Michel Parigot and Lutz Straßburger (eds.)

Proceedings of
Structures and Deduction 2009

ESSLLI'09 Workshop

Bordeaux, July 20–24, 2009

Preface

The topic of this workshop is the application of algebraic, geometric, and combinatorial methods in proof theory. In recent years many researchers have proposed approaches to understand and reduce ”syntactic beaucracy” in the presentation of proofs. Examples are proof nets, atomic flows, new deductive systems based on deep inference, and new algebraic semantics for proofs. These efforts have also led to new methods of proof normalisation and new results in proof complexity.

The workshop is relevant to a wide range of people. The list of topics includes among others: algebraic semantics of proofs, game semantics, proof nets, deep inference, tableaux systems, category theory, deduction modulo, cut elimination, complexity theory, etc.

The goal of the workshop is twofold: first, to bring together researchers from various fields who share the interest of understanding and dealing with structural properties of proofs and second, to provide an opportunity for PhD students and researchers to present and discuss their work with colleagues who work in the broad subject areas that are represented at ESSLLI.

The workshop is intended to be a sequel of the ICALP-workshop SD05 in Lisbon 2005.

Bordeaux,
July 2009

Michel Parigot
Lutz Straßburger

Programme Committee

Contents

An interesting link between type theory and topology

François Lamarche

LORIA and INRIA Nancy – Grand Est

Henri Poincaré invented both homology and homotopy theory around 1899. The spaces he used for homotopy were pieces of $\mathbb{R}^n$ glued together; the general concept of topological spaces had not been invented yet. He could not do the same for homology, so he resorted to simple combinatorial objects to define the necessary spaces, which he called simplicial complexes, and where computations were easy anyway. It is only in the forties that Eilenberg showed how do define homology directly on topological spaces, and in the early fifties that combinatorial objects were found—that we now call simplicial sets, and that are also due to Eilenberg and his students—which are sufficiently powerful to enable direct definitions and quite a few computations in homotopy.

The most fundamental concept of homotopy is the notion of path between two points in a space. For several years it has been tempting to give a logical meaning to this notion of path, namely as "a constructive proof that two points are equal". One way to formalize this intuitive notion would be to interpret the Martin-Löf type-theoretical equality predicate in a suitable category of spaces. The first positive results in this direction have been given by Michael Warren, working under the supervision of Steve Awodey. They are quite technical and depend very much on modern abstract homotopy theory à la Quillen.

I will present a very concrete model, which I also think is the simplest that has been discovered so far. It is based on a well-known specialization of simplicial set, for which I will define a concrete path object which doubles as an identity type, along with a suitable notion of dependent type.

I will try keep things as elementary as possible.

(Joint work with Robert Hein)

On the Lambek Calculus with One Division and One Primitive Type

Stepan Kuznetsov

Department of Mathematical Logic and Theory of Algorithms,
Faculty of Mechanics and Mathematics, Moscow State University
skuzn@inbox.ru

Abstract. The following theorem is proved: a formal language without the empty word is context-free if and only if it is generated by some $\mathrm{L}(\setminus; p_1)$-grammar, where $\mathrm{L}(\setminus; p_1)$ is the Lambek calculus with one division and one primitive type. To do that, we use a substitution of types which reduces derivability in $\mathrm{L}(\setminus)$ to derivability in $\mathrm{L}(\setminus; p_1)$.

1 Introduction

We consider a fragment of the Lambek calculus introduced in [3]—the calculus $\mathrm{L}(\setminus)$ (the Lambek calculus with one division). The set $\mathrm{Pr} = \{p_1, p_2, p_3, \ldots\}$ is called the set of *primitive types*. *Types* of $\mathrm{L}(\setminus)$ are built from primitive types using one binary connective $\setminus$, called *left division*; we shall denote the set of all types by $\mathrm{Tp}(\setminus)$. The set of those types that contain only $p_1, \ldots, p_N$ is denoted by $\mathrm{Tp}(\setminus; p_1, \ldots, p_N)$. Capital letters $(A, B, \ldots)$ range over types. Capital Greek letters range over finite (possibly empty) sequences of types; Λ stands for the empty sequence. Expressions of the form $\Gamma \to C$ are called *sequents* of $\mathrm{L}(\setminus)$.

Axioms: $A \to A$.

Rules:

$$\frac{A\Pi \to B}{\Pi \to A \setminus B}\ (\to\setminus) \text{ where } \Pi \neq \Lambda \qquad\qquad \frac{\Pi \to A \quad \Gamma B \Delta \to C}{\Gamma \Pi (A \setminus B) \Delta \to C}\ (\setminus\to)$$

The calculus $\mathrm{L}(\setminus; p_1, \ldots, p_N)$ is the conservative fragment of $\mathrm{L}(\setminus)$ that uses only types from $\mathrm{Tp}(\setminus; p_1, \ldots, p_N)$.

Example 1. $\mathrm{L}(\setminus) \vdash (p_1 \setminus p_2)\ ((p_3 \setminus p_2) \setminus p_4) \to ((p_3 \setminus p_1) \setminus p_4)$:

$$\dfrac{\dfrac{\dfrac{p_3 \to p_3 \qquad \dfrac{p_1 \to p_1 \qquad p_2 \to p_2}{p_1\ (p_1 \setminus p_2) \to p_2}\ (\setminus\to)}{p_3\ (p_3 \setminus p_1)\ (p_1 \setminus p_2) \to p_2}\ (\setminus\to)}{(p_3 \setminus p_1)\ (p_1 \setminus p_2) \to (p_3 \setminus p_2)}\ (\to\setminus) \qquad p_4 \to p_4}{\dfrac{(p_3 \setminus p_1)\ (p_1 \setminus p_2)\ ((p_3 \setminus p_2) \setminus p_4) \to p_4}{(p_1 \setminus p_2)\ ((p_3 \setminus p_2) \setminus p_4) \to ((p_3 \setminus p_1) \setminus p_4)}\ (\to\setminus)}\ (\setminus\to)$$

We also consider the calculus MCLL$'$—a variant of the *multiplicative fragment of cyclic linear logic without constants* (LLNC from [4], CLL* from [5]). Elements of a countable set $\mathrm{Var} = \{p_1, p_2, p_3, \ldots\}$ are called *variables*; $\mathrm{At} \leftrightharpoons \mathrm{Var} \cup \{\bar{q} \mid q \in \mathrm{Var}\}$ is the set of *atoms* (the symbol "$\leftrightharpoons$" here and further means "equals by definition"). Formulae of MCLL$'$ are built from atoms using two binary connectives: $\parr$ (multiplicative disjunction, "*par*") and $\otimes$ (multiplicative conjunction, "*tensor*"). We denote the set of all formulae by Fm and the set of formulae containing only variables $p_1, \ldots, p_N$ by $\mathrm{Fm}(p_1, \ldots, p_N)$. Capital Latin letters range over formulae. Capital Greek letters denote finite sequences of formulae; Λ stands for the empty sequence. Sequents of MCLL$'$ are of the form $\to \Gamma$. The *linear negation* is introduced externally as an inductively defined mapping $(\cdot)^{\perp} \colon \mathrm{Fm} \to \mathrm{Fm}$: $p_i^{\perp} \leftrightharpoons \bar{p}_i$, $\bar{p}_i^{\perp} \leftrightharpoons p_i$, $(A \parr B)^{\perp} \leftrightharpoons B^{\perp} \otimes A^{\perp}$, $(A \otimes B)^{\perp} \leftrightharpoons B^{\perp} \parr A^{\perp}$.

Axioms: $\to p_i\ \bar{p}_i$.

Rules:

$$\frac{\to \Gamma A B \Delta}{\to \Gamma (A \parr B) \Delta}\ (\to\parr) \text{ where } \Gamma\Delta \neq \Lambda \qquad \frac{\to \Gamma A \quad \to B \Delta}{\to \Gamma (A \otimes B) \Delta}\ (\to\otimes) \qquad \frac{\to \Gamma\Delta}{\to \Delta\Gamma}\ (\mathrm{rot})$$

It is easy to see that if $\mathrm{MCLL}' \vdash\ \to \Gamma$, then Γ consists of at least two formulae.

We define the *standard translation* from $\mathrm{Tp}(\backslash)$ to Fm: $\widehat{p_i} \leftrightharpoons p_i$, $\widehat{A \backslash B} \leftrightharpoons \widehat{A}^{\perp} \parr \widehat{B}$. In terms of this translation $\mathrm{L}(\backslash)$ is a conservative fragment of MCLL$'$: $\mathrm{L}(\backslash) \vdash A_1 \ldots A_n \to B \iff \mathrm{MCLL} \vdash\ \to \widehat{A}_n^{\perp} \ldots \widehat{A}_1^{\perp} \widehat{B}$ (see [5]).

Example 2. We shall here use the notation $A^{\frown}$ instead of $\widehat{A}$. By definition we have $\big((p_3 \backslash p_1) \backslash p_4\big)^{\frown} = (\bar{p}_1 \otimes p_3) \parr p_4$, $\big((p_3 \backslash p_2) \backslash p_4\big)^{\frown\perp} = \bar{p}_4 \otimes (\bar{p}_3 \parr p_2)$, and $\big(p_1 \backslash p_2\big)^{\frown\perp} = \bar{p}_2 \otimes p_1$. Hence due to Example 1 and because $\mathrm{L}(\backslash)$ is a fragment of MCLL$'$ we have $\mathrm{MCLL}' \vdash\ \to (\bar{p}_4 \otimes (\bar{p}_3 \parr p_2))\ (\bar{p}_2 \otimes p_1)\ ((\bar{p}_1 \otimes p_3) \parr p_4)$. The explicit derivation is as follows:

$$\cfrac{\cfrac{\cfrac{\to p_4\ \bar{p}_4 \qquad \cfrac{\cfrac{\cfrac{\cfrac{\to p_2\ \bar{p}_2 \qquad \to p_1\ \bar{p}_1}{\to p_2\ (\bar{p}_2 \otimes p_1)\ \bar{p}_1}\,(\to\otimes) \qquad \to p_3\ \bar{p}_3}{\to p_2\ (\bar{p}_2 \otimes p_1)\ (\bar{p}_1 \otimes p_3)\ \bar{p}_3}\,(\to\otimes)}{\to \bar{p}_3\ p_2\ (\bar{p}_2 \otimes p_1)\ (\bar{p}_1 \otimes p_3)}\,(\mathrm{rot})}{\to (\bar{p}_3 \parr p_2)\ (\bar{p}_2 \otimes p_1)\ (\bar{p}_1 \otimes p_3)}\,(\to\parr)}{\to p_4\ (\bar{p}_4 \otimes (\bar{p}_3 \parr p_2))\ (\bar{p}_2 \otimes p_1)\ (\bar{p}_1 \otimes p_3)}\,(\to\otimes)}{\to (\bar{p}_4 \otimes (\bar{p}_3 \parr p_2))\ (\bar{p}_2 \otimes p_1)\ (\bar{p}_1 \otimes p_3)\ p_4}\,(\mathrm{rot})}{\to (\bar{p}_4 \otimes (\bar{p}_3 \parr p_2))\ (\bar{p}_2 \otimes p_1)\ ((\bar{p}_1 \otimes p_3) \parr p_4)}\,(\to\parr)$$

In this paper we shall prove the following theorem (compare with [4], Proposition 2.1):

Theorem 1. *There exist types $A_1, \ldots, A_N \in \mathrm{Tp}(\backslash; p_1)$ such that for any sequent $\to \Gamma$, where $\Gamma = B_1 \ldots B_m$ and $B_1, \ldots, B_m \in \mathrm{Fm}(p_1, \ldots, p_N)$, we have*

$$\mathrm{MCLL}' \vdash\ \to \Gamma \iff \mathrm{MCLL}'(p_1) \vdash\ \to \Gamma[p_1 \leftarrow \widehat{A}_1, \ldots, p_N \leftarrow \widehat{A}_N].$$

The notation $\Gamma[p_1 \leftarrow \widehat{A}_1, \ldots, p_N \leftarrow \widehat{A}_N]$ means Γ with $\widehat{A}_1, \ldots, \widehat{A}_N$ substituted for $p_1, \ldots, p_N$ (and $\widehat{A}_1^\perp, \ldots, \widehat{A}_N^\perp$ for $\bar{p}_1, \ldots, \bar{p}_N$).

It easily follows from this theorem that the same property holds for $\mathrm{L}(\backslash)$ (due to conservativity):

Theorem 2. *There exist types $A_1, \ldots, A_N \in \mathrm{Tp}(\backslash; p_1)$ such that for any sequent $\Pi \to C$, where $\Pi = B_1 \ldots B_m$ and $B_1, \ldots, B_m, C \in \mathrm{Tp}(\, p_1, \ldots, p_N)$, we have*

$$\mathrm{L}(\backslash) \vdash \Pi \to C \iff \mathrm{L}(\backslash; p_1) \vdash (\Pi \to C)[p_1 \leftarrow A_1, \ldots, p_N \leftarrow A_N].$$

Let us prove Theorem 1. We define $p \leftrightharpoons p_1$ and $(D_1 \cdot \ldots \cdot D_m) \backslash C \leftrightharpoons D_m \backslash (D_{m-1} \backslash \ldots \backslash (D_2 \backslash (D_1 \backslash C)) \ldots)$. For given N we introduce types $A_1, \ldots, A_N$: $A_k \leftrightharpoons \bigl(p^{k+1} \cdot \bigl(((p \cdot p) \backslash p) \backslash p\bigr) \cdot p^{N-k+1}\bigr) \backslash p$ (where $p^1 \leftrightharpoons p$, $p^{m+1} \leftrightharpoons p^m \cdot p$).

Example 3. Let $N = 2$. We have

$$A_1 = \bigl(p \cdot p \cdot \bigl(((p \cdot p) \backslash p) \backslash p\bigr) \cdot p \cdot p\bigr) \backslash p,$$
$$A_2 = \bigl(p \cdot p \cdot p \cdot \bigl(((p \cdot p) \backslash p) \backslash p\bigr) \cdot p\bigr) \backslash p.$$

Since $\mathrm{L}(\backslash) \not\vdash p_1 \backslash p_2 \to p_2 \backslash p_1$, by Theorem 2 we have $\mathrm{L}(\backslash) \not\vdash A_1 \backslash A_2 \to A_2 \backslash A_1$.

The "$\Longrightarrow$" implication in Theorem 1 is trivial. We shall prove the other one. Let $R_k \leftrightharpoons \widehat{A}_k$. It is easy to see that

$$R_k = \underbrace{\bar{p} \parr \ldots \parr \bar{p}}_{N-k+1} \parr \bigl(\bar{p} \otimes (\bar{p} \parr \bar{p} \parr p_*)\bigr) \parr \underbrace{\bar{p} \parr \ldots \parr \bar{p}}_{k+1} \parr p_+;$$

$$R_k^\perp = \bar{p}_+ \otimes \underbrace{p \otimes \ldots \otimes p}_{k+1} \otimes \bigl((\bar{p}_* \otimes p \otimes p) \parr p\bigr) \otimes \underbrace{p \otimes \ldots \otimes p}_{N-k+1}.$$

(We assume that $\parr$s associate to the right and $\otimes$s associate to the left.) The subscripts $*$ and $+$ here mark concrete occurrences of p and $\bar{p}$ for further reference. Let us call R_k a *positive* formula and $R_k^\perp$ a *negative* formula.

2 Strong Proof Nets

We introduce the notion of *strong proof net*—a modification of proof nets from [7] for MCLL$'$. First we build a relation structure $\boldsymbol{\Omega}_\Gamma = \langle \Omega_\Gamma, <_\Gamma, \prec_\Gamma \rangle$. Let $\Gamma = A_1 \ldots A_m$. We put $\diamond$ signs before A_1 and between A_i and A_{i+1} $(i = 1, \ldots, m-1)$ ($\diamond$ is a new formal symbol, $\diamond \notin \mathrm{Fm}$): $\diamond A_1 \diamond A_2 \diamond \ldots \diamond A_m$. In this string we number all symbols except brackets (an atom is considered one symbol) and denote the set of pairs $\langle \text{symbol}, \text{number} \rangle$ by Ω_Γ. Elements of Ω_Γ are called *occurrences* of the corresponding symbols into Γ and denoted by lowercase Greek letters. For $\alpha = \langle s_1, k_1 \rangle$, $\beta = \langle s_2, k_2 \rangle \in \Omega_\Gamma$ we define $\alpha <_\Gamma \beta \iff k_1 < k_2$.

For a subformula of Γ the corresponding subset of Ω_Γ is called the occurrence of this subformula ($A_1, A_2, \ldots, A_m$ are also considered subformulae). If X is a subformula occurrence, then we denote by $\mathrm{l}(X)$ the occurrence of the connective or $\diamond$ immediately to the left of X and by $\mathrm{r}(X)$ the connective or $\diamond$ occurrence immediately to the right of X (if there is no such occurrence, $\mathrm{r}(X)$ is defined cyclically as the leftmost occurrence of $\diamond$).

The transitive closure of the union of parse trees of the formulae $A_1, \dots, A_m$ (where the vertices are occurrences, that is, elements of Ω_Γ) is called $\prec_\Gamma$.

We denote the set of all occurrences of $\invamp$ by $\Omega_\Gamma^{\invamp}$, of $\otimes$ by $\Omega_\Gamma^{\otimes}$, of $\diamond$ by $\Omega_\Gamma^{\diamond}$, of $p_1, p_2, \dots$ by $\Omega_\Gamma^{\mathrm{At}^+}$, of $\bar{p}_1, \bar{p}_2, \dots$ by $\Omega_\Gamma^{\mathrm{At}^-}$; $\Omega_\Gamma^{\invamp\diamond} \leftrightharpoons \Omega_\Gamma^{\invamp} \cup \Omega_\Gamma^{\diamond}$, $\Omega_\Gamma^{\otimes\invamp\diamond} \leftrightharpoons \Omega_\Gamma^{\otimes} \cup \Omega_\Gamma^{\invamp\diamond}$, $\Omega_\Gamma^{\mathrm{At}} \leftrightharpoons \Omega_\Gamma^{\mathrm{At}^+} \cup \Omega_\Gamma^{\mathrm{At}^-}$. For $X \subseteq \Omega_\Gamma$ we define $\#(X) \leftrightharpoons |X \cap \Omega_\Gamma^{\mathrm{At}^+}| - |X \cap \Omega_\Gamma^{\mathrm{At}^-}|$.

For $\alpha, \beta \in \Omega_\Gamma$ we define $\mathrm{In}(\alpha, \beta) \leftrightharpoons \{\gamma \mid \alpha <_\Gamma \gamma <_\Gamma \beta \text{ or } \beta <_\Gamma \gamma <_\Gamma \alpha\}$, $\mathrm{Out}(\alpha, \beta) \leftrightharpoons \Omega_\Gamma - (\mathrm{In}(\alpha, \beta) \cup \{\alpha, \beta\})$ (the minus sign here means set-theoretic difference).

Definition 1. *An oriented graph $\langle \Omega_\Gamma, \mathcal{C} \rangle$, where $\mathcal{C} \subseteq \Omega_\Gamma \times \Omega_\Gamma$, is called $<_\Gamma$-*planar *if for any $\langle \alpha, \beta \rangle \in \mathcal{C}$ and $\langle \gamma, \delta \rangle \in \mathcal{C}$ if $\{\alpha, \beta\} \cap \{\gamma, \delta\} = \emptyset$ then $\gamma \in \mathrm{In}(\alpha, \beta) \iff \delta \in \mathrm{In}(\alpha, \beta)$. Geometrically this means that the graph can be drawn in the upper semiplane without intersections if its vertices lie on the semiplane's border in the $<_\Gamma$ order.*

Definition 2. *A* strong proof net *is a triple $\mathfrak{N} = \langle \boldsymbol{\Omega}_\Gamma, \mathcal{A}, \mathcal{E} \rangle$, where $\mathcal{A} \subset \Omega_\Gamma \times \Omega_\Gamma$, $\mathcal{E} \subset \Omega_\Gamma \times \Omega_\Gamma$, that satisfies the following axioms:*

1. *$|\Omega_\Gamma^{\invamp\diamond}| - |\Omega_\Gamma^{\otimes}| = 2$;*
2. *$\mathcal{A}$ is a total function from $\Omega_\Gamma^{\otimes}$ to $\Omega_\Gamma^{\invamp\diamond}$;*
3. *$\mathcal{E}$ is a bijective function from $\Omega_\Gamma^{\mathrm{At}^+}$ to $\Omega_\Gamma^{\mathrm{At}^-}$, and if α is an occurrence of p_i then $\mathcal{E}(\alpha)$ is an occurrence of $\bar{p}_i$;*
4. *the graph $\langle \Omega_\Gamma, \mathcal{A} \cup \mathcal{E} \rangle$ is $<_\Gamma$-planar;*
5. *the graph $\langle \Omega_\Gamma, \mathcal{A} \cup \prec_\Gamma \rangle$ is acyclic;*
6. *for any subformula occurrence $X \subset \Omega_\Gamma$ we have $\widetilde{\mathcal{A}}(\mathrm{l}(X)) \neq \widetilde{\mathcal{A}}(\mathrm{r}(X))$, where the mapping $\widetilde{\mathcal{A}} \colon \Omega_\Gamma^{\otimes\invamp\diamond} \to \Omega_\Gamma^{\invamp\diamond}$ is defined as follows:*

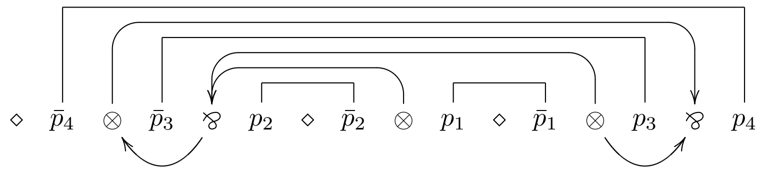
$$\widetilde{\mathcal{A}}(\alpha) \leftrightharpoons \begin{cases} \alpha & \text{if } \alpha \in \Omega_\Gamma^{\invamp\diamond}, \\ \mathcal{A}(\alpha) & \text{if } \alpha \in \Omega_\Gamma^{\otimes}. \end{cases}$$

Strong proof nets give a derivability criterion for MCLL':

Theorem 3. $\mathrm{MCLL}' \vdash\ \to \Gamma$ *if and only if there exists a strong proof net for $\to \Gamma$.*

This result is analogous to Theorem 7.12 from [7]. In fact, removing axiom 6 from the definition of strong proof net leads to the notion of *proof net* (not strong), that gives a derivability criterion for MCLL—a conservative fragment of SPNCL' from [7]. The calculus MCLL has the same axioms and rules as MCLL', but without the restriction $\Gamma\Delta \neq \Lambda$ in $(\to \invamp)$.

