule, also when commutativity does not apply, since \bar{a} is an atom, and $\bar{\bar{a}} = a$.)
- If $\text{ai}_1\downarrow$ is the last rule applied in Π , such that

$$\text{ai}_1\downarrow \frac{S\{R\}}{S\{R, [a, \bar{a}]\}} \quad , \quad \text{there are the following possibilities for } Q : \text{ If } \\ = \frac{}{Q}$$

- $Q = S[R, a, \bar{a}]$; take $\text{ai}_{11}\downarrow$.
- $Q = S(R, [a, \bar{a}])$; take $\text{ai}_{21}\downarrow$.
- $Q = S[a, \bar{a}, R]$; take $\text{ai}_{12}\downarrow$.
- $Q = S([a, \bar{a}], R)$; take $\text{ai}_{22}\downarrow$.
- $Q = S[a, R, \bar{a}]$; take $\text{ai}_{13}\downarrow$.
- $Q = S\langle R; [a, \bar{a}] \rangle$; take $\text{ai}_3\downarrow$.

- $Q = S\langle [a, \bar{a}]; R \rangle$; take $\text{ai}_4 \downarrow$.

– If s_1 is the last rule applied in Π , such that

$$\stackrel{\text{s}_1}{=} \frac{S\langle [R, W], T \rangle}{S\langle [R, T], W \rangle}, \text{ there are the following possibilities for } Q : \text{ If}$$

- $Q = S\langle [R, T], W \rangle$; take s_{11a} .
- $Q = S\langle [T, R], W \rangle$; take s_{12a} .
- $Q = S\langle W, [R, T] \rangle$; take s_{13a} .
- $Q = S\langle W, [T, R] \rangle$; take s_{14a} .
- $Q = S\langle [R, T, U], W \rangle$; take s_{15a} .
- $Q = S\langle W, [R, T, U] \rangle$; take s_{16a} .
- $Q = S'\langle [R, T], P, W \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take s_{11b} .
- $Q = S'\langle [T, R], P, W \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take s_{12b} .
- $Q = S'\langle W, P, [R, T] \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take s_{13b} .
- $Q = S'\langle W, P, [T, R] \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take s_{14b} .
- $Q = S'\langle [R, T, U], P, W \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take s_{15b} .
- $Q = S'\langle W, P, [R, T, U] \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take s_{16b} .

– If $\text{q}_1 \downarrow$ is the last rule applied in Π , such that

$$\stackrel{\text{q}_1 \downarrow}{=} \frac{S\langle [R, T]; [U, V] \rangle}{S\langle [R; U], [T; V] \rangle}, \text{ there are the following possibilities for } Q : \text{ If}$$

- $Q = S\langle [R; U], [T; V] \rangle$; take $\text{q}_{11} \downarrow$.
- $Q = S'\langle [R; U], P, [T; V] \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take $\text{q}_{12} \downarrow$.

– If $\text{q}_2 \downarrow$ is the last rule applied in Π , such that

$$\stackrel{\text{q}_2 \downarrow}{=} \frac{S\langle R; T \rangle}{S\langle [R, T] \rangle}, \text{ there are the following possibilities for } Q : \text{ If}$$

- $Q = S\langle [R, T] \rangle$; take $\text{q}_{21} \downarrow$.
- $Q = S\langle [T, R] \rangle$; take $\text{q}_{22} \downarrow$.
- $Q = S'\langle [R, P, T] \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take $\text{q}_{23} \downarrow$.
- $Q = S'\langle [T, P, R] \rangle$ such that $S\{ \ } = S'\{ \ }, P$; take $\text{q}_{24} \downarrow$.

– If $\mathfrak{q}_3\downarrow$ is the last rule applied in Π , such that

$$\mathfrak{q}_3\downarrow \frac{S\langle [W, T]; U \rangle}{S[W, \langle T; U \rangle]} = \frac{\quad}{Q}, \text{ there are the following possibilities for } Q : \text{ If}$$

- $Q = S[W, \langle T; U \rangle]; \text{ take } \mathfrak{q}_{31}\downarrow.$
- $Q = S[\langle T; U \rangle, W]; \text{ take } \mathfrak{q}_{32}\downarrow.$
- $Q = S'[W, P, \langle T; U \rangle]$ such that $S\{ \} = S'[\{ \}, P]; \text{ take } \mathfrak{q}_{33}\downarrow.$
- $Q = S'[\langle T; U \rangle, P, W]$ such that $S\{ \} = S'[\{ \}, P]; \text{ take } \mathfrak{q}_{34}\downarrow.$

– If $\mathfrak{q}_4\downarrow$ is the last rule applied in Π , such that

$$\mathfrak{q}_4\downarrow \frac{S\langle T; [W, U] \rangle}{S[W, \langle T; U \rangle]} = \frac{\quad}{Q}, \text{ there are the following possibilities for } Q : \text{ If}$$

- $Q = S[W, \langle T; U \rangle]; \text{ take } \mathfrak{q}_{41}\downarrow.$
- $Q = S[\langle T; U \rangle, W]; \text{ take } \mathfrak{q}_{42}\downarrow.$
- $Q = S'[W, P, \langle T; U \rangle]$ such that $S\{ \} = S'[\{ \}, P]; \text{ take } \mathfrak{q}_{43}\downarrow.$
- $Q = S'[\langle T; U \rangle, P, W]$ such that $S\{ \} = S'[\{ \}, P]; \text{ take } \mathfrak{q}_{44}\downarrow.$

5 Reducing the Nondeterminism

In a proof search episode, inference rules can be applied to a structure in many different ways, however only few of these applications can lead to a proof. For example, to the structure $[(a, b), \bar{a}, \bar{b}]$ switch rule can be applied bottom-up in twelve different ways, but only two of these instances can lead to a proof. With the below definition, we will redesign the inference rules such that the instances of the inference rules which do not provide a proof will not be possible. For an extensive exposure to these ideas the reader is referred to [10].

Definition 4. *Given a structure S , the notation $\text{at } S$ indicates the set of all the atoms appearing in S . Let lazy interaction switch be the rule*

$$\text{lis} \frac{S([R, W], T)}{S[(R, T), W]},$$

where structure W is not a proper par structure and $\text{at } \bar{W} \cap \text{at } R \neq \emptyset$. The following rules are called interaction seq rule 1, lazy interaction seq rule 3, and lazy interaction seq rule 4, respectively,

$$\text{iq}_1\downarrow \frac{S\langle [R, T]; [U, V] \rangle}{S[\langle R; U \rangle, \langle T; V \rangle]} \quad \text{liq}_3\downarrow \frac{S\langle [R, W]; T \rangle}{S[W, \langle R; T \rangle]} \quad \text{liq}_4\downarrow \frac{S\langle T; [R, W] \rangle}{S[W, \langle T; R \rangle]}$$

where structure W is not a proper par structure and, in $\text{iq}_1\downarrow$, $\text{at}\bar{R} \cap \text{at}T \neq \emptyset$ and $\text{at}\bar{U} \cap \text{at}V \neq \emptyset$; in $\text{liq}_3\downarrow$ and in $\text{liq}_4\downarrow$, $\text{at}\bar{R} \cap \text{at}W \neq \emptyset$. The system resulting from replacing the rules s_1 , $q_1\downarrow$, $q_3\downarrow$, and $q_4\downarrow$, in BVu with the rule lis , $\text{iq}_1\downarrow$, $q_2\downarrow$, $\text{liq}_3\downarrow$, and $\text{liq}_4\downarrow$ is called interaction system BV , or BVi .

Theorem 2. [10] *System BV and system BVi are equivalent.*

With the below definition, we will combine the ideas from systems BVc and BVi in a single system, that is, we will impose the restrictions on the rules of BVi analogously on the inference rules of system BVc . This way, we will obtain a system where the equalities for unit and commutativity are redundant, and non-determinism is reduced.

Definition 5. *Let commutativity-free interaction system BV or system BVci be the system obtained by imposing the following restrictions on system BVc : in the rules s_{11a} , s_{12a} , s_{13a} , s_{14a} , s_{11b} , s_{12b} , s_{13b} , s_{14b} we have $\text{at}\bar{R} \cap \text{at}W \neq \emptyset$; in the rules s_{15a} , s_{16a} , s_{15b} , s_{16b} we have $\text{at}(\bar{R}, \bar{U}) \cap \text{at}W \neq \emptyset$; in the rules $q_{11}\downarrow$, $q_{12}\downarrow$ we have $\text{at}\bar{R} \cap \text{at}T \neq \emptyset$ and $\text{at}\bar{U} \cap \text{at}V \neq \emptyset$; in the rules $q_{31}\downarrow$, $q_{32}\downarrow$, $q_{33}\downarrow$, $q_{34}\downarrow$ we have $\text{at}\bar{W} \cap \text{at}T \neq \emptyset$; in the rules $q_{41}\downarrow$, $q_{42}\downarrow$, $q_{43}\downarrow$, $q_{44}\downarrow$ we have $\text{at}\bar{W} \cap \text{at}U \neq \emptyset$.*

Theorem 3. [10] *System BV and system BVci are equivalent.*

Proof: Follows immediately from Proposition 2 and Theorem 2.

6 From Inference Rules to Term Rewriting Rules

The systems in the calculus of structures can be represented as term rewriting systems modulo equational theories.⁵ In such a representation, the notion of a structure is replaced by a notion of term, considering terms over variables. Thus, bottom up application of an inference rule is represented as a rewriting rule that rewrites the conclusion to the premise of the inference rule. Similarly, inference rules with conditions are represented as conditional rewriting rules. For instance, consider the following rewrite rules for the inference rules switch and interaction seq rule 1, respectively:

$$\begin{aligned} s & : [(R, T), U] \rightarrow ((R, U), T) \\ \text{iq}_1\downarrow & : [\langle R; U \rangle, \langle T; V \rangle] \rightarrow \langle [R, T]; [U, V] \rangle \text{ if } \text{at}\bar{R} \cap \text{at}T \wedge \text{at}\bar{U} \cap \text{at}V \end{aligned}$$

Such rewrite rules are applied modulo the equational theory underlying the proof theoretical system. Because we use structures as terms the equalities for context closure and singleton become redundant. This leaves us with only equalities for associativity for system BVci when expressed as term rewriting system. In the next section, by resorting to a list representation of n-ary terms which captures associativity, we will present an implementation of the term rewriting system for system BVci .

⁵ For an indepth exposure on the correspondence between systems of the calculus of structures and term rewriting systems the reader is referred to [7].

7 Implementation in TOM

TOM is a language extension which adds syntactic and associative pattern-matching facilities to existing languages like Java, C, and OCaml. This hybrid approach is particularly well-suited when describing transformations of structured entities like trees/terms and XML documents, for example. In this work, we use TOM, combined with Java, to implement our prototype.

An interesting feature of the language is to provide support for matching modulo sophisticated theories. In particular, pattern-matching modulo associativity and neutral element (also known as list-matching) is both useful and efficient to model the exploration of a search space.

For expository reasons, we assume that TOM only adds two new constructs: `%match` and *back-quote* (`'`). The first construct is similar to the `match` primitive of ML and related languages: given a term (called subject) and a list of pairs pattern-action, the `match` primitive selects a pattern that matches the subject and performs the associated action. The second construct is a mechanism that allows one to easily build ground terms over a defined signature. This operator, called *back-quote*, is followed by a well-formed term, written in prefix notation.

A main originality of this system is to be data-structure independent. This means that a *mapping* has to be defined to connect algebraic data-structure, on which pattern matching is performed, to low-level data-structures, that correspond to the implementation. Most of the time, TOM is used in conjunction with the ApiGen system [19], which generates abstract syntax tree implementations and a mapping, from a given datatype definition. The input format for ApiGen is a concise language defining sorts and constructors for the abstract syntax. The output is an efficient, in time and memory, Java implementation for this datatype. This implementation is characterized by strong typing and maximal sub-term sharing, providing both memory efficiency and constant-time equality checking.

For an interested reader, design and implementation issues related to TOM are presented in [13, 12].

7.1 Data structures

A main difficulty, when implementing the systems of the calculus of structures, is to find a good representation for the *par*, *cop*, and *seq* structures ($[R, T]$, (R, T) and $\langle R; T \rangle$). In our implementation of BVci, we considered these constructors as unary operators which take a *list of structures* as argument. Using ApiGen, the considered data-type can be described by the following signature:

```
module Struct
  public sorts Struc StrucPar StrucCop StrucSeq
  abstract syntax
    a -> Struc
    b -> Struc
    ...other atom constants
```

```

neg(Struc)          -> Struc
par(StrucPar)      -> Struc
cop(StrucCop)      -> Struc
seq(StrucSeq)      -> Struc
concPar( Struc* )  -> StrucPar
concCop( Struc* ) -> StrucCop
concSeq( Struc* ) -> StrucSeq

```

The grammar rule `par(StrucPar) -> Struc` defines a unary operator `par` of sort `Struc` which takes a `StrucPar` as unique argument. The grammar rule `concPar(Struc*) -> StrucPar` defines the `concPar` operator of sort `StrucPar`. The special syntax `Struc*` indicates that `concPar` is a *list-operator* which takes a list of `Struc` as argument. Thus, by combining `par` and `concPar` it becomes possible to represent the structure $[a, [b, c]]$ by `par(concPar(a,b,c))`. Note that structures are flattened. In TOM, list-operators are interesting because their arity is not fixed. Thus, `concPar(a,b,c)` corresponds to a list of 3 elements, `concPar(a)` corresponds to a list of single element, namely `a`, whereas `concPar()` denotes the empty list. (R, T) and $\langle R; T \rangle$ are represented in a similar way, using `cop`, `seq`, `concCop`, and `concSeq`.

A problem with this approach is that we can manipulate objects, like `par(concPar())`, which do not necessarily correspond to intended structures. It is also possible to have several representations for the same structure. Hence, `par(concPar(a))` and `cop(concCop(a))` both denote the structure `a`. To avoid such situations, we have encoded, in the defined mapping, a notion of canonical form which avoids building uninteresting terms. Thus, we ensure that

- `[]`, `\langle \rangle` and `()` are reduced when containing only one sub-structure:
 $par(concPar(x)) \rightarrow x$
- nested structures are flattened:
 $par(concPar(..., par(concPar(x_1, ..., x_n)), ...)) \rightarrow par(concPar(..., x_1, ..., x_n, ...))$
- subterms are sorted (according to a given total lexical order $<$):
 $concPar(..., x_i, ..., x_j, ...) \rightarrow concPar(..., x_j, ..., x_i, ...)$ if $x_j < x_i$.

This notion of canonical form allows us to efficiently check if two terms represents the same structure with respect to commutativity of those connectors.

7.2 Rewrite rules

The rewrite rules define the deduction steps in system `BVci`. They are implemented by a *match* construct which matches a sub-term with the left-hand side of the rewrite rule. Then the right-hand side of the rule builds the deduced structure.

For instance, the rules $[(R, T), U] \rightarrow ([R, U], T)$ and $[(R, T), U] \rightarrow ([T, U], R)$ are encoded by the following match construct.

```

%match(Struc t) {
  par(concPar(X1*, cop(concCop(R*, T*)), X2*, U, X3*)) -> {

```

```

if('T*.isEmpty() || 'R*.isEmpty() ) {
} else {
  StrucPar context = 'concPar(X1*,X2*,X3*);
  if(canReact('R*,'U)) {
    StrucPar parR = cop2par('R*);
    // transform a StrucCop into a StrucPar
    Struc elt1 = 'par(concPar(
      cop(concCop(par(concPar(parR*,U)),T*)),context*));
    c.add(elt1);
  }
  if(canReact('T*,'U)) {
    StrucPar parT = cop2par('T*);
    Struc elt2 = 'par(concPar(
      cop(concCop(par(concPar(parT*,U)),R*)),context*));
    c.add(elt2);
  } } } }

```

We ensure that we do not execute the right-hand side of the rule if either R or T are empty lists. The other tests implements the restrictions on the application of the rules detailed in Section 5 for reducing the non-determinism. This is done by using an auxiliary predicate function `canReact(a,b)` which collects all atoms in structures `a` and `b` and returns `true` only if `a` contains at least one atom which is contained in a negated form in `b`. This function can be made efficient by using the features of the host language of TOM, in our case, by using an efficient hash-set implementation in Java. The remaining rules are expressed in a similar way.

7.3 Strategy

When designing a proof search procedure, implementing the set of inference rules is very important, but this is only one part of the job. The second part consists in defining a strategy which describes how to apply the rules. In rule based systems like ELAN or MAUDE, it is very easy to describe such strategies, using primitive operators or meta-level capabilities. However, in some cases, it may be difficult to express strategies which take time and space into consideration. In ELAN for example, the search is implemented using a backtracking mechanism. This is a good approach to implement depth-first search strategies. However, while efficient in space, such a strategy may lead to explore infinite branches and non-terminating programs. On the other hand, breadth-first search, as in MAUDE, usually terminates when a proof exists, but the memory needed can be considerably huge. In languages like ELAN, the backtracking based mechanism makes the definition of strategies difficult.

In TOM, there is no particular support for implementing search space exploration strategies. Thus, the search space has to be handled explicitly. On one hand, this leads to more complex implementations, but on the other, this allows us to define very fine and efficient search strategies.

In our implementation, we have implemented a *global search* strategy which combines the advantages of both depth- and breadth-first search strategies: given an ordered list of elements, we select the first term and compute the set of its successors by applying all rules at every position. Implementing a breadth-first search strategy can be done by adding this resulting set of elements at the end of the list. To implement a depth-first search strategy, the set has to be inserted at the beginning of the list. In our case, the elements of the set are inserted and inter-mixed in the initial list, according to the given order. In one sense, the order implements a heuristic which characterizes the *interesting* structures that have to be explored first, since they may lead to the proof in an efficient way. This mechanism is iterated until the main list contains the unit element.

The order used for those lists is the main parameter of the method, and can be changed at will to find a suitable order for fast proof finding.

8 Summary and Discussions:

We have presented a proof search implementation of system **BV** of the calculus of structures by resorting to the correspondence between the systems of the calculus of structures and term rewriting systems modulo equational theories. The term rewriting rules corresponding to inference rules of system **BV** are applied modulo an equational theory which admits associativity, commutativity and a unit for different logical operators. By making the role played by the equalities for unit and commutativity in the application of the inference rules explicit, we presented a system equivalent to system **BV** where these equalities become redundant. This way, we expressed associativity in a list representation.

Our implementation, in Java, uses the pattern matching preprocessor TOM in order to integrate term rewriting features into Java. By exploiting the expressive power due to Java, we provided a search algorithm which combines different search strategies and heuristic search. The source code of the implementation is available at the TOM distribution⁶. A representative applet of this implementation can be found at <http://tom.loria.fr/examples/structures/BV.html>.

Our implementation respects a common scheme which is shared by all the systems of the calculus of structures for classical logic, modal logics, and linear logic. For this reason, our implementation can be easily generalized for implementing different tools for these systems, also by employing different search strategies at will.

In [8], Kahramanoğulları presents an implementation of system **BV** in MAUDE by using the *search* function of this system which implements breadth-first search on the search space of term rewriting systems modulo equational theories. Although this implementation benefits from the simple high-level language of MAUDE, implementation of a certain strategy, different from breadth-first search, remains complicated due to the complex meta-level language. However, in the implementation presented in this paper, the availability of the Java language provides a great ease on implementing different search strategies.

⁶ <http://tom.loria.fr>

References

1. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*, volume 1. Cambridge University Press, 1998.
2. Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. PhD thesis, Technische Universität Dresden, 2003.
3. Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 347–361. Springer-Verlag, 2001.
4. Paola Bruscoli. A purely logical account of sequentiality in proof search. In Peter J. Stuckey, editor, *Logic Programming, 18th International Conference*, volume 2401 of *Lecture Notes in Computer Science*, pages 302–316. Springer-Verlag, 2002.
5. Alessio Guglielmi. A system of interaction and structure. Technical Report WV-02-10, TU Dresden, 2002. To app. in ACM Transactions on Computational Logic.
6. Alessio Guglielmi. Mismatch. Available on the web at <http://iccl.tu-dresden.de/~guglielm/p/AG9.pdf>, 2003.
7. Steffen Hölldobler and Ozan Kahramanoğulları. From the calculus of structures to term rewriting systems. Technical Report WV-04-03, TU Dresden, 2004.
8. Ozan Kahramanoğulları. Implementing system BV of the calculus of structures in maude. In L. A. i Alemany and P. Égré, editors, *Proc. of the ESSLLI-2004 Student Session*, pages 117–127, Université Henri Poincaré, Nancy, France, 2004.
9. Ozan Kahramanoğulları. System BV without the equalities for unit. In C. Aykanat, T. Dayar, and I. Körpeoğlu, editors, *Proc. of the 19th Int. Symp. on Computer and Information Sciences, ISCIS'04*, volume 3280 of *LNCS*. Springer, 2004.
10. Ozan Kahramanoğulları. Reducing the non-determinism in the calculus of structures. Technical report, TU Dresden, 2005. Available at <http://www.informatik.uni-leipzig.de/~ozan/reducingNondet.pdf>.
11. Ozan Kahramanoğulları. System BV is NP-complete. to appear in Proceedings of WoLLIC'05, <http://www.informatik.uni-leipzig.de/~ozan/BVnpc.pdf>, 2005.
12. Claude Kirchner, Pierre-Etienne Moreau, and Antoine Reilles. Formal validation of pattern matching code. 2005. To appear in PPDP '05: Proceedings of the 7th ACM SIGPLAN Int. Conf. on Principles and practice of declarative programming.
13. Pierre-Etienne Moreau, Christophe Ringeissen, and Marian Vittek. A Pattern Matching Compiler for Multiple Target Languages. In G. Hedin, editor, *12th Conference on Compiler Construction, Warsaw (Poland)*, volume 2622 of *LNCS*, pages 61–76. Springer-Verlag, May 2003.
14. Charles Stewart and Phiniki Stouppa. A systematic proof theory for several modal logics. Technical Report WV-03-08, TU Dresden, 2003. Accepted at Advances in Modal Logic 2004, to app. in proceedings published by King's College Publications.
15. Lutz Straßburger. A local system for linear logic. In Matthias Baaz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2002*, volume 2514 of *LNAI*, pages 388–402. Springer-Verlag, 2002.
16. Lutz Straßburger. *Linear Logic and Noncommutativity in the Calculus of Structures*. PhD thesis, TU Dresden, 2003.
17. Alwen Fernanto Tiu. Properties of a logical system in the calculus of structures. Technical Report WV-01-06, Technische Universität Dresden, 2001.
18. Alwen Fernanto Tiu. A local system for intuitionistic logic: Preliminary results. <http://www.loria.fr/~tiu/localint.pdf>, 2005.
19. Mark van den Brand, Pierre-Etienne Moreau, and Jurgen Vinju. A generator of efficient strongly typed abstract syntax trees in java. Technical report SEN-E0306, ISSN 1386-369X, CWI, Amsterdam (Holland), November 2003.

$$\begin{array}{c}
ax \overline{[a, \bar{a}]} \\
\text{ai}_{11} \downarrow \frac{S\{R\}}{S[R, a, \bar{a}]} \quad \text{ai}_{12} \downarrow \frac{S\{R\}}{S[a, \bar{a}, R]} \quad \text{ai}_{13} \downarrow \frac{S\{R\}}{S[a, R, \bar{a}]} \\
\text{ai}_{21} \downarrow \frac{S\{R\}}{S(R, [a, \bar{a}])} \quad \text{ai}_{22} \downarrow \frac{S\{R\}}{S([a, \bar{a}], R)} \quad \text{ai}_3 \downarrow \frac{S\{R\}}{S\langle R; [a, \bar{a}] \rangle} \quad \text{ai}_4 \downarrow \frac{S\{R\}}{S\langle [a, \bar{a}]; R \rangle} \\
\text{s}_{11a} \frac{S([R, W], T)}{S([R, T], W)} \quad \text{s}_{12a} \frac{S([R, W], T)}{S([T, R], W)} \quad \text{s}_{13a} \frac{S([R, W], T)}{S[W, (R, T)]} \quad \text{s}_{14a} \frac{S([R, W], T)}{S[W, (T, R)]} \\
\text{s}_{15a} \frac{S([(R, U), W], T)}{S[(R, T, U), W]} \quad \text{s}_{16a} \frac{S([(R, U), W], T)}{S[W, (R, T, U)]} \\
\text{s}_{11b} \frac{S([(R, W], T), P]}{S[(R, T), P, W]} \quad \text{s}_{12b} \frac{S([(R, W], T), P]}{S[(T, R), P, W]} \\
\text{s}_{13b} \frac{S([(R, W], T), P]}{S[W, P, (R, T)]} \quad \text{s}_{14b} \frac{S([(R, W], T), P]}{S[W, P, (T, R)]} \\
\text{s}_{15b} \frac{S([(R, U), W], T), P]}{S[(R, T, U), P, W]} \quad \text{s}_{16b} \frac{S([(R, U), W], T), P]}{S[W, P, (R, T, U)]} \\
\text{q}_{11} \downarrow \frac{S\langle [R, T]; [U, V] \rangle}{S\langle [R; U], \langle T; V \rangle \rangle} \quad \text{q}_{12} \downarrow \frac{S\langle \langle [R, T]; [U, V] \rangle, P \rangle}{S\langle [R; U], P, \langle T; V \rangle \rangle} \\
\text{q}_{21} \downarrow \frac{S\langle R; T \rangle}{S[R, T]} \quad \text{q}_{22} \downarrow \frac{S\langle R; T \rangle}{S[T, R]} \quad \text{q}_{23} \downarrow \frac{S\langle [R; T], P \rangle}{S[R, P, T]} \quad \text{q}_{24} \downarrow \frac{S\langle \langle R; T \rangle, P \rangle}{S[T, P, R]} \\
\text{q}_{31} \downarrow \frac{S\langle [W, T]; U \rangle}{S[W, \langle T; U \rangle]} \quad \text{q}_{32} \downarrow \frac{S\langle [W, T]; U \rangle}{S[\langle T; U \rangle, W]} \quad \text{q}_{33} \downarrow \frac{S\langle \langle [W, T]; U \rangle, P \rangle}{S[W, P, \langle T; U \rangle]} \quad \text{q}_{34} \downarrow \frac{S\langle \langle [W, T]; U \rangle, P \rangle}{S[\langle T; U \rangle, P, W]} \\
\text{q}_{41} \downarrow \frac{S\langle T; [W, U] \rangle}{S[W, \langle T; U \rangle]} \quad \text{q}_{42} \downarrow \frac{S\langle T; [W, U] \rangle}{S[\langle T; U \rangle, W]} \quad \text{q}_{43} \downarrow \frac{S\langle \langle T; [W, U] \rangle, P \rangle}{S[W, P, \langle T; U \rangle]} \quad \text{q}_{44} \downarrow \frac{S\langle \langle T; [W, U] \rangle, P \rangle}{S[\langle T; U \rangle, P, W]}
\end{array}$$

Fig. 4. System BVC

Abstract derivations, equational logic and interpolation (extended abstract)

Gerard R. Renardel de Lavalette

Department of Mathematics and Computing Science
University of Groningen, the Netherlands

Abstract. We define abstract derivations for equational logic and use them to prove the interpolation property.

1 Introduction

In this paper, we introduce a notion of abstract derivation for equational logic and use it to prove the interpolation property. The work reported here has the character of an experiment, intended to sharpen and test our initially rather vague ideas about abstract derivations. These ideas came from a feeling of dissatisfaction about the low level of abstraction in traditional proof theory, where derivations are trees consisting of sequents, i.e. strings of symbols, with great redundancy by repeating in every proof step the parts of a sequent that do not change. As a consequence, operations on derivations (normalization, e.g. via cut elimination, interpolant extraction) only admit a precise definition in local terms, on the level of proof steps, and global properties are left to intuition. More particular, in [4] and [15] we were able to prove interpolation for several fragments of intuitionistic propositional logic, but we admit that full understanding of what is really happening on a global level in our proofs is lacking, because of the reasons sketched above.

The choice for equational logic as a basis for the elaboration of our ideas was motivated by two reasons. Firstly, the prooftheoretical proof for interpolation in equational logic by Rodenburg in [12] is rather involved, especially when compared with the algebraic proof by the same author in [11], and constitutes a challenge for proof theory to try to fill the gap with model theory. Secondly, equational logic is a very general and rather strong system: it is undecidable, and virtually all propositional, modal and linear logics can be embedded in it. Moreover, proof theory for equational logic seems to be a rather underdeveloped area of research.

The experiment in abstract derivation design is by no means finished yet, but we think the time has come to report on the first results, in the hope that this may lead to constructive comments. After a short sketch of the ideas behind abstract derivations and a survey of interpolation in equational logic, we present the main definitions, prove soundness, derive some relevant properties and prove the interpolation theorem. We end with some suggestions for further research.

Because of space constraints, most proofs have been omitted in this extended abstract. See [5] for a full version.

1.1 Abstract derivations

The notion of abstract derivation we present here is based on an abstract view on the traditional notion of derivation for equational logic. We see two fundamental and general principles at work here. Firstly the idea of *matching*: e.g. $f(s) \equiv f(t)$ follows from $s \equiv t$ since the f in $f(s)$ matches the f in $f(t)$. Secondly the idea of *abstraction*, or its dual instantiation: $r(t) \equiv s(t)$ follows from $r(x) \equiv s(x)$ since the latter equation is to be read as ' $r(x)$ and $s(x)$ are equal for all x '. We consider the fact that \equiv is an equivalence relation to be less fundamental for the proof system: it is of course essential for a logic called equational, but another interesting logic may be obtained if we would replace \equiv by \leq , a partial order.

Now the idea of abstract derivation can be explained as follows. Start with an abstract representation of terms: the obvious choice is trees consisting of nodes labeled by signature elements (is there an alternative?). Then abstract equations become pairs of nodes, and binary relations on nodes are sets of equations. The proof rules become operations on relations, so e.g. R^e , the least equivalence relation containing R , corresponds with application of the proof rules of reflexivity, symmetry and transitivity. For the congruence rule, we put $\text{cong}(R) = M \cap \text{lift}(R)$. Here $\text{lift}(R)$ is the collection of pairs (k, l) such that k and l represent terms with an equal number of direct subterms, and for all k', l' representing corresponding direct subterms we have $(k', l') \in R$. M is the matching relation: we have that $(k, l) \in M$ implies that k and l are labeled by the same signature element, but the converse implication does not hold in general. The role of M is to regulate the development of the abstract derivation. In order to deal with the instantiation rule, we work with abstractions, a kind of inverse of substitutions. Abstractions α are partial mappings on nodes yielding variables, and when applied to a term structure T they yield a new term structure T_α where some terms are replaced by variables. Now an abstract derivation is a tuple $D = \langle T, E, M, \alpha \rangle$ where T is an abstract term structure, E is a relation on nodes in T representing a set of equations, M is a matching relation, and α is an abstraction. An operator $\text{der} = \mu R. (E \cup \text{cong}(R))^e$, defined in terms of T and M and applied to E , yields the set $\text{der}(E)$ of equations that are derived in D from E , where E is now seen as a relation on nodes in the term structure T_α . Moreover, D is required to be wellfounded, and the abstraction α should be justified by D . See Definition 5 for the details.

For the moment, we choose to represent the steps in the proof and its conclusion implicitly: only the premiss E is present in the representation, its consequences after n proof steps can be *computed* by applying the operator $\lambda R. (E \cup \text{cong}(R))^e$ n times. For our experiment with interpolation, this representation will do, but for other purposes it may be useful to work with other, more explicit variants.

For conciseness' sake, we will often drop the epitheton *abstract*, and refer to derivations when we mean abstract derivations.

1.2 Equational logic

A *signature* SIG is a collection of constants and function symbols. The arity of a signature element $s \in \text{SIG}$ is given by $\text{arity}(s) \in \mathbb{N}$. VAR is an infinite collection of variable symbols, with $\text{SIG} \cap \text{VAR} = \emptyset$. For variables $x \in \text{VAR}$, $\text{arity}(x) = 0$. Terms built from signature elements and variables are defined as usual; we write $\text{sig}(t)$ for the collection of signature elements that occur in term t . $[x := s]t$ denotes substitution of term s for all occurrences of variable x in term t , and $[x_i := s_i]_{i < n} t$ is used for simultaneous substitution. Equations between terms are denoted by $s \equiv t$: they are the formulas φ of equational logic. Sequents are of the form $\Gamma \vdash \varphi$ where Γ is a collection of equations. As usual, the derivability relation \vdash is the least relation satisfying

$$\begin{array}{ll}
\text{assumption} & \Gamma \vdash \varphi \text{ if } \varphi \in \Gamma \\
\text{reflexivity} & \Gamma \vdash t \equiv t \\
\text{symmetry} & \Gamma \vdash s \equiv t \Rightarrow \Gamma \vdash t \equiv s \\
\text{transitivity} & \Gamma \vdash r \equiv s \ \& \ \Gamma \vdash s \equiv t \Rightarrow \Gamma \vdash r \equiv t \\
\text{congruence} & \Gamma \vdash s_0 \equiv t_0 \ \& \ \dots \ \& \ \Gamma \vdash s_{n-1} \equiv t_{n-1} \\
& \Rightarrow \Gamma \vdash f(s_0, \dots, s_{n-1}) \equiv f(t_0, \dots, t_{n-1}) \\
\text{instantiation} & \Gamma \vdash \varphi \Rightarrow \Gamma \vdash [x := t]\varphi
\end{array}$$

The instantiation rule can be generalized to

$$\begin{array}{l}
\Gamma \vdash t_0 \equiv t_1 \ \& \ \Gamma \vdash t_1 \equiv t_2 \ \& \ \dots \ \& \ \Gamma \vdash t_{n-1} \equiv t_n \ \& \ \Gamma \vdash \varphi(x, \dots, x) \\
\Rightarrow \Gamma \vdash \varphi(t_0, \dots, t_n)
\end{array}$$

which says that we may substitute provably equal terms for different occurrences of the same variable. This principle underlies Definition 5 of abstract derivation.

1.3 Interpolation

Interpolation is the following property;

$$\begin{array}{l}
\text{if } \Gamma, \Delta \vdash \varphi, \text{ then there is a collection of formulae } \Theta \text{ such that} \\
\Gamma \vdash \theta \text{ for every } \theta \in \Theta, \ \Theta, \Delta \vdash \varphi, \text{ and } \text{sig}(\Theta) \subseteq \text{sig}(\Gamma) \cap \text{sig}(\Delta \cup \{\varphi\}).
\end{array}$$

Θ is called the *interpolant*. The first formulation and proof are by Craig in [3], first for classical predicate logic without function symbols; the case with function symbols is reduced to the former case by replacing function symbols by predicates. Craig's proof is prooftheoretical and proceeds via proof normalization. Since then, interpolation has been shown for many logics, with either prooftheoretic or modeltheoretic means. All prooftheoretic proofs work with proof normalization, usually obtained via cut elimination. This can be seen as a disadvantage, since proof normalization may lead to an exponential increase in size.

Interpolation as defined above does not hold for equational logic, as is shown in the following counterexample: take

$$\Gamma = \{f(a) \equiv b, f(c) \equiv d\}, \Delta = \{a \equiv c\}, \varphi = (b \equiv d).$$

The only possible interpolant would be something like $(a \equiv c) \rightarrow (b \equiv d)$, but this is not expressible in equational logic.

However, the weaker version of interpolation with $\Delta = \emptyset$ is valid for equational logic. It was first proved by Rodenburg in [11] with a rather short and perspicuous algebraic proof, using Birkhoff's HSP theorem. (Birkhoff's theorem states that a class of algebras K is equational, i.e. characterized by a set of equations, iff it is a variety, i.e. closed under homomorphic images, subalgebras, and direct products; see e.g. [8].) Rodenburg's proof is not constructive in the sense that the proof does not contain an effective method to construct the interpolant. In [12], Rodenburg gives a prooftheoretical proof of the same theorem, which is constructive but not very perspicuous. The proof transforms a derivation in equational logic step by step into a derivation in a related system, from which an interpolant can be obtained easily. (As Rodenburg points out, an apparently different interpolant construction for equational logic is already implicit in the proof of the main theorem of [9] by Pigozzi.)

A constructive proof of interpolation for equational logic can also be extracted from the prooftheoretical proof for interpolation for predicate logic with function symbols given by Felscher in [6]. This proof (which is also rather involved) does not eliminate function symbols, but uses what is called *Takeuti's Lemma* (Felscher gives the obscure reference [14]) to eliminate spurious function symbols from a candidate interpolant. Takeuti's Lemma indicates when, in a derivation, a term t can be replaced by a variable x , so it is a kind of inverse substitution property. Our Lemma 5 is related to Takeuti's Lemma.

Let us see what has to be done for the construction of an interpolant for a provable sequent $\Gamma \vdash \varphi$ in equational logic. The general situation is: there are enough candidate interpolants Θ , e.g. Γ or $\{\varphi\}$, but in general they do not satisfy the signature condition $\text{sig}(\Theta) \subseteq \text{sig}(\Gamma) \cap \text{sig}(\varphi)$. This is caused by the proof rules, which may add signature elements to or eliminate them from the conclusion (observe that all proof rules leave the premiss Γ intact and only modify the conclusion φ). The congruence rule and the instantiation rule may add signature elements. In the transitivity rule, the term s is eliminated, and possibly some signature elements that occur in it. In that sense, the transitivity rule is comparable with the cut rule of propositional and predicate logic. There is no Transitivity Elimination Theorem, however, which would help us here. But we do have a partial result in that direction: Lemma 3, which splits an abstract derivation in two parts: in the first part no signature elements are added, and the second part contains no transitivity steps. With this lemma, we obtain interpolation for a weakening of equational logic, obtained by removing the instantiation rule. For interpolation in full equational logic, we need Lemma 5 to eliminate signature elements that were introduced in the conclusion by the instantiation rule, in a way comparable to Felscher's use of Takeuti's Lemma as described in the previous paragraph.