Example 4. The following figure shows the strong proof net for the sequent from Example 2:


$\diamond$ $\bar{p}_4$ $\otimes$ $\bar{p}_3$ $\invamp$ p_2 $\diamond$ $\bar{p}_2$ $\otimes$ p_1 $\diamond$ $\bar{p}_1$ $\otimes$ p_3 $\invamp$ p_4

Here (and further in the figures illustrating fragments of strong proof nets) the graphs $\mathcal{A}$ and $\mathcal{E}$ are drawn in the upper semiplane, and the relation $\prec_\Gamma$ (restricted to $\Omega_\Gamma^{\otimes \parr \diamond}$) is drawn in the lower semiplane.

Lemma 1. *If there exists a strong proof net $\mathfrak{N} = \langle \boldsymbol{\Omega}_\Gamma, \mathcal{A}, \mathcal{E} \rangle$, then Γ contains at least two formulae.*

Proof. Suppose the contrary: $\Gamma = A_1$. Let X be the occurrence of A_1 (as a subformula of Γ). We have $\mathrm{l}(X) = \mathrm{r}(X) = \langle \diamond, 0 \rangle$, whence $\widetilde{\mathcal{A}}(\mathrm{l}(X)) = \mathrm{l}(X) = \mathrm{r}(X) = \widetilde{\mathcal{A}}(\mathrm{r}(X))$. Contradiction with axiom 6. $\square$

Proof (of Theorem 3). Here we give only a sketch of the proof. The “only if” part is proved by constructing the strong proof net inductively from the derivation of $\to \Gamma$. To prove the “if” part we proceed by induction on the number of connective occurrences in Ω_Γ.

The induction base is trivial: in the case $\Omega_\Gamma^{\parr} \cup \Omega_\Gamma^{\otimes} = \emptyset$ our sequent can be either $\to p_i \bar{p}_i$ (axiom) or $\to \bar{p}_i p_i$ (derivable from $\to p_i \bar{p}_i$ by one application of (rot)).

Induction step. We define a new binary relation $\ll$ as the restriction of $\prec_\Gamma \cup \mathcal{A}$ to $\Omega_\Gamma^{\parr} \cup \Omega_\Gamma^{\otimes}$. Due to the acyclicity of $\prec_\Gamma \cup \mathcal{A}$ there exists an element $\gamma \in \Omega_\Gamma^{\parr} \cup \Omega_\Gamma^{\otimes}$ that is maximal with respect to $\ll$. If $\gamma \in \Omega_\Gamma^{\parr}$, then we replace it with $\diamond$ (thus getting a strong proof net with fewer connective occurrences), use the induction hypothesis, and apply the rule $(\to \parr)$. The restriction of this rule is satisfied due to Lemma 1.

Now let $\gamma \in \Omega_\Gamma^{\otimes}$, $\beta = \mathcal{A}(\gamma)$. We have $\beta \in \Omega_\Gamma^{\diamond}$, because γ is maximal with respect to $\ll$. We can assume that $\beta = \langle \diamond, 0 \rangle$ (that is, β is the leftmost occurrence of $\diamond$): in the other case we apply (rot) and do a cyclic transformation of the net.

We have $\Gamma = \Phi\,(A \otimes B)\,\Psi$, and the arc $\langle \gamma, \beta \rangle \in \mathcal{A}$ leads from the occurrence of $\otimes$ to the leftmost occurrence of $\diamond$. This arc divides the upper semiplane into two parts. We take the upper part as the strong proof net for $\to B\Psi$ and the lower part for $\to \Phi A$. Axioms 2, 3, 4, 5, and 6 are checked trivially. Axiom 1 is proved in [7], Lemma 7.10.

Now, using the induction hypothesis for $\to \Phi A$ and $\to B\Psi$, we conclude that these sequents are derivable in MCLL$'$. Therefore MCLL$' \vdash \to \Phi\,(A \otimes B)\,\Psi$ by application of $(\to \otimes)$. $\square$

We shall consider $\mathcal{E}$ a non-oriented graph on Ω_Γ; the edges of $\mathcal{E}$ will be called *links*. Intuitively, links connect occurrences of atoms that come from one axiom leaf in the derivation tree. For a link $\mathbf{C}$ with vertices α and β we define $\mathrm{In}(\mathbf{C}) \leftrightharpoons \mathrm{In}(\alpha, \beta)$ and $\mathrm{Out}(\mathbf{C}) \leftrightharpoons \mathrm{Out}(\alpha, \beta)$. The graph $\langle \Omega_\Gamma, \mathcal{E} \rangle$ is $<_\Gamma$-planar, whence $\#(\mathrm{In}(\mathbf{C})) = \#(\mathrm{Out}(\mathbf{C})) = 0$. Brackets divide the upper semiplane into *regions*. Each link has the *inner* and the *outer* region. Evidently, each region must contain at least one occurrence of $\parr$ or $\diamond$ (because $\mathcal{A} \cup \mathcal{E}$ is $<_\Gamma$-planar). On the other hand, it is easy to see that $|\Omega_\Gamma^{\parr \diamond}|$ is equal to the number of regions, so each region contains only one occurrence of $\parr$ or $\diamond$.

Let X and Y be two subformula occurrences such that $X \cap Y = \emptyset$. We define the *fragment from X to Y* as $\{\alpha \in \Omega_\Gamma \mid X <_\Gamma \{\alpha\} <_\Gamma Y\}$ if $X <_\Gamma Y$ and as $\{\alpha \in \Omega_\Gamma \mid \{\alpha\} <_\Gamma X \text{ or } Y <_\Gamma \{\alpha\}\}$ if $Y <_\Gamma X$ (other cases are impossible).

If $\mathbf{C}$ is a link and $\mathscr{K}$ is a subset of Ω_Γ, we define $\mathscr{D}(\mathbf{C},\mathscr{K}) = \mathrm{In}(\mathbf{C})$, if $\mathscr{K} \subseteq \mathrm{In}(\mathbf{C})$ and $\mathscr{D}(\mathbf{C},\mathscr{K}) = \mathrm{Out}(\mathbf{C})$ otherwise (if $\mathscr{K}$ is a fragment from one subformula occurrence to another, and it doesn't contain vertices of $\mathbf{C}$, then in this case we have $\mathscr{K} \subseteq \mathrm{Out}(\mathbf{C})$, thus getting $\mathscr{K} \subseteq \mathscr{D}(\mathbf{C},\mathscr{K})$ always). It is easy to see that $\#(\mathscr{D}(\mathbf{C},\mathscr{K})) = 0$.

Further we shall sometimes omit the word "occurrence".

3 Proof of Theorem 1

We have $\mathrm{MCLL}'(p) \vdash \to \Gamma[p_1 \leftarrow R_1, \dots, p_N \leftarrow R_N]$. This sequent contains formulae from $\mathrm{Fm}(p)$ (since Γ contains only variables $p_1, \dots, p_N$), thus this statement is equivalent to $\mathrm{MCLL}' \vdash \to \Gamma[p_1 \leftarrow R_1, \dots, p_N \leftarrow R_N]$. Hence there exists a strong proof net $\mathfrak{N}$ for this sequent. We shall modify $\mathfrak{N}$ to get a strong proof net $\mathfrak{N}'$ for $\to \Gamma$.

First we shall prove some lemmata about $\mathfrak{N}$.

Lemma 2. *The number of positive formula occurrences is equal to the number of negative formula occurrences.*

Proof. Suppose there are m_1 positive formula occurrences and m_2 negative formula occurrences. Then we have $0 = \#(\Omega_\Gamma) = (m_2 - m_1)(N+3)$, whence $m_2 = m_1$. □

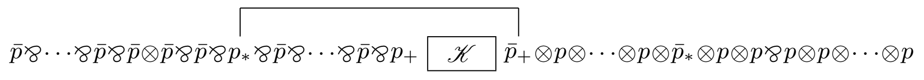
Lemma 3. *Any occurrence of p_* from R_k is connected to an occurrence of $\bar{p}_*$ from some $R_{k'}^{\perp}$ (possibly, $k \neq k'$).*

Proof. Suppose the contrary. Let $\mathbf{C}$ be a link from some p_* that leads not to $\bar{p}_*$. We consider 3 cases:

Case 1: $\mathbf{C}$ leads to an occurrence of $\bar{p}$ from the same R_k. Since $\#(\mathrm{In}(\mathbf{C})) = 0$, this is the neighbour occurrence of $\bar{p}$. But then there are two $⅋$ in the outer region of $\mathbf{C}$. Contradiction.

Case 2: $\mathbf{C}$ leads to an occurrence of $\bar{p}$ from another $R_{k'}$. There are $⅋$ connectives on both sides of p_* and at least on one side of any $\bar{p}$ from $R_{k'}$, therefore either in the inner or in the outer region of $\mathbf{C}$ there are two $⅋$ connectives. Contradiction.

Case 3: $\mathbf{C}$ leads to $\bar{p}_+$ (from some $R_{k'}^{\perp}$):

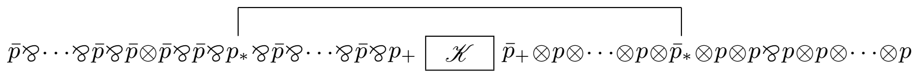

$\bar{p}⅋\cdots⅋\bar{p}⅋\bar{p}\otimes\bar{p}⅋\bar{p}⅋p_*⅋\bar{p}⅋\cdots⅋\bar{p}⅋p_+ \boxed{\mathscr{K}} \bar{p}_+\otimes p\otimes\cdots\otimes p\otimes\bar{p}_*\otimes p\otimes p⅋p\otimes p\otimes\cdots\otimes p$

Let $\mathscr{K}$ be the fragment from R_k to $R_{k'}^{\perp}$. The fragment $\mathscr{K}$ consists of several occurrences of positive and negative formulae and connectives between them. Let m_1 be the number of positive formulae there and m_2 be the number of the negative ones. Then we get $0 = \#(\mathscr{D}(\mathbf{C},\mathscr{K})) = 1 - (k+1) + \#(\mathscr{K}) = -k + (m_2 - m_1)(N+3)$, therefore $k = (m_2 - m_1)(N+3)$. This is absurd, because $1 \leq k \leq N$. □

Lemma 4. *Any occurrence of $\bar{p}_*$ from $R_k^\perp$ is connected to an occurrence of p_* from some $R_{k'}$.*

Lemma 5. *If an occurrence of p_* from R_k is connected to an occurrence of $\bar{p}_*$ from $R_{k'}^\perp$, then $k = k'$.*

Proof.

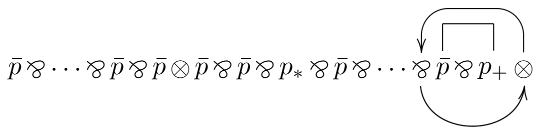

$\bar{p}⅋\cdots⅋\bar{p}⅋\bar{p}\otimes\bar{p}⅋\bar{p}⅋p_*⅋\bar{p}⅋\cdots⅋\bar{p}⅋p_+$ $\boxed{\mathscr{K}}$ $\bar{p}_+\otimes p\otimes\cdots\otimes p\otimes\bar{p}_*\otimes p\otimes p⅋p\otimes p\otimes\cdots\otimes p$

Let $\mathscr{K}$ be the fragment from R_k to $R_{k'}^\perp$. Occurrences of positive formulae inside $\mathscr{K}$ are in one-to-one correspondence with occurrences of negative formulae there (by the links connecting p_* and $\bar{p}_*$), thus $\#(\mathscr{K}) = 0$. Therefore $0 = \#(\mathscr{D}(\mathbf{C}, \mathscr{K})) = k' - k + \#(\mathscr{K})$. $k = k'$. □

Lemma 6. *Any occurrence of p_+ from R_k is connected to an occurrence of $\bar{p}_+$ from some $R_{k'}^\perp$ (possibly, $k \neq k'$).*

Proof. We consider several cases:

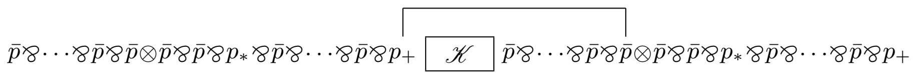
Case 1: the occurrence of p_+ is connected to an occurrence of $\bar{p}$ from the same R_k by a link $\mathbf{C}$. Since $\#(\mathrm{In}(\mathbf{C})) = 0$, it is the rightmost occurrence of $\bar{p}$:


$\bar{p}⅋\cdots⅋\bar{p}⅋\bar{p}\otimes\bar{p}⅋\bar{p}⅋p_*⅋\bar{p}⅋\cdots⅋\bar{p}⅋p_+\otimes$

But then immediately to the right of R_k there is an occurrence τ of $\otimes$ (otherwise there would be two occurrences of ⅋ in one region) connected by an $\mathcal{A}$-arc with the occurrence π of ⅋ on the left side of $\mathbf{C}$ (due to $<_\Gamma$-planarity of $\mathcal{A} \cup \mathcal{E}$). On the other hand, $\pi \prec_\Gamma \tau$. Contradiction with the acyclicity of $\mathcal{A} \cup \prec_\Gamma$.

Case 2: p_+ is connected by a link $\mathbf{C}$ to an occurrence of $\bar{p}$ from another $R_{k'}$, but not the third to the left from p_*. In this case either in the inner (if $R_{k'}$ lies to the left from R_k) or in the outer region of $\mathbf{C}$ there are two occurrences of ⅋. Contradiction.

Case 3: p_+ is connected to the third to the left from p_* occurrence of $\bar{p}$ in R_k by a link $\mathbf{C}$:


$\bar{p}⅋\cdots⅋\bar{p}⅋\bar{p}\otimes\bar{p}⅋\bar{p}⅋p_*⅋\bar{p}⅋\cdots⅋\bar{p}⅋p_+$ $\boxed{\mathscr{K}}$ $\bar{p}⅋\cdots⅋\bar{p}⅋\bar{p}\otimes\bar{p}⅋\bar{p}⅋p_*⅋\bar{p}⅋\cdots⅋\bar{p}⅋p_+$

Let $\mathscr{K}$ be the fragment from R_k to $R_{k'}$. The same argument as in lemma 5 shows that $\#(\mathscr{K}) = 0$. But then $\#(\mathscr{D}(\mathbf{C}, \mathscr{K})) = -(N - k' + 1) + \#(\mathscr{K}) \neq 0$. Contradiction.

Case 4: p_+ is connected to $\bar{p}_*$ from some $R_{k'}^\perp$. Contradiction with lemma 4: $\bar{p}_*$ is connected to an occurrence of p_*, but not p_+.

So the only possible situation is the **5-th case:** p_+ is connected to an occurrence of $\bar{p}_+$ from some $R_{k'}^\perp$. □

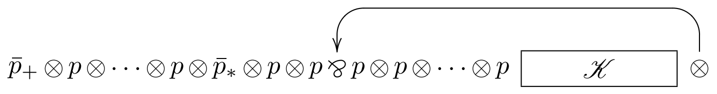
Lemma 7. *Any occurrence of $\bar{p}_+$ from $R_k^\perp$ is connected to an occurrence of p_+ from some $R_{k'}$.*

Let us call an occurrence of a connective *old* if it is not inside an occurrence of a positive or negative formula (thus this occurrence comes from the original sequent $\to \varGamma$).

Lemma 8. *If the occurrence τ of $\otimes$ is old, then $\mathcal{A}(\tau)$ is also old.*

Proof. Suppose the contrary. Let $\mathcal{A}(\tau)$ be not old and let $\mathcal{A}(\tau) <_\varGamma \tau$ (in the other case we proceed symmetrically with respect to the arc $\langle \tau, \mathcal{A}(\tau)\rangle$). Consider several cases:

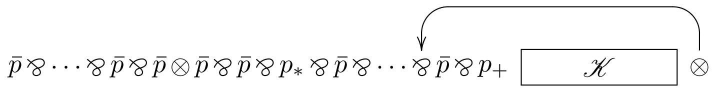
Case 1: $\mathcal{A}(\tau)$ lies inside some $R_k^\perp$:


$\bar{p}_+ \otimes p \otimes \cdots \otimes p \otimes \bar{p}_* \otimes p \otimes p \parr p \otimes p \otimes \cdots \otimes p$
$\mathscr{K}$
$\otimes$

Let $\mathscr{D} = \mathrm{In}(\tau, \mathcal{A}(\tau))$ and $\mathscr{K}$ be the fragment of $\varOmega_\varGamma$ between $R_k^\perp$ and τ. We have $\#(\mathscr{K}) = 0$ and $\#(\mathscr{D}) = 0$. Contradiction.

Case 2: $\mathcal{A}(\tau)$ is an occurrence of $\parr$ inside R_k, but not the second from the right side. From the two sets $\mathrm{In}(\tau, \mathcal{A}(\tau))$ and $\mathrm{Out}(\tau, \mathcal{A}(\tau))$ we take the one not containing p_* from this R_k and call it $\mathscr{D}$. For $\mathscr{K}$ we take the subset of $\varOmega_\varGamma$ containing all elements of $\mathscr{D}$, except those from R_k. Now we proceed exactly as in case 1.

Case 3: $\mathcal{A}(\tau)$ is the second from the right side occurrence of $\parr$ in R_k:


$\bar{p} \parr \cdots \parr \bar{p} \parr \bar{p} \otimes \bar{p} \parr \bar{p} \parr \bar{p} \parr p_* \parr \bar{p} \parr \cdots \parr \bar{p} \parr p_+$
$\mathscr{K}$
$\otimes$

We define $\mathscr{D}$ and $\mathscr{K}$ as in case 1. The numbers of p_* and $\bar{p}_*$ in $\mathscr{K}$ are equal. Therefore, the number of p_+ and $\bar{p}_+$ in $\mathscr{K}$ are also equal. On the other hand, the same is true for $\mathscr{D}$. Contradiction: the number of $\bar{p}_+$ in $\mathscr{D}$ is the same as in $\mathscr{K}$, but the number of p_+ is greater by one. □

Now we define a strong proof net $\mathfrak{N}'$ for the original sequent $\to \varGamma$: $\mathfrak{N}' \leftrightharpoons \langle \boldsymbol{\varOmega}_\varGamma, \mathcal{A}', \mathcal{E}' \rangle$. Here $\mathcal{A}'$ contains the arcs of $\mathcal{A}$ that start at old $\otimes$ occurrences, and edges of $\mathcal{E}'$ connect those occurrences of p_k and $\bar{p}_k$ for which the occurrences of p_* and $\bar{p}_*$ from the corresponding R_k and $R_k^\perp$ are connected by edges of $\mathcal{E}$. It easily follows from the lemmata above that $\mathfrak{N}'$ is a strong proof net for $\to \varGamma$, therefore $\mathrm{MCLL} \vdash \to \varGamma$. Q. E. D.

4 Grammars

We call an *alphabet* an arbitrary finite non-empty set. The set of all non-empty words over the alphabet $\varSigma$ (i. e., finite sequences of elements of $\varSigma$) is denoted by $\varSigma^+$. Any subset of $\varSigma^+$ is called a *formal language* (without the empty word) over $\varSigma$.

Definition 3. *An* L($\backslash$)-grammar *is a triple* $\mathcal{G} = \langle \Sigma, H, \rhd \rangle$*, where* Σ *is an alphabet,* $H \in \mathrm{Tp}(\backslash)$*, and* $\rhd$ *is a finite correspondence between* $\mathrm{Tp}(\backslash)$ *and* Σ *(i. e.,* $\rhd \subset \mathrm{Tp}(\backslash) \times \Sigma$*). The* language generated by $\mathcal{G}$ *is the set of all words* $a_1 \dots a_n$ *over* Σ *for which there exist types* $B_1, \dots, B_n$ *such that* $\mathrm{L}(\backslash) \vdash B_1 \dots B_n \to H$ *and* $B_i \rhd a_i$ *for all* $i \le n$*. We shall denote this language by* $\mathcal{L}(\mathcal{G})$*.*

Definition 4. *A* context-free grammar without ϵ-rules *is a quadruple* $G = \langle N, \Sigma, P, S \rangle$*, where* N *and* Σ *are two disjoint alphabets,* $P \subset N \times (N \cup \Sigma)^+$*,* P *is finite, and* $S \in N$*. We define a binary relation* $\Rightarrow_G$ *as follows: for all* $\omega, \psi \in (N \cup \Sigma)^*$ *we have* $\omega \Rightarrow_G \psi$ *if and only if* $\omega = \eta A \theta$*,* $\psi = \eta \beta \theta$*, and* $\langle A, \beta \rangle \in P$ *for some* $A \in N$*,* $\beta, \eta, \theta \in (N \cup \Sigma)^*$*. The binary relation* $\Rightarrow_G^*$ *is the reflexive transitive closure of* $\Rightarrow_G$*. The language* $\mathcal{L}(G) \rightleftharpoons \{w \in \Sigma^+ \mid S \Rightarrow_G^* w\}$ *is the* language generated by G*. Such languages are called context-free.*

These two notions of formal grammar are equivalent in the following sense:

Theorem 4. *A formal language is context-free if and only if it is generated by some* L($\backslash$)*-grammar.*

The "if" part follows from Gaifman's theorem [1] as shown in [2], Proposition 1. The "only if" part is a trivial corollary of Pentus' theorem [6]. (Theorem 2 from [2] is not sufficient, because our definition of L($\backslash$)-grammars allows non-primitive types to be used as H.)

L($\backslash; p_1, \dots, p_N$)-grammars are defined exactly as L($\backslash$)-grammars, but with L($\backslash; p_1, \dots, p_N$) instead of L($\backslash$).

Theorem 5. *A formal language is context-free if and only if it is generated by some* L($\backslash; p_1$)*-grammar.*

Proof. The "if" part follows from Theorem 4 due to conservativity.

Let us prove the "only if" part. For a given context-free language Theorem 4 gives us an L($\backslash$)-grammar. Let it be $\mathcal{G} = \langle \Sigma, H, \rhd \rangle$. Let N be the maximal subscript of a primitive type used in $\mathcal{G}$. Then $\mathcal{G}$ is an L($\backslash; p_1, \dots, p_N$)-grammar. Let $H' = H[p_1 \leftarrow A_1, \dots, p_N \leftarrow A_N]$ and $\rhd' = \{\langle B[p_1 \leftarrow A_1, \dots, p_N \leftarrow A_N], a \rangle \mid B \rhd a\}$, where $A_1, \dots, A_N$ are taken from Theorem 2. Now for the L($\backslash; p_1$)-grammar $\mathcal{G}' = \langle \Sigma, H', \rhd' \rangle$ we have $\mathcal{L}(\mathcal{G}') = \mathcal{L}(\mathcal{G})$ due to Theorem 2. $\square$

Acknowledgments I would like to thank Prof. M. Pentus for guiding me into the subject and constant attention to my work. I am grateful to the participants of the seminar "Logical Problems in Computer Science" at the Department of Mathematical Logic and Theory of Algorithms, Faculty of Mathematics and Mechanics, MSU for their help in finding and eliminating mistakes in the proof.

References

1. Bar-Hillel, Y., Gaifman, C., and Shamir, E.: On categorial and phrase-structure grammars. Bull. Res. Council Israel Sect. F, 9F:1–16 (1960)

2. Buszkowski, W.: The equivalence of unidirectional Lambek categorial grammars and context-free grammars. Zeitschr. für math. Logik und Grundl. der Math. 31, 369–384 (1985)
3. Lambek, J.: The mathematics of sentence structure. American Math. Monthly 65(3), 154–170 (1958)
4. Métayer, F.: Polynomial equivalence among systems LLNC, LLNC$_a$ and LLNC$_0$. Theor. Comput. Sci. 227(1), 221–229 (1999)
5. Pentus, M.: Equivalent types in Lambek calculus and linear logic. Preprint No. 2 of the Department of Math. Logic, Steklov Math. Institute, Series Logic and Comput. Sci., Moscow (1992)
6. Pentus, M.: Lambek grammars are context free. Proc. of the 8th Annual IEEE Symposium on Logic in Computer Science, 429–433. IEEE Computer Society Press, Los Alamitos, California (1993)
7. Pentus, M.: Free monoid completeness of the Lambek calculus allowing empty premises. Logic Colloquium '96: proc. of the colloquium held in San Sebastian, Spain, July 9–15, 1996. Editors J. M. Larrazabal, D. Lascar and G. Mints. LNL, vol. 12, pp. 171–209. Springer, Berlin etc. (1998)

Resolution Refinements for Cut-Elimination based on Reductive Methods [⋆]

Stefan Hetzl[2], Alexander Leitsch[1], Tomer Libal[1], Daniel Weller[1], and Bruno Woltzenlogel Paleo[1]

[1] {leitsch, shaolin, weller, bruno}@logic.at
Institute of Computer Languages (E185),
Vienna University of Technology,
Favoritenstraße 9, 1040 Vienna, Austria

[2] hetzl@lix.polytechnique.fr
INRIA Saclay –Île-de-France
École Polytechnique – LIX
91128 Palaiseau, France

Abstract. Traditional reductive cut-elimination and *CERES* seem to be methods of entirely different nature and hence hard to compare. This short paper describes ongoing research that aims at comparing and possibly combining them in ways that retain that best features of each method.

1 Introduction

Cut-elimination theorems and algorithms that actually perform the elimination of cuts from proofs are among the most prominent results and techniques of proof theory and of logic in general. Originally devised as a way to prove consistency [8], cut-elimination also plays a major role in: automated theorem proving, where the sub-formula property corollary allows a bottom-up construction of proofs; analysis of mathematical proofs, where the elimination of cuts corresponds to the elimination of undesired mathematical lemmas [2]; extraction of interpolants via Maehara's lemma, which requires cut-free proofs [11]; semantics and identity of proofs, where confluence of cut-elimination is important [10].

Therefore, it is important to compare different cut-elimination algorithms and devise new and hopefully better ones, as such improvements can potentially have implications for several areas of proof theory.

In this paper, in particular, we compare reductive cut-elimination methods and cut-elimination by resolution (*CERES*) (defined in Sections 2 and 3) and we propose a way to combine them via resolution refinements (Section 4). These refinements restrict the atomic cut normal forms (ACNFs, which are not cut-free, but whose cuts are at most atomic) obtainable by *CERES* essentially to those that are obtainable by reductive methods. The long range aim is to implement and use various refinements in the analysis of formalized mathematical proofs.

⋆ Supported by the Austrian Science Fund (project P19875)

2 Reductive Cut-Elimination Methods

The standard method of cut-elimination is that of Gentzen defined in his famous "Hauptsatz" [8]. The method is essentially a nondeterministic algorithm extracted from his (constructive) proof. Its characteristic feature is a rewriting system that rewrites proofs by shifting cut inferences upwards (rank reduction) and by reducing the complexity of cut-formulas when these are the main formulas of the inferences immediately above the cut (grade reduction). The result is a proof containing cuts that occur on top of the proof and whose cut-formulas are at most atomic. These atomic cuts can be simply eliminated, because their conclusion sequents are equal to their premise sequents.

3 *CERES*

The resolution-based method *CERES* for cut-elimination in classical logic has been defined in [6] and further developed in [3] and [9].

The method inductively defines a set of pairs (with a *clause* in the first component and a *projection* (to this clause) in the second component) C_ν for every node ν in a **skolemized**[3] proof φ:

- If ν is an occurrence of an axiom sequent $S(\nu)$, S' is the subsequent of $S(\nu)$ containing only the ancestors of cut-formula occurrences and S is the whole axiom, then $C_\nu = \{\langle S', S\rangle\}$.
- Let ν' be the predecessor of ν in a unary inference ρ.
  Let $C_{\nu'} = \{\langle c_1, \psi_1\rangle, \ldots, \langle c_n, \psi_n\rangle\}$.
  (a) The auxiliary formulas of ν' are ancestors of cut-formula occurrences. Then
  $$C_\nu = C_{\nu'}$$
  (b) The auxiliary formulas of ν' are not ancestors of cut-formula occurrences. Then
  $$C_\nu = \{\langle c_1, \rho(\psi_1)\rangle, \ldots, \langle c_n, \rho(\psi_n)\rangle\}$$
  where $\rho(\psi)$ denotes the derivation that is obtained from ψ by applying ρ to its end-sequent.
- Let ν_1, ν_2 be the predecessors of ν in a binary inference ρ.
  (a) The auxiliary formulas of ν_1, ν_2 are ancestors of cut-formula occurrences. Then
  $$C_\nu = C_{\nu_1} \cup C_{\nu_2}.$$
  (b) The auxiliary formulas of ν_1, ν_2 are not ancestors of cut-formula occurrences. Then
  $$C_\nu = C_{\nu_1} \times C_{\nu_2}.$$

[3] A skolemized proof is a proof with an end-sequent in skolem normal form. For any proof φ there is a proof φ' such that φ' is skolemized and the end-sequent of φ' is a structural skolem normal form of the end-sequent of φ [5].

where

$$C \times \mathcal{D} = \{\langle c \circ d, \rho(\psi, \chi)\rangle \mid \langle c, \psi\rangle \in C, \langle d, \chi\rangle \in D\}$$

where $c \circ d$ is the merge of clauses and $\rho(\psi, \chi)$ denotes the derivation that is obtained from the derivations ψ and χ by applying the binary inference ρ.

The *characteristic clause set* $\mathrm{CL}(\varphi)$ of φ is defined as C_{ν_0}, where ν_0 is the root.

$\mathrm{CL}(\varphi)$ is always unsatisfiable [6]. Therefore, there is a resolution refutation of $\mathrm{CL}(\varphi)$, which can be grounded and then used as a skeleton where each leaf clause receives its corresponding instantiated projection on top. Finally, the resolution and factoring inferences can be replaced by cuts and contractions, respectively, yielding a proof of the end-sequent of φ in ACNF (possibly with the addition of contractions in the bottom of the proof).

The main advantage of *CERES* over reductive cut-elimination methods is that it is implicitly capable of detecting redundancies in the input proof φ and eliminating them. Therefore, *CERES* can, in the best cases, produce ACNFs that are non-elementarily smaller than ACNFs produced by reductive methods [6]. More precisely, there are proofs φ such that, for all ACNFs $\varphi_{\triangleright_{GT}}$ obtained via Gentzen's or Tait's reductive cut-elimination methods, there exists an ACNF $\varphi_{\mathbf{CERes}}$ obtained via *CERES* such that:

$$\frac{|\varphi_{\triangleright_{GT}}|}{|\varphi_{\mathbf{CERes}}|} = O(\overbrace{2^{2^{\cdots}}}^{|\varphi|})$$

where $|\psi|$ denotes the size of the proof ψ.

For this reason, *CERES* is computationally superior than reductive cut-elimination methods, especially considering that the converse (i.e. proofs whose ACNFs via reductive cut-elimination would be non-elementary smaller than ACNFs via *CERES*) is not possible [6]. However, the price paid by *CERES* is its increased non-confluence (i.e. *CERES* can usually produce more ACNFs than reductive cut-elimination methods) and a correspondingly large search space for refutations. In some cases, such as for Fuerstenberg's proof of the infinitude of primes, current theorem provers like Otter and Prover9 were unable to refute the characteristic clause set [4].

4 Resolution Refinements for Cut-Elimination

In order to tackle the problem of the large search space for refutations, the chosen approach was the development of resolution refinements that reduce the number of *CERES* ACNFs that are not reductive ACNFs[4].

[4] Strictly speaking, ACNFs produced by CERes are structurally very different from ACNFs produced by reductive cut-elimination methods. In the former the atomic cuts occur in the bottom, while in the latter they occur in the top of the ACNF. However, the ACNFs can be compared with respect to their *canonic refutation* [7]. In this paper,

The following examples show some kinds of proofs whose characteristic clause sets admit refutations that lead to ACNFs that are not obtainable with reductive cut-elimination methods. Each example motivates the development of a different refinement.

4.1 Blocking the Resolution of Literals from Different Cuts

Consider the proof φ below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P\alpha \vdash P\alpha}{P\alpha, \neg P\alpha \vdash P\alpha}\,w_l}{P\alpha \vdash \neg P\alpha \rightarrow P\alpha}\,\rightarrow_r}{\forall x Px \vdash \neg P\alpha \rightarrow P\alpha}\,\forall_l}{\forall x Px \vdash \forall x(\neg Px \rightarrow Px)}\,\forall_r \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Ps \vdash Ps}{\vdash \neg Ps, Ps}\,\neg_r \qquad Ps \vdash Ps}{\neg Ps \rightarrow Ps \vdash Ps, Ps}\,\rightarrow_l}{\neg Ps \rightarrow Ps \vdash Ps}\,c_r}{\forall x(\neg Px \rightarrow Px) \vdash Ps}\,\forall_l \qquad \dfrac{Ps \vdash Ps}{Ps \vdash \exists y Py}\,\exists_r}{\forall x(\neg Px \rightarrow Px) \vdash \exists y Py}\,cut}{\forall x Px \vdash \exists y Py}\,cut$$

Its characteristic clause set is:

$$\mathrm{CL}(\varphi) \equiv \{\underbrace{\vdash P\alpha}_{c_1};\underbrace{Ps \vdash Ps}_{c_2};\underbrace{Ps \vdash Ps}_{c_3};\underbrace{Ps \vdash}_{c_4}\}$$

This characteristic clause admits the following resolution refutation δ:

$$\dfrac{c_1 \qquad c_4}{\vdash}\,R$$

δ is used as a skeleton for an ACNF that will have an atomic cut where the cut formula occurrences are Ps from c_4 and the instance Ps from the occurrence $P\alpha$ from c_1. But, if reductive cut-elimination methods had been used, this could not happen, because reductive cut-elimination methods are local. As the cuts are shifted upwards, grade reduction always keeps cut-formula occurrences of a cut paired (via the new cuts) with cut-formula occurrences of the same original cut. But the literals of c_1 come from ancestors of the lowermost cut, while the literals of c_4 come from ancestors of the uppermost cut. δ is effectively pairing cut formula occurrences from different cuts.

In order to prevent this class of refutations exemplified by δ, the ancestors of cut formula occurrences can be annotated with labels in such a way that two occurrences have the same label iff they are *cut-linked*.

Definition 1 (Cut-Linkage). *Two (sub)formula occurrences ν_1 and ν_2 in a proof φ are* cut-linked *if and only if there is a cut ρ such that ν_1 is an ancestor of ν_i and ν_2 is an ancestor of ν_j where ν_i and ν_j are auxiliary occurrences of ρ.*

The labels for cut-linkage in φ are shown below:

two ACNFs obtained from the same input proof are considered equal if they have the same canonic refutation.

$$\dfrac{\varphi_1 \qquad \varphi_2}{\forall x Px \vdash \exists y Py}\, cut$$

where φ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{P\alpha \vdash [P\alpha]_1}{P\alpha, \neg[P\alpha]_1 \vdash [P\alpha]_1}\, w_l}{P\alpha \vdash [\neg P\alpha \to P\alpha]_1}\, \to_r}{\forall x Px \vdash [\neg P\alpha \to P\alpha]_1}\, \forall_l}{\forall x Px \vdash \forall x[(\neg Px \to Px)]_1}\, \forall_r$$

and φ_2 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[Ps]_1 \vdash [Ps]_2}{\vdash \neg[Ps]_1, [Ps]_2}\, \neg_r \qquad [Ps]_1 \vdash [Ps]_2}{[\neg Ps \to Ps]_1 \vdash [Ps]_2, [Ps]_2}\, \to_l}{[\neg Ps \to Ps]_1 \vdash [Ps]_2}\, c_r}{\forall x[(\neg Px \to Px)]_1 \vdash [Ps]_2}\, \forall_l \qquad \dfrac{[Ps]_2 \vdash Ps}{[Ps]_2 \vdash \exists y Py}\, \exists_r}{\forall x[(\neg Px \to Px)]_1 \vdash \exists y Py}\, cut$$

The new characteristic clause set is essentially the same as before, but now the literals have the labels that indicate the cuts from which they originate:

$$\mathrm{CL}(\varphi) \equiv \{\underbrace{\vdash [P\alpha]_1}_{c_1};\underbrace{[Ps]_1 \vdash [Ps]_2}_{c_2};\underbrace{[Ps]_1 \vdash [Ps]_2}_{c_3};\underbrace{[Ps]_2 \vdash}_{c_4}\}$$

We define *R_{cl}-resolution* as resolution restricted in such a way that two literals can only be resolved if they have the same labels (i.e. if they originated from the same cut). It is clear that δ is not an R_{cl}-refutation.

4.2 Blocking the Resolution of Literals from the Same Branch of a Cut

The previously described refinement of R_{cl}-refutation still can produce refutations whose corresponding ACNFs would not be obtainable by reductive cut-elimination methods. This can occur, for example, when the proof contains only one cut but the cut-formula is valid, as shown in the proof φ below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P\alpha \vdash [P\alpha]_1}{P\alpha \vdash [\neg P\alpha]_1, [P\alpha]_1}\, w_r}{P\alpha \vdash [\neg P\alpha \lor P\alpha]_1}\, \lor_r}{\forall x Px \vdash [\neg P\alpha \lor P\alpha]_1}\, \forall_l}{\forall x Px \vdash [\forall x(\neg Px \lor Px)]_1}\, \forall_r \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Pt \vdash [Pt]_1}{[\neg Pt]_1, Pt \vdash}\, \neg_l}{[\neg Pt]_1 \vdash \neg Pt}\, \neg_r \qquad [Pt]_1 \vdash Pt}{[\neg Pt \lor Pt]_1 \vdash Pt, \neg Pt}\, \lor_l}{[\neg Pt \lor Pt]_1 \vdash Pt \lor \neg Pt}\, \lor_r}{[\forall x(\neg Px \lor Px)]_1 \vdash Pt \lor \neg Pt}\, \forall_l}{\forall x Px \vdash Pt \lor \neg Pt}\, cut$$

Its characteristic clause set is:

$$\mathrm{CL}(\varphi) \equiv \{\underbrace{\vdash [P\alpha]_1}_{c_1};\underbrace{\vdash [Pt]_1}_{c_2};\underbrace{[Pt]_1 \vdash}_{c_3}\}$$

Note that all clauses of the set have the same label, because φ has only one cut. This means that, for this case, any unrestricted R-refutation would also be an R_{cl}-refutation. Hence R_{cl}-resolution does not really help in this case.

Let δ be the R_{cl}-refutation below:

$$\dfrac{c_2 \qquad c_3}{\vdash}\ R_{cl}$$

Both c_2 and c_3 contain literals originating from the right branch of the cut. By executing reductive cut-elimination on φ, on the other hand, the final atomic cut will necessarily have an instance of $P\alpha$ and the Pt from the second branch of the $\vee_l$ rule as its cut-formula occurrences. In general, in reductive cut-elimination methods the atomic cuts of the resulting ACNF must pair occurrences originating from different branches of the original cuts. This is so, because in the grade reduction rewrite rules, it is never the case that the new cuts pair occurrences that are subformulas of the same cut-formula occurrences.

To prevent refutation as δ above, *side-labels* l and r can be added to the ancestors of cut formula occurrences, indicating whether they are ancestor from the left or from the right cut formula occurrence. This is shown below:

$$
\begin{prooftree}
\AxiomC{$P\alpha \vdash [P\alpha]^l_1$}
\RightLabel{w_r}
\UnaryInfC{$P\alpha \vdash [\neg P\alpha]^l_1, [P\alpha]_1$}
\RightLabel{$\vee_r$}
\UnaryInfC{$P\alpha \vdash [(\neg P\alpha \vee P\alpha)]^l_1$}
\RightLabel{$\forall_l$}
\UnaryInfC{$\forall x Px \vdash [(\neg P\alpha \vee P\alpha)]^l_1$}
\RightLabel{$\forall_r$}
\UnaryInfC{$\forall x Px \vdash [\forall x(\neg Px \vee Px)]^l_1$}
\AxiomC{$Pt \vdash [Pt]^r_1$}
\RightLabel{$\neg_l$}
\UnaryInfC{$[\neg Pt]^r_1, Pt \vdash$}
\RightLabel{$\neg_r$}
\UnaryInfC{$[\neg Pt]^r_1 \vdash \neg Pt$}
\AxiomC{$[Pt]^r_1 \vdash Pt$}
\RightLabel{$\vee_l$}
\BinaryInfC{$[(\neg Pt \vee Pt)]^r_1 \vdash Pt, \neg Pt$}
\RightLabel{$\vee_r$}
\UnaryInfC{$[(\neg Pt \vee Pt)]^r_1 \vdash Pt \vee \neg Pt$}
\RightLabel{$\forall_l$}
\UnaryInfC{$[\forall x(\neg Px \vee Px)]^r_1 \vdash Pt \vee \neg Pt$}
\RightLabel{cut}
\BinaryInfC{$\forall x Px \vdash Pt \vee \neg Pt$}
\end{prooftree}
$$

$$\mathrm{CL}(\varphi) \equiv \{\underbrace{\vdash [P\alpha]^l_1}_{c_1};\underbrace{\vdash [Pt]^r_1}_{c_2};\underbrace{[Pt]^r_1 \vdash}_{c_3}\}$$

R_{cls}-resolution is defined as R_{cl}-resolution with the additional constraint that two literals can only be resolved if their side labels are different. From the side-labelled $\mathrm{CL}(\varphi)$ above, one can see that the R_{cl}-refutation δ is not an R_{cls}-refutation, because the literals of c_2 and c_3 have the same side-label r and hence cannot be resolved with each other.