2 Preliminaries

For any set X , X^* denotes the collection of finite sequences of elements of X . If $xs \in X^*$, then $\text{lth}(xs)$ is the length of xs , and xs_i (where $0 \leq i < \text{lth}(xs)$) denotes the i -th element of xs . We write $x \in xs$ to denote that x occurs in the sequence xs , and the empty sequence is denoted by $()$.

If $R \subseteq X \times Y$ is a relation between X and Y , then we can extend R to $R^\otimes \subseteq X^* \times Y^*$ by defining

$$R^\otimes = \{(xs, ys) \mid \text{lth}(xs) = \text{lth}(ys) \wedge \forall i < \text{lth}(xs) (xs_i, ys_i) \in R\}$$

and likewise for functions, so

$$f^\otimes(x_0, \dots, x_{n-1}) = (f(x_0), \dots, f(x_{n-1}))$$

Composition of relations is defined as usual: $R \cdot S = \{(x, z) \mid \exists y(xRy \wedge ySz)\}$. If $f : X \rightarrow Y$, then $\ker(f) \subseteq X^2$ is the equivalence relation defined by

$$\ker(f) = \{(x, y) \mid f(x) = f(y)\}.$$

$\text{car}(R)$, the *carrier* of R , is the least set X with $R \subseteq X^2$, so

$$\text{car}(R) = \{x \mid \exists y((x, y) \in R \vee (y, x) \in R)\}$$

If f, g are partial functions, then we define $f:g$ (*f before g*) by

$$f:g = f \cup (g \upharpoonright (\text{dom}(g) - \text{dom}(f)))$$

Here \upharpoonright denotes restriction: $f \upharpoonright X = f \cap (X \times \text{rg}(f))$.

As usual, we write R^+ for the transitive closure of R , and R^* for the reflexive transitive closure. Moreover, R^e is the smallest equivalence relation containing R , so $R^e = (R \cup R^{-1})^*$.

Let $R \subseteq X^2$ and $Y \subseteq X$: we define

$$R \text{ respects } Y \text{ iff } R \subseteq Y^2 \cup (X - Y)^2$$

so if $(x, y) \in R$ then $x \in Y$ iff $y \in Y$. If $f : X \rightarrow Z$ is a partial function,

$$R \text{ respects } f \text{ iff } R \subseteq \ker(f) \cup (X - \text{dom}(f))^2$$

3 Term structures and abstractions

3.1 Forests

The idea to represent terms by trees is straightforward. The context for an abstract derivation will be a *forest*, i.e. a collection of trees that represent terms. A forest is a pair $F = \langle K, \text{arg} \rangle$, where K is a collection of nodes, and $\text{arg} : K \rightarrow K^*$; moreover, the relation $[\text{arg}]$ on K , defined by

$$[\text{arg}] = \{(k, l) \mid k \in \text{arg}(l)\}$$

is wellfounded. As a consequence, we have the following *induction principle for forests*: for all $L \subseteq K$,

$$\forall k(\arg(k) \subseteq L \rightarrow k \in L) \rightarrow L = K \quad (1)$$

The subnode relation \leq is defined as the transitive closure of $[\arg]^{-1}$. The *arity* of a node k is defined as the length of its sequence of children: $\text{arity}(k) = \text{lth}(\arg(k))$. When $0 \leq i < \text{arity}(k)$, we may write $\arg_i(k)$ for $(\arg(k))_i$, the i -th element of $\arg(k)$: so $\arg(k) = (\arg_0(k), \dots, \arg_{\text{arity}(k)-1}(k))$. A *path* in F is a finite nonempty sequence $(k_0, \dots, k_n) \in K^+$ satisfying $(k_i, k_{i+1}) \in [\arg]$ for $0 \leq i < n$. We call $L \subseteq K$ *arg-closed* if $\arg[L] \subseteq L^*$. For *arg-closed* l , we defined the restriction \arg_L of \arg by $\arg_L = \arg \cap (L \times L^*)$. It is obvious that $\langle L, \arg_L \rangle$ is a forest.

Parallel to (1), we have a recursion principle for forests: if $f : (K \otimes X) \rightarrow X$, where $K \otimes X = \{(k, (x_0, \dots, x_{n-1})) \mid n = \text{arity}(k), x_0, \dots, x_{n-1} \in X\}$, then there is a unique $g : K \rightarrow X$ with

$$g(k) = f(k)(g^{\otimes}(\arg(k)))$$

For later use, we define the operator lift on relations $R \subseteq K^2$ by

$$\text{lift}(R) = \arg \cdot R^{\otimes} \cdot \arg^{-1}$$

As a consequence, we see that

$$(k, l) \in \text{lift}(R) \Leftrightarrow \text{arity}(k) = \text{arity}(l) \wedge \forall i < \text{arity}(k) (\arg_i(k), \arg_i(l)) \in R$$

lift will be used for the congruence step in the derivations we will define later on.

Definition 1 (Term structures). A term structure over SIG is a triple $T = \langle K, \arg, \sigma \rangle$, where $\langle K, \arg \rangle$ is a forest, and $\sigma : K \rightarrow \text{SIG} \cup \text{VAR}$ preserves arity.

So a term structure is a forest where every node is labeled with a signature element or a variable, in such a way that the arity of node and label correspond. It is clear that every node k represents a term: to obtain this term, we define (with recursion) the term operator *term* by

$$\text{term}(k) = (\sigma(k))(\text{term}^{\otimes}(\arg(k)))$$

The formula operator *form* is defined in terms of *term*:

$$\text{form}(k, l) = (\text{term}(k) \equiv \text{term}(l))$$

term_T is extended to sets of nodes by $\text{term}(L) = \{\text{term}(k) \mid k \in L\}$; idem for *sig*. *form* is extended analogously to relations: $\text{form}(R) = \{\text{form}(k, l) \mid (k, l) \in R\}$.

The signature $\text{sig}(k)$ of a node k is the signature of the term represented by k , so $\text{sig}(k) = \text{sig}(\text{term}(k))$. We trust that this overloading of *sig* will not cause confusion. Observe that *sig* satisfies

$$\text{sig}(k) = (\{\sigma(k)\} \cap \text{SIG}) \cup \bigcup \text{sig}^{\otimes}(\arg(k))$$

We also define, for $\Sigma \subseteq \text{SIG}$:

$$\begin{aligned} K_\Sigma &= \{k \in K \mid \sigma(k) \in \Sigma\} = \sigma^{-1}[\Sigma] \\ K_{\overline{\Sigma}} &= \{k \in K \mid \text{sig}(k) \subseteq \Sigma\} = \text{sig}^{-1}[\wp(\Sigma)] \end{aligned}$$

Observe that $K_{\overline{\Sigma}}$ and K_Σ are different and even incomparable. K_Σ contains all nodes that represent terms with their principal signature element in Σ , while $K_{\overline{\Sigma}}$ contains nodes representing terms that are made from variables and elements of Σ . So $k \in K_{\overline{\Sigma}} - K_\Sigma$ if $\sigma(k) \in \text{VAR}$, and $k \in K_\Sigma - K_{\overline{\Sigma}}$ if $\sigma(k) \in \Sigma$, $\text{arg}(k) = (l)$ and $\sigma(l) \in \text{SIG} - \Sigma$.

For technical reasons, we work with abstractions, which are the dual of substitutions.

Definition 2 (Abstraction). Let $T = \langle K, \text{arg}, \sigma \rangle$ be a term structure. A partial mapping $\alpha : K \rightarrow \text{VAR}$ is an abstraction of T if $\text{rg}(\alpha) \cap \text{rg}(\sigma) = \emptyset$. T_α , the result of applying α to T , is defined as $T_\alpha = \langle K, \text{arg}_\alpha, \alpha : \sigma \rangle$, where arg_α is defined by

$$\begin{aligned} \text{arg}_\alpha(k) &= () \quad \text{if } k \in \text{dom}(\alpha) \\ &= \text{arg}(k) \quad \text{if } k \notin \text{dom}(\alpha) \end{aligned}$$

By the definition of $\alpha : \sigma$ (see section 2), we have

$$\begin{aligned} (\alpha : \sigma)(k) &= \alpha(k) \quad \text{if } k \in \text{dom}(\alpha) \\ &= \sigma(k) \quad \text{if } k \notin \text{dom}(\alpha) \end{aligned}$$

The idea behind the definition of abstraction is that, for $k \in \text{dom}(\alpha)$, the term $\text{term}(k)$ represented by k is replaced by the variable $\alpha(k)$. The restriction $\text{rg}(\alpha) \cap \text{rg}(\sigma) = \emptyset$ ensures that the variables in $\text{rg}(\alpha)$ are fresh.

Combining abstraction with the signature function sig , we have the following properties. Let sig_α be the signature function of T_α then, for all $k \in K$:

$$\text{sig}_\alpha(k) \subseteq \text{sig}(k) \tag{2}$$

$$K_\Sigma \subseteq \text{dom}(\alpha) \Rightarrow \text{sig}_\alpha(k) \cap \Sigma = \emptyset \tag{3}$$

(2) follows from the fact that $\text{sig}(k) = \emptyset$ if $\sigma(k) \in \text{VAR}$. For (3), we argue as follows: if $\text{sig}_\alpha(k) \cap \Sigma \neq \emptyset$, then there is an l with $\sigma(l) \in \Sigma$ and a path from k to l which does not meet $\text{dom}(\alpha)$; this yields contradiction, for $l \in \text{dom}(\alpha)$, since $K_\Sigma \subseteq \text{dom}(\alpha)$.

4 Derivations

Now we can define derivations, the central notion of this paper. We proceed in two steps: first we introduce basic derivations, which correspond with proofs in equational logic without the substitution rule; to deal with substitution, we add abstractions and come to the full definition of derivation.

Definition 3 (Basic derivation and derivability). A triple $B = \langle T, E, M \rangle$ is a basic derivation if $T = \langle K, \arg, \sigma \rangle$ is a term structure, $E \subseteq K^2$, and $M \subseteq \ker(\sigma)$ is an equivalence relation on K . E is called the equality relation, and M the matching relation of B . The derivability operator der is defined by:

$$\text{der}(E) = \mu R. (E \cup (M \cap \text{lift}(R)))^e$$

We also define cong (congruence) on $\wp(K^2)$ by

$$\text{cong}(R) = M \cap \text{lift}_B(R)$$

so $\text{cong}(R) = M \cap \arg \cdot R^\otimes \cdot \arg^{-1}$, and $\text{der}(E) = \mu R. (E \cup \text{cong}(R))^e$. We write $E \vdash_B G$ when B is a basic derivation with $G \subseteq \text{der}_B(E)$.

So a basic derivation is a term structure with an equality relation and a matching relation. The equality relation indicates which pairs of (abstract) terms are given to be equal. The matching relation is a restriction on the possible pairs of terms that may be proved to be equal via congruence (i.e. via equality of corresponding direct subterms). One might expect $M = \ker(\sigma)$ here: when two terms have the same leading signature element and the corresponding direct subterms are equal, then they are equal. However, this leads to a notion of derivability that is too strong for our purposes, in the sense that it may spoil a notion of wellfoundedness of derivations that we shall introduce below. We use M for a kind of locality restriction: a congruence step in an abstract derivation, establishing the equality of two nodes with identical signature elements and equal corresponding direct subterms, is only allowed when these two nodes are, in some sense, on the same level in the derivation.

By definition, $\text{der}(E)$ is the least *equivalence* relation containing E and closed under cong :

$$\text{der}(E) = (E \cup \text{cong}(\text{der}(E)))^e \quad (4)$$

$$\text{if } (E \cup \text{cong}(R))^e \subseteq R, \text{ then } \text{der}(E) \subseteq R \quad (5)$$

We shall refer to these as the *defining property* and the *minimality property* of der , respectively. Moreover, we have

$$E \subseteq \text{der}(E) = \text{der}(\text{der}(E)) \subseteq (E \cup M)^e \quad (6)$$

$E \subseteq \text{der}(E) \subseteq \text{der}(\text{der}(E))$ is obvious; $\text{der}(E) \subseteq (M \cup E)^e$ follows from $(E \cup \text{cong}((E \cup M)^e))^e = (E \cup (M \cap \text{lift}((E \cup M)^e)))^e \subseteq (E \cup M)^e$ and the minimality property of der . $\text{der}(\text{der}(E)) \subseteq \text{der}(E)$ follows via the minimality property of der from $(\text{der}(E) \cup \text{cong}(\text{der}(E)))^e \subseteq \text{der}(E)$.

Definition 4 (Wellfounded basic derivations). Let $B = \langle T, E, M \rangle$ be a basic derivation. B is wellfounded if \prec is wellfounded, where \prec is defined by $\text{der}(E) \cdot [\arg]^{-1}$. If B is wellfounded, we have the following induction principle: for all $L \subseteq K$,

$$\forall k (\arg[k] \subseteq L \rightarrow k \in L) \rightarrow L = K, \quad (7)$$

where $[k] = \{l \mid (k, l) \in \text{der}(E)\}$.

So $\arg[k] = \{\arg_i(l) \mid (k, l) \in \text{der}(E) \ \& \ i < \text{arity}(l)\}$, and

$$\prec_B = \{(k, \arg_i(l)) \mid (k, l) \in \text{der}(E) \ \& \ i < \text{arity}(l)\}$$

The induction principle (7) will be used in the proof of the Soundness Theorem.

Definition 5 (Derivation). *A derivation D is a pair $\langle B, \alpha \rangle$ where B is a well-founded basic derivation and $\alpha : K \rightarrow \text{VAR}$ is an abstraction of T that is justified by B , i.e. satisfies*

$$\ker(\alpha) \subseteq \ker(\sigma) \cap \text{lift}(\text{der}(E)) \quad (8)$$

The idea is that B proves, in some sense, the equalities implicit in α . We write $E \vdash_D G$ when D is a derivation with $G \subseteq \text{der}(E)$.

So a derivation is a wellfounded basic derivation B together with an abstraction α such that B proves, in some sense, $\ker(\alpha)$, i.e. the equalities that are implicit in α . Observe the particular form of (8), the definition of justification: $\ker(\alpha) \subseteq \text{der}(E)$ would be more natural. However, both (8) and the wellfoundedness requirement are essential in the proof of the Soundness Theorem: together, they ensure the provable equality of terms before this equality is used in the rest of the derivation.

This definition of derivation, especially the justification condition (8), models a version of the strong instantiation rule that we mentioned at the end of Subsection 1.2. An alternative definition of derivation is possible which stays closer to the ordinary instantiation rule: replace the justification condition (8) by $\ker(\alpha) \subseteq \ker(\text{term})$, and drop the wellfoundedness condition (since it is not required for proving soundness). However, proving interpolation for this notion of derivation is harder in this case: the obvious abstraction required for the interpolant does not satisfy this stronger notion of justification.

For later use, we introduce the notion of parsimony.

Definition 6 (Parsimony). *Basic derivation $B = \langle T, E, M \rangle$ is parsimonious if $M \subseteq \text{lift}(\text{der}(E))$; derivation $\langle B, \alpha \rangle$ is parsimonious whenever B is.*

The idea is that, in a parsimonious derivation, the matching relation M is minimal: all pairs $(k, l) \in M$ are needed to establish $\text{der}(E)$. This is made explicit in the first part of the next lemma. The second part shows that we may always assume that a derivation is parsimonious: for if it is not, we can make it parsimonious without changing $\text{der}(E)$.

Lemma 1 (Parsimony). *Let $B = \langle T, E, M \rangle$ be a basic derivation.*

1. *If B is parsimonious, then $\text{der}(E) = (E \cup M)^e$.*
2. *There is a parsimonious basic derivation $B' = \langle T, E, M' \rangle$ with derivation operator der' such that $\text{der}(E) = \text{der}'(E)$.*

Proof. (1) is rather easy, and for (2) take $M' = \text{cong}(\text{der}(E))$. Details are omitted.

5 Interpretation

In this section, we define the interpretation of abstract derivations and prove a Soundness Theorem. Let a signature SIG be given. A signature interpretation of SIG in universe U is a mapping $I : \text{SIG} \cup \text{VAR} \rightarrow \bigcup_n (U^n \rightarrow U)$ that respects arity, i.e. if $I(f) \in (U^{\text{arity}(f)} \rightarrow U)$. We write $\text{Int}(\text{SIG}, U)$ for the collection of all interpretations of SIG in U . The relation \sim on $\text{Int}(\text{SIG}, U)$ is defined by

$$I \sim J \Leftrightarrow \forall f \in \text{SIG} \ I(f) = J(f)$$

Now let some term structure $T = \langle K, \text{arg}, \sigma \rangle$ be given, with $\text{rg}(\sigma) \subseteq \text{SIG} \cup \text{VAR}$. A model $\mathcal{M} = \langle U, I \rangle$ for T consists of a universe $U \neq \emptyset$ and a signature interpretation $I \in \text{Int}(\text{SIG}, U)$. With recursion, we define $\bar{I}_\sigma : K \rightarrow U$:

$$\bar{I}_\sigma(k) = I(\sigma(k))(\bar{I}_\sigma^\otimes(\text{arg}(k)))$$

For $F, G \subseteq K^2$ and abstraction α , we define several notions of validity:

$$\begin{aligned} \langle U, I \rangle \models G & \quad \text{iff } G \subseteq \ker(\bar{I}_\sigma) \\ \langle U, I \rangle \models_\alpha G & \quad \text{iff } G \subseteq \ker(\bar{I}_{\alpha:\sigma}) \\ \langle U, I \rangle \models_\alpha \forall G & \quad \text{iff } \forall J \sim I (\langle U, J \rangle \models_\alpha G) \\ F \models G & \quad \text{iff } \forall \mathcal{M} (\mathcal{M} \models F \Rightarrow \mathcal{M} \models G) \\ \forall F \models_\alpha G & \quad \text{iff } \forall \mathcal{M} (\mathcal{M} \models_\alpha \forall F \Rightarrow \mathcal{M} \models G) \end{aligned}$$

Theorem 1 (Basic soundness). *Basic derivations are sound: if B is a basic derivation, then $E \models \text{der}(E)$.*

Proof. We have to show, for arbitrary $\mathcal{M} = \langle U, I \rangle$: $\mathcal{M} \models E \Rightarrow \mathcal{M} \models \text{der}(E)$, i.e.

$$E \subseteq \ker(\bar{I}_\sigma) \Rightarrow \text{der}(E) \subseteq \ker(\bar{I}_\sigma)$$

So assume $E \subseteq \ker(\bar{I}_\sigma)$. Now

$$\begin{aligned} & \text{der}(E) \subseteq \ker(\bar{I}_\sigma) \\ \Leftarrow & \quad \{\text{minimality property of der}\} \\ & (E \cup \text{cong}(\ker(\bar{I}_\sigma))^e \subseteq \ker(\bar{I}_\sigma) \\ \Leftrightarrow & \quad \{\ker(\bar{I}_\sigma) \text{ is an equivalence relation, and } E \subseteq \ker(\bar{I}_\sigma) \text{ is given}\} \\ & M \cap \text{lift}(\ker(\bar{I}_\sigma)) \subseteq \ker(\bar{I}_\sigma) \\ \Leftarrow & \quad \{M \subseteq \ker(\sigma)\} \\ & \ker(\sigma) \cap \text{lift}(\ker(\bar{I}_\sigma)) \subseteq \ker(\bar{I}_\sigma) \\ \Leftrightarrow & \quad \{\text{property of interpretations}\} \\ & \text{true} \end{aligned}$$

Theorem 2 (Soundness). *Derivations are sound: if $\langle B, \alpha \rangle$ is a derivation, then $\forall E \models_\alpha \text{der}(E)$.*

The proof is omitted.

6 Some important lemmata

In this section, we prove some properties of abstract derivations. They provide insight in the nature of derivations, and they will be applied in the proof of the Interpolation theorem given in the next section. The proofs of the lemmata presented here are rather involved generally, and we postpone them to the Appendix.

We start with introducing a notion of weak derivability, where only congruence steps are allowed, ignoring the proof rules for reflexivity, symmetry and transitivity. The *weak derivability operator* der^- is defined by:

$$\text{der}^-(E) = \mu R.(E \cup \text{cong}(R))$$

By definition, $\text{der}^-(E)$ is the least relation containing E and closed under cong . So the defining and the minimality property of der^- are

$$\text{der}^-(E) = E \cup \text{cong}(\text{der}^-(E)) \quad (9)$$

$$\text{if } E \cup \text{cong}(R) \subseteq R, \text{ then } \text{der}^-(E) \subseteq R \quad (10)$$

Moreover, we have $E \subseteq \text{der}^-(E) \subseteq \text{der}(E)$. In the next lemma, we show that, contrary to derivability, weak derivability commutes with restriction to arg -closed substructures.

Lemma 2 (der⁻ and intersection). *Let $B = \langle T, E, M \rangle$ be a basic derivation, and let $L \subseteq K$ be arg -closed. Then $\text{der}^-(E) \cap L^2 \subseteq \text{der}^-(E \cap L^2)$.*

The proof is omitted.

The following counterexample shows that Lemma 2 does not hold for der :

$$K = \{a, b, c\}, L = \{a, c\}, E = \{(a, b), (b, c)\}$$

(we identify nodes and signature elements). Now

$$\text{der}(E) \cap L^2 = K^2 \cap L^2 = L^2 \neq \{(a, a), (b, b), (c, c)\} = \text{der}(\emptyset) = \text{der}(E \cap L^2)$$

The next Lemma indicates how we can reduce der to der^- without affecting the signature.

Lemma 3 (reduction of der to der^-). *Let $B = \langle T, E, M \rangle$ be a basic derivation. Then there is a relation $E' \subseteq \text{der}(E)$ satisfying $\text{der}(E) = \text{der}^-(E')$ and $\text{sig}(E') \subseteq \text{sig}(E)$.*

Proof. $E' = \text{der}(E) - \text{cong}(\text{der}(E))$ does the trick. Details are omitted.

Lemma 3 can be paraphrased as follows: E' is part of the $\text{sig}(E)$ -consequences of E , and all consequences of E follows from E' via only congruence steps.

The two previous lemmata are sufficient to prove interpolation for basic derivations, as we shall see in the next section. For the full interpolation result, we need two properties involving abstractions. The first is about the result of applying an abstraction to a basic derivation. We start with a definition.

Definition 7 (Applicability and neutrality). Let $B = \langle T, E, M \rangle$ be a basic derivation and α an abstraction of T . The result of applying α to B is defined as $B_\alpha = \langle T_\alpha, E, M \rangle$. We also define

α is applicable to B if M respects α (i.e. $M \subseteq \ker(\alpha) \cup (K - \text{dom}(\alpha))^2$);
 α is neutral for B if $\text{der}(E) = \text{der}_\alpha(E)$.

It is easy to see that if α is applicable to B , then B_α is a basic derivation: for $M \subseteq \ker(\alpha) \cup (K - \text{dom}(\alpha))^2$ and $M \subseteq \ker(\sigma)$ imply that $M \subseteq \ker(\alpha : \sigma)$. The next Lemma states that an applicable abstraction has no influence on the derivability operator.

Lemma 4 (applicable implies neutral). If abstraction α is applicable to basic derivation B and B is parsimonious, then α is neutral for B .

The proof is omitted.

The lemma addresses only parsimonious derivations; however, as we have observed above Lemma 1, this restriction is not essential. Observe that we did not require that B justifies α .

The last lemma is an existence lemma about abstractions. As we mentioned in the Introduction, it is related to Takeuti's Lemma. For its formulation we need the notion of separation, which we define now.

Definition 8 (Separation). Let $\langle K, \text{arg} \rangle$ be a forest with $L, L_1, L_2 \subseteq K$. We say that L separates L_1 from L_2 if every path from L_1 to L_2 meets L : so if (k_0, \dots, k_n) is a path in T with $k_0 \in L_1$ and $k_n \in L_2$, then $k_i \in L$ for some i with $0 \leq i < n$.

Let α, β be abstractions. We say that α separates L from β if $\text{dom}(\alpha)$ separates L from $\text{dom}(\beta)$, i.e. every path from L to $\text{dom}(\beta)$ meets $\text{dom}(\alpha)$.

The idea behind separation is that, from the perspective of L , the effect of applying β is hidden by α . This is made explicit in the following property:

$$\alpha \text{ separates } L \text{ from } \beta \Rightarrow \forall k \in L \text{ term}_\alpha(k) = \text{term}_{\beta:\alpha}(k) \quad (11)$$

To see that this holds, we argue as follows: if $\text{term}_{\beta:\alpha}(k)$ and $\text{term}_\alpha(k)$ differ somewhere, then there is a path that starts in k , avoids $\text{dom}(\alpha)$ and ends in $k' \in \text{dom}(\beta)$. But this contradicts the fact that α separates L from β .

Lemma 5 (existence of eliminating abstraction). Let $\langle B, \alpha \rangle$ be a parsimonious derivation, and let Σ be some signature. Then there is an abstraction $\beta : K_\Sigma \rightarrow \text{VAR}$ that satisfies the following conditions

1. B justifies β (so $\langle B, \beta \rangle$ is a derivation);
2. β is applicable to B and B_β justifies α (so $\langle B_\beta, \alpha \rangle$ is a derivation);
3. if $\text{sig}_\alpha(E) \cap \Sigma = \emptyset$, then α separates $\text{car}(E)$ from β .

Proof. Define $\beta : K_\Sigma \rightarrow (\text{VAR} - \text{rg}(\sigma))$ implicitly by

$$\ker(\beta) = ((M \cup \ker(\alpha)) \cap K_\Sigma^2)^+$$

Observe that $((M \cup \ker(\alpha)) \cap K_\Sigma^2)^+$ is an equivalence relation on K_Σ , since $M \cup \ker(\alpha)$ is reflexive and symmetric, and $\text{car}(M) = K$. Such a β can always be found, since VAR is infinite. β is an abstraction, for $\text{rg}(\beta) \cap \text{rg}(\sigma) = \emptyset$. Details are omitted.

7 Interpolation

Definition 9 (Interpolation). Let $B = \langle K, \text{arg}, \sigma, E, M \rangle$ be a basic derivation, and assume that $E \vdash_B G$, i.e. $G \subseteq \text{der}(E)$. A basic interpolant for E and G in B is an $I \subseteq K^2$ with

1. $E \vdash_B I$ and $I \vdash_B G$;
2. $\text{sig}(I) \subseteq \text{sig}(E) \cap \text{sig}(G)$.

Moreover, if $\langle B, \alpha \rangle$ is a derivation, then $\langle J, \beta \rangle$ is an interpolant for E and G in $\langle B, \alpha \rangle$ if

1. β is an abstraction justified by B and separated from E by α ;
2. $\langle B_\beta, \alpha \rangle$ is a derivation, $E \vdash_{B_\beta} J$ and $J \vdash_B G$;
3. $\text{sig}_\beta(J) \subseteq \text{sig}_\alpha(E) \cap \text{sig}(G)$.

This translates to the usual notion of interpolation in the context of equational logic, via the mapping form:

$$\begin{aligned} \text{form}(E) \vdash \text{form}(I) \text{ and } \text{form}(I) \vdash \text{form}(G) \\ \text{sig}(\text{form}(I)) \subseteq \text{sig}(\text{form}(E)) \cap \text{sig}(\text{form}(G)) \end{aligned}$$

$$\begin{aligned} \text{form}_\alpha(E) \vdash \text{form}_\beta(J) \text{ and } \text{form}_\beta(J) \vdash \text{form}(G) \\ \text{sig}(\text{form}_\beta(J)) \subseteq \text{sig}(\text{form}_\alpha(E)) \cap \text{sig}(\text{form}(G)) \end{aligned}$$

In the last two lines, we used that $\text{form}_\alpha(E) = \text{form}_{\beta:\alpha}(E)$; this follows via (11) from the fact that α separates β from E .

Now we formulate the Interpolation Theorem for abstract derivations. The lemmata of the previous section are applied in its proof.

Theorem 3 (Interpolation). Let B be a basic derivation with $E \vdash_B G$. Then there is a basic interpolant I for E and G in B , which even satisfies $I \vdash_{\bar{B}} G$. Moreover, if $\langle B, \alpha \rangle$ is a derivation, then there is an abstraction β such that $\langle I, \beta \rangle$ is an interpolant for E and G in $\langle B, \alpha \rangle$.

Proof. Let $\Pi = \text{sig}(G)$. Define

$$I := \text{der}(E) - \text{cong}(\text{der}(E)) \cap K_{\frac{2}{\Pi}}$$

We check the conditions for basic interpolation.

1. $E \vdash_B I$ is evident, for $I \subseteq \text{der}(E)$. $I \vdash_B^- G$, i.e. $G \subseteq \text{der}^-(I)$, is shown as follows:

$$\begin{aligned}
& G \\
\subseteq & \quad \{G \subseteq \text{der}(E) \text{ and } \text{sig}(G) = \Pi\} \\
& \text{der}(E) \cap K_{\Pi}^2 \\
= & \quad \{\text{Lemma 3}\} \\
& \text{der}^-(\text{der}(E) - \text{cong}(\text{der}(E))) \cap K_{\Pi}^2 \\
\subseteq & \quad \{\text{Lemma 2}\} \\
& \text{der}^-(\text{der}(E) - \text{cong}(\text{der}(E))) \cap K_{\Pi}^2 \\
= & \quad \{\text{definition of } I\} \\
& \text{der}^-(I)
\end{aligned}$$

2. By Lemma 3, $\text{sig}(\text{der}(E) - \text{cong}(\text{der}(E))) \subseteq \text{sig}(E)$, so $\text{sig}(I) \subseteq \text{sig}(E) \cap \Pi = \text{sig}(E) \cap \text{sig}(G)$.

So I is indeed a basic interpolant.

Now assume that $\langle B, \alpha \rangle$ is a derivation. Without loss of generality (Lemma 1) we assume that B is parsimonious. Let $\Sigma = \text{sig}(E)$, $\Sigma^- = \text{sig}_{\alpha}(E)$. Let β be the abstraction of Lemma 5 with $\text{dom}(\beta) = K_{\Sigma - \Sigma^- \cap \Pi}$. We show that (I, β) is an interpolant by checking the conditions for interpolation.

1. By Lemma 5.1, B justifies β ; by $\text{sig}_{\alpha}(E) \cap (\Sigma - \Sigma^- \cap \Pi) = \Sigma^- \cap (\Sigma - \Sigma^- \cap \Pi) = \emptyset$ and Lemma 5.3, α separates E from β .
2. $E \vdash_B I$ and $I \vdash_B^- G$ are proved above. By Lemma 5.2, β is applicable to B and B_{β} justifies α , so by Lemma 4 $\text{der}_B(E) = \text{der}_{B_{\beta}}(E)$, and we have $E \vdash_{B_{\beta}} I$.
3. Above, we showed that $\text{sig}(I) \subseteq \text{sig}(E) \cap \text{sig}(G) = \Sigma \cap \Pi$. Since $\text{dom}(\beta) = K_{\Sigma - \Sigma^- \cap \Pi}$, we have by (2) and (3) that $\text{sig}_{\beta}(I) \subseteq \text{sig}(I) - (\Sigma - \Sigma^- \cap \Pi)$, so $\text{sig}_{\beta}(I) \subseteq (\Sigma \cap \Pi) - (\Sigma - \Sigma^- \cap \Pi) = \Sigma^- \cap \Pi = \text{sig}_{\alpha}(E) \cap \text{sig}(G)$.

8 Concluding remarks

We presented abstract derivations for equational logic and established some interesting properties which we applied in a proof of the Interpolation theorem. The proof is constructive in that the interpolant is given explicitly, using rather perspicuous global operations on the relations that represent equations. We see this as an advantage over other proofs for the same theorem, which are either not constructive or proceed via incremental proof transformations. It came as a surprise that, in the proof of interpolation, the underlying term structure is not modified. Another surprise (now a negative one) was the involvedness of many of the proofs of the lemmata we needed, especially since their formulation is quite simple.

As we explained in the Introduction, we consider the development of this result as an experiment in working out the proper definitions of abstract derivations. We find the present outcome of the experiment satisfactory, but it is

not finished yet. The main thing that has to be done is to firmly establish the correspondence between traditional derivations in equational logic and abstract derivations. Going from abstract derivations to traditional derivations is the easy direction: for the other direction, most of the work will lie in proving that the resulting abstract derivation is wellfounded.

We finish with an unordered list of ideas for further research. We chose equational logic for its general nature, in the sense that equality is a fundamental notion, and many logics can be naturally embedded in equational logic. But it remains to be investigated what happens with derivations in these embeddings. It may also be interesting to extend the notion of abstract terms and to allow for bags or sets instead of sequences as the datatype for the immediate subterms of a compound term: by doing so, properties like commutativity, associativity and idempotency can be 'hardwired' in the abstract terms. Another idea is to replace term structures by sequent structures (after all, sequents can be considered as a kind of generalized terms) so as to investigate sequent-based derivation systems. The representation of quantifiers is an open question. Furthermore, there is conditional equational logic, where implications $(s_0 \equiv t_0 \wedge \dots s_{n-1} \equiv t_{n-1}) \rightarrow s \equiv t$ are allowed: Rodenburg proved interpolation in [10] via an adaptation of the algebraic proof for equational logic in [11], but a prooftheoretic proof has not been given yet. Another direction for research is proof complexity. In principle, abstract derivations allow for efficient representation of proofs by sharing of subterms: the question is how efficient this representation is. Are they comparable with extended Frege systems, as defined by Cook and Reckhow in [2]? Are there general methods to reduce the size of derivations? The relation between abstract derivations and other alternative representation of proofs, like proof nets (see [7]) and deep derivations (see [1], [13]), is another open question.

Three anonymous referees and Piet Rodenburg are acknowledged for their constructive criticism on an earlier version of this paper.

References

1. Kai Brünnler. *Deep inference and symmetry in classical proofs*. Logos Verlag, Berlin, 2004.
2. Stephen A. Cook and Robert A. Reckhow. Time bounded random access machines. *Journal of Computer and System Sciences*, 7:354–375, 1973.
3. W.W. Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *Journal of Symbolic Logic*, 22:250–268, 1957.
4. Gerard R. Renardel de Lavalette. Interpolation in fragments of intuitionistic propositional logic. *Journal of Symbolic Logic*, 54:1419 – 1430, 1989.
5. Gerard R. Renardel de Lavalette. Abstract derivations, equational logic and interpolation. <http://www.cs.rug.nl/~gr1/pub/lisbon2005full.pdf>, 2005.
6. Walter Felscher. On interpolation when function symbols are present. *Archiv für mathematische Logik und Grundlagenforschung*, 17:145–158, 1976.
7. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
8. G. Grätzer. *Universal Algebra (2nd edition)*. Springer-Verlag, New York, 1979.

9. Don Pigozzi. The join of equational theories. *Colloquium Mathematicum*, 30:15–25, 1974.
10. Piet Rodenburg. Interpolation in conditional equational logic. *Fundamenta Informaticae*, 15:80–85, 1991.
11. Piet Rodenburg. A simple algebraic proof of the equational interpolation theorem. *Algebra Universalis*, 28:48–51, 1991.
12. Piet Rodenburg. Interpolation in equational logic. Technical report, University of Amsterdam, Department of Mathematics and Computer Science, Programming Research Group, January 1992. Report P9201.
13. Charles Stewart and Phiniki Stouppa. A systematical proof theory for several modal logics (extended abstract). In Renate Schmidt, Ian Pratt-Hartmann, and Mark Reynolds, editors, *AiML2004 — Advances in Modal Logic (conference proceedings)*, pages 357–371. Department of Computer Science, University of Manchester, Technical Report Series UMCS-04-9-1, 2004.
14. Gaisi Takeuti. Lecture notes on proof theory (mimeographed). Urbana, 1971.
15. A. Visser, J. van Benthem, D. de Jongh, and G.R. Renardel de Lavalette. NNIL, a study in intuitionistic propositional logic. In A. Ponse, M. de Rijke, and Y. Venema, editors, *Modal Logic and Process Algebra*, pages 289 – 326. CSLI Publications, Stanford (USA), 1995.

Intersection Types: a Proof-Theoretical Approach

Elaine Pimentel^{1,2}, Simona Ronchi Della Rocca¹, and Luca Roversi^{1*}

¹ Dipartimento di Scienze dell'Informazione, Università di Torino, Italy
{pimentel,ronchi,roversi}@di.unito.it

² Departamento de Matemática, UFMG, Brasil
{elaine@mat.ufmg.br}

Abstract. In this work we present a proof-theoretical justification for **IT** by means of the logical system Intersection Synchronous Logic (**ISL**). **ISL** builds equivalence classes of deductions of the implicative and conjunctive fragment of **NJ**. **ISL** results from decomposing intuitionistic conjunction into two connectives: a *synchronous* conjunction, that can be used only among equivalent deductions of **NJ**, and an *asynchronous* one, that can be applied among any sets of deductions of **NJ**. A term decoration of **ISL** exists so that it matches both: the **IT** assignment system, when only the synchronous conjunction is used, and the simple types assignment with pairs and projections, when the asynchronous conjunction is used. Moreover, the proof of strong normalization property for **ISL** is a simple consequence of the same property in **NJ** and hence strong normalization for **IT** comes for free.

1 Introduction

The intersection type assignment system (**IT**) [4] is a deductive system that assigns formulae (built from the intuitionistic implication \rightarrow and the intersection \cap) as types to the untyped λ -calculus. **IT** has been used as an investigation tool for a large variety of problems, like, for example, characterizations of the strongly normalizing λ -terms [11].