4.3 Blocking the Resolution of Literals from Different Positions in Cuts

Still it is possible to have R_{cls}-refutations whose corresponding ACNFs are not obtainable via reductive cut-elimination methods. This fact can be exemplified by the proof φ below:

$$
\begin{prooftree}
\AxiomC{$P\alpha \vdash [P\alpha]^l_1$}
\RightLabel{$\forall_l$}
\UnaryInfC{$\forall z Pz \vdash [P\alpha]^l_1$}
\RightLabel{$\forall_r$}
\UnaryInfC{$\forall z Pz \vdash [\forall x Px]^l_1$}
\RightLabel{w_r}
\UnaryInfC{$\forall z Pz \vdash [\forall x Px]^l_1, [\forall y Py]^l_1$}
\RightLabel{$\vee_r$}
\UnaryInfC{$\forall z Pz \vdash [(\forall x Px \vee \forall y Py)]^l_1$}
\AxiomC{$[Pt]^r_1 \vdash Pt$}
\RightLabel{$\exists_r$}
\UnaryInfC{$[Pt]^r_1 \vdash \exists w Pw$}
\RightLabel{$\forall_l$}
\UnaryInfC{$[\forall x Px]^r_1 \vdash \exists w Pw$}
\AxiomC{$[Ps]^r_1 \vdash Ps$}
\RightLabel{$\exists_r$}
\UnaryInfC{$[Ps]^r_1 \vdash \exists w Pw$}
\RightLabel{$\forall_l$}
\UnaryInfC{$[\forall y Py]^r_1 \vdash \exists w Pw$}
\RightLabel{$\vee_l$}
\BinaryInfC{$[(\forall x Px \vee \forall y Py)]^r_1 \vdash \exists w Pw, \exists w Pw$}
\RightLabel{c_r}
\UnaryInfC{$[(\forall x Px \vee \forall y Py)]^r_1 \vdash \exists w Pw$}
\RightLabel{cut}
\BinaryInfC{$\forall z Pz \vdash \exists w Pw$}
\end{prooftree}
$$

Its characteristic clause set is:

$$\mathrm{CL}(\varphi) \equiv \{\underbrace{\vdash [P\alpha]^l_1}_{c_1}; \underbrace{[Pt]^r_1 \vdash}_{c_2}; \underbrace{[Ps]^r_1 \vdash}_{c_3}\}$$

And it admits the following R_{cls}-refutation δ:

$$\dfrac{c_1 \qquad c_3}{\vdash}\ R_{cls}$$

No ACNF produced by reductive cut-elimination would have an atomic cut whose cut formula occurrences come from $P\alpha$ and Ps. Instead, $P\alpha$ would be resolved with Pt, because $P\alpha$ and Pt originate from the left disjunct of the cut-formula, while Ps originates from the right disjunct, and grade reduction mantains this structure when it creates new cuts of smaller formula complexity.

To forbid refutations like δ, a more strict labeling, called atomic cut linkage, of the cut-formula ancestors can be devised.

Definition 2 (Atomic Cut-Linkage). *Two atomic (sub)formula occurrences ν_1 and ν_2 in a proof φ are* atomically cut-linked *if and only if there is a cut ρ such that ν_1 is an ancestor of $\lfloor \nu_i \rfloor_\pi$ and ν_2 is an ancestor of $\lfloor \nu_j \rfloor_\pi$ where π is the position of an atomic sub-formula and ν_i and ν_j are auxiliary occurrences of ρ.*

In the proof below, atomic subformula occurrences of cut ancestors are given labels such that if two occurrences have the same label, then they are atomic cut-linked:

$$
\begin{prooftree}
\AxiomC{$P\alpha \vdash [P\alpha]_1$}
\RightLabel{$\forall_l$}
\UnaryInfC{$\forall z Pz \vdash [P\alpha]_1$}
\RightLabel{$\forall_r$}
\UnaryInfC{$\forall z Pz \vdash \forall x [Px]_1$}
\RightLabel{w_r}
\UnaryInfC{$\forall z Pz \vdash \forall x [Px]_1, \forall y [Py]_2$}
\RightLabel{$\vee_r$}
\UnaryInfC{$\forall z Pz \vdash (\forall x [Px]_1 \vee \forall y [Py]_2)$}
\AxiomC{$[Pt]_1 \vdash Pt$}
\RightLabel{$\exists_r$}
\UnaryInfC{$[Pt]_1 \vdash \exists w Pw$}
\RightLabel{$\forall_l$}
\UnaryInfC{$\forall x [Px]_1 \vdash \exists w Pw$}
\AxiomC{$[Ps]_2 \vdash Ps$}
\RightLabel{$\exists_r$}
\UnaryInfC{$[Ps]_2 \vdash \exists w Pw$}
\RightLabel{$\forall_l$}
\UnaryInfC{$\forall y [Py]_2 \vdash \exists w Pw$}
\RightLabel{$\vee_l$}
\BinaryInfC{$(\forall x [Px]_1 \vee \forall y [Py]_2) \vdash \exists w Pw, \exists w Pw$}
\RightLabel{c_r}
\UnaryInfC{$(\forall x [Px]_1 \vee \forall y [Py]_2) \vdash \exists w Pw$}
\RightLabel{cut}
\BinaryInfC{$\forall z Pz \vdash \exists w Pw$}
\end{prooftree}
$$

The characteristic clause set with the atomic cut-linkage labels is:

$$\mathrm{CL}(\varphi) \equiv \{\underbrace{\vdash [P\alpha]_1}_{c_1}; \underbrace{[Pt]_1 \vdash}_{c_2}; \underbrace{[Ps]_2 \vdash}_{c_3}\}$$

R_{acl}-resolution is defined as R-resolution with the restriction that two literals can only be resolved if they have the same atomic cut-linkage label. Clearly, as desired, δ is not an R_{acl}-refutation.

We can now give a uniform definition of the refined resolution rules

Definition 3 (Refined Resolution and Factoring Rules). *The* resolution rule R *shown below:*

$$\cdot \frac{\Gamma_1 \vdash \Delta_1, A_1 \qquad A_2, \Gamma_2 \vdash \Delta_2}{(\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2)mgu(A_1, A_2)} R$$

where $\Gamma_1 \vdash \Delta_1, A_1$ and $A_2, \Gamma_2 \vdash \Delta_2$ are variable-disjoint clauses, is a:

– Cut-Linkage Refined Resolution Rule R_{cl} *iff A_1 and A_2 are cut-linked.*
– Cut-Linkage/Sides Refined Resolution Rule R_{cls} *iff A_1 and A_2 are cut-linked and from opposite sides (branches) of a cut.*
– Atomic Cut-Linkage Refined Resolution Rule R_{acl} *iff A_1 and A_2 are atomically cut-linked.*

Analogously, the restricted rules of factoring *should also be restricted so that, if $A_1, \ldots, A_n$ are the factorized atoms, then the factoring is a:*

– Cut-Linkage Factoring F_{cl} *iff $A_1, \ldots, A_n$ are pairwise cut-linked.*
– Cut-Linkage/Sides Factoring F_{cls} *iff $A_1, \ldots, A_n$ are pairwise cut-linked and from the same side (branch) of a cut.*
– Atomic Cut-Linkage Refined Resolution Rule F_{acl} *iff $A_1, \ldots, A_n$ are pairwise atomically cut-linked.*

5 Refined Refutability

The original proof of the refutability of the characteristic clause set shows that the characteristic clause set is unsatisfiable by constructing a refutation in sequent calculus **LK** [6]. Then it relies on the completeness of the unrestricted resolution calculus, which guarantees that refutations of unsatisfiable clause sets exist.

However, with the restrictions imposed by the refinements, one cannot rely on the completeness of resolution anymore. Indeed, for *arbitrary* clause sets with *arbitrary* labels, R_{cl}-resolution, R_{cls}-resolution and R_{acl}-resolution are clearly incomplete. Nevertheless, Theorem 1 shows that for *characteristic* clause sets extracted from proofs with the labeling done in the *specific* ways defined in Section 4, refined refutations always exist.

Theorem 1 (Refutability of the Characteristic Clause Set). *For any proof φ,* $\mathrm{CL}(\varphi)$ *is R-refutable, R_{cl}-refutable, R_{cls}-refutable and R_{acl}-refutable.*

Proof. The full detailed proof is under development in the unfinished PhD thesis of Bruno Woltzenlogel Paleo. What follows is a basic informal outline of the ideas of the proof.

Firstly, it can be noted that every R_{acl}-refutation is an R_{cls}-refutation, every R_{cls}-refutation is an R_{cl}-refutation, and every R_{cl}-refutation is an R-refutation. Hence, it suffices to show that CL(φ) is R_{acl}-refutable.

We prove the refutability by constructing an R_{acl}-refutation, as follows:

- Let φ' be an atomic cut normal form of φ obtained by reductive methods (i.e. by applying rank and grade reduction, but no elimination of atomic cuts).
- Lemma 1 (Subsumption of the Characteristic Clause Sets under cut reduction):
  Show that CL(φ') is subsumed by CL(φ) [7].
- Lemma 2 (Invariance of the Atomic Cut Linkage Labeling under cut-reduction): Show that, when a cut reduction is performed, the labels of the cut-formula occurrences of the new cuts are the same, so that in φ', the atomic cuts always resolve two atoms that have the same labels.
- Lemma 3 (Canonic Refutation):
  Show that CL(φ') admits a canonic refutation [7], which can be extracted from φ' roughly by taking the cut-relevant part of φ', which is composed of atomic cut inferences only, and transforming these atomic cuts into resolution inferences. Let δ' be this canonic refutation.
- By Lemma 1, δ' can be lifted to a R-refutation δ of CL(φ), because CL(φ) subsumes CL(φ').
- By Lemma 2, δ is an R_{acl}-refutation of CL(φ).

6 Conclusions and Future Work

The refinements defined in this paper correspond, in various degrees, to the simulation of reductive methods within *CERES*. This allows a tradeoff between confluence of cut-elimination and size of the ACNFs, and consequently also some control on the search space for refutations.

The structural differences between ACNFs produced by *CERES* and reductive methods indicate some possible directions for future work in this area. It is noticeable that an ACNF produced by *CERES* ends with a series of contraction inferences. The duplications of formula occurrences (which are eventually contracted in the end) occur because of three different reasons:

1. Duplications of subproofs are intrinsic to the proccess of (reductive) cut-elimination in classical logics (due to rank reduction over contraction inferences).
2. Parts of the input proof are duplicated to appear in many projections. This is necessary to allow projections to be plugged on top of the refutation of the characteristic clause set. It seems that the contractions that exist due to this source of duplications could be avoided either by a more careful

construction and combination of the projections with the refutation or by a postprocessing step in which the contractions would be shifted upwards until they meet the weakening inferences that introduce one of their auxiliary formula occurrences, in which case both the contraction and the weakening could be eliminated.

3. The construction of the characteristic clause set may be seen as a standard CNF-transformation of the *characteristic clause term* [7]. The standard CNF transformation (which distributes disjunction over conjunction, or in this case products over sums in the charateristic clause term) can cause an exponential increase in size. The contractions associated with this source of duplications are intrinsic to cut-elimination by resolution using standard CNF-transformation, but it might be fruitful to investigate the possibility of using structural CNF-transformations, for which these duplications do not occur [1]. However, this approach would imply a radical change in the concepts of characteristic clause set and projections, because structural CNF-transformation adds fresh predicate symbols to the signature of the input proof.

An improvement of *CERES* that eliminates the second and third sources of contractions mentioned above would not only improve the efficiency of the method but also make it suitable for substructural logics in which contraction rules are not available. A deeper understanding of the relation between *CERES* and reductive methods seems to be crucial to achieve this improvement.

References

1. M. Baaz, U. Egly, and A. Leitsch. Normal form transformations. In A. Voronkov A. Robinson, editor, *Handbook of Automated Reasoning*, pages 275–333. Elsevier, 2001.
2. Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Cut-Elimination: Experiments with CERES. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR) 2004*, volume 3452 of *Lecture Notes in Computer Science*, pages 481–495. Springer, 2005.
3. Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Proof Transformation by CERES. In Jonathan M. Borwein and William M. Farmer, editors, *Mathematical Knowledge Management (MKM) 2006*, volume 4108 of *Lecture Notes in Artificial Intelligence*, pages 82–93. Springer, 2006.
4. Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Ceres: An analysis of fürstenberg's proof of the infinity of primes. *Theor. Comput. Sci.*, 403(2-3):160–175, 2008.
5. Matthias Baaz and Alexander Leitsch. Cut normal forms and proof complexity. *Annals of Pure and Applied Logic*, 97:127–177, 1999.
6. Matthias Baaz and Alexander Leitsch. Cut-elimination and Redundancy-elimination by Resolution. *Journal of Symbolic Computation*, 29(2):149–176, 2000.
7. Matthias Baaz and Alexander Leitsch. Towards a clausal analysis of cut-elimination. *Journal of Symbolic Computation*, 41:381–410, 2006.
8. G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1934–1935.

9. Stefan Hetzl. *Characteristic Clause Sets and Proof Transformations*. PhD thesis, Vienna University of Technology, 2007.
10. Lutz Straßburger. What is a logic, and what is a proof? In Jean-Yves Beziau, editor, *Logica Universalis*, pages 135–145. Birkhäuser, 2005. Updated version at http://www.lix.polytechnique.fr/~lutz/papers/WhatLogicProof.pdf.
11. G. Takeuti. *Proof Theory*. North-Holland, Amsterdam, 2 edition, 1987.

An Algorithmic Interpretation of a Deep Inference System

Kai Brünnler and Richard McKinley

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

In our paper [1] we set out to find an algorithmic interpretation of deep inference [3]. Here we mean an algorithmic interpretation in the same sense as the lambda-calculus is an algorithmic interpretation of natural deduction. Starting from natural deduction for the conjunction-implication fragment of intuitionistic logic we design a corresponding deep inference system together with reduction rules on proofs that allow a fine-grained simulation of beta-reduction. The principal way of composing our proof terms is not function application, as in the lambda calculus, but is function composition, as in composition of arrows in a category. So it is a system of *categorical combinators* and similar to some categorical combinators that Curien designed in the eighties, in order to serve as a target for the compilation of functional programming languages [2]. A very accessible introduction to those combinators and how they led to the development of explicit substitution calculi, like the $\lambda\sigma$-calculus, can be found in Hardin [4].

The difference between our combinators and Curien's is in the presentation of the defining adjunctions of a cartesian closed category. In our presentation proof terms can be thought of graphically: they are built using vertical composition (the usual composition of morphisms) and horizontal composition (the connectives).

In the talk we will present material from our paper [1] but also speak about ongoing work towards proving two conjectures that arise from this paper, which concern confluence and (a kind of) strong normalisation for the reduction system.

References

1. Kai Brünnler and Richard McKinley. An algorithmic interpretation of a deep inference system. In I. Cervesato, H. Veith, and A. Voronkov, editors, *LPAR 2008*, volume 5330 of *Lecture Notes in Computer Science*, pages 482—496. Springer-Verlag, 2008. `http://www.iam.unibe.ch/~kai/Papers/2008aidis.pdf`.
2. Pierre-Louis Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* Research Notes in Theoretical Computer Science. Birkhäuser, 2nd edition, 1993.
3. Alessio Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1):1–64, 2007.
4. Therese Hardin. From categorical combinators to $\lambda\sigma$-calculi, a quest for confluence. Technical report, INRIA Rocquencourt, 1992. Available from `http://hal.inria.fr/inria-00077017/`.

A DPLL Proof Procedure for Propositional Iterated Schemata

Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier

LIG, CNRS/Grenoble INP
Bâtiment IMAG C - 220, rue de la Chimie,
38400 Saint Martin d'Hères, France
Vincent.Aravantinos@imag.fr, Ricardo.Caferra@imag.fr,
Nicolas.Peltier@imag.fr

Abstract. We investigate *iterated schemata* whose syntax integrates arithmetic parameters, indexed propositional variables (e.g. P_i), and iterated conjunctions/disjunctions (e.g. $\bigwedge_{i=1}^{n} P_i$, where n is a *parameter*). Using this formalism gives some extra information about the structure of the problem, which can be used in order to prove such a schema *for all values of the parameters.* The structure of the formula is used as a guide for the structure of the proof. We start by defining a Davis-Putnam-Logemann-Loveland based procedure for iterated schemata that we show to be sound and complete (w.r.t. satisfiability). There cannot exist a complete calculus for unrestricted schemata but we show, by extending our procedure, that cycles can be detected in the proof tree, thus allowing to prove some schemata that are neither provable nor refutable in the initial calculus. An example shows how this method allows to tackle non-trivial problems. We give evidence that the proposed calculus is a useful tool for identifying classes of schemata for which complete calculus exist.

1 Introduction

Problems formalised in propositional logic are frequently the instances of more general problems. More expressive languages can convey this generality, but this often requires a reformulation that loses the *structure* of the original problem. *Schemata* can be used to generalize these statements while preserving the structure of the problem.

In this paper, we present a proof procedure for schemata of propositional formulae. We handle schemata containing *iterated conjunctions and disjunctions* (e.g. $\bigvee_{i=1}^{n} p_i$) ranging over sets of natural numbers, and depending on *integer parameters.* A typical example is the unsatisfiability of the schema corresponding to the pigeonhole problem:

$$\left(\bigwedge_{i=1}^{n} \bigvee_{j=1}^{n-1} P_{i,j}\right) \wedge \left(\bigwedge_{k=1}^{n-1} \bigwedge_{\substack{i,j=1 \\ i \neq j}}^{n} (\neg P_{i,k} \vee \neg P_{j,k})\right)$$

To the best of our knowledge, there is no similar work neither in logic nor in automated deduction. Some formalisms have been proposed for denoting *term* schemata and performing inference on them [1–3], but the extension to the logical level has never been considered.

Iteration schemata are ubiquitous in mathematics and in computer science. They occur frequently for instance in circuit verification where the formulae modeling the circuits are frequently parameterized by a natural number n (e.g. denoting the number of bits). The use of iterated schemata is also extremely useful for the formalization of mathematical proofs, because it allows one to express infinite proof sequences, which avoids explicit use of the induction principle. This has been used to avoid working with more expressive logical formalisms [4].

For every finite $n \in \mathbb{N}$, these schemata can be expanded into a propositional formula (in the standard sense) which can be decided by a SAT-solver (at least theoretically). However, proving that the schema is valid (resp. unsatisfiable) *for every value of* n is much more difficult. In general it requires some form of mathematical induction. Designing proof procedures able to reason directly which such schemata (without grounding them) would significantly increase the expressive power and allow for shorter, more meaningful and structured proofs.

In this paper, we restrict ourselves to schemata of *propositional* formulae (i.e. the underlying language is classical propositional logic). No sound and refutationally complete proof procedures can exist for propositional schemata (see Theorem 1). However it is easy to define proof procedures that are complete w.r.t. satisfiability i.e. that terminate iff the considered formula has a model (similarly to first-order logic w.r.t. finite models).

In a previous work [5] we have presented a first proof procedure for propositional schemata in a tableaux-like style (called STAB, for Schemata TABleaux). New decomposition rules have been designed in order to handle iterations and a looping rule has been proposed to detect and avoid cycles in the derivation. For some subclass of formulae, the cycle detection rule is powerful enough to ensure termination, yielding a decision procedure for this class of schemata.

In this paper, we present another proof procedure for propositional schemata which is based on different principles. It is more general than STAB in the sense that it terminates more often, but it is also more complex and less intuitive. It can be seen as an extension of the Davis-Putnam-Logemann-Loveland procedure [6]. This extension is not straightforward because the number of literals is unbounded (it depends on the value of the integer parameters).

In the context of propositional formulae schemata, it turns out that the use of a DPLL procedure has some important advantages w.r.t. tableaux. In contrast to the “shallow” rules composing STAB, the new calculus may operate at deep positions in the schema. This feature turns out to be *essential* for detecting cycles and proving unsatisfiability when the schema at hand contains nested iterations. In particular, this has the important consequence that the simultaneous simplification of an “unbounded” number of subformulae is possible – a useful form of “meta-inference”. This is shown informally on a simple example.

2 Schemata of Propositional formulae

In the rest of the paper indexed propositions are written $P_i, P_{i,j}, P_{i,j,k}, \dots$ and integer variables are written $i, j, k, \dots$ or $n, p, q, \dots$ Schemata are denoted by $S, S_1, \dots$, sets of schemata by $\mathcal{S}$, interpretations by $\mathcal{I}, \mathcal{J}$, tableaux by $\mathcal{T}, \mathcal{T}_0, \dots$ and nodes in a tableau $\alpha, \beta, \dots$

2.1 Syntax

Definition 1 (Linear a-terms). *Let $\mathcal{IV}$ be an infinite set of* integer variables*. The set of* linear a-terms, *written $\mathcal{LT}$ is the smallest set containing $\mathcal{IV}$, $\mathbb{Z}$ and s.t. for all $k \in \mathbb{Z}$, $t_1, t_2 \in \mathcal{LT}$: $t_1 + t_2 \in \mathcal{LT}$ and $k.t_1 \in \mathcal{LT}$. For all $t \in \mathcal{LT}$, $\mathcal{V}\mathrm{ar}(t)$ is the set of integer variables that occur in t. A* ground term *is a term t s.t. $\mathcal{V}\mathrm{ar}(t) = \emptyset$. A* linear constraint *is a first-order formula whose atoms are of the form $t_1 \bullet t_2$ (t_1 and t_2 linear a-tems) where $\bullet \in \{=, <, >, \leq, \geq\}$.*

Definition 2 (Indexed propositions). *Let $(\mathcal{P}_k)_{k \in \mathbb{N}}$ be a family of symbols. For all $k \in \mathbb{N}$, $P \in \mathcal{P}_k$, and $t_1, \dots, t_k \in \mathcal{LT}$, $P_{t_1,\dots,t_k}$ is an* indexed proposition*. An indexed proposition $P_{t_1,\dots,t_k}$ s.t. $t_1, \dots, t_k \in \mathbb{Z}$ is called a* propositional variable*; a propositional variable or its negation is a* literal*.*

Definition 3 (Schemata). *The set of* formula schemata *is the smallest set s.t.*

– *$\top$, $\bot$ are formula schemata.*
– *Each indexed proposition is a formula schema.*
– *If S_1, S_2 are formula schemata then $S_1 \vee S_2$, $S_1 \wedge S_2$ and $\neg S_1$ are formula schemata.*
– *If S is a formula schema, $i \in \mathcal{IV}$, $t_1, t_2 \in \mathcal{LT}$ not containing i, and C_i a linear constraint containing at least $t_1 \leq i \leq t_2$ then $\bigwedge_{i|C_i} S$ and $\bigvee_{i|C_i} S$ are formula schemata (such schemata are called* iterations*).*

In contrast to our previous work [5] we allow formulae containing disjunctions and conjunctions specified by finite sets of arithmetic constraints. The condition $t_1 \leq i \leq t_2$ guarantees that the set is finite for every interpretation of the integer variables, which ensures that every ground instance of the schema can be transformed into an equivalent propositional formula (e.g. $\bigvee_{i=1}^{\infty} \dots$ is forbidden). When C_i is of the form $t_1 \leq i \wedge i \leq t_2$, we write $\bigwedge_{i=t_1}^{t_2} S$ for $\bigwedge_{i|C_i} S$. We write $S[T]$ to mean that the schema T occurs in the construction of the schema S. An iteration $\bigwedge_{i|C} S$ is said to be *empty* if C is unsatisfiable.