The main goal of this work is to give a proof-theoretical justification to **IT**.¹ To this aim a basis step is to clarify, within a pure logical system, the difference between the connectives intersection (\cap) and intuitionistic conjunction (\wedge), by imposing constraints on the use of the logical and structural rules of intuitionistic logic.

* All authors supported by MIUR national project PROTOCOLLO. Pimentel is also supported by CNPq.

¹ This goal sounds very much alike, for example, to the one of giving a proof-theoretical characterization of linear functions. To that purpose one could use λ -terms with exactly a single occurrence of every free and bound name. However, proof-theoretically equivalently, the same set, under the “derivations-as-programs” analogy, is characterized by a deductive system of second order propositional logic without weakening and contraction.

Recall that derivations of **IT** form a strict subset of derivations of the implicative and conjunctive fragment of Intuitionistic logic (**NJ**), in the sense that the λ -terms to which **IT** gives types to are used as meta-theoretical modalities. More specifically, for every $\Pi : \Gamma \vdash_{\mathbf{IT}} M : \sigma$ of **IT**, the term M records where \rightarrow -introductions and eliminations are used inside Π . Then the intersection can be introduced only between formulas typing the *same* term. Hence the rule for the introduction of the intersection ($\cap I$) can be seen, roughly speaking, as a “mistaken decoration” of the rule for the introduction of the conjunction ($\wedge I$) of **NJ**, where pairs are forgotten:

$$\frac{\Gamma \vdash N : \sigma \quad \Gamma \vdash M : \tau}{\Gamma \vdash (M, N) : \sigma \wedge \tau} (\wedge I) \qquad \frac{\Gamma \vdash_{\mathbf{IT}} M : \sigma \quad \Gamma \vdash_{\mathbf{IT}} M : \tau}{\Gamma \vdash_{\mathbf{IT}} M : \sigma \cap \tau} (\cap I)$$

In order to evidence, at the level of λ -terms, the difference between the usual conjunction \wedge of **NJ** and the intersection \cap of **IT**, we start by defining a non standard decoration for **NJ** (called **NJr**) that has explicit structural rules and where the original conjunction \wedge is split into two conjunctions \cap and $\&$, in the spirit of Linear Logic [6]. Note that this decomposition cannot be expressed directly inside **NJ** without collapsing \cap and $\&$.

From **NJr**, we build **ISL**, a logical system that internalizes this decomposition, maintaining explicit the structural rules. The rules of **ISL** inductively build *equivalence classes*. The basis of the inductive definition is any set of relevant axioms. The inductive steps, represented by the rules of **ISL**, preserve the structural invariant as follows: (A) \rightarrow -introduction and elimination must be applied *synchronously* on all the components of a set of already equivalent deductions; (B) weakening must be applied *synchronously* on all the assumptions of the judgments of already equivalent deductions of **NJ**; (C) the order of assumptions matters as far as two derivations must be declared structurally equivalent, or not. So the explicit use of the exchange rule is required. Without the careful management of structural equivalences, **ISL** would collapse to **NJ**—see Subsection 4.1 for a better understanding of the role played by structural rules in this setting.

Given the notion of equivalence classes above, the intersection operator of **IT** becomes a conjunction of **ISL** that can be introduced only between the components of a given class. That conjunction is dubbed as *synchronous*, to recall the kind of building process we use for the derivations it is applied to.

We should conclude by saying that the present work gives a complete proof-theoretical justification for **IT** since **ISL**: (i) highlights the role of structural rules to delineate **IT** inside intuitionistic logic; (ii) reinterprets the intersection operator \cap of **IT** in terms of an operator that can be used among sets of structurally equivalent deductions of intuitionistic logic; (iii) reformulates the tree structures that **IL** [14] required in order to characterize **IT**. The reformulation is in terms of (simultaneous) logical and structural operations on the equivalence classes. Finally, **ISL** is technically good, since it enjoys strong normalization and sub-formula properties.

The rest of the paper is organized as follows: Section 2 recalls the implicative and conjunctive fragment of Intuitionistic Logic (**NJ**) and introduces the system

NJr, a refinement of the standard decoration for **NJ**. The Intersection Types Assignment System (**IT**), with explicit weakening and exchange rules, is then introduced as a subsystem of **NJr**. Section 3 introduces **ISL**, which embodies all our intuitions into a formal system. Section 4 formalizes how **ISL**, **NJ** and **IT** correspond. Also in this section, we give a technical justification for the conditions on the explicit structural rules, required to reformulate **IT** in terms of **ISL**. Section 5 proves that **ISL** is a good deductive system and describes the behavior of the two **ISL** conjunctions with respect to the implication. Finally, Section 6 describes the relationship between this and some related work.

2 Splitting the conjunction

In this section we first recall the implicative and conjunctive fragment of Intuitionistic Logic (**NJ**) in natural deduction style. We then present **NJr**, a type assignment system based on the splitting of the conjunction into two connectives, each one grasping a particular aspect of its behavior. Finally, the Intersection Types Assignment System (**IT**) will be presented as a subsystem of **NJr**.

Definition 1 ($\{\wedge \rightarrow\}$ -fragment of **NJ**).

- i) The formulae of the implicative and conjunctive fragment of **NJ** are generated by the grammar: $\sigma ::= a \mid \sigma \rightarrow \sigma \mid \sigma \wedge \sigma$, where a belongs to a denumerable set of constants. As usual, \rightarrow is right-associative while \wedge is left-associative. Formulae of **NJ** will be denoted by Greek small letters.
- ii) We will denote by $\mathcal{F}_{\mathbf{NJ}}$ the set of formulae of **NJ**. A context is a finite sequence $\sigma_1, \dots, \sigma_m$ of formulae. Contexts are denoted by Γ and Δ .
- iii) The implicative and conjunctive fragment of **NJ** proves statements $\Gamma \vdash_{\mathbf{NJ}} \sigma$, where Γ is a context and σ a formula. It consists of the following rules:

$$\begin{array}{l}
(A) \frac{}{\sigma \vdash_{\mathbf{NJ}} \sigma} \qquad (W) \frac{\Gamma \vdash_{\mathbf{NJ}} \sigma}{\Gamma, \tau \vdash_{\mathbf{NJ}} \sigma} \\
(X) \frac{\Gamma_1, \sigma_1, \sigma_2, \Gamma_2 \vdash_{\mathbf{NJ}} \sigma}{\Gamma_1, \sigma_2, \sigma_1, \Gamma_2 \vdash_{\mathbf{NJ}} \sigma} \qquad (\wedge I) \frac{\Gamma \vdash_{\mathbf{NJ}} \sigma \quad \Gamma \vdash_{\mathbf{NJ}} \tau}{\Gamma \vdash_{\mathbf{NJ}} \sigma \wedge \tau} \\
(\wedge E^l) \frac{\Gamma \vdash_{\mathbf{NJ}} \sigma \wedge \tau}{\Gamma \vdash_{\mathbf{NJ}} \sigma} \qquad (\wedge E^r) \frac{\Gamma \vdash_{\mathbf{NJ}} \sigma \wedge \tau}{\Gamma \vdash_{\mathbf{NJ}} \tau} \\
(\rightarrow I) \frac{\Gamma, \sigma \vdash_{\mathbf{NJ}} \tau}{\Gamma \vdash_{\mathbf{NJ}} \sigma \rightarrow \tau} \qquad (\rightarrow E) \frac{\Gamma \vdash_{\mathbf{NJ}} \sigma \rightarrow \tau \quad \Gamma \vdash_{\mathbf{NJ}} \sigma}{\Gamma \vdash_{\mathbf{NJ}} \tau}
\end{array}$$

$\Pi : \Gamma \vdash_{\mathbf{NJ}} \sigma$ means that the deduction Π concludes by proving $\Gamma \vdash_{\mathbf{NJ}} \sigma$. $\vdash_{\mathbf{NJ}} \sigma$ is a short notation for $\emptyset \vdash_{\mathbf{NJ}} \sigma$.

By somewhat abusing the name, **NJ** will always name the implicative and conjunctive fragment of **NJ**.

NJr is a type assignment for λ -terms with pairs. It splits the original conjunction \wedge of **NJ**

$$(\wedge I) \frac{\Gamma \vdash_{\mathbf{NJ}} \sigma \quad \Gamma \vdash_{\mathbf{NJ}} \tau}{\Gamma \vdash_{\mathbf{NJ}} \sigma \wedge \tau}$$

into two conjunctions, depending on the form of the λ -terms M and N that could be typed by the premises $\Gamma \vdash_{\mathbf{NJ}} \sigma$ and $\Gamma \vdash_{\mathbf{NJ}} \tau$. In particular, when the conclusion of the two premises is the type of the same λ -term, it is possible to replace \wedge by a *synchronous conjunction* (\cap) that keeps giving type to M , identical to N . Otherwise, the refinement of \wedge consists of the *asynchronous conjunction* ($\&$) that gives type to the pair (M, N) .

Definition 2 (NJr).

- i) The grammar defining the set of formulae of **NJr** ($\mathcal{F}_{\mathbf{NJr}}$) is obtained from that of Definition 1 by replacing the rule $\sigma ::= \sigma \wedge \sigma$ by the following rules: $\sigma ::= \sigma \cap \sigma \mid \sigma \& \sigma$, where \cap and $\&$ are, respectively synchronous and asynchronous conjunctions.
- ii) A **NJr**-context is a finite sequence of pairs $x_1 : \sigma_1, \dots, x_n : \sigma_n$ that assigns formulae to variables so that $i \neq j$ implies $x_i \neq x_j$. By abusing the notation, **NJr**-contexts will be denoted by Γ . If $\Gamma = x_1 : \sigma_1, \dots, x_n : \sigma_n$, then $\text{dom}(\Gamma) = \{x_1, \dots, x_n\}$.
- iii) Terms of the λ -calculus (Λ) are defined by the following grammar: $M ::= x \mid \lambda x.M \mid MM$, where x belongs to a countable set of variables. Terms of the λ -calculus with pairs (Λ_p) are obtained by adjoining to the previous grammar the following terms: $M ::= (M, M) \mid \pi_l(M) \mid \pi_r(M)$. As usual, terms will be considered modulo α -conversion and application is left associative.
- iii) **NJr** derives judgments $\Gamma \vdash_{\mathbf{NJr}} M : \sigma$ where $M \in \Lambda_p$, Γ is an **NJr**-context, and σ is a formula. The rules of **NJr** are:

$$\begin{array}{l}
(A) \frac{}{x : \sigma \vdash_{\mathbf{NJr}} x : \sigma} \\
(W) \frac{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \quad x \notin \text{dom}(\Gamma)}{\Gamma, x : \tau \vdash_{\mathbf{NJr}} M : \sigma} \\
(\cap I) \frac{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \quad \Gamma \vdash_{\mathbf{NJr}} M : \tau}{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \cap \tau} \\
(\cap E^l) \frac{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \cap \tau}{\Gamma \vdash_{\mathbf{NJr}} M : \sigma} \\
(\& E^l) \frac{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \& \tau}{\Gamma \vdash_{\mathbf{NJr}} \pi_l(M) : \sigma} \\
(\rightarrow I) \frac{\Gamma, x : \sigma \vdash_{\mathbf{NJr}} M : \tau}{\Gamma \vdash_{\mathbf{NJr}} \lambda x.M : \sigma \rightarrow \tau} \\
(X) \frac{\Gamma_1, x : \sigma_1, y : \sigma_2, \Gamma_2 \vdash_{\mathbf{NJr}} M : \sigma}{\Gamma_1, y : \sigma_2, x : \sigma_1, \Gamma_2 \vdash_{\mathbf{NJr}} M : \sigma} \\
(\& I) \frac{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \quad \Gamma \vdash_{\mathbf{NJr}} N : \tau}{\Gamma \vdash_{\mathbf{NJr}} (M, N) : \sigma \& \tau} \\
(\cap E^r) \frac{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \cap \tau}{\Gamma \vdash_{\mathbf{NJr}} M : \tau} \\
(\& E^r) \frac{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \& \tau}{\Gamma \vdash_{\mathbf{NJr}} \pi_r(M) : \tau} \\
(\rightarrow E) \frac{\Gamma \vdash_{\mathbf{NJr}} M : \sigma \rightarrow \tau \quad \Gamma \vdash_{\mathbf{NJr}} N : \sigma}{\Gamma \vdash_{\mathbf{NJr}} MN : \tau}
\end{array}$$

$\Pi : \Gamma \vdash_{\mathbf{NJr}} \sigma$ means that the deduction Π concludes by proving $\Gamma \vdash_{\mathbf{NJr}} \sigma$.

Intuitively, **NJr** identifies derivations of **NJ** which are *synchronous* with respect to the introduction and the elimination of the implication. In other words, \cap merges sub-derivations where \rightarrow is introduced or eliminated in the “same points”.

Formally, let $e : \mathcal{F}_{\mathbf{NJr}} \longrightarrow \mathcal{F}_{\mathbf{NJ}}$ be defined as follows:

$$e(a) = a, \quad e(\sigma \rightarrow \tau) = e(\sigma) \rightarrow e(\tau) \quad e(\sigma \& \tau) = e(\sigma \cap \tau) = e(\sigma) \wedge e(\tau)$$

The function e can be extended to contexts in the obvious way:

$$e(x_1 : \sigma_1, \dots, x_n : \sigma_n) = e(\sigma_1), \dots, e(\sigma_n)$$

Moreover, let E be an erasure function from \mathbf{NJr} -proofs to \mathbf{NJ} -proofs, that erases all type information from \mathbf{NJr} -proofs and collapses the introduction and elimination rules of (\cap) and $(\&)$ to the corresponding rules for (\wedge) .

The following theorem shows the relation between \mathbf{NJ} and \mathbf{NJr} :

Theorem 1. *i) If $\Pi : \Gamma \vdash_{\mathbf{NJr}} M : \sigma$ then $E(\Pi) : e(\Gamma) \vdash_{\mathbf{NJ}} e(\sigma)$.
ii) If $\Pi : \Gamma \vdash_{\mathbf{NJ}} \sigma$ then there is a proof $\Pi' : \Gamma' \vdash_{\mathbf{NJr}} M : \sigma'$ such that $E(\Pi) = \Pi$, $e(\Gamma') = \Gamma$ and $e(\sigma') = \sigma$.*

Now we can define the Intersection Type Assignment System \mathbf{IT} as a sub-system of \mathbf{NJr} where only synchronous conjunction is used.

Definition 3 (IT).

- i) The set $\mathcal{F}_{\mathbf{IT}}$ of types of \mathbf{IT} is the subset of $\mathcal{F}_{\mathbf{NJr}}$ generated by the grammar:
 $\sigma ::= a \mid \sigma \rightarrow \sigma \mid \sigma \cap \sigma$, where a belongs to a denumerable set of constants.*
- ii) The Intersection Type Assignment System \mathbf{IT} proves statements of the shape:
 $\Gamma \vdash_{\mathbf{IT}} M : \sigma$ where M is a λ -term, Γ is an \mathbf{IT} -context, assigning types to variables, and σ is a type. The rules of the system are the rules of \mathbf{NJr} but $(\&I)$, $(\&E^l)$ and $(\&E^r)$.
 $\Pi : \Gamma \vdash_{\mathbf{IT}} M : \sigma$ means that the deduction Π concludes by proving $\Gamma \vdash_{\mathbf{IT}} \sigma$.*

Note that also the Curry's type assignment for Λ_p can be seen as a sub-system of \mathbf{NJr} , where only the asynchronous conjunction is used.

2.1 An Example

The difference between synchronous and asynchronous conjunction, being related to a meta-condition on the form of proofs, cannot be expressed inside \mathbf{NJ} . The following example can be useful for better understanding this point.

Let $\sigma = ((\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha) \& (\alpha \rightarrow \alpha)$ and let us consider the following derivation:

$$\frac{\Pi_2'' : \vdash_{\mathbf{NJr}} \lambda x. \pi_1(x) \pi_2(x) : \sigma \rightarrow \alpha \rightarrow \alpha \quad \Pi_1'' : \vdash_{\mathbf{NJr}} (\lambda x. x, \lambda x. x) : \sigma}{\vdash_{\mathbf{NJr}} (\lambda x. \pi_1(x) \pi_2(x)) (\lambda x. x, \lambda x. x) : \alpha \rightarrow \alpha} (\rightarrow E)$$

where Π_1'' is:

$$\frac{\frac{\frac{}{x : \alpha \rightarrow \alpha \vdash_{\mathbf{NJr}} x : \alpha \rightarrow \alpha} (A)}{\vdash_{\mathbf{NJr}} \lambda x. x : (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha} (\rightarrow I) \quad \frac{\frac{}{x : \alpha \vdash_{\mathbf{NJr}} x : \alpha} (A)}{\vdash_{\mathbf{NJr}} \lambda x. x : \alpha \rightarrow \alpha} (\rightarrow I)}{\vdash_{\mathbf{NJr}} (\lambda x. x, \lambda x. x) : ((\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha) \& (\alpha \rightarrow \alpha)} (\&I)}$$

and Π_2'' is:

$$\frac{\frac{\frac{}{x : \sigma \vdash_{\mathbf{NJr}} x : \sigma} (A)}{x : \sigma \vdash_{\mathbf{NJr}} \pi_1(x) : (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha} (\&E^l) \quad \frac{\frac{}{x : \sigma \vdash_{\mathbf{NJr}} x : \sigma} (A)}{x : \sigma \vdash_{\mathbf{NJr}} \pi_2(x) : \alpha \rightarrow \alpha} (\&E^r)}{x : \sigma \vdash_{\mathbf{NJr}} \pi_1(x)\pi_2(x) : \alpha \rightarrow \alpha} (\rightarrow E)}{\vdash_{\mathbf{NJr}} \lambda x.\pi_1(x)\pi_2(x) : \sigma \rightarrow \alpha \rightarrow \alpha} (\rightarrow I)$$

In the derivation Π_1'' , the conjunction $\&$ has been introduced. But since the two terms typing the premises are syntactically the same, we could replace it by \cap , so obtaining the following derivation, where $\tau = ((\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha) \cap (\alpha \rightarrow \alpha)$:

$$\frac{\Pi_2' : \vdash_{\mathbf{NJr}} (\lambda x.xx) : \tau \rightarrow \alpha \rightarrow \alpha \quad \Pi_1' : \vdash_{\mathbf{NJr}} \lambda x.x : \tau}{\vdash_{\mathbf{NJr}} (\lambda x.xx)(\lambda x.x) : \alpha \rightarrow \alpha} (\rightarrow E)$$

where Π_1' is:

$$\frac{\frac{\frac{}{x : \alpha \rightarrow \alpha \vdash_{\mathbf{NJr}} x : \alpha \rightarrow \alpha} (A)}{\vdash_{\mathbf{NJr}} \lambda x.x : (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha} (\rightarrow I) \quad \frac{\frac{}{x : \alpha \vdash_{\mathbf{NJr}} x : \alpha} (A)}{\vdash_{\mathbf{NJr}} \lambda x.x : \alpha \rightarrow \alpha} (\rightarrow I)}{\vdash_{\mathbf{NJr}} \lambda x.x : ((\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha) \cap (\alpha \rightarrow \alpha)} (\cap I)}$$

and Π_2' is:

$$\frac{\frac{\frac{}{x : \tau \vdash_{\mathbf{NJr}} x : \tau} (A)}{x : \tau \vdash_{\mathbf{NJr}} x : (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha} (\cap E^l) \quad \frac{\frac{}{x : \tau \vdash_{\mathbf{NJr}} x : \tau} (A)}{x : \tau \vdash_{\mathbf{NJr}} x : \alpha \rightarrow \alpha} (\cap E^r)}{x : \tau \vdash_{\mathbf{NJr}} xx : \alpha \rightarrow \alpha} (\rightarrow E)}{\vdash_{\mathbf{NJr}} \lambda x.xx : \tau \rightarrow \alpha \rightarrow \alpha} (\rightarrow I)$$

Both the previous derivations correspond, in the sense of Theorem 1, to the following derivation in **NJ**:

$$\frac{\Pi_2 : \vdash_{\mathbf{NJ}} \rho \rightarrow \alpha \rightarrow \alpha \quad \Pi_1 : \vdash_{\mathbf{NJ}} \rho}{\vdash_{\mathbf{NJ}} \alpha \rightarrow \alpha} (\rightarrow E)$$

where: $\rho = e(\sigma) = e(\tau) = ((\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha) \wedge (\alpha \rightarrow \alpha)$ and Π_1, Π_2 are, respectively:

$$\frac{\frac{\frac{}{\alpha \rightarrow \alpha \vdash_{\mathbf{NJ}} \alpha \rightarrow \alpha} (A)}{\vdash_{\mathbf{NJ}} (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha} (\rightarrow I) \quad \frac{\frac{}{\alpha \vdash_{\mathbf{NJ}} \alpha} (A)}{\vdash_{\mathbf{NJ}} \alpha \rightarrow \alpha} (\rightarrow I)}{\vdash_{\mathbf{NJ}} ((\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha) \wedge (\alpha \rightarrow \alpha)} (\wedge I)}$$

and

$$\frac{\frac{\frac{}{\sigma \vdash_{\mathbf{NJ}} \sigma} (A)}{\sigma \vdash_{\mathbf{NJ}} (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha} (\wedge E^l) \quad \frac{\frac{}{\sigma \vdash_{\mathbf{NJ}} \sigma} (A)}{\sigma \vdash_{\mathbf{NJ}} \alpha \rightarrow \alpha} (\wedge E^r)}{\frac{\sigma \vdash_{\mathbf{NJ}} \alpha \rightarrow \alpha}{\vdash_{\mathbf{NJ}} \sigma \rightarrow \alpha \rightarrow \alpha} (\rightarrow I)} (\rightarrow E)$$

3 The logical system ISL

The logical system **ISL**, that will be presented below, internalizes at the logical level the two different behaviors of the conjunction, the synchronous and asynchronous one. The notion of molecule is the one we use to obtain the result. A molecule is the technical tool to represent the equivalence classes, informally described in the introduction.

- Definition 4.** *i) Formulae of **ISL** are formulae of **NJr**. Contexts are finite sequences of such formulae, ranged over by Δ, Γ .*
ii) An atom is a pair $(\Gamma; \alpha)$, where Γ (the context) is a finite sequence of formulas. \mathcal{A}, \mathcal{B} will range over atoms.
iii) A finite multiset of atoms, such that the contexts in all atoms have the same cardinality is called a molecule. $[\mathcal{A}_1, \dots, \mathcal{A}_n]$ denotes a molecule consisting of the atoms $\mathcal{A}_1, \dots, \mathcal{A}_n$. \mathcal{M}, \mathcal{N} will range over molecules. \cup is multiset union.
*iv) **ISL** derives molecules by the following rules:*

$$\frac{}{[(\alpha_i; \alpha_i) \mid 1 \leq i \leq r]} (A) \quad \frac{\mathcal{M} \cup \mathcal{N}}{\mathcal{M}} (P)$$

$$\frac{[(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i, \alpha_i; \beta_i) \mid 1 \leq i \leq r]} (W) \quad \frac{[(\Gamma_1^i, \beta_i, \alpha_i, \Gamma_2^i; \sigma_i) \mid 1 \leq i \leq r]}{[(\Gamma_1^i, \alpha_i, \beta_i, \Gamma_2^i; \sigma_i) \mid 1 \leq i \leq r]} (X)$$

$$\frac{[(\Gamma_i, \alpha_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i \rightarrow \beta_i) \mid 1 \leq i \leq r]} (\rightarrow I)$$

$$\frac{[(\Gamma_i; \alpha_i \rightarrow \beta_i) \mid 1 \leq i \leq r] \quad [(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]} (\rightarrow E)$$

$$\frac{[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r] \quad [(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i \& \beta_i) \mid 1 \leq i \leq r]} (\&I)$$

$$\frac{[(\Gamma_i; \alpha_i \& \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]} (\&E_L) \quad \frac{[(\Gamma_i; \alpha_i \& \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]} (\&E_R)$$

$$\frac{\mathcal{M} \cup [(\Gamma; \alpha), (\Gamma; \beta)]}{\mathcal{M} \cup [(\Gamma; \alpha \cap \beta)]} (\cap I)$$

$$\frac{\mathcal{M} \cup [(\Gamma; \alpha \cap \beta)]}{\mathcal{M} \cup [(\Gamma; \alpha)]} (\cap E_L) \quad \frac{\mathcal{M} \cup [(\Gamma; \alpha \cap \beta)]}{\mathcal{M} \cup [(\Gamma; \beta)]} (\cap E_R)$$

- v) $\vdash_{\mathbf{ISL}} \mathcal{M}$ denotes the existence of an **ISL** deduction rooted at \mathcal{M} .*

A molecule contains sets of deductions of **NJ** on which some building steps are performed *synchronously*. On the other side, two different molecules, the elements of which have been built *asynchronously*, cannot belong to the same molecule and can be merged only through the introduction of a conjunction or an elimination of an implication.

Making a parallel with the so called *hypersequents*, this means that the intersection is an *internal* connective, while the conjunction and implication are *external*. More about the relationship between molecules and hypersequents (in fact, with hyperformulae since **ISL** is in natural deduction style) can be seen in Section 6.

Example 1. Let τ denote $\alpha \rightarrow \alpha$, ρ denote $(\tau \rightarrow \tau) \& \tau$, and σ denote $(\tau \rightarrow \tau) \cap \tau$. The deductions of the Section 2.1 can be developed inside **ISL**, without the need of λ -terms:

$$\begin{array}{c}
\frac{\frac{\frac{\overline{[(\tau; \tau), (\alpha; \alpha)]} (A)}{[(\emptyset; \tau \rightarrow \tau), (\emptyset; \tau)]} (\rightarrow I)}{[(\emptyset; \sigma)]} (\cap I)}{[(\emptyset; \tau)]} (\cap I) \quad \frac{\frac{\overline{[(\sigma; \sigma)]} (A)}{[(\sigma; \tau \rightarrow \tau)]} (\cap E_L) \quad \frac{\overline{[(\sigma; \sigma)]} (A)}{[(\sigma; \tau)]} (\cap E_R)}{[(\sigma; \tau)]} (\rightarrow E)}{[(\emptyset; \sigma \rightarrow \tau)]} (\rightarrow I)}{[(\emptyset; \tau)]} (\rightarrow E)
\end{array}$$

$$\begin{array}{c}
\frac{\frac{\frac{\overline{[(\tau; \tau)]} (A)}{[(\emptyset; \tau \rightarrow \tau)]} (\rightarrow I) \quad \frac{\overline{[(\alpha; \alpha)]} (A)}{[(\emptyset; \tau)]} (\rightarrow I)}{[(\emptyset; \rho)]} (\& I)}{[(\emptyset; \tau)]} (\& I) \quad \frac{\frac{\overline{[(\rho; \rho)]} (A)}{[(\rho; \tau \rightarrow \tau)]} (\& E_L) \quad \frac{\overline{[(\rho; \rho)]} (A)}{[(\rho; \tau)]} (\& E_R)}{[(\rho; \tau)]} (\rightarrow E)}{[(\emptyset; \rho \rightarrow \tau)]} (\rightarrow I)}{[(\emptyset; \tau)]} (\rightarrow E)
\end{array}$$

4 ISL, NJ and IT

This section states the formal correspondence between **ISL**, **NJ** and **IT**. This is done decorating proofs of **ISL** by means of λ -terms. The decoration is similar to the one described in [14] and it is inspired by the so called ‘‘Curry-Howard isomorphism’’: every deduction Π is associated to a λ -term in order to keep track of some structural properties of Π .

Note that this decoration is not standard: the λ -term associated to Π is *untyped*, and does not encode the *whole* structure of Π , but only the order of occurrences of the rules for the implication and conjunction.

- Definition 5.** *i)* Let $\Gamma \equiv \beta_1, \dots, \beta_n$ be a context. A decoration of Γ , with respect to a sequence of different variables x_1, \dots, x_n , is $(\Gamma)^{x_1, \dots, x_n} \equiv x_1 : \beta_1, \dots, x_n : \beta_n$. The symbol s denotes a sequence of variables, different from each other.
- ii)* Every Π that proves the molecule $\mathcal{M} = [(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]$ can be decorated so that the result is a type assignment that proves $M_s : (\mathcal{M})_s$, where $(\mathcal{M})_s \equiv [((\Gamma_i)^s; \beta_i) \mid 1 \leq i \leq r]$, and M_s is a λ -term. The decoration procedure is inductively defined in Figure 1.
- iii)* $\vdash_{\mathbf{ISL}}^* M : (\mathcal{M})_s$ denotes the existence of a decorated proof of **ISL** rooted at $M : (\mathcal{M})_s$.

Observe that the decoration is defined in a parametric way with respect to the sequence of variables s .

$$\begin{aligned}
& - \Pi : \frac{[(\alpha_i; \alpha_i) \mid 1 \leq i \leq m]}{(A)} \Rightarrow \frac{M_x \equiv x : [(x : \alpha_i; \alpha_i) \mid 1 \leq i \leq m]}{(A^*);} \\
& - \Pi : \frac{\Pi_1 : [(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i, \alpha_i; \beta_i) \mid 1 \leq i \leq r]} (W) \Rightarrow \\
& \quad \frac{M_s(\Pi_1) : [((\Gamma_i)^s; \beta_i) \mid 1 \leq i \leq r] \quad x \notin \text{dom}(\Gamma)^s}{M_{s,x}(\Pi) \equiv M_s(\Pi_1) : [((\Gamma_i)^s, x : \alpha_i; \beta_i) \mid 1 \leq i \leq r]} (W^*); \\
& - \Pi : \frac{\Pi_1 : [(\Gamma_1^i, \beta_i, \alpha_i, \Gamma_2^i; \sigma_i) \mid 1 \leq i \leq r]}{[(\Gamma_1^i, \alpha_i, \beta_i, \Gamma_2^i; \sigma_i) \mid 1 \leq i \leq r]} (X) \Rightarrow \\
& \quad \frac{M_{s_1,y,x,s_2}(\Pi_1) : [((\Gamma_1^i)^{s_1}, y : \beta_i, x : \alpha_i, (\Gamma_2^i)^{s_2}; \sigma_i) \mid 1 \leq i \leq r]}{M_{s_1,x,y,s_2}(\Pi) \equiv M_{s_1,y,x,s_2}(\Pi_1) : [((\Gamma_1^i)^{s_1}, x : \alpha_i, y : \beta_i, (\Gamma_2^i)^{s_2}; \sigma_i) \mid 1 \leq i \leq r]} (X^*); \\
& - \Pi : \frac{\Pi_1 : [(\Gamma_i, \alpha_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i \rightarrow \beta_i) \mid 1 \leq i \leq r]} (\rightarrow I) \Rightarrow \\
& \quad \frac{M_{s,x}(\Pi_1) : [((\Gamma_i)^s, x : \alpha_i; \beta_i) \mid 1 \leq i \leq r]}{M_s(\Pi) \equiv \lambda x. M_{s,x}(\Pi_1) : [((\Gamma_i)^s; \alpha_i \rightarrow \beta_i) \mid 1 \leq i \leq r]} (\rightarrow I^*); \\
& - \Pi : \frac{\Pi_1 : [(\Gamma_i; \alpha_i \rightarrow \beta_i) \mid 1 \leq i \leq r] \quad \Pi_2 : [(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]} (\rightarrow E) \Rightarrow \\
& \quad \frac{M_1 : [((\Gamma_i)^s; \alpha_i \rightarrow \beta_i) \mid 1 \leq i \leq r] \quad M_2 : [((\Gamma_i)^s; \alpha_i) \mid 1 \leq i \leq r]}{M_s(\Pi) \equiv M_1 M_2 : [((\Gamma_i)^s; \beta_i) \mid 1 \leq i \leq r]} (\rightarrow E^*), \\
& \quad \text{where } M_1 \equiv M_s(\Pi_1), M_2 \equiv M_s(\Pi_2); \\
& - \Pi : \frac{\Pi_1 : [(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r] \quad \Pi_2 : [(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i \& \beta_i) \mid 1 \leq i \leq r]} (\&I) \Rightarrow \\
& \quad \frac{M_s(\Pi_1) : [((\Gamma_i)^s; \alpha_i) \mid 1 \leq i \leq r] \quad M_s(\Pi_2) : [((\Gamma_i)^s; \beta_i) \mid 1 \leq i \leq r]}{M_s(\Pi) \equiv (M_s(\Pi_1), M_s(\Pi_2)) : [((\Gamma_i)^s; \alpha_i \& \beta_i) \mid 1 \leq i \leq r]} (\&I^*); \\
& - \Pi : \frac{\Pi_1 : [(\Gamma_i; \alpha_i^l \& \alpha_i^r) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]} (\&E_X) \Rightarrow \\
& \quad \frac{M_s(\Pi_1) : [((\Gamma_i)^s; \alpha_i^l \& \alpha_i^r) \mid 1 \leq i \leq r]}{M_s(\Pi) = \pi_t(M_s(\Pi_1)) : [((\Gamma_i)^s; \alpha_i) \mid 1 \leq i \leq r]} (\&E_X^*) \\
& \quad \text{where } X \in \{L, R\}, \text{ and, if } X = L \text{ then } t = l \text{ else } t = r; \\
& - \Pi : \frac{\Pi_1 : \mathcal{M}_1}{\mathcal{M}_2} (R) \Rightarrow \frac{M_s(\Pi_1) : (\mathcal{M}_1)^s}{M_s(\Pi) = M_s(\Pi_1) : (\mathcal{M}_2)^s} (R^*) \\
& \quad \text{where } R \in \{(\cap I), (\cap E_L), (\cap E_R), (P)\};
\end{aligned}$$

Fig. 1. The decoration of ISL.

The next theorem, together with Theorem 1, proves that **ISL** is as powerful as **NJ**. Moreover, it puts into evidence the fact that a molecule represents a set of proofs of **NJ**, built *synchronously*. The proofs of this and the next theorem are by induction on derivations.

Theorem 2 (ISL and NJ). *Let $\mathcal{M} = [(\Gamma_1; \alpha_1), \dots, (\Gamma_m; \alpha_m)]$. Then $\vdash_{\mathbf{ISL}}^* M : (\mathcal{M})_s$ if and only if $(\Gamma_i)^s \vdash_{\mathbf{NJ}} M : \alpha_i$ for all $1 \leq i \leq m$.*

ISL can be proposed as the logic for **IT**, thanks to the following theorem:

Theorem 3 (ISL and IT).

- i) *Let $\mathcal{M} = [(\Gamma_1; \alpha_1), \dots, (\Gamma_m; \alpha_m)]$, where α_i and all types in Γ_i belong to $\mathcal{F}_{\mathbf{IT}}$, and let $\vdash_{\mathbf{ISL}}^* M : (\mathcal{M})_s$ with $M \in \Lambda$, for some s . Then $(\Gamma_i)^s \vdash_{\mathbf{IT}} M : \alpha_i$.*
- ii) *$\Gamma \vdash_{\mathbf{IT}} M : \alpha$ implies $\vdash_{\mathbf{ISL}}^* M : [(\Gamma; \alpha)]$.*

Corollary 1. ***IT** collects the synchronous behavior of **NJ**.*

4.1 The role of structural rules

There is in the literature a lot of intersection types assignment systems. Here clearly we want to consider a "minimal" version, in the sense that only the rules dealing with the two connectives \rightarrow and \cap occur (while there are systems with various kinds of subtyping and eta-rules) and also there is no universal type. The reason for this choice is clear, being this a foundational investigation, and being these extra features not motivated from a logical point of view. But also in this minimal version **IT** is usually presented in a different style, i.e. contexts are *sets* of pairs $\{x_1 : \sigma_1, \dots, x_n : \sigma_n\}$, and the three rules (A),(W),(X) are replaced by:

$$(A) \frac{x : \sigma \in \Gamma}{\Gamma \vdash_{\mathbf{NJ}} x : \sigma}$$

The two formulations are equivalent. But the design of **ISL**, and consequently a logical account of **IT**, needs explicit structural rules.

In fact, let **ISL'** be defined from **ISL** by considering contexts as sets and by replacing the rules (A) and (W) by the axiom:

$$\overline{[(\Gamma_i \cup \{\alpha_i\}; \alpha_i) \mid 1 \leq i \leq r]} \quad (A')$$

Then the following molecules could be proved:

$$[(\{\alpha \cap \beta \rightarrow \gamma\}; \alpha \rightarrow \beta \rightarrow \gamma)] \text{ and } [(\{\alpha \rightarrow \beta \rightarrow \gamma\}; \alpha \cap \beta \rightarrow \gamma)]$$

hence collapsing \cap to \wedge (see also Section 5). This shows that implicit weakening cannot be used in the definition of **ISL**.

On the other hand, let \mathbf{ISL}'' be defined from \mathbf{ISL} by using contexts as sets (instead of sequences) but maintaining the explicit weakening rule (thus still having a linear axiom). Then it would be possible to derive:

$$\frac{\frac{\frac{[(\{\alpha\}; \alpha), (\{\beta\}; \beta)]}{[(\{\alpha, \beta\}; \alpha), (\{\beta\}; \beta)]} (W)}{[(\{\alpha, \beta\}; \alpha), (\{\alpha, \beta\}; \beta)]} (W)}{[(\{\alpha, \beta\}; \alpha \cap \beta)]} (\cap I)$$

The proof above does not correspond to any derivation of \mathbf{IT} . Indeed, assume the two atoms $(\{\alpha, \beta\}; \alpha)$ and $(\{\alpha, \beta\}; \beta)$ represent the two judgments $x : \alpha, y : \beta \vdash_{\mathbf{IT}} x : \alpha$ and $x : \alpha, y : \beta \vdash_{\mathbf{IT}} y : \beta$. They have the same context, being, however, labelled by different terms. So \cap cannot be introduced.

Hence, in order to capture correctly the behavior of the intersection connective, we need *both* contexts as sequences and explicit structural rules.