Example 1.

$$S = Q_1 \wedge \bigwedge_{i=1}^{n} \left(P_i \wedge \bigvee_{\substack{1 \leq j \leq n+1 \\ i \neq j}} \neg Q_j \vee Q_{j+1} \right) \quad \text{is a formula schema.}$$

Q_1, P_i, Q_j and Q_{j+1} are indexed propositions. The only iterations occurring in S are $\bigwedge_{i=1}^{n} \left(P_i \wedge \bigvee_{\substack{1 \leq j \leq n+1 \\ i \neq j}} \neg Q_j \vee Q_{j+1} \right)$ and $\bigvee_{\substack{1 \leq j \leq n+1 \\ i \neq j}} \neg Q_j \vee Q_{j+1}$.

An occurrence of a variable i is *bound* in S if S contains an iteration of the form $\bigoplus_{i|C_i} S_i$ ($\bigoplus \in \{\bigwedge, \bigvee\}$). An occurrence of i is *free* (or is a *parameter* of S) if it occurs in S, but not in the scope of an iteration $\bigoplus_{i|C_i} S_i$. *Substitutions* on integer variables are defined as usual. We write $\{t_1/i_1, \ldots, t_k/i_k\}$ the substitution mapping $i_1, \ldots, i_k$ to $t_1, \ldots, t_k$ respectively. From now on we assume that for every schema S no variable can be simultaneously free and bound in S, and if $\bigoplus_{i|C_i} S_i$ and $\bigotimes_{j|D_j} S_j$ (where $\bigoplus, \bigotimes \in \{\bigwedge, \bigvee\}$) are two distinct iterations occurring in S then i and j are distinct. Such a schema is said *rectified*. Let a rectified schema S contain iterations $\bigoplus_{i_1|C_1} S_1, \ldots, \bigoplus_{i_p|C_p} S_p$, then $C_1 \wedge \cdots \wedge C_p$ is called the *constraint context* of S, written $\text{Context}(S)$.

2.2 Semantics

Definition 4 (Semantics). *An* interpretation of the schemata language *is a function mapping every propositional variable to a truth value* $\mathbf{T}$ *or* $\mathbf{F}$ *and every integer variable to an integer. Then the* semantic $[\![S]\!]_{\mathcal{I}}$ *of a propositional schema in an interpretation* $\mathcal{I}$ *is inductively defined as:*

- $[\![P_{t_1,\ldots,t_k}]\!]_{\mathcal{I}} = \mathcal{I}(P_{\mathcal{I}(t_1),\ldots,\mathcal{I}(t_k)})$ *where the interpretation of arithmetic expressions is defined as usual.*
- $[\![\neg\Phi]\!]_{\mathcal{I}} = \mathbf{T}$ *iff* $[\![\Phi]\!]_{\mathcal{I}} = \mathbf{F}$.
- $[\![\Phi \vee \Phi']\!]_{\mathcal{I}} = \mathbf{T}$ *iff* $[\![\Phi]\!]_{\mathcal{I}} = \mathbf{T}$ *or* $[\![\Phi']\!]_{\mathcal{I}} = \mathbf{T}$.
- $[\![\Phi \wedge \Phi']\!]_{\mathcal{I}} = \mathbf{T}$ *iff* $[\![\Phi]\!]_{\mathcal{I}} = \mathbf{T}$ *and* $[\![\Phi']\!]_{\mathcal{I}} = \mathbf{T}$.
- $[\![\bigvee_{i|C_i} S]\!]_{\mathcal{I}} = \mathbf{T}$ *iff there is a* $k \in \mathbb{Z}$ *s.t.* $\mathcal{I}(C_i[k/i]) = \mathbf{T}$ *(usual first-order interpretation) and* $[\![S]\!]_{\mathcal{J}} = \mathbf{T}$ *where* $\mathcal{J}$ *is s.t.* $\mathcal{J}(i) = k$ *and* $\mathcal{J}(j) = \mathcal{I}(j)$ *for* $j \neq i$.
- $[\![\bigwedge_{i|C_i} S]\!]_{\mathcal{I}} = \mathbf{T}$ *iff for every integer* k *s.t.* $\mathcal{I}(C_i[k/i]) = \mathbf{T}$*:* $[\![S]\!]_{\mathcal{J}} = \mathbf{T}$ *where* $\mathcal{J}$ *is defined the same way as for* $\bigvee$.

A schema S *is* satisfiable *iff there is an interpretation* $\mathcal{I}$ *s.t.* $[\![S]\!]_{\mathcal{I}} = \mathbf{T}$*. Then* $\mathcal{I}$ *is called a* model *of* S.

It is trivially semi-decidable to determine if a schema is satisfiable:

Proposition 1. *The set of satisfiable schemata is recursively enumerable.*

Proof. The set of instances of a schema is recursively enumerable and it is decidable whether each instance is (propositionally) satisfiable. See [5] for details. □

However this is not more than semi-decidable as the following result shows:

Theorem 1. *The set of satisfiable schemata is not recursive.*

Proof. By reduction to Post's correspondence problem. See [5] for details. □

Hence schemata calculi cannot be *refutationally complete*.

3 A Proof Procedure: DPLL*

We provide now a set of (sound) deduction rules (in the spirit of the Davis-Putnam-Logemann-Loveland procedure for propositional logic) that ensure completeness w.r.t. satisfiability (we know from Theorem 1 we cannot ensure refutational completeness). Compared to the naive procedure sketched in the proof of Proposition 1, DPLL* is much more efficient and terminates more often.

Definition 5 (Tableau). *A* tableau *is a tree $\mathcal{T}$ s.t. each node α occurring in $\mathcal{T}$ is labeled by a triple $(S_{\mathcal{T}}(\alpha), \mathcal{L}_{\mathcal{T}}(\alpha), \mathcal{C}_{\mathcal{T}}(\alpha))$ containing respectively a schema, a set of literals, and a linear constraint involving only parameters.*

As usual a tableau is generated from another tableau by applying some extension rules. Let $r : \dfrac{(S, \mathcal{L}, \mathcal{C})}{C_1 \mid C_2}$ be a rule. Let α be a leaf of a tree $\mathcal{T}$. If the triple labelling it matches $(S, \mathcal{L}, \mathcal{C})$ then we can *extend* the tableau by adding to α two children labeled with $C_1\sigma$ and $C_2\sigma$ where σ is the matching substitution. A leaf is *closed* iff its schema is $\bot$ or its parameter constraints are unsatisfiable (this can be detected using decision procedures for arithmetic without multiplication [7]). Let α be a node of a tableau $\mathcal{T}$, we write $\mathrm{Context}(\alpha)$, called the *constraint context* of α, the constraint $\mathcal{C}_{\mathcal{T}}(\alpha) \wedge \mathrm{Context}(S_{\mathcal{T}}(\alpha))$. We provide the formal definition of the rules and then explain their underlying intuition.

Definition 6 (Extension rules). *α denotes any leaf on which an extension rule is intended to be applied. The* extension rules *are:*

- *Splitting rules:*
  - Propositional splitting.

$$\frac{(S, \mathcal{L}, \mathcal{C})}{(S, \mathcal{L} \cup P_{e_1,\dots,e_k}, \mathcal{C}) \;\Big|\; (S, \mathcal{L} \cup \neg P_{e_1,\dots,e_k}, \mathcal{C})}$$

If the only variables of $e_1, \dots, e_k$ are parameters of S and:

$$\exists\, \mathcal{BV}(S) \cdot \mathrm{Context}(\alpha) \wedge \bigvee_{\substack{P_{f_1,\dots,f_k} \\ \text{occurs in } S}} (e_1 = f_1 \wedge \cdots \wedge e_k = f_k)$$

(i.e. $P_{e_1,\dots,e_k}$ occurs in S in every interpretation) where $\mathcal{BV}(S)$ is the set of bound variables of S, ordered to form a tuple, and:

$$\mathcal{C} \Rightarrow \bigwedge_{\substack{P_{f_1,\dots,f_k} \in \mathcal{L} \\ \text{or } \neg P_{f_1,\dots,f_k} \in \mathcal{L}}} e_1 \neq f_1 \vee \cdots \vee e_k \neq f_k$$

(i.e. $P_{e_1,\dots,e_k}$ does not occur in $\mathcal{L}$ in any interpretation) are valid.

  - Constraint splitting. *If the only free variables of C are i and parameters of S, and $\mathcal{C} \wedge \forall i \cdot \neg C$ is satisfiable:*

$$\frac{(S[\bigoplus_{i|C} S'], \mathcal{L}, \mathcal{C})}{(S[\bigoplus_{i|C} S'], \mathcal{L}, \mathcal{C} \wedge \exists i \cdot C) \;\Big|\; (S[e], \mathcal{L}, \mathcal{C} \wedge \forall i \cdot \neg C)}$$

where $(e, \bigoplus) \in \{(\top, \bigwedge), (\bot, \bigvee)\}$,

- Splitting into intervals. *If the only free variables of C are i and parameters of S:*

$$\frac{(S[\bigoplus_{i|C \wedge k \times i \lhd t_1 \wedge l \times i \lhd t_2} S'], \mathcal{L}, \mathcal{C})}{\begin{array}{r|r}(S[\bigoplus_{i|C \wedge k \times i \lhd t_1} S'], \mathcal{L}, & (S[\bigoplus_{i|C \wedge l \times i \lhd t_2} S'], \mathcal{L}, \\ \mathcal{C} \wedge l \times t_1 \lhd k \times t_2) & \mathcal{C} \wedge l \times t_1 \not\lhd k \times t_2)\end{array}}$$

where $l, k \in \mathbb{N}$, $\bigoplus \in \{\bigwedge, \bigvee\}$, $\lhd \in \{<, \leq, \geq, >\}$.

– *Rewriting:*

$$\frac{(S, \mathcal{L}, \mathcal{C})}{(S', \mathcal{L}, \mathcal{C})}$$

where $S \to S'$ *by the following rewrite system:*

- Unfolding. *If* $\text{Context}(\alpha) \Rightarrow C[I/i]$ *is valid:*

$$\bigoplus_{i|C} T \to T[I/i] \oplus \bigoplus_{i|C \wedge i \neq I} T \quad \textit{where } (\oplus, \bigoplus) \in \left\{(\wedge, \bigwedge), (\vee, \bigvee)\right\}$$

- Emptiness detection. *If* C' *does not involve any variable distinct from* i *and bound in* S *and if* $\text{Context}(\alpha) \wedge \forall i' \cdot \neg C'$ *is satisfiable:*

$$\bigoplus_{i|C} T[\bigotimes_{i'|C'} T'] \to \bigoplus_{i|C \wedge \exists i' \cdot C'} T[\bigotimes_{i'|C'} T'] \oplus \bigoplus_{i|C \wedge \forall i' \cdot \neg C'} T[e]$$

where $(\oplus, \bigoplus) \in \{(\wedge; \bigwedge), (\vee; \bigvee)\}$ *and* $(\bigotimes, e) \in \{(\bigwedge; \top), (\bigvee; \bot)\}$.

- Evaluation. *If* $P_{t_1,\dots,t_k} \in \mathcal{L}$ *(resp. or* $\neg P_{t_1,\dots,t_k} \in \mathcal{L}$*), in which case* $\bigoplus = \bigwedge$ *(resp.* $\bigoplus = \bigvee$*), and if* $\text{Context}(\alpha) \wedge s_1 = t_1 \wedge \cdots \wedge s_k = t_k$ *is satisfiable:*

$$P_{s_1,\dots,s_k} \to \bigoplus_{i|(s_1 \neq t_1 \vee \cdots \vee s_k \neq t_k) \wedge i = 0} P_{s_1,\dots,s_k}$$

where i is a fresh variable. It is not used but we assign it 0 to satisfy the condition that it has to be in an interval in Definition 3.

- Algebraic simplification.

$$\neg\top \to \bot \quad T \wedge \top \to T \quad T \wedge \bot \to \bot \quad \bigwedge_{i|C_i} \top \to \top \quad T \wedge T \to T$$

$$\neg\bot \to \top \quad T \vee \top \to \top \quad T \vee \bot \to T \quad \bigvee_{i|C_i} \bot \to \bot \quad T \vee T \to T$$

if $\text{Context}(\alpha) \Rightarrow \exists i \cdot C_i$ *is valid:* $\quad \bigwedge_{i|C_i} \bot \to \bot \quad \bigvee_{i|C_i} \top \to \top$

if $\text{Context}(\alpha) \wedge \exists i \cdot C_i$ *is unsatisfiable:* $\quad \bigwedge_{i|C_i} T \to \top \quad \bigvee_{i|C_i} T \to \bot$

if $\text{Context}(\alpha) \Rightarrow \exists i \cdot C_i$ *is valid and T does not contain i:*

$$\bigwedge_{i|C_i} T \to T \quad \bigvee_{i|C_i} T \to T$$

Intuition of the Rules. DPLL* browses the possible interpretations depending on the shape of the current schema. As interpretations assign a truth value to each atom and a number to each parameter, this browsing is achieved through *Propositional splitting* and *Constraint splitting* (indeed a constraint implicitly defines a set of numbers possibly assigned to parameters).

Schemata represent infinite sets of structurally similar propositional formulae. This similarity enables us to treat simultaneously all those formulae. We decide to interpret an atom as soon as we know that it occurs in all those formulae, which is specifically the case when this atom occurs explicitly in the schema (by "explicitly", we mean e.g. that P_1 occurs explicitly in $P_1 \wedge \bigwedge_{i=2}^{n} P_i$ whereas it occurs implicitly in $\bigwedge_{i=1}^{n} P_i$). But this condition is not sufficient: an atom may occur explicitly in an empty iteration, in which case it does not actually occur; hence the first side condition in *Propositional splitting* to ensure that this is not the case. This is the role of *Unfolding* to extract a particular rank of an iteration in order to make appear some atom explicitly. This rule only applies if we know that the rank indeed exists for *all instances* of the schema. (e.g. if we intend to make P_1 occur explicitly in $\bigwedge_{i=1}^{n} P_i$, *Unfolding* will apply only if $n \geq 1$). Of course most of the time we cannot ensure such a knowledge: it depends on the fact that the iteration is empty or not. *Constraint splitting* and *Emptiness detection* are there to make the branching accordingly to those two cases (in the previous example *Constraint splitting* will apply and branch on two cases: $n < 1$ and $n \geq 1$). This is generally not sufficient as an iteration may be known empty without us knowing *which rank* makes it non-empty e.g. if the constraint of the variable is $k < i < l \wedge p < i < q$ then how can we know if it is the rank k or p which indeed exists (k, l, p, q are supposed unrelated) ? In such cases, the *Splitting into intervals* rule adds some constraints on the involved expressions so that this knowledge is ensured (in the example, it will branch on the cases $l < q$ and $l \geq q$, and then on $k < p$ and $k \geq p$).

Once an atom (resp. its negation) occurs explicitly and is selected for insertion in $\mathcal{L}$, we can substitute it with $\top$ (resp. $\bot$). However though the atom has an explicit occurrence it may also have implicit occurrences (e.g. P_1 in $\bigwedge_{i=1}^{n} P_i$) in which case we cannot just substitute $\top$ to it. *Evaluation* deals with this by wrapping with an iteration the indexed propositions for which this atom may be an instance (P_i in our example). The only aim of this wrapping is to ensure that the indexed proposition differs from the atom (still in our example, the schema is turned into $\bigwedge_{i=1}^{n} \bigwedge_{j|i\neq 1 \wedge j=0} P_i$). *Emptiness detection* and *Constraint splitting* then distinguish the case where they indeed differ from the case where they equal (in the example *Emptiness detection* applies: $\bigwedge_{i|1\leq i\leq n\wedge \exists j\cdot(i\neq 1\wedge j=0)} \bigwedge_{j|i\neq 1\wedge j=0} P_i \wedge \bigwedge_{i|1\leq i\leq n\wedge \forall j\cdot(i=1\vee j\neq 0)} \top$). *Algebraic simplification* finally makes the simplifications allowed by this case distinction (which gives in the example: $\bigwedge_{i|1\leq i\leq n\wedge \exists j\cdot(i\neq 1\wedge j=0)} \bigwedge_{j|i\neq 1\wedge j=0} P_i \wedge \top$, and then: $\bigwedge_{i|1\leq i\leq n\wedge \exists j\cdot(i\neq 1\wedge j=0)} \bigwedge_{j|i\neq 1\wedge j=0} P_i$). Though this choice of rules may seem intricate, it is a simple and powerful way of propagating constraints about nested iterations along the schema. The resulting schemata may be simplified to allow reader-friendly presentation (e.g. the last schema may be simplified as $\bigwedge_{i=2}^{n} P_i$).

DPLL. Eluding the work on iterations, we see that DPLL* simulates DPLL in the case where the schema is a propositional formula. The selection of an atom (*Propositional splitting*), the traversal of the formula to check for its occurrences (*Evaluation*), the substitution with its value when we indeed fall on an occurrence (*Constraint splitting*,*Emptiness detection*), and the subsequent simplifications (*Algebraic simplification*) are a faithful description of the DPLL procedure.

4 Soundness and Completeness

Definition 7 (Tableau semantics). *A node α of a tableau $\mathcal{T}$ is satisfied in an interpretation $\mathcal{I}$, written $\mathcal{I} \models \alpha$, iff $\mathcal{I} \models S_{\mathcal{T}}(\alpha)$ (schemata semantics), $\mathcal{I} \models \mathcal{C}_{\mathcal{T}}(\alpha)$ (linear constraints, i.e. first-order, semantics), and for all $L \in \mathcal{L}_{\mathcal{T}}(\alpha)$, $\mathcal{I}(L) = \mathbf{T}$. A tableau $\mathcal{T}$ is satisfied in $\mathcal{I}$ iff it contains a leaf α s.t. $\mathcal{I} \models \alpha$.*

Lemma 1. *If $\mathcal{T}'$ is a tableau obtained by applying a rule on a leaf α of a tableau $\mathcal{T}$ then $\mathcal{I} \models \alpha$ iff there exists a leaf β of $\mathcal{T}'$ s.t. β is a child of α in $\mathcal{T}'$ and $\mathcal{I} \models \beta$.*

Proof. (Sketch) By inspection of the extension rules. □

Lemma 2. *If a leaf α in $\mathcal{T}$ is irreducible and not closed then $\mathcal{T}$ is satisfiable.*

Proof. By irreducibility of *Constraint splitting* and *Emptiness detection*, all iterations occurring in $S_{\mathcal{T}}(\alpha)$ are non-empty. For a given root iteration S of constraint C we know by irreducibility of *Splitting into intervals* and elimination of quantifiers in linear arithmetic that we can restrict C to a disjunction of inequalities $t_1 \leq i \leq t_2$. Once this done, for any such t_1, $C[t_1/i]$ is valid and thus *Unfolding* applies which is not possible by irreducibility. Hence there is no root iteration, and thus no iteration at all. So $S_{\mathcal{T}}(\alpha)$ is constructed only on $\wedge, \vee, \neg$ and atoms. If an atom A occurs then it is in $\mathcal{L}_{\mathcal{T}}(\alpha)$ by irreducibility of *Propositional splitting*. So A occurs in a schema of the form $\bigoplus_{i|C} A$ by *Evaluation* which is impossible as there is no iteration. Hence no atom occurs in $S_{\mathcal{T}}(\alpha)$. Then $S_{\mathcal{T}}(\alpha)$ is either $\bot$, impossible as the branch is not closed, or $\top$. $\top$ being satisfiable, we conclude with Lemma 1 that the initial tableau is satisfiable. □

Theorem 2 (Soundness). *Let $\mathcal{T}$ be a tableau. If a tableau $\mathcal{T}'$ is obtained from $\mathcal{T}$ by application of the extension rules, and if $\mathcal{T}'$ contains an irreducible and not closed leaf then $\mathcal{T}$ is satisfiable.*

Proof. This follows immediately from Lemmata 1 and 2. □

We prove DPLL* is complete w.r.t. satisfiability. Let $\mathcal{I}$ be an interpretation, $\mathcal{T}$ a tableau and α a leaf in $\mathcal{T}$ labelled by $(S, \mathcal{L}, \mathcal{C})$. We set the following measures:

1. $m^1_{\mathcal{I}}(S)$ is defined by induction on the structure of S:
   - $m^1_{\mathcal{I}}(P) \stackrel{\text{def}}{=} 1$ if P is an indexed proposition.
   - $m^1_{\mathcal{I}}(\neg S) \stackrel{\text{def}}{=} m^1_{\mathcal{I}}(S) + 1$.
   - $m^1_{\mathcal{I}}(S_1 \lhd S_2) \stackrel{\text{def}}{=} m^1_{\mathcal{I}}(S_1) + m^1_{\mathcal{I}}(S_2)$ if $\lhd \in \{\vee, \wedge\}$.