5 Properties of ISL

This section proves that \mathbf{ISL} enjoys properties expected for logical systems, like strong normalization and sub-formula. Both proofs follow the method described in [14] showing that, in fact, these properties are inherited from \mathbf{NJ} . We also discuss the behavior of the two \mathbf{ISL} conjunctions with respect to the implication.

Strong normalization. We start by noting that the rule (P) can be eliminated, and that the weakening can be moved up on the proof, being it applied just after the axioms.

Lemma 1. *Let Π be a derivation for \mathcal{M} . Then there is a derivation Π' of \mathcal{M} without any occurrences of the rule (P) and all applications of the weakening rule follow the axioms.*

Definition 6. *A derivation is said to be in pre-normal form if it doesn't have any occurrences of the rule (P) and all applications of the weakening rule follow the axioms.*

Definition 7. *Let Π be a derivation.*

i. A \cap -redex of Π is one of the sequences:

$$\frac{\frac{\mathcal{M} \cup [(\Gamma; \alpha), (\Gamma; \beta)]}{\mathcal{M} \cup [(\Gamma; \alpha \cap \beta)]} (\cap I)}{\mathcal{M} \cup [(\Gamma; \alpha)]} (\cap E_L) \qquad \frac{\frac{\mathcal{M} \cup [(\Gamma; \alpha), (\Gamma; \beta)]}{\mathcal{M} \cup [(\Gamma; \alpha \cap \beta)]} (\cap I)}{\mathcal{M} \cup [(\Gamma; \beta)]} (\cap E_R)$$

ii. A $\&$ -redex of Π is the sequence:

$$\frac{\frac{[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r] \quad [(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i \& \beta_i) \mid 1 \leq i \leq r]} (\& I)}{[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]} (\& E_L)$$

or the similar sequence for the right case.

iii. A \rightarrow -redex of Π is the sequence:

$$\frac{\frac{[(\Gamma_i, \alpha_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i \rightarrow \beta_i) \mid 1 \leq i \leq r]} (\rightarrow I)}{[(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]} (\rightarrow E) \quad \frac{[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]} (\rightarrow E)$$

It is easy to see that the exchange rule can be moved up when between redexes. This result, together with Lemma 1, shows that the structural rules do not interfere with the normalization process.

Lemma 2. Let $\diamond \in \{\rightarrow, \&, \cap\}$ and let SR be a certain number of occurrences of the rule (X) . Then

$$\frac{\frac{\mathcal{G}}{\mathcal{G}'} (\diamond I)}{\frac{\mathcal{G}''}{\mathcal{G}'''} (\diamond E)} (SR) \text{ can be rewritten as } \frac{\frac{\mathcal{G}}{\mathcal{G}^{iv}} (SR)}{\frac{\mathcal{G}^v}{\mathcal{G}'''} (\diamond E)} (\diamond I)$$

where \mathcal{G}^{iv} and \mathcal{G}^v are formed accordingly.

The following Lemma is proved by structural induction.

Lemma 3 (Substitution lemma). Let Π_0 be a proof of $[(\Gamma_i, \alpha_i; \beta_i) \mid 1 \leq i \leq r]$ and Π_1 be a proof of $[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]$. Suppose that both Π_1 and Π_2 are in pre-normal form. Let $S(\Pi_0, \Pi_1)$ be the deductive structure obtained from Π_0 by substituting all axioms $[(\alpha_i; \alpha_i) \mid 1 \leq i \leq r]$ by Π_1 (using weakening and exchange rules where necessary in order to re-arrange contexts), and by eliminating all occurrences of weakening over α_i . Then $S(\Pi_1, \Pi_0) : [(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]$.

Definition 8. Let Π be a derivation in pre-normal form.

i. A \cap -rewriting step on Π is:

$$\frac{\frac{\mathcal{M} \cup [(\Gamma; \alpha), (\Gamma; \beta)]}{\mathcal{M} \cup [(\Gamma; \alpha \cap \beta)]} (\cap I)}{\frac{\mathcal{M} \cup [(\Gamma; \alpha)]}{\mathcal{M} \cup [(\Gamma; \alpha)]} (\cap E_L)} \hookrightarrow \frac{\mathcal{M} \cup [(\Gamma; \alpha), (\Gamma; \beta)]}{\mathcal{M} \cup [(\Gamma; \alpha)]} (P)$$

The $(\cap E_R)$ case is analogous.

ii. A $\&$ -rewriting step on Π is:

$$\frac{\frac{\Pi_1 : [(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r] \quad \Pi_2 : [(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i \& \beta_i) \mid 1 \leq i \leq r]} (\& I)}{[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]} (\& E_L)}{\hookrightarrow \Pi_1 : [(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]}$$

The $(\& E_R)$ case is analogous.

iii. A \rightarrow -rewriting step on Π is:

$$\frac{\frac{\Pi_0 : [(\Gamma_i, \alpha_i; \beta_i) \mid 1 \leq i \leq r]}{[(\Gamma_i; \alpha_i \rightarrow \beta_i) \mid 1 \leq i \leq r]} (\rightarrow I)}{[(\Gamma_i; \beta_i) \mid 1 \leq i \leq r]} (\rightarrow E) \quad \Pi_1 : [(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]}{\hookrightarrow S(\Pi_1, \Pi_0)}$$

Theorem 4. *ISL is strongly normalizable.*

Proof Lemmas 1, 2 show that (P) can be eliminated and that the other structural rules can be moved up from redexes, so they do not play a significant role in the normalization process. Consider a sequence of normalization steps in **ISL**: $\Pi_1 \rightarrow \dots \rightarrow \Pi_n$. By Theorem 2, there is a one-to-many correspondence between proofs in **ISL** and proofs in **NJ**. This means that the every redex of a derivation Π_i in the sequence above can be translated to redexes in **NJ** and the number of redexes of Π_i is bounded by the number of analogous redexes of any projection of Π in **NJ**. Hence the result follows from the fact that **NJ** has the property of strong normalization. ■

The proof above shows how strong normalizability in **ISL** is a simple consequence of the same property in **NJ**. This is, indeed, a very interesting result since strong normalization for **IT** comes for free while most of the known proofs of this fact uses very complicated techniques, like the Tait-Girard reducibility predicates [7, 15].

Sub-formula property. Sub-formulae in **ISL** are defined as follows:

Definition 9 (Sub-formula). *Let α be a formula of ISL. Then:*

- i. α is a sub-formula of α .*
- ii. If $\beta \diamond \gamma$ is a sub-formula of α , then so are β and γ for $\diamond \in \{\&, \cap, \rightarrow\}$.*

Definition 10 (Sub-formula property). *Let Π be a ISL derivation of the molecule $[(\Gamma_i; \alpha_i) \mid 1 \leq i \leq r]$. Π enjoys the sub-formula property, written $\mathbf{sf}(\Pi)$, if every formula appearing in Π is a sub-formula of one of those occurring in $\Gamma_i \cup \{\alpha_i\}$.*

Theorem 5 (Sub-formula property). *Let Π be a ISL proof in normal form. Then $\mathbf{sf}(\Pi)$.*

Proof The proof is an easy extension of the same property for **NJ**, given the relationship between **NJ** and **ISL** described by Theorem 2. ■

The adjoint property. In **NJ**, the conjunction (\wedge) is the adjoint of the implication, that is, the formulae:

$$\alpha \wedge \beta \rightarrow \gamma \text{ and } \alpha \rightarrow \beta \rightarrow \gamma$$

are isomorphic. The question that arises then is if the conjunctions of **ISL** ($\&, \cap$) also have this property.

It is easy to see that the answer is *yes* for the asynchronous conjunction: the molecules

$$[(\emptyset; \alpha \& \beta \rightarrow \gamma)] \text{ and } [(\emptyset; \alpha \rightarrow \beta \rightarrow \gamma)]$$

are provable equivalent in **ISL**.

However, the answer is *no* for the synchronous conjunction (\cap). In fact, in the formula $\alpha \cap \beta \rightarrow \gamma$ it is implicit that α and β are dependent in the sense

presented on Section 4, that is, these formulae are labelled by the same λ -term, while in the formula $\alpha \rightarrow \beta \rightarrow \gamma$, α and β are completely independent.

This same kind of behavior is observed, for example, in Linear Logic where the additive conjunction ($\&$) is the adjoint of the implication, while the multiplicative one (\otimes) is not.

6 Related work

The idea of studying the relationship between the intersection and intuitionistic conjunction connectives is not new. In fact, this kind of discussion started with Pottinger's observation [11] that \cap does not correspond to the traditional conjunction (this was later formally proved by Hindley [8]). This subject was further motivated in [1, 2]. But still, the study of the behavior of these two connectives were always restricted to type assignment systems.

The first attempt of giving a logical foundation for **IT** appears in [16], where a new type inference system equivalent to **IT** was defined. This system, called TA_{\wedge}^* avoids the traditional introduction rule for the intersection, and the logic L_{\wedge} in a Hilbert-style axiom based formulation was proposed in such a way that combinators in the type assignment system can be associated to logical proofs. This approach is indeed very interesting, and it follows in many ways the ideas already in [11]. Still, the intersection type inference is investigated in the context of combinatory logic instead of λ -calculus and the presentation of the resultant logic is axiomatic. This work was further extended in order to support also union types [5].

In [3], hyperformulae were used in order to obtain the logic HL presented in standard natural deduction style, hence abandoning the axiomatic framework. Molecules are very much alike hyperformulae, the differences consisting in the fact that a context inside an atom (sequent) is a list of formulae (and hence the ordering is crucial), the existence in HL of a distinguished formula ε (the empty formula) and explicit substitutions.² This makes the syntax of HL more complicated than the one presented here, but still easier to handle than *kits* appearing in [14] (see comment below).

However, in the logic HL hyperformulae cannot interact. That means they just have internal rules. This interaction is achieved in **ISL** by introducing the conjunction that can merge arbitrary construction processes. In other words, **ISL** has external rules as well.

Another approach on the logical foundation for **IT** is given in [14], where **IL** has been introduced. Roughly speaking, **ISL** can be viewed as **IL** enriched with conjunction. But, although inspired in this former work, the notation designed for **ISL** is completely different from that presented for **IL**, where kits (i.e., trees labelled by formulas) were used in order to keep track of the structure of proofs. The presence of trees introduces a beautiful geometry within the logical system, but at the same time it makes the definition of derivations harder to manipulate,

² This permitted the implication to become an internal connective as well.

in the sense that it is necessary the introduction of classes of equivalence between proofs in order to define valid derivations. It turns out that kits aren't really necessary: controlling the order of the leaves is enough in this case. That made it possible to choose a much simpler approach based on molecules, where we don't record the shape of proofs, but only group the equivalent ones, step by step.

In any case, the logical systems proposed so far admit the presence of only one between intersection and conjunction, giving the idea that it was impossible to mix them in the same setting. The main contribution of this work is to present a logical system in natural deduction style in which conjunction and intersection can be represented and hence making it possible to characterize, at the proof theoretical level, the behavior of these two connectives. In this way, the intersection \cap leaves the stigma of being a truly proof-functional connective (as described in [10]) in order to become a connective with synchronous behavior, contrasting with the asynchronous nature of the conjunction.

A logic for **IT** always gives, as sub-product, a typed version of λ with intersection types, through a complete decoration of proofs [13]. But typed versions of **IT** can be obviously defined following a non logical approach: examples are in [9, 12, 17].

References

1. Alessi, F. and Barbanera, F. Strong conjunction and intersection types. In *16th International Symposium on Mathematical Foundation of Computer Science (MFCS91)*, volume Lecture Notes in Computer Science 520. Springer-Verlag, 1991.
2. Barbanera, F. and Martini, S. Proof-functional connectives and realizability. *Archive for Mathematical Logic*, 33:189–211, 1994.
3. Capitani, B., Loreti, M. and Venneri B. Hyperformulae, parallel deductions and intersection types. *Electronic Notes in Theoretical Computer Science*, 50(2), 2001.
4. Coppo, M. and Dezani-Ciancaglini, M. An extension of the basic functionality theory for the λ -calculus. *Notre Dame J. Formal Logic*, 21(4):685–693, 1980.
5. Dezani-Ciancaglini, M., Ghilezan, S. and Venneri, B. The “relevance” of intersection and union types. *Notre Dame J. Formal Logic*, 38(2):246–269, 1997.
6. Girard, J-Y. Linear Logic. *Theoretical Computer Science*, 50:1-102, 1987.
7. Girard, J-Y., Lafont, Y. and Taylor, P. Proofs and types. *Cambridge University Press*, 1989.
8. Hindley, J.R. Coppo Dezani types do not correspond to propositional logic. *Theoret. Comput. Sci.*, 28(1-2):235–236, 1984.
9. Liquori, L. and Ronchi Della Rocca, S. Toward an Intersection-Typed System la Church. Presented at ITRS'04, to appear.
10. Lopez-Escobar, E. K. G. Proof-functional connectives. *Methods of Mathematical Logic, Proceedings of the 6th Latin-American Symposium on Mathematical Logic, Caracas, 1983 LNCS*, 1130:208–221, 1985.
11. Pottinger, G. A type assignment for the strongly normalizable λ -terms. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 561–577. Academic Press, London, 1980.
12. Reynolds, J. C. Design of the programming language Forsythe. In P. O'Hearn and R. D. Tennent editors, *Algol-like Languages*, Birkhauser, 1996.

13. Ronchi Della Rocca, S. Typed Intersection Lambda Calculus. In *LTRS 2002*, volume 70(1) of Electronic Notes in Computer Science. Elsevier, 2002.
14. Ronchi della Rocca, S., Roversi, L. Intersection Logic. In *Proceedings of CSL'01*, volume 2142 of LNCS, pages 414-428. Springer-Verlag, 2001.
15. Tait, W.W. Intensional interpretation of functions of finite type I. *Journal of Symbolic Logic*, 32:198–212, 1967.
16. Venneri, B. Intersection types as logical formulae. *J. Logic Comput.*, 4(2):109–124, 1994.
17. Wells, J. B. and Haack, C. Branching types. In *Programming Languages & Systems, 11th European Symp. Programming*, volume 2305 of LNCS, pages 115-132. Springer-Verlag, 2002.

Cut Elimination in Propositional Based Logics

João Rasga

Center for Logic and Computation, Department of Mathematics,
IST, Lisbon, Portugal

`jfr@math.ist.utl.pt`

Abstract. Sufficient conditions for sequent calculi for propositional based logics to enjoy cut elimination are established. These conditions are satisfied by a wide class of sequent calculi encompassing among others calculi for classical and intuitionistic logic, modal logic S4, and classical and intuitionistic linear logic and some of their fragments. The conditions can be checked in finite time and define relations between the rules and the provisos so that the calculus can enjoy cut elimination. A general result of cut elimination is obtained for any calculus satisfying those conditions.

1 Introduction

Cut elimination has been studied in sequent calculi for a wide variety of logics, but most of the times in a case by case basis [6, 5, 2, 4, 8].

Herein conditions for sequent calculi in a certain class to enjoy cut elimination are investigated and a general cut elimination result is obtained for the sequent calculi satisfying that conditions.¹ These conditions can be checked in finite time and are satisfied by sequent calculi for a wide variety of logics including classical and intuitionistic logic, modal logic S4, and classical and intuitionistic linear logic and some of its fragments. Proving cut elimination for a class of sequent calculi has several advantages: i) bring to clarity relationships between the rules and the provisos in the calculi, or between other aspects of the calculi, that guarantee cut elimination, ii) a unique general proof of cut elimination for the sequent calculi in that class, and iii) to show cut elimination for sequent calculi not already known whether to enjoy cut elimination.

Another interesting aspect made clear by this work is the relation between the presence of structural rules, the multiplicative and the additive character of the introduction rules for a connective, and the number of premises that are used in the cut elimination process when the cut formula is introduced by that rules.

In Section 2, the notions necessary to the definition of the sequent calculi considered in our study of cut elimination are introduced. In section 3, sufficient conditions for a calculus to enjoy cut elimination are presented. The general result stating that any calculus satisfying those conditions enjoys cut elimination

¹ This work is the conference version of [9].

is introduced in Section 4. Finally in Section 5, a brief discussion of related work is made.

2 Sequent Calculi

A general result of cut elimination is provided in Section 4 for any calculus satisfying a collection of conditions checkable in finite time. In order to define that conditions it is necessary to refer to common aspects of the calculi. So, in this section, it is defined in a general form notions like rule, proviso, deduction, sequent calculus, in order for them to apply to several calculi, if necessary.

A *signature* C is a family $\{C_k : k \in \mathbb{N}\}$ where each C_k is a countable set of *connectives* of arity k . All the sets are assumed to be pairwise disjoint. We assume given once and for all two denumerable sets: the set $\{\xi_i : i \in \mathbb{N}\}$ of *formula meta-variables*, and the set $\{\Gamma_i : i \in \mathbb{N}\}$ of *multiset meta-variables*. Meta-variables will be only used in rules and in provisos, and their role is to indicate the places where a formula or a multiset of formulas, can and should appear when using that rule or proviso in a deduction. The *language of formulas* over a signature and the sets of meta-variables is inductively defined in the usual way. A *sequent* is a pair $\langle \Psi, \Delta \rangle$, written $\Psi \rightarrow \Delta$ where Ψ and Δ are finite multisets of formulas. A *proviso* π is a map that given a substitution of the meta-variables returns 1 or 0, meaning that the substitution satisfies or does not satisfy the proviso. For instance the proviso Γ is \bullet *closed*, where \bullet is a connective, is satisfied by any substitution that assigns to Γ a multiset of formulas having as the main connective \bullet . A *rule* is a triple $\langle \{s_1, \dots, s_p\}, s, \pi \rangle$ written $\frac{s_1 \dots s_p}{s} \triangleleft \pi$ where s and s_1, \dots, s_p are sequents and π is a proviso. A *sequent calculus* \mathcal{C} is a pair $\langle C, R \rangle$ where C is a signature and R is a finite set of rules. The *deduction* of a sequent s from a set S of sequents in the context of a sequent calculus \mathcal{C} , written $S \vdash_{\mathcal{C}} s$ is defined in the usual way as a labelled tree, see [12].

In the sequel, boolean combinations of the following provisos will be considered: $|\Gamma| \leq a$, denoted by *cardinality proviso*, where Γ is a multiset meta-variable and a is a natural, and the proviso Γ is \bullet *closed*, denoted by \bullet *closure proviso*, where Γ is a multiset meta-variable and \bullet is a connective. Moreover, we denote by $\pi_{||}$ the boolean combination of cardinality provisos, which may be empty, and, given a unary connective c we will denote by $c(\xi_i)$ a formula whose main connective is c . When convenient we will refer to the \bullet *closure proviso* simply by *closure proviso*. Each of these provisos when appearing in rules are such that its meta-variables appear in the rule.

A *cardinality proviso is imposed by a sequent calculus* or in other words a *sequent calculus has or imposes a cardinality proviso* when the number of formulae at one side or both sides of any sequent in a rule in the calculus is limited by cardinality provisos present in the rules to a same number for all the rules.

2.1 Rules

The class of the calculi to be considered for cut elimination is characterized by having rules of a certain specific type. We now describe the types of rules that can be used and the allowed variants for each type, starting by the axiom rule. An *axiom* rule has the form

$$\frac{}{\Gamma_1, \xi_1 \rightarrow \xi_1, \Gamma_2} \triangleleft \pi_{||},$$

and is named Ax. A *left weakening rule for a connective c* has the form

$$\frac{\Gamma_1 \rightarrow \Gamma_2}{\Gamma_1, c(\xi_1) \rightarrow \Gamma_2} \triangleleft \pi_{||},$$

and is named Lw *c*. Similarly for a *right weakening rule for a connective c*. A *left weakening rule* is a similar rule but without any restriction on the principal formula. Similarly for a *right weakening rule*. A *left contraction rule for a connective c* has the form

$$\frac{\Gamma_1, c(\xi_1), c(\xi_1) \rightarrow \Gamma_2}{\Gamma_1, c(\xi_1) \rightarrow \Gamma_2} \triangleleft \pi_{||},$$

and is named Lc *c*. Similarly for a *right contraction rule for c*. A *left contraction rule* is a similar rule but without any restriction on the principal formula. Similarly for a *right contraction rule*. The *cut rule* has the form

$$\frac{\Gamma_1 \rightarrow \Gamma_2, \xi_1 \quad \xi_1, \Gamma'_1 \rightarrow \Gamma'_2}{\Gamma_1, \Gamma'_1 \rightarrow \Gamma_2, \Gamma'_2} \triangleleft \pi_{||}$$

and is named Cut. The *multicut rule* has the form

$$\frac{\Gamma_1 \rightarrow \Gamma_2, \xi_1^m \quad \xi_1^n, \Gamma'_1 \rightarrow \Gamma'_2}{\Gamma_1, \Gamma'_1 \rightarrow \Gamma_2, \Gamma'_2} \triangleleft \pi_{||}, \quad m, n > 0$$

and is named Multicut. The *left multicut rule* has the form

$$\frac{\Gamma_1 \rightarrow \Gamma_2, \xi_1 \quad \xi_1^n, \Gamma'_1 \rightarrow \Gamma'_2}{\Gamma_1, \Gamma'_1 \rightarrow \Gamma_2, \Gamma'_2} \triangleleft \pi_{||}, \quad n > 0$$

and is named LMulticut. Similarly for the *right multicut rule*. The *left multicut rule for a connective c* has the form

$$\frac{\Gamma_1 \rightarrow \Gamma_2, c(\xi_1) \quad c(\xi_1)^n, \Gamma'_1 \rightarrow \Gamma'_2}{\Gamma_1, \Gamma'_1 \rightarrow \Gamma_2, \Gamma'_2} \triangleleft \pi_{||}, \quad n > 1$$

and is named LMulticut *c*. Similarly for a *right multicut rule over c*. In the sequel, we may designate any of the cut rules presented above simply by a *cut rule* or when referring to all of them by *the cut rules*. Moreover the rule Multicut may be designated by *the general multicut rule*. An *additive left introduction rule for a connective c* has the form

$$\frac{\Gamma_1, \Psi_1 \rightarrow \Delta_1, \Gamma_2 \quad \dots \quad \Gamma_1, \Psi_k \rightarrow \Delta_k, \Gamma_2}{\Gamma_1, c(\xi_1, \dots, \xi_n) \rightarrow \Gamma_2} \triangleleft \pi$$

where k is greater than or equal to 0 and the multiset meta-variables in the premises are the multiset meta-variables in the conclusion. A *multiplicative left introduction rule for c* has the form

$$\frac{\Gamma_{11}, \Psi_1 \rightarrow \Delta_1, \Gamma_{21} \quad \dots \quad \Gamma_{1k}, \Psi_k \rightarrow \Delta_k, \Gamma_{2k}}{\Gamma_{11}, \dots, \Gamma_{1k}, c(\xi_1, \dots, \xi_n) \rightarrow \Gamma_{21}, \dots, \Gamma_{2k}} \triangleleft \pi$$

where k is greater than or equal to 0 and the multiset meta-variables in the conclusion are obtained from the multiset meta-variables in the premises. Both types of rules are designated by L_c and satisfy the following conditions:

- Ψ_i and Δ_i for $i = 1, \dots, k$ are multisets of formula meta-variables in ξ_1, \dots, ξ_n ;
- a formula meta-variable appears in more than one premise only if it appears in the same side in each of the premises;

and similarly for an *additive* or a *multiplicative right introduction rule for c* . We use the term introduction in the name of these rules to avoid the situations where it is not clear which type of rule for connectives is being referred.

When we want to stress that a rule is not for a connective we will designate it by *generic rule*. In general when there is no ambiguity we will omit that designation. So by a contraction rule it should be understood a contraction rule that it is not for a connective. When we want to stress that a rule can be either generic or for a connective, we will say it explicitly. When we want to refer to a rule for a connective it is explicitly referred the fact that the rule is for a connective.

3 Conditions for Cut Elimination

The conditions for a calculus to enjoy cut elimination are presented in this section. The calculi satisfying these conditions are denoted by cut suitable calculi. To simplify the presentation, we introduce first the conditions imposing that the rules in the calculus are not redundant except when necessary for cut elimination, then we specify the conditions for pairs of introduction rules, and finally we present all the conditions in the definition of cut suitable calculus.

To simplify the presentation, in the sequel, when writing conditions like *the left (right) multicut rule is in the calculus only if the calculus has a cardinality proviso for the right (left) side* it is meant the following two conditions: *the left multicut rule is in the calculus only if the calculus has a cardinality proviso for the right side* and *the right multicut rule is in the calculus only if the calculus has a cardinality proviso for the left side*.

The first conditions presented, impose that a calculus, named a suitable calculus, has no redundancies in terms of rules except when necessary for cut elimination. These conditions can be checked in finite time for any sequent calculus.

Definition 1. A sequent calculus is *suitable* if

S1 a rule in the calculus is of a type described in section 2;

- S2 the cut rules in the calculus are such that
- 1 a generic multicut rule is in the calculus only if no other cut rule is present;
 - 2 the generic multicut rules are not simultaneously in the calculus;
 - 3 the left (right) multicut rule for a connective is present only if the multicut rule and the left (right) multicut rule are not present;
- S3 the contraction rules in the calculus are such that
- 1 left (right) contraction rule r is present only if the calculus has no cardinality proviso over the left (right) side imposing that the number of formulas is at most 1 - similarly if r is for a connective;
 - 2 the left (right) contraction rule for a connective is present only if the left (right) contraction rule is not present;
- S4 the left (right) weakening rule for a connective is in the calculus only if the left (right) weakening rule is not present.

Introduction rules for a connective at opposite sides should satisfy specific constraints in order for a cut in a deduction to be propagated for the premises, when the cut formula is introduced by these rules.

Definition 2. Consider a calculus with the cut rule or with a generic multicut rule. A right and a left introduction rules for a same connective are a *cut suitable pair* whenever there is a sequence (s_1, \dots, s_u) with no repetitions, of premises without multiset meta-variables, of both rules, for u greater than 0, such that

$$\frac{\frac{\frac{s_1 \quad s_2}{s_{1,2}} \text{ cut} \quad s_3}{s_{1,2,3}} \text{ cut} \quad \vdots \quad s_u}{s_{1,\dots,u}} \text{ cut}$$

is a deduction where $s_{1,\dots,u}$ is the empty sequent \rightarrow , and if the left (right) introduction rule, denoted by r , does not have neither closure provisos nor provisos imposing that both contexts are empty then, setting

$$\text{np} = \text{number of premises of } r, \text{ without multiset variables, in } (s_1, \dots, s_u)$$

and

$$\text{cnp} = \begin{cases} \text{number of premises of } r & \text{if } r \text{ is multiplicative} \\ 1 & \text{if } r \text{ is additive} \end{cases}$$

we have that if $\text{np} < \text{cnp}$ then the calculus has the right (left) weakening rule, and also the left (right) weakening rule if the calculus does not impose a cardinality proviso over the left (right) side. The same for the contraction rule if $\text{np} > \text{cnp}$.

The sequence (s_1, \dots, s_u) is denoted by *cut sequence*. The existence of this sequence tries to capture the idea that the premises s_1, \dots, s_u lead to a contradiction, or, in other words, that there is a classical refutation of the clauses corresponding to s_1, \dots, s_u . The condition on the relation between the number of premises in the cut sequence, the multiplicative and additive character of the

rules, and the presence of the structural rules of weakening and contraction is important to guarantee that it is always possible when eliminating cuts to obtain the sequent, in the original deduction, that results from the cut. Note that each condition in the definition of cut suitable pair can be checked in finite time.

We now illustrate Definition 2 by concluding that the conjunction rules

$$\text{L}\wedge_i \frac{\Gamma_1, \xi_i \rightarrow \Gamma_2}{\Gamma_1, \xi_1 \wedge \xi_2 \rightarrow \Gamma_2} \triangleleft |\Gamma_2| \leq 1 \quad (i = 1, 2) \qquad \text{R}\wedge \frac{\Gamma_1 \rightarrow \xi_1, \Gamma_2 \quad \Gamma_1 \rightarrow \xi_2, \Gamma_2}{\Gamma_1 \rightarrow \xi_1 \wedge \xi_2, \Gamma_2} \triangleleft |\Gamma_2| = 0$$

in the intuitionistic calculus G1i described in [12], with the left multicut rule, constitute a cut suitable pair with cut sequence (s_1, s_2) where s_1 is the sequent $\xi_1 \rightarrow$ corresponding to premise $\Gamma_1, \xi_1 \rightarrow \Gamma_2$ in $\text{L}\wedge_1$ and s_2 is the sequent $\rightarrow \xi_1$ corresponding to premise $\Gamma_1 \rightarrow \xi_1, \Gamma_2$ in $\text{R}\wedge$. This happens because

- $\text{R}\wedge$ does not have neither closure provisos nor provisos imposing that both contexts are empty;
- the number of sequents corresponding to premises of $\text{R}\wedge$ in the cut sequence is 1;
- the value of cnp corresponding to $\text{R}\wedge$ is 1, since $\text{R}\wedge$ is additive;

and, for $i = 1, 2$,

- $\text{L}\wedge_i$ does not have neither closure provisos nor provisos imposing that both contexts are empty;
- the number of sequents corresponding to premises of $\text{L}\wedge_i$ in the cut sequence is 1;
- the value of cnp corresponding to $\text{L}\wedge_i$ is 1, since $\text{L}\wedge_i$ is additive.

3.1 Cut suitable calculus

The conditions allowing that a calculus enjoy cut elimination, are put together in the definition of a cut suitable calculus.

Definition 3. A sequent calculus is *cut suitable* if it is suitable and

- C1 if the calculus has a cardinality proviso for side l then that proviso imposes that the number of formulas at that side of each sequent in a rule is at most 1;
- C2 the axiom rule is in the calculus only if it has a cardinality proviso imposing that the contexts are empty if the left or the right weakening rule is not in the calculus;
- C3 contraction, multicut, and weakening rules, either for a connective or generic, and the cut rule, are such that
 - 1 the general multicut rule is in the calculus iff the contraction rules are present;
 - 2 the left (right) multicut rule is in the calculus iff the left (right) contraction rule is in the calculus and the right (left) contraction rule is not;

- 3 the left (right) multicut rule is in the calculus only if the calculus has a cardinality proviso for the right (left) side;
 - 4 the left (right) multicut rule for a connective is in the calculus iff the left (right) contraction rule for that connective is also in the calculus;
 - 5 the left (right) multicut rule for a connective c is in the calculus only if (i) introduction rules with closure provisos, the axiom rule and the generic weakening rule, are the only rules in the calculus that have a principal formula in the right (left) side of the conclusion that may have c as main connective, (ii) the cut rule is present, and (iii) the right (left) multicut rule is not in the calculus;
 - 6 multicut rules for connectives are not in the calculus only if the left and right weakening rules are in the calculus;
 - 7 the left weakening rule is in the calculus iff the right weakening is in the calculus;
 - 8 the left (right) weakening rule for a connective c is in the calculus only if (i) the right (left) introduction rules for c have closure provisos, and (ii) the right (left) weakening rule for c is not in the calculus;
 - 9 the contraction rules for a connective are not simultaneously in the calculus;
 - 10 the cardinality provisos in these rules are the ones imposed by the calculus;
- C4 a left (right) introduction rule r for a connective c is in the calculus only if
- 1 r does not have premises only if r does not have closure provisos over the l side if the weakening rule over side l is not in the calculus;
 - 2 either r does not have any cardinality proviso, or has the cardinality proviso imposed by the calculus, or has a cardinality proviso imposing that all the contexts are empty;
 - 3 all the left (right) introduction rules for c in the calculus have the same provisos;
- C5 a rule r with closure proviso(s) is in the calculus only if its number is at most 2, and
- 1 if it is 1 and the proviso is c left (right) then i) r is a right (left) introduction rule for c , and ii) there is a restriction in the calculus imposing that the number of formulas at the right (left) side is 1;
 - 2 if it is 2 then the provisos are over opposite sides, and if one is c left and the other is c' right then i) either r is a right introduction rule for c or a left introduction rule for c' , ii) if r is a right (left) introduction rule then all the left (right) introduction rules for c' (c) in the calculus have the same provisos;
 - 3 if r has a c left (right) closure proviso then there are in the calculus i) a left (right) contraction rule, generic or for c , and ii) a left (right) weakening rule, generic or for c ;
- C6 each pair of rules in the calculus with closure provisos is such that either for each side they have the same closure provisos or the provisos are all over different connectives;

C7 each pair of rules in the calculus formed by a right and a left introduction rule for the same connective is cut suitable.

The conditions in the definition of a cut suitable calculus avoid the problematic cases of the cut elimination proof.

4 Cut Elimination

In this section cut elimination is stated for any cut suitable calculus. The proof follows the Tait style proof of cut elimination [11] as described in [12] and the proofs of the lemmas are outlined.

We start by introducing some basic definitions needed in the cut elimination proof. Our reference is [12]. The *depth* of a formula φ , denoted by $|\varphi|$, is the maximum length of a branch in its construction tree minus 1. The *depth* of a deduction \mathcal{D} is the maximum length of a branch in \mathcal{D} minus 1. The *level* of a cut is the sum of the depths of the deductions of the premises. The *rank* of a cut over a formula φ is $|\varphi| + 1$. The *cutrank* of a deduction \mathcal{D} , denoted by $\text{cr}(\mathcal{D})$, is the maximum of the ranks of cuts in \mathcal{D} over formulae. If there are no cuts in \mathcal{D} over formulae, the cutrank is 0.

Lemma 1. Given a deduction \mathcal{D}° for $\vdash_{\mathcal{C}} s$ where s is obtained by a cut from deductions \mathcal{D} and \mathcal{D}' with a lower cutrank than \mathcal{D}° , in the context of a cut suitable calculus \mathcal{C} , then there is a deduction \mathcal{D}^\bullet for $\vdash_{\mathcal{C}} s$ with lower cutrank than \mathcal{D}° .

The proof of this lemma follows by complete induction on the level of the cuts. The base and the step follows by case analysis using the induction hypothesis and the cut suitable conditions.

The next lemma extends the previous one by proving a similar result for any deductions and so not only for deductions ending in a cut with greater rank than the cutrank of the deductions of the premises. That is it shows that for any deduction with non null cutrank there is a deduction with lower cutrank.

Lemma 2. Given a deduction \mathcal{D} for $\vdash_{\mathcal{C}} s$ with non null cutrank, where \mathcal{C} is a cut suitable calculus, then there is a deduction \mathcal{D}^\bullet for $\vdash_{\mathcal{C}} s$ with lower cutrank than \mathcal{D} .

The proof follows straightforwardly by complete induction on the depth of the deduction.

The cut elimination theorem can be seen as resulting by the successive application of the cutrank reduction lemma until a deduction without cuts is obtained.

Theorem 1. Given a deduction for $\vdash_{\mathcal{C}} s$ where \mathcal{C} is a cut suitable calculus then there is a deduction for $\vdash_{\mathcal{C}} s$ without cuts.

The proof follows straightforwardly by complete induction on the cutrank of the deduction.

5 Related Work

Herein we studied sufficient conditions for sequent calculi for propositional based logics to enjoy cut elimination. There are only a few works in the literature dedicating some attention to this issue, like [4, 7, 13] where sufficient conditions for a display calculus to enjoy cut elimination are described, and [10] where these conditions are studied in the context of substructural logics.

Interesting results appear also in [1] and [3], where a condition similar to the existence of a cut sequence for each pair of introduction rules, see Definition 2, is present. This condition, nevertheless, do not contemplate the relation between the number of premises in the cut sequence and the multiplicative and additive character of the rules.

6 Acknowledgments

The author wishes to express his gratitude to Cristina Sernadas for the suggestions made on an earlier version of this work, as well as to the participants of the Logic and Computation Seminar of the CLC for the useful feedback on a related presentation. This work was partially supported by FCT and FEDER through POCI, namely via the QuantLog POCI/MAT/55796/2004 Project.

References

1. A. Avron and I. Lev. Non-deterministic multiple-valued structures. *Journal of Logic and Computation*, 15:241–261, 2005.
2. M. Baaz and A. Leitsch. Cut normal forms and proof complexity. *Annals of Pure and Applied Logic*, 97:127–177, 1999.
3. M. Baaz and A. Leitsch. Cut-elimination and redundancy-elimination by resolution. *Journal of Symbolic Computation*, 29:149–176, 2000.
4. N. Belnap. Display logic. *Journal of Philosophical Logic*, 11:375–417, 1982.
5. S. R. Buss. An introduction to proof theory. In *Handbook of Proof Theory*, pages 1–78. Elsevier, 1998.
6. G. Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934–35. English translation in "The collected papers of Gerhard Gentzen", M. E. Szabo ed., North-Holland, 1969, pages 68–131.
7. R. Goré. Substructural logics on display. *Logic Journal of the IGPL*, 6(3):451–504, 1998.
8. P. Mateus, J. Rasga, and C. Sernadas. Modal sequent calculi labelled with truth values: Cut elimination. *Logic Journal of the IGPL*, 13(2):173–199, 2005.
9. J. Rasga. Cut elimination for a class of propositional based logics. Technical report, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2005. Submitted for publication. Preprint available from <http://slc.math.ist.utl.pt/jfr.html>.
10. G. Restall. *An Introduction to Substructural Logics*. Routledge, 2000.
11. W. W. Tait. Normal derivability in classical logic. In J. Barwise, editor, *The Syntax and Semantics of Infinitary Languages*, volume 72 of *LNM*, pages 204–236. Springer, 1968.

12. A. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, 1996.
13. H. Wansing. Strong cut-elimination in display logic. *Reports on Mathematical Logic*, 29:117–131, 1995. First German-Polish Workshop on Logic & Logical Philosophy (Bachotek, 1995).