  - $m^1_{\mathcal{I}}(\bigoplus_{i|C_i} S) \stackrel{\text{def}}{=} (\#E)^2 \times (\Sigma_{j\in E} m^1_{\mathcal{J}_j}(S)$ where $\bigoplus \in \{\bigwedge, \bigvee\}$, $E = \{i \mid \mathcal{I} \models C_i\}$, $\#E$ is the size of E and $\mathcal{J}_j$ is an interpretation defined exactly as $\mathcal{I}$, except that $[\![i]\!]_{\mathcal{J}_j} \stackrel{\text{def}}{=} j$.

2. $m^2_{\mathcal{I}}(\alpha) \stackrel{\text{def}}{=} \#(\mathcal{A}toms(F) \setminus \mathcal{A})$ where F is the propositional formula obtained by substituting all the parameters of S by their corresponding value under $\mathcal{I}$, $\mathcal{A}toms(F)$ is the set of atoms that occur in F, and $\mathcal{A}$ is the set of atoms that occur in $\mathcal{L}$.
3. $m^3(\alpha)$ is the number of pairs $(P_{s_1,\dots,s_k}, P_{t_1,\dots,t_k})$ s.t. $P_{s_1,\dots,s_k}$ occurs in S, $P_{t_1,\dots,t_k}$ or $\neg P_{t_1,\dots,t_k}$ occurs in $\mathcal{L}$, and $\text{Context}(\alpha) \wedge \bigwedge_{i=1}^n t_i = s_i$ is satisfiable.
4. $m^4(\alpha)$ is the number of iterations $\bigoplus_{i|C} S'$ of $S_{\mathcal{T}}(\alpha)$ s.t. $\text{Context}(\alpha) \wedge \forall i \cdot \neg C$ is satisfiable.
5. $m^5(S)$ is the number of iterations $\bigoplus_{i|C \wedge i \lhd t_1 \wedge i \lhd t_2} S'$ of S ($\lhd \in \{<, \leq, \geq, >\}$).

$m_{\mathcal{I}}(\alpha, \mathcal{T})$ is defined as $(m^1_{\mathcal{I}}(S), m^2_{\mathcal{I}}(\alpha), m^3(\alpha), m^4(\alpha), m^5(S))$ ordered using the lexicographic extension of the usual ordering on natural numbers.

Lemma 3. *Let $\mathcal{I}$ be an interpretation. Let $\mathcal{T}$ be a tableau. If $\mathcal{T}'$ is deduced from $\mathcal{T}$ by applying an extension rule on a leaf α s.t. $\mathcal{I} \models \alpha$, then for every child β of α in $\mathcal{T}'$ s.t. $\mathcal{I} \models \beta$, we have $m_{\mathcal{I}}(\beta, \mathcal{T}') < m_{\mathcal{I}}(\alpha, \mathcal{T})$.*

Proof. By inspection of the extension rules:

Rule	$m^1_{\mathcal{I}}(S)$	$m^2_{\mathcal{I}}(\alpha)$	$m^3(\alpha)$	$m^4(\alpha)$	$m^5(S)$
Propositional splitting	$\leq$	$<$			
Constraint splitting	$\leq$	$\leq$	$\leq$	$<$	
Splitting into intervals	$\leq$	$\leq$	$\leq$	$\leq$	$<$
Unfolding	$<$				
Emptiness detection	$\leq$	$\leq$	$\leq$	$<$	
Evaluation	$\leq$	$\leq$	$<$		
Algebraic simplification	$<$				

$\leq$ (resp. $<$) means that the corresponding measure *does not increase* (resp. *strictly decreases*) by application of the rule. □

A *derivation* is a (possibly infinite) sequence of tableaux $(\mathcal{T}_n)_{n\in I}$ s.t. I is either $[0..n]$ or $\mathbb{N}$ and s.t. for all $i \in I\setminus\{0\}$, $\mathcal{T}_i$ is obtained from $\mathcal{T}_{i-1}$ by applying one of the extension rules. A derivation is *fair* if either there is $i \in I$ s.t. $\mathcal{T}_i$ contains an irreducible not closed leaf or if for every $i \in I$ and α a not closed leaf in $\mathcal{T}_i$ there is $j \geq i$ s.t. a rule applies on α in $\mathcal{T}_j$ (i.e. no leaf can be indefinitely freezed).

Theorem 3 (Model Completeness). *Let $\mathcal{T}_0$ be a satisfiable tableau and let $\mathcal{I}$ be a model of $\mathcal{T}_0$. If $(\mathcal{T}_n)_{n\in I}$ is a fair derivation then there are $k \in I$ and a leaf α_k in $\mathcal{T}_k$ s.t. α_k is irreducible and not closed.*

Proof. By Lemma 1, for all $i \in I$, $\mathcal{T}_i$ contains a leaf α_i s.t. $\mathcal{I} \models \alpha_i$. Let $k \in I$ s.t. $m_{\mathcal{I}}(\alpha_k, \mathcal{T}_k)$ is minimal (k exists since $m_{\mathcal{I}}(\alpha_i, \mathcal{T}_i)$ defines a well-founded order). Assume a rule is applied on α_k in the derivation, on some tableau $\mathcal{T}_l$. By Lemma 1 there is a child β of α_k s.t. $\mathcal{I} \models \beta$. By Lemma 3 we have $m_{\mathcal{I}}(\beta, \mathcal{T}_l) < m_{\mathcal{I}}(\alpha_k, \mathcal{T}_k)$ which is impossible. Thus no rule is applied on α_k. Since the derivation is fair, α_k is irreducible (or there is another leaf that is irreducible). □

5 Termination

DPLL* does not terminate in general (Theorem 1). The reason is that the truth value of a schema depends on a potentially infinite number of atoms. But it is often the case that when simplifying a schema according to the value given to an atom, the newly obtained schema has already been seen (up to some transformation) i.e. the procedure is *looping*. This description of what happens corresponds to a procedural view of mathematical induction which naturally arises when doing the proof manually. The cycle detection rule ("looping") is basically equivalent to the corresponding one in STAB, it is included here for the sake of completeness.

We start by giving a quite general definition of looping, that we know from Theorem 1 that it has no chance to be decidable. *We assume all parameters are interpreted as positive integers.* This can be specified by rewriting a schema S into $\bigvee_{i|n_1 \geq 0 \wedge \dots \wedge n_k \geq 0 \wedge i=0} S$ where $n_1, \dots, n_k$ are the parameters of S.

Definition 8 (Looping). *Let α, β be two nodes of a tableau $\mathcal{T}$, and $n_1, \dots, n_k$ be the parameters of $S_{\mathcal{T}}(\alpha)$. Then β* loops *on α if there are $p_1, \dots, p_k \in \mathbb{N}$ s.t. one at least is positive and for every model $\mathcal{I}$ of β, $\mathcal{I} \models S_{\mathcal{T}}(\alpha)\{n_1 - p_1/n_1, \dots, n_k - p_k/n_k\}$, $\mathcal{I} \models \mathcal{C}_{\mathcal{T}}(\alpha)\{n_1 - p_1/n_1, \dots, n_k - p_k/n_k\}$, and for all $L \in \mathcal{L}_{\mathcal{T}}(\alpha)$, $\mathcal{I}(L\{n_1 - p_1/n_1, \dots, n_k - p_k/n_k\}) = \mathbf{T}$.*

When a leaf loops, it is treated as a closed branch (though it is not necessarily unsatisfiable), we say that it is *blocked*.

Theorem 2 trivially remains true but the proof of Theorem 3 must be adapted:

Theorem 4 (Model Completeness). *Let $\mathcal{T}_0$ be a satisfiable tableau and $\mathcal{I}$ be a model of $\mathcal{T}_0$. If $(\mathcal{T}_n)_{n \in I}$ is a fair derivation then there exist $k \in I$ and a leaf α_k in $\mathcal{T}_k$ s.t. α_k is irreducible and neither closed nor blocked.*

Proof. The proof is basically the same as in [5]. □

In order to apply the looping rule in practice one has to compute the natural numbers $p_1, \dots, p_k$ and check that the implication holds. This problem is obviously undecidable. Thus we define a stronger relation between two nodes which is decidable and allows to check that $p_1, \dots, p_k$ exist. The underlying idea is the following: let $S = \bigvee_{i=a}^{b} S_i$ and $S' = \bigvee_{j=c}^{d} S_j$. To check that $S \Rightarrow S'$, it is sufficient that for all $i \in [a..b]$ there exists $j \in [c..d]$ s.t. $S_i \Rightarrow S_j$. If S_i, S_j are indexed propositions then $S_i \Rightarrow S_j$ holds if $i = j$ (thus the above condition is equivalent to $[a..b] \subseteq [c..d]$). Formally we inductively construct a linear arithmetic formula from the structure of both schemata for which we want to check the looping. This relation can be seen as a form of subsumption between schemata.

Proposition 2. *To check that a leaf β loops on a node α in a tableau $\mathcal{T}$, it is sufficient to check that the arithmetic formula $\exists p_1, \dots, p_k \cdot (p_1 > 0 \vee \dots \vee p_k > 0) \wedge \forall n_1, \dots, n_k \cdot F_{S_{\mathcal{T}}(\beta) \Rightarrow S_{\mathcal{T}}(\alpha)\{n_1 - p_1/n_1, \dots, n_k - p_k/n_k\}} \wedge \mathcal{C}_{\mathcal{T}}(\beta) \Rightarrow \mathcal{C}_{\mathcal{T}}(\alpha)\{n_1 - p_1/n_1, \dots, n_k - p_k/n_k\}$ is valid, where $F_{S' \Rightarrow S}$ is inductively defined as follows:*

- *$F_{S' \Rightarrow S} \overset{def}{=} S' \Rightarrow S$ if S', S are constraints.*

– $F_{S'_1 \vee S'_2 \Rightarrow S} \overset{def}{=} F_{S' \Rightarrow S_1 \wedge S_2} \overset{def}{=} F_{S'_1 \Rightarrow S} \wedge F_{S'_2 \Rightarrow S}$
– $F_{S'_1 \wedge S'_2 \Rightarrow S_1} \overset{def}{=} F_{S' \Rightarrow S_1 \vee S_2} \overset{def}{=} F_{S'_1 \Rightarrow S'} \vee F_{S'_2 \Rightarrow S}$
– $F_{\bigvee_{i|C} S' \Rightarrow S} \overset{def}{=} F_{S' \Rightarrow \bigwedge_{i|C} S} \overset{def}{=} \forall i \cdot\ C \Rightarrow F_{S' \Rightarrow S}$
– $F_{\bigwedge_{i|C} S' \Rightarrow S} \overset{def}{=} F_{S' \Rightarrow \bigvee_{i|C} S} \overset{def}{=} \exists i \cdot\ C \wedge F_{S' \Rightarrow S}$
– $F_{\neg S' \Rightarrow \neg S} \overset{def}{=} F_{S' \Rightarrow S}$
– $F_{P_{t_1,\ldots,t_n} \Rightarrow P_{s_1,\ldots,s_n}} \overset{def}{=} (t_1 = s_1 \wedge \ldots \wedge t_n = s_n)$
– $F_{S' \Rightarrow S} \overset{def}{=} \bot$ *otherwise.*

s.t. rules presented first have higher priority in case of ambiguity.

It follows from Proposition 3, proved by an easy induction on S, S':

Proposition 3. *Every model of both $F_{S \Rightarrow S'}$ and S is a model of S'.*

6 Examples

6.1 The n-bit Adder

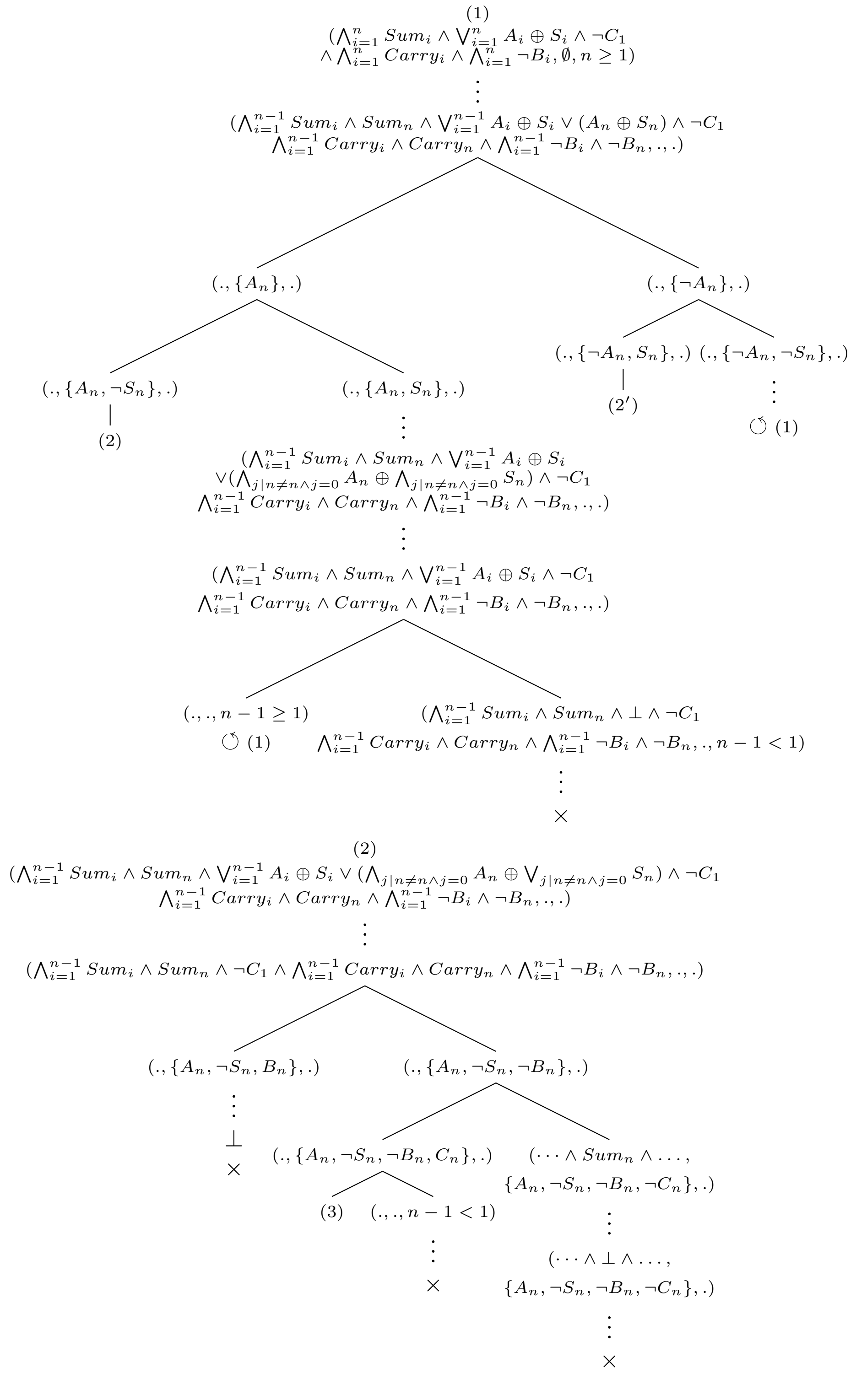
We express an n-bits adder circuit with propositional schemata. Such a circuit takes as input two numbers in bitwise representation: $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$. Then S_i is the i^{th} bit of their sum. We use a classical implementation composed of n 1-bit adders. Each adder generates a carry that is propagated to the next adder. We write C_{i+1} the carry generated by the i^{th} adder. C_1 is always false. We set the notations $Sum_i \overset{\text{def}}{=} S_i \Leftrightarrow (A_i \oplus B_i) \oplus C_i$ and $Carry_i \overset{\text{def}}{=} C_{i+1} \Leftrightarrow (A_i \wedge B_i) \vee (C_i \wedge A_i) \vee (C_i \wedge B_i)$ where $\oplus$ denotes the exclusive or. Then the schema $Adder \overset{\text{def}}{=} \bigwedge_{i=1}^{n} Sum_i \wedge \bigwedge_{i=1}^{n} Carry_i \wedge \neg C_1$ where $n \geq 1$ expresses this implementation of the adder circuit.

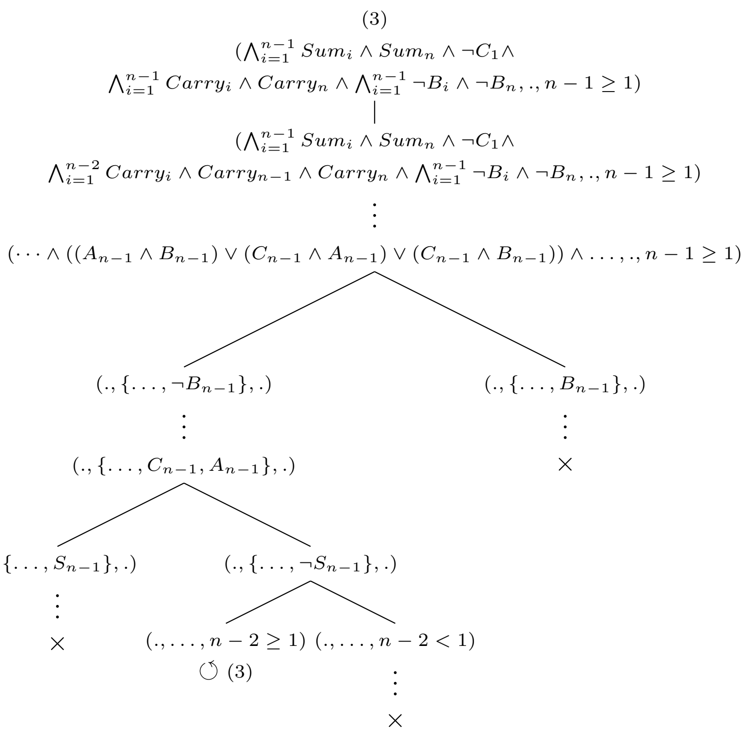
We want to prove that $A + 0 = A$. A SAT-solver can easily prove that this formula is unsatisfiable for a fixed n (say $n = 9$). We show how to prove it for all $n \in \mathbb{N}$. This simple example has been chosen for the sake of conciseness, commutativity or associativity of the n-bits adder could have been proven too.

We express the fact that the second operand is null: $\bigwedge_{i=1}^{n} \neg B_i$, and the fact that the result equals the first operand: $\bigwedge_{i=1}^{n} A_i \Leftrightarrow S_i$. Which gives by refutation: $\bigvee_{i=1}^{n} A_i \oplus S_i$. So we want to refute $Adder \wedge \bigwedge_{i=1}^{n} \neg B_i \wedge \bigvee_{i=1}^{n} A_i \oplus S_i$.

We sketch the closed tableau obtained by DPLL* to enable the comparison with STAB – which can handle this example, see [5]. It happens that the global structure of both proofs is the same. DPLL* decomposes the proof at a finer grain, thus the resulting proof is longer than for STAB.

We use the conventions that closed leaves are marked by $\times$, and $\circlearrowleft(\alpha)$ expresses the fact that the leaf loops on the node α. "." denotes "the same value as previously in this position". The following figure is only a sketch of the real tableau: several rules are often applied at once. Furthermore, to stick to DPLL*, $\Rightarrow$, $\Leftrightarrow$ and $\oplus$ have to be rewritten so as to use only $\wedge$, $\vee$ and $\neg$.


(1)
$(\bigwedge_{i=1}^{n} Sum_i \wedge \bigvee_{i=1}^{n} A_i \oplus S_i \wedge \neg C_1$
$\wedge \bigwedge_{i=1}^{n} Carry_i \wedge \bigwedge_{i=1}^{n} \neg B_i, \emptyset, n \geq 1)$
$(\bigwedge_{i=1}^{n-1} Sum_i \wedge Sum_n \wedge \bigvee_{i=1}^{n-1} A_i \oplus S_i \vee (A_n \oplus S_n) \wedge \neg C_1$
$\bigwedge_{i=1}^{n-1} Carry_i \wedge Carry_n \wedge \bigwedge_{i=1}^{n-1} \neg B_i \wedge \neg B_n, ., .)$
$(., \{A_n\}, .)$
$(., \{\neg A_n\}, .)$
$(., \{A_n, \neg S_n\}, .)$
(2)
$(., \{A_n, S_n\}, .)$
$(., \{\neg A_n, S_n\}, .)$
(2′)
$(., \{\neg A_n, \neg S_n\}, .)$
↻ (1)
$(\bigwedge_{i=1}^{n-1} Sum_i \wedge Sum_n \wedge \bigvee_{i=1}^{n-1} A_i \oplus S_i$
$\vee(\bigwedge_{j|n\neq n \wedge j=0} A_n \oplus \bigwedge_{j|n\neq n\wedge j=0} S_n) \wedge \neg C_1$
$\bigwedge_{i=1}^{n-1} Carry_i \wedge Carry_n \wedge \bigwedge_{i=1}^{n-1} \neg B_i \wedge \neg B_n, ., .)$
$(\bigwedge_{i=1}^{n-1} Sum_i \wedge Sum_n \wedge \bigvee_{i=1}^{n-1} A_i \oplus S_i \wedge \neg C_1$
$\bigwedge_{i=1}^{n-1} Carry_i \wedge Carry_n \wedge \bigwedge_{i=1}^{n-1} \neg B_i \wedge \neg B_n, ., .)$
$(., ., n-1 \geq 1)$
↻ (1)
$(\bigwedge_{i=1}^{n-1} Sum_i \wedge Sum_n \wedge \bot \wedge \neg C_1$
$\bigwedge_{i=1}^{n-1} Carry_i \wedge Carry_n \wedge \bigwedge_{i=1}^{n-1} \neg B_i \wedge \neg B_n, ., n-1 < 1)$
$\times$
(2)
$(\bigwedge_{i=1}^{n-1} Sum_i \wedge Sum_n \wedge \bigvee_{i=1}^{n-1} A_i \oplus S_i \vee (\bigwedge_{j|n\neq n\wedge j=0} A_n \oplus \bigvee_{j|n\neq n\wedge j=0} S_n) \wedge \neg C_1$
$\bigwedge_{i=1}^{n-1} Carry_i \wedge Carry_n \wedge \bigwedge_{i=1}^{n-1} \neg B_i \wedge \neg B_n, ., .)$
$(\bigwedge_{i=1}^{n-1} Sum_i \wedge Sum_n \wedge \neg C_1 \wedge \bigwedge_{i=1}^{n-1} Carry_i \wedge Carry_n \wedge \bigwedge_{i=1}^{n-1} \neg B_i \wedge \neg B_n, ., .)$
$(., \{A_n, \neg S_n, B_n\}, .)$
$\bot$
$\times$
$(., \{A_n, \neg S_n, \neg B_n\}, .)$
$(., \{A_n, \neg S_n, \neg B_n, C_n\}, .)$
(3)
$(., ., n-1 < 1)$
$\times$
$(\cdots \wedge Sum_n \wedge \ldots,$
$\{A_n, \neg S_n, \neg B_n, \neg C_n\}, .)$
$(\cdots \wedge \bot \wedge \ldots,$
$\{A_n, \neg S_n, \neg B_n, \neg C_n\}, .)$
$\times$


(3)
$(\bigwedge_{i=1}^{n-1} Sum_i \wedge Sum_n \wedge \neg C_1 \wedge \bigwedge_{i=1}^{n-1} Carry_i \wedge Carry_n \wedge \bigwedge_{i=1}^{n-1} \neg B_i \wedge \neg B_n, ., n-1 \geq 1)$
$(\bigwedge_{i=1}^{n-1} Sum_i \wedge Sum_n \wedge \neg C_1 \wedge \bigwedge_{i=1}^{n-2} Carry_i \wedge Carry_{n-1} \wedge Carry_n \wedge \bigwedge_{i=1}^{n-1} \neg B_i \wedge \neg B_n, ., n-1 \geq 1)$
$(\cdots \wedge ((A_{n-1} \wedge B_{n-1}) \vee (C_{n-1} \wedge A_{n-1}) \vee (C_{n-1} \wedge B_{n-1})) \wedge \ldots, ., n-1 \geq 1)$
$(., \{\ldots, \neg B_{n-1}\}, .)$
$(., \{\ldots, B_{n-1}\}, .)$
$\times$
$(., \{\ldots, C_{n-1}, A_{n-1}\}, .)$
$(., \{\ldots, S_{n-1}\}, .)$
$\times$
$(., \{\ldots, \neg S_{n-1}\}, .)$
$(., \ldots, n-2 \geq 1)$
$\circlearrowleft$ (3)
$(., \ldots, n-2 < 1)$
$\times$

And (2′) is very similar to (2).

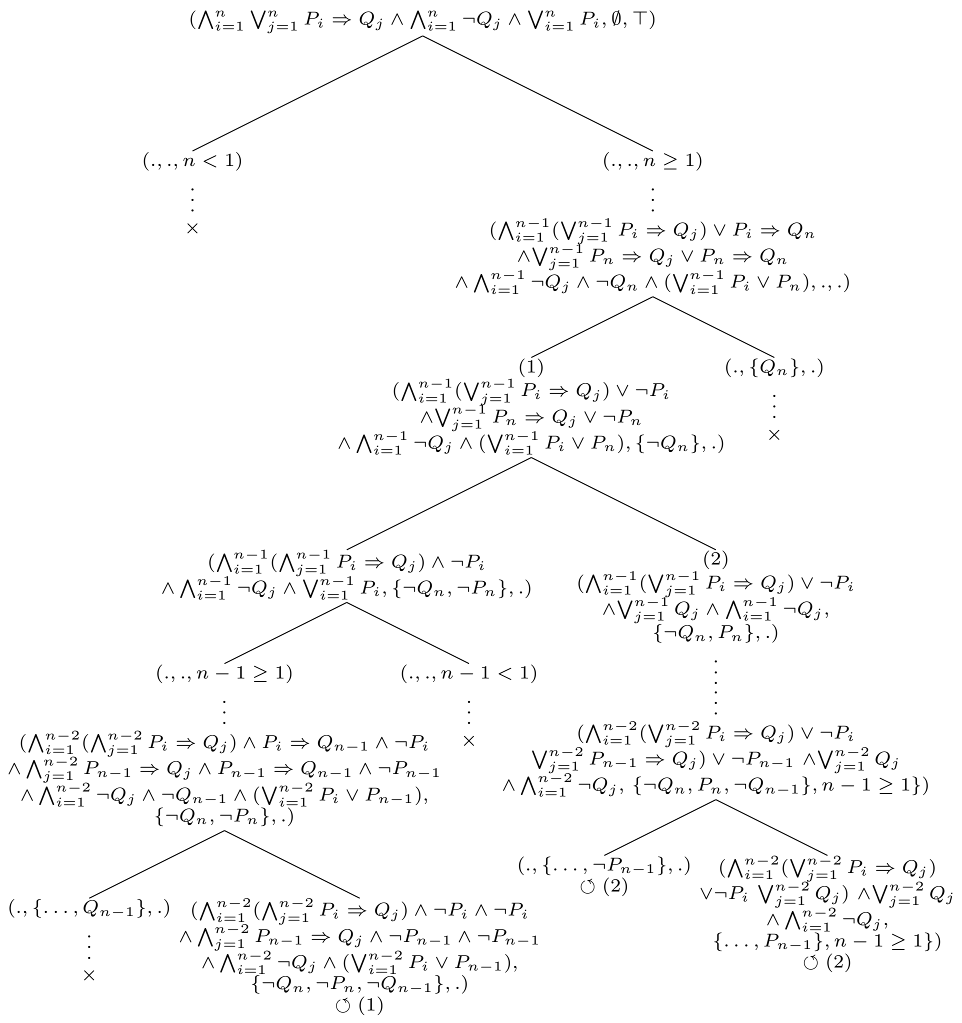
6.2 An example that terminates on DPLL* but not on STAB

The main reason for developing DPLL* was that STAB could not deal with nested iterations (e.g. $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} P_i \vee Q_j$). Indeed STAB only works at the root of formulae, in particular it can unfold only the outermost iteration but cannot modify any inner iteration. On the other side DPLL* is designed to select an atom and give it a value, *wherever this atom occurs in the schema* which allows to detect more cycles in the proof tree. We now give an example of a schema that can be proved with DPLL* but not with STAB:

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} P_i \Rightarrow Q_j \wedge \bigwedge_{i=1}^{n} \neg Q_i \wedge \bigvee_{i=1}^{n} P_i$$

This schema diverges under STAB (with a *restricted*, decidable, looping, similar to the one found in Proposition 2). The innermost iteration will never be unfolded by STAB thus its bounds will never decrease which is needed for a cycle to occur. And a cycle is indeed required as, for the schema to be refuted, we have to refute the case where some P_N is true, which is easily done, *and the case where* $\bigvee_{i \neq N} P_i$ *is true*, which can only be done through a cycle detection (as it is impossible for STAB to treat simultaneously all P_i's). We let the details to the reader.

We now sketch a refutation of this schema by DPLL*:


$(\bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} P_i \Rightarrow Q_j \wedge \bigwedge_{i=1}^{n} \neg Q_j \wedge \bigvee_{i=1}^{n} P_i, \emptyset, \top)$
$(.,.,n < 1)$
$\times$
$(.,.,n \geq 1)$
$(\bigwedge_{i=1}^{n-1}(\bigvee_{j=1}^{n-1} P_i \Rightarrow Q_j) \vee P_i \Rightarrow Q_n \wedge \bigvee_{j=1}^{n-1} P_n \Rightarrow Q_j \vee P_n \Rightarrow Q_n \wedge \bigwedge_{i=1}^{n-1} \neg Q_j \wedge \neg Q_n \wedge (\bigvee_{i=1}^{n-1} P_i \vee P_n), ., .)$
(1)
$(\bigwedge_{i=1}^{n-1}(\bigvee_{j=1}^{n-1} P_i \Rightarrow Q_j) \vee \neg P_i \wedge \bigvee_{j=1}^{n-1} P_n \Rightarrow Q_j \vee \neg P_n \wedge \bigwedge_{i=1}^{n-1} \neg Q_j \wedge (\bigvee_{i=1}^{n-1} P_i \vee P_n), \{\neg Q_n\}, .)$
$(., \{Q_n\}, .)$
$\times$
$(\bigwedge_{i=1}^{n-1}(\bigwedge_{j=1}^{n-1} P_i \Rightarrow Q_j) \wedge \neg P_i \wedge \bigwedge_{i=1}^{n-1} \neg Q_j \wedge \bigvee_{i=1}^{n-1} P_i, \{\neg Q_n, \neg P_n\}, .)$
(2)
$(\bigwedge_{i=1}^{n-1}(\bigvee_{j=1}^{n-1} P_i \Rightarrow Q_j) \vee \neg P_i \wedge \bigvee_{j=1}^{n-1} Q_j \wedge \bigwedge_{i=1}^{n-1} \neg Q_j, \{\neg Q_n, P_n\}, .)$
$(.,.,n-1 \geq 1)$
$(.,.,n-1 < 1)$
$\times$
$(\bigwedge_{i=1}^{n-2}(\bigwedge_{j=1}^{n-2} P_i \Rightarrow Q_j) \wedge P_i \Rightarrow Q_{n-1} \wedge \neg P_i \wedge \bigwedge_{j=1}^{n-2} P_{n-1} \Rightarrow Q_j \wedge P_{n-1} \Rightarrow Q_{n-1} \wedge \neg P_{n-1} \wedge \bigwedge_{i=1}^{n-2} \neg Q_j \wedge \neg Q_{n-1} \wedge (\bigvee_{i=1}^{n-2} P_i \vee P_{n-1}), \{\neg Q_n, \neg P_n\}, .)$
$(\bigwedge_{i=1}^{n-2}(\bigvee_{j=1}^{n-2} P_i \Rightarrow Q_j) \vee \neg P_i \; \bigvee_{j=1}^{n-2} P_{n-1} \Rightarrow Q_j) \vee \neg P_{n-1} \wedge \bigvee_{j=1}^{n-2} Q_j \wedge \bigwedge_{i=1}^{n-2} \neg Q_j, \{\neg Q_n, P_n, \neg Q_{n-1}\}, n-1 \geq 1\})$
$(., \{\ldots, Q_{n-1}\}, .)$
$\times$
$(\bigwedge_{i=1}^{n-2}(\bigwedge_{j=1}^{n-2} P_i \Rightarrow Q_j) \wedge \neg P_i \wedge \neg P_i \wedge \bigwedge_{j=1}^{n-2} P_{n-1} \Rightarrow Q_j \wedge \neg P_{n-1} \wedge \neg P_{n-1} \wedge \bigwedge_{i=1}^{n-2} \neg Q_j \wedge (\bigvee_{i=1}^{n-2} P_i \vee P_{n-1}), \{\neg Q_n, \neg P_n, \neg Q_{n-1}\}, .)$
$\circlearrowleft$ (1)
$(., \{\ldots, \neg P_{n-1}\}, .)$
$\circlearrowleft$ (2)
$(\bigwedge_{i=1}^{n-2}(\bigvee_{j=1}^{n-2} P_i \Rightarrow Q_j) \vee \neg P_i \; \bigvee_{j=1}^{n-2} Q_j) \wedge \bigvee_{j=1}^{n-2} Q_j \wedge \bigwedge_{i=1}^{n-2} \neg Q_j, \{\ldots, P_{n-1}\}, n-1 \geq 1\})$
$\circlearrowleft$ (2)

7 Conclusion

We have presented a new calculus, called DPLL*, for reasoning on schemata of propositional formulae and proved that it is sound and complete w.r.t. satisfiability. We extended this calculus to detect cycles in the proof tree, thus allowing refutation of non-trivial conjectures as shown in the examples. We have shown evidence that DPLL* allows to prove some schemata that were not provable with a previous calculus proposed by the authors, called STAB, based on shallow inference rules. In our opinion, this work is a first step opening several research ways:

- The main motivation to develop DPLL* is that STAB is not well-suited to handle nested iterations, a frequently needed feature. Section 6.2 gives ev-

idence of this. The next step is now to *precisely identify a syntactic class (containing imbrications of iterations) for which* DPLL* *terminates.*
– At the moment both those schemata calculi are just “calculi to handle schemata of propositional formulae” but one can wonder about “schemata of calculi to handle propositional formulae”. Formalising this intuition can shed some light on the relationship between the structure of propositional formulae and the structure of their proofs.
– A natural continuation of this work is to extend DPLL* to first-order logic. First-order schemata can introduce extensions of first-order logic with hopefully attractive features.
– This paradigm of “getting more abstraction while preserving structure” applies in many symbolic computation procedures, and particularly in type inference for functional languages. In this context, the so-called “generic programming” brings a gain of abstraction in types and functions by describing classes of functions *by induction on the type of their input* (see e.g. [8]). We believe that this has close connections with the ideas and techniques of the present paper and deserves to be studied.
– Similarly, in interactive theorem proving, the user often would like to give to the proof engine the information that the proof it is carrying out is analogous to some other proof. A *proof schema* seems adequate to do that. The languages and techniques presented here could be of interest for expressing theses similarities in the proof languages.

References

1. Chen, H., Hsiang, J., Kong, H.: On finite representations of infinite sequences of terms. In: Conditional and Typed Rewriting Systems, 2nd International Workshop, Springer, LNCS 516 (1990) 100–114
2. Peltier, N.: A General Method for Using Terms Schematizations in Automated Deduction. In: Proceedings of the International Joint Conference on Automated Reasoning (IJCAR'01), Springer LNCS 2083 (2001) 578–593
3. Hermann, M., Galbavý, R.: Unification of Infinite Sets of Terms schematized by Primal Grammars. Theoretical Computer Science **176**(1–2) (1997) 111–158
4. Hetzl, S., Leitsch, A., Weller, D., Woltzenlogel Paleo, B.: Proof analysis with HLK, CERES and ProofTool: Current status and future directions. In Sutcliffe G., Colton S., S.S., ed.: Workshop on Empirically Successful Automated Reasoning for Mathematics (ESARM). (July 2008) 21–41
5. Aravantinos, V., Caferra, R., Peltier, N.: A Schemata Calculus For Propositional Logic. In: Proceedings of the 18^{th} International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'09), Springer-Verlag (2009) 32–46
6. Davis, M., Logemann, G., Loveland, D.: A Machine Program for Theorem Proving. Communication of the ACM **5** (1962) 394–397
7. Cooper, D.: Theorem proving in arithmetic without multiplication. In Meltzer, B., Michie, D., eds.: Machine Intelligence 7. Edinburgh University Press (1972) 91–99
8. Hinze, R.: A new approach to generic functional programming. In: The 27th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press (2000) 119–132

Concurrency and Permutability in the Sequent Calculus

May 2009

Nicolas Guenot, LIX — École Polytechnique
`nguenot@lix.polytechnique.fr`

Abstract. In proof theory, the sequent calculus has been the most widely used logical formalism since its inception by Gentzen. However, it uses a syntax involving a lot of bureaucracy, thus detaining similar proofs from being written as a unique object. Indeed, different rule instances in a proof can be permuted, if the order of the corresponding inference steps does not matter. What we present here is a work in progress, aiming at a general framework for studying permutability properties of inference rule instances. In order to achieve this, we introduce the notion of *permutability graph*, which provides a unified syntax for equivalent proofs up to permutations. Eventually, it can be used to prove completeness for proof transformations, especially those based on permutations of inference rule instances such as *focusing*, as discussed at the end of this paper.

1 — Introduction

In the sequent calculus, proofs are trees of inference rule instances, structured by the order in which inference steps are used, and creating branches when two unrelated paths of the inference process need to be glued together. However, this order is often arbitrary, thus disallowing a clear reading of the proof by hiding the natural concurrency that exists between different inference steps. This is an inconsistent way of dealing with concurrency, since branches are explicitly separated, while some of the other rule instances within the same branch could be used in parallel without essentially changing the proof.

We are interested in the permutability properties of inference rule instances, and the dependencies between steps of the inference process that they induce, in any given logic — in particular, our analysis is valid for systems of usual logics, namely LK, LJ and LL. In order to work on permutability within proof objects, we introduce the notion of *permutability graph*, providing a unified presentation of most equivalent proofs *modulo* permutations. This is obviously related to proof-nets [Gir96], that allow for a canonical representation of proofs in linear logic, but the scope and goals of our analysis are quite different.

These graphs can be used to prove completeness for proof transformations based on rule instances permutations, since they represent an invariant of proofs through permutations. This was actually the original motivation for introducing these graphs, and all of this work should be considered as a continuation of the *focalization graphs* used in [MS07] to prove completeness for *focusing*. Indeed, after the permutability graph of a given proof has been built, any equivalent proof can be produced from it by a *sequentialisation* process, which can be controlled and create a focused proof. This is related to the sequentialisation theorem for proof-nets, and is close to the creation of tree-like strategies from ℓ-nets as presented in [CF05].

However, we try here to give a more general analysis, and this work, which is still in progress, aims at exploring with permutability the ideas of concurrency and sequentiality that can be found in proof objects. This is why this work is mainly based on the permutability properties of inference rule instances, and we will only illustrate the use of this approach in the end by discussing a proof of completeness for focusing in MLL.

2 — Permuting Inference Rules

In the sequent calculus, permuting two inference rule instances is very easy, in an informal way. However, the analysis of permutability presented here requires to be precise, so that we need some definitions for basic logical objects.

First, we assume that we are given a language of formulas, which is defined by binary connectives on a set of atoms $\{a, b, c, \cdots\}$. We work with occurrences of such formulas rather than directly with formulas, in order to distinguish between copies, and thus handle additive and exponential behaviours. An *occurrence* A_n of a formula A is simply this formula labeled with an integer. Moreover, there should be no two occurrences of the same formula with the same index within a sequent, and inference rules should handle duplication as illustrated below by the additive conjunction rule of linear logic :

$$\& \frac{\vdash A_5, B_3 \quad \vdash A_6, C_4}{\vdash A_1, (B_3 \& C_4)_2}$$

Then, we shall define the objects we are trying to permute : an *instance* χ of an inference rule scheme is derived from this rule scheme by instantiating variable formulas with actual occurrences. Given a rule instance χ, some of the occurrences in its conclusion are special, since they are the ones on which χ is applied, that could not be removed while keeping the instance correct with respect to its scheme — they are called the *principal occurrences*, and the set of such occurrences will be denoted by $\mathcal{P}(\chi)$ in the following.

Finally, we define an equivalence relation over occurrences, so that $A_n \approx A_m$ for any $n, m \in \mathbb{N}$. This can be immediately extended to sets of occurrences, and then to inference rule instances, as shown in the definition below.

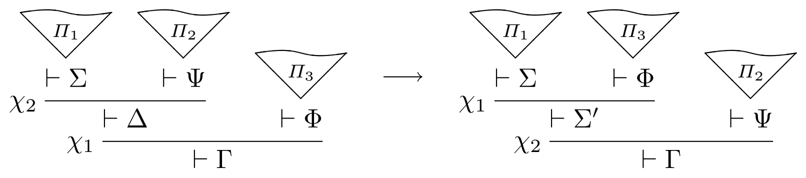
Definition 1 *Two instances χ_1 and χ_2 of the same inference rule scheme are said to be* twins, *which is denoted by $\chi_1 \approx \chi_2$, if and only if $\mathcal{P}(\chi_1) \approx \mathcal{P}(\chi_2)$.*

Then, the permutation of two rule instances can be defined by exhibition of the corresponding transformation on the rule instances at the bottom of the proof, which is trivially extended to instances located anywhere in the proof.

2.1 — Conservative Permutations

We start with a definition that expresses the most simple permutation case, where no instance is duplicated, keeping the set of rule instances used in the proof unchanged — in a proof Π, we will denote this set by $\mathcal{I}_\Pi$ in the following.

Definition 2 *A proof Π' is a* conservative permutation *of some proof Π if and only if Π' is derived from Π by any variant of the following transformation :*



Example 1 The two MLL proofs below are equivalent because the right one is built by permuting the $\parr$ rule instance below a $\otimes$ instance in the left one, using a conservative permutation. In MLL, this is the only possible kind of permutation.

$$\otimes\,\dfrac{\parr\,\dfrac{\otimes\,\dfrac{\mathsf{id}\,\dfrac{}{\vdash a, a^{\perp}} \qquad \mathsf{id}\,\dfrac{}{\vdash b, b^{\perp}}}{\vdash a, b, a^{\perp} \otimes b^{\perp}}}{\vdash a \parr b, a^{\perp} \otimes b^{\perp}} \qquad 1\,\dfrac{}{\vdash 1}}{\vdash a \parr b, (a^{\perp} \otimes b^{\perp}) \otimes 1} \quad \equiv \quad \parr\,\dfrac{\otimes\,\dfrac{\otimes\,\dfrac{\mathsf{id}\,\dfrac{}{\vdash a, a^{\perp}} \qquad \mathsf{id}\,\dfrac{}{\vdash b, b^{\perp}}}{\vdash a, b, a^{\perp} \otimes b^{\perp}} \qquad 1\,\dfrac{}{\vdash 1}}{\vdash a, b, (a^{\perp} \otimes b^{\perp}) \otimes 1}}{\vdash a \parr b, (a^{\perp} \otimes b^{\perp}) \otimes 1}$$

The basic step of conservative permutations can be seen as the permutation of an inference rule instance χ_2 under another rule instance χ_1. Then, the existence of cases where the conservative permutation of two rule instances fails induces a dependency relation between all instances within a given proof. This relation, defined below, will be the starting point of our analysis of the permutability properties of rule instances in proof objects.

Definition 3 *Given a proof Π, and $\chi_1, \chi_2 \in \mathcal{I}_{\Pi}$, the* conservative dependency *relation $\sqsubset_{\Pi}$ is defined so that $\chi_1 \sqsubset_{\Pi} \chi_2$ if and only if, for any conservative permutation Π' of Π, χ_2 is located above χ_1 in the proof tree Π'.*

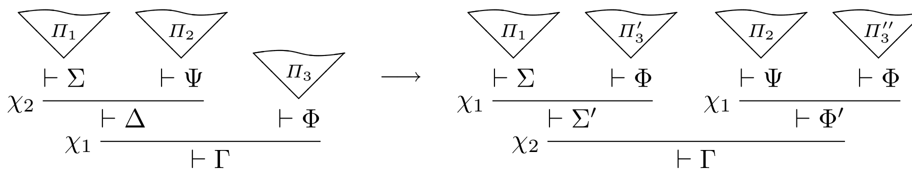
In the following, when working on a given proof Π, we will use the transitive reduction of the $\sqsubset_{\Pi}$ relation, denoted by $<_{\Pi}$, in order to keep the graph of rule instances dependencies as close as possible to the tree structure of proof objects. Moreover, it should be noticed that this relation always exists, and is unique, because $\sqsubset_{\Pi}$ is antisymmetric and finite.

Remark 2 Given a proof Π, its conservative dependency relation $\sqsubset_{\Pi}$ is an invariant of all conservative permutations of rule instances, since it is defined by quantification over these permutations. Thus, all conservative permutations of a given proof have the same permutability properties.

2.2 — General Permutations

A general definition can be given for permutations, allowing to duplicate rule instances in the process, and thus to handle the additive and exponential rule instances, where contraction is involved. This is done by relaxing the condition on the instances that can be used in the resulting proof — this requires to extend the twin relation to proofs, which is trivial — and by allowing to copy the instance moved upward in the tree, as done in the following definition.

Definition 4 *A proof Π' is a* permutation *of some proof Π if and only if Π' is derived from Π by any variant of the following transformation, where Π_3' and Π_3'' are twins of the original Π_3 proof :*



Remark 3 The most general notion of permutation we consider here does not allow erasures of inference rule instances, so that axiomatic rules can never be permuted down. This is the reason why it will be impossible here to capture all possible proofs of the sequent calculus. We will thus restrict our study to *extensive proofs*, where axiomatic rules are only applied on sequents that cannot be handled by any other non-axiomatic inference rule.

Again, the cases of permutation failure induce a dependency relation, but in order to handle duplications, we need to group some rule instances. This is done by using the *merging* relation, which is defined on rule instances of a proof Π so that $\chi_1 \asymp \chi_2$ when $\chi_1 \approx \chi_2$ and there exists a permutation of Π in which only one twin of these instances appear. Then, we will denote by $\mathcal{I}^+_\Pi$ the set of equivalence classes of rule instances generated by the merging relation, applied on all inference rule instances used in this proof.

Definition 5 *Given a proof Π, and $X_1, X_2 \in \mathcal{I}^+_\Pi$, the* dependency *relation $\sqsubset^+_\Pi$ is defined so that $X_1 \sqsubset^+_\Pi X_2$ if and only if, for any permutation Π' of Π, $\chi_1 \in X_1$ and $\chi_2 \in X_2$, χ_2 is located above χ_1 in the proof tree Π'.*

As before, when working on a proof Π, we will use the transitive reduction of the $\sqsubset^+_\Pi$ relation, denoted by $<^+_\Pi$, which always exists and is unique, because the dependency relation defined here is antisymmetric and finite.

Remark 4 The meaning of the merging relation is to group rule instances that can move together within a proof and can be duplicated or merged when need, in order to permuter under or over duplicating rule instances. Then, the general case of the dependency relation is used to express the fact that the dependencies are always consistent among instances of the same equivalence class.

3 — Permutability Graphs

In order to find a satisfying representation for proofs of various logics where the natural concurrency between inference rule applications is exposed, we can use the dependency relation induced by these objects, as described above. Indeed, the most basic role of a proof is to describe all of the deductive steps required to prove a given logic formula, along with at least the necessary part of the ordering between these inference steps.

We devise here a representation for proofs based on graphs, where nodes are rule instances of a proof and links are dependencies that exist between these instances. Because of the strong geometric structure of dependencies, this gives enough information to build a proof in the sequent calculus, while there is no useless sequentialisation information. We start in the restricted setting where no duplication of inference rule instances can happen during any possible permutation — the case of conservative permutations.

3.1 — Conservative Permutability Graphs

Representing equivalent proofs *modulo* conservative permutations is simple since any such permutation applied on a proof produces another proof with the same set of rule instances. This representation is built as a graph $\langle \mathcal{N}, \mathcal{L} \rangle$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ the set of links, described by a relation, as done below.

Definition 6 *The* (conservative) permutability graph *of a proof Π is a directed graph built on the set of rule instances, and defined as $\mathcal{G}_\Pi = \langle \mathcal{I}_\Pi, <_\Pi \rangle$.*

Remark 5 Given a proof Π and its graph $\mathcal{G}_\Pi$, any conservative permutation Π' of Π, with a graph $\mathcal{G}_{\Pi'}$, is such that $\mathcal{G}_{\Pi'} = \mathcal{G}_\Pi$. Indeed, the sets $\mathcal{I}_\Pi$ and $\mathcal{I}_{\Pi'}$ are the same since permutations are conservative, and we noticed in remark 2 that the dependency relation is also invariant through permutations.

In this representation, the non-permutability induced by dependencies is viewed as an *enabling* relation, expliciting the fact that the realisation of an inference step χ_1 requires the previous realisation of another inference step χ_2. As shown below, this relation always induces directed acyclic graphs.

Proposition 1 *For any proof Π, the graph $\mathcal{G}_\Pi$ is acyclic.*

Proof. Let Π be a proof, with $\chi_1, \chi_2 \in \mathcal{I}_\Pi$. If $\chi_2 \sqsubset_\Pi \chi_1$, then χ_1 is above χ_2 in the proof Π, by definition this relation. The other way around, if $\chi_1 \sqsubset_\Pi \chi_2$, then χ_2 is above χ_1 in Π. Thus, such paths in both directions between χ_1 and χ_2 cannot exist in $\mathcal{G}_\Pi$, which is therefore acyclic. $\square$

Beyond acyclicity, permutability graphs have stronger geometric properties, which corresponds to the geometric structure of the represented proofs. Indeed, the branching structure of inference rule instances in a proof is kept in the graph, and can be translated in the language of graph theory as done below.

Definition 7 *Given a permutability graph $\mathcal{G} = \langle \mathcal{I}_\mathcal{G}, <_\mathcal{G} \rangle$, and an instance $\chi \in \mathcal{I}_\mathcal{G}$, a subgraph $\mathcal{B}$ of $\mathcal{I}_\mathcal{G}$ is said to be a* branch *of χ if and only if it is a maximal weakly connected graph such that for any $\chi' \in \mathcal{B}$, $\chi \sqsubset_\mathcal{G} \chi'$.*

This notion of branch, imported from proofs in the setting of permutability graphs, is interesting because one can count different branches of a node, and deduce the branching structure of the corresponding rule instance.

Definition 8 *Given a permutability graph $\mathcal{G} = \langle \mathcal{I}_\mathcal{G}, <_\mathcal{G} \rangle$, and an instance $\chi \in \mathcal{I}_\mathcal{G}$, the* multiplicity *$\mu_\mathcal{G}(\chi)$ of χ is the number of branches of χ in $\mathcal{G}$.*

The interest of counting branches above a given node is then ensured by the correspondence between the situation within proof trees, and permutability graphs used to represent these proofs, as shown below.

Proposition 2 *Given a proof Π, its permutability graph $\mathcal{G}_\Pi = \langle \mathcal{I}_\Pi, <_\Pi \rangle$, and an instance $\chi \in \mathcal{I}_\Pi$, the multiplicity $\mu_\mathcal{G}(\chi)$ is equal to the arity of χ in Π.*

Proof. Let Π be a proof, $\mathcal{G}_\Pi = \langle \mathcal{I}_\Pi, <_\Pi \rangle$ its permutability graph, and $\chi \in \mathcal{I}_\Pi$ an instance such that $\mu_\mathcal{G}(\chi) = n$. First, let $\chi_1, \chi_2 \in \mathcal{I}_\Pi$ be two rule instances located above χ in Π and all of its conservative permutations. If they are in different branches of χ in Π, there is no dependency between them and thus they cannot belong to the same branch of χ in $\mathcal{G}_\Pi$. Then, if they belong to different branches of χ in $\mathcal{G}_\Pi$, then they cannot be located in the same branch of χ in Π, by contradiction. Indeed, it would imply the existence of $\chi_3 \in \mathcal{I}_\Pi$ such that $\chi \sqsubset_\Pi \chi_3$, $\chi_3 \sqsubset_\Pi \chi_1$ and $\chi_3 \sqsubset_\Pi \chi_2$, which means that χ_1 and χ_2 would be in the same branch of χ in $\mathcal{G}_\Pi$. Therefore, there is exactly as many branches of χ in Π as in its permutability graph $\mathcal{G}_\Pi$. $\square$

Remark 6 Following the idea of dependencies as enabling relations, it can be useful to compare permutability graphs with *event structures*, introduced by Winskel [Win86] in order to model causality and concurrency. Indeed, a proof can be considered as an event structure, where rule instances are the events and the conflict relation is induced by the branches of nodes. Indeed, rule instances that belong to different branches of a node should never be grouped in the same branch of this node in any proof extracted from the graph.

Example 7 We provide here an example proof of MLL, with the corresponding conservative permutability graph, given on the right. Branches are suggested using dotted lines, which represent the conflict they induce between nodes.

$$\otimes_1 \dfrac{\parr_2 \dfrac{\otimes_4 \dfrac{\mathsf{id}_5 \dfrac{}{\vdash a, a^\perp} \qquad \mathsf{id}_6 \dfrac{}{\vdash b, b^\perp}}{\vdash a, b, a^\perp \otimes b^\perp}}{\vdash a \parr b, a^\perp \otimes b^\perp} \qquad 1_3 \dfrac{}{\vdash 1}}{\vdash a \parr b, a^\perp \otimes b^\perp \otimes 1} \quad \longrightarrow$$

$$\begin{tikzcd}
 & \mathsf{id}_6 & & & \\
\mathsf{id}_5 \arrow[ur, dotted, no head] & & \otimes_4 \arrow[ul] \arrow[ll] \arrow[rr, dotted, no head] & & 1_3 \\
 & \parr_2 \arrow[ur] & & \otimes_1 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

3.2 — General Permutability Graphs

In order to extend the notion of permutability graph to the general case of permutations, we need to consider groups of rule instances, and work with the general case of the dependency relation we defined. We can follow the scheme of the conservative case to build a similar structure for this extended case.

Definition 9 *The* compact permutability graph *of a proof Π is a directed graph built on equivalence classes of rule instances, and defined as $\mathcal{C}^+_\Pi = \langle \mathcal{I}^+_\Pi, <^+_\Pi \rangle$.*

Example 8 Here is given an example MALL proof, for which the corresponding compact permutability graph, given on the right, requires the use of equivalence classes of inference rule instances. The branches suggested by the dotted line are common to both the $\&_1$ and $\perp_2$ rule instances.

$$\&_1\,\dfrac{\perp_2\,\dfrac{\mathsf{id}_4\,\dfrac{}{\vdash a, a^\perp}}{\vdash a, \perp, a^\perp} \qquad \perp_3\,\dfrac{\mathsf{id}_5\,\dfrac{}{\vdash a, a^\perp}}{\vdash a, \perp, a^\perp}}{\vdash a, \perp, a^\perp \parr a^\perp} \quad \longrightarrow \quad \begin{tikzcd} & \perp_2 \arrow[dl] \arrow[dr] & \\ \mathsf{id}_4 \arrow[rr, dotted, no head] & & \mathsf{id}_5 \\ & \&_1 \arrow[ul] \arrow[ur] & \end{tikzcd}$$

As can be seen in example 8, the problem with compact permutability graphs is the sharing of branches among different nodes. Indeed, sequentialising such nodes might require duplications, and the order between duplicated nodes could be different in different branches. This would induce cycles in the sequentialised versions of the graph, so that we need to find a structure that simplifies the way of managing shared branches and duplications.

Definition 10 *Given a compact graph $\mathcal{C}^+_\Pi = \langle \mathcal{I}^+_\Pi, <^+_\Pi \rangle$, and nodes $X_1, X_2 \in \mathcal{I}^+_\Pi$, two subsets $\mathcal{B}_1$ and $\mathcal{B}_2$ of $\mathcal{I}^+_\Pi$ are said to be* incompatible *if and only if both of them are branches of X_1 and also branches of X_2.*

Now, we can build a graph that avoids compatibility problems, by removing in each case of branch incompatibility one of the two branches. Such a subset of the original nodes set, where no branches can be incompatible, is said to be *compatible*, and is used in the following definition.

Definition 11 *Given a compact graph $\mathcal{C}^+_\Pi = \langle \mathcal{I}^+_\Pi, <^+_\Pi \rangle$, a subgraph of $\mathcal{C}_\Pi$ is said to be a* slice *if and only if it is induced by a compatible subset of $\mathcal{I}^+_\Pi$.*

Finally, the general version of permutability graphs will be a structure that yields a simple sequentialisation, is built by gathering slices of the corresponding compact graph, and behaves like a collection of conservative graphs.

Definition 12 *The* (general) permutability graph *$\mathcal{G}^+_\Pi$ of a proof Π is the set of all valid slices of its compact permutability graph $\mathcal{C}^+_\Pi$.*

Remark 9 Once again, a proof Π and a permutation Π' of Π have the same graph $\mathcal{G}^+$. This is also valid for their compact graph $\mathcal{C}^+$. Indeed, the nodes sets are the same since they are generated by the merging relation on the same intial set of inference rule instances, and the dependency relation is invariant through general permutations because of its definition.

Permutability graphs in the general case can be manipulated through their slices, which are similar to conservative graphs — we removed incompatible branches to avoid duplication problems — and have some of their properties.

Proposition 3 *For any proof Π, all slices in $\mathcal{G}^+_\Pi$ are acyclic.*

Proof. We can use the same reasoning as in the conservative case to show that the compact graph $\mathcal{C}^+_\Pi$ of a proof Π is acyclic. Then, all the slices in $\mathcal{G}^+_\Pi$ are subgraphs of the compact graph, so that they are acyclic as well. $\square$

Remark 10 In the compact graph of a proof, the multiplicity of a node is no longer equal to its arity in the corresponding proofs, because of duplications. However, slices partially solve this problem, so that this property holds, except in the case of nodes that induce some branch incompatibility — and in this case, merging different slices will eventually induce the right multiplicity.

3.3 — Building and Using Permutability Graphs

We now have a generic representation for proofs based on graphs, with two levels of analysis, depending on the kind of permutations we consider as valid. However, some important points of our methodology have been left unspecified, so that we need to clarify the way permutability graphs are intended to be built, and used for proof representation as well as for proving completeness of proof transformations based on inference rule instances permutations.

Building dependencies. The definition of permutability graphs has been almost entirely abstracted with respect to the construction of the permutability relations on which it relies. Indeed, the dependency relation is defined using a construction process that gives very little information, so that for example, a rule instance depends on another if and only if we observe that the permutation of these instances never happens. This implies to build all equivalent proofs when building the graph of a proof, which seems redundant if we want to eventually build a permutation of the initial proof.

Despite its very general and heavy definition, the dependency relation can be build by only looking at the given proof, in most logical systems. Indeed, we can define dependency in terms of principal occurrences of rule instances at least for LK, LJ, and LL without exponentials. For example, the most simple dependency case happens when the principal occurrence of a rule instance is a suboccurrence of the principal occurrence of another instance, and there are similar definitions for splitting and duplication dependencies. Using such techniques would allow for an efficient construction algorithm for permutability graphs.

Conservativity and duplications. The level of analysis required, and thus the choice of conservative or general permutability graphs, depends on the goals of such a work on proofs. There are three cases to consider :

- *without duplications, both versions collapse (e.g. the case of* MLL*).*
- *with duplications, if the size/complexity of proofs matters.*
- *with duplications, maximal identification of proofs, inducing a more compact representation, but not respecting complexity.*

In order to use permutability graphs as a canonical representation for proof objects, it is therefore required to choose first the features they need, according to the properties of the resulting objects — especially regarding the issue of size and complexity of proofs —, except in weak logics where duplication is impossible, such as MLL. Indeed, the general version of these graphs induces the identification of many proofs, up to duplications that can create an exponential blow-up. However, when using permutability graphs to prove completeness for proof transformations, the choice depends on the transformation. For example, focusing requires general graphs, since it can produce equivalent proofs that are much bigger that the original one.

4 — Back to the Sequent Calculus : Sequentialisation

Because the original motivation for introducing permutability graphs was the study of proof transformations based on permutations, the next step is naturally to prove that a valid proof tree of the sequent calculus can always be built from such a graph. This is of course very close to the key result of sequentialisation in the settings of proof-nets, and we will use here a methodology inspired by the proof of sequentialisation given in [GF08].

In order to build a valid proof tree of the sequent calculus, we need to add more sequentiality to permutability graphs, since the ordering of inference steps is not refined enough. Such partially sequentialised objects can be uniformly defined based on permutability graphs either in the conservative or general case, where each slice will be sequentialised, instead of a whole graph.

Definition 13 *Given a graph $\mathcal{G} = \langle \mathcal{I}_\mathcal{G}, <_\mathcal{G} \rangle$, a* sequential relation $\rhd_\mathcal{G}$ *on $\mathcal{G}$ is an irreflexive relation such that $\mu_\mathcal{G} = \mu_{\mathcal{G}'}$, where $\mathcal{G}' = \langle \mathcal{I}_\mathcal{G}, <_\mathcal{G} \cup \rhd_\mathcal{G} \rangle$.*

The idea behind the whole sequentialisation process is very simple : we just pick two nodes for which no order is specified, and check that sequentialise them will not merge different branches, which would break the structure of the proof. Then, we add a sequential link between them, in one of the directions allowed by the condition on multiplicity. The objects created during this operation are then a straightforward extension of permutability graphs.

Definition 14 *Given a graph $\mathcal{G}$, a* sequentialisation *of $\mathcal{G}$ is a graph $\mathcal{S} = \langle \mathcal{I}_\mathcal{G}, <^{\rhd}_\mathcal{G} \rangle$, where $<^{\rhd}_\mathcal{G}$ is the transitive reduction of $<_\mathcal{G} \cup \rhd_\mathcal{G}$, and $\rhd_\mathcal{G}$ is a sequential relation.*

Usual permutability graphs can then be considered as the most basic case of sequentialisation, when no sequential link has been added. Beyond this, there is a whole spectrum of partially refined objects, well defined because the transitive reduction of new links added to dependencies exists and is unique, since this is an antisymmetric and finite relation. The objects we are eventually interested in are maximal sequentialisations, where no more link can be added.

Definition 15 *A* full sequentialisation *of a given graph $\mathcal{G}$ is a graph $\mathcal{F} = \langle \mathcal{I}_\mathcal{G}, <_\mathcal{F} \rangle$ for which there is no valid sequential relation $<^{\rhd}_\mathcal{F}$ on $\mathcal{F}$ such that $<_\mathcal{F} \subset <^{\rhd}_\mathcal{F}$.*

Full sequentialisations allow us to end the permutability graph refinement process, by finally providing a modified version of the initial graph which holds all the sequentialisation information of a proof tree in the sequent calculus. In order to prove that a valid sequent calculus proof can finally be built, we need to prove a sequentialisation theorem, starting in the conservative case.

4.1 — Conservative Sequentialisation

The proof of our sequentialisation theorem relies in the conservative case on the invariance of the multiplicity of nodes. which ensures that the arity of rule instances is correct in the resulting proof, and also on the transformation of the directed acyclic graph into a tree, after full sequentialisation. The key lemma exhibits the fact that a sequent calculus proof is a permutability graph where sequential links have been maximally added.

Lemma 4 *A conservative full sequentialisation $\mathcal{F}$ is a sequent calculus proof.*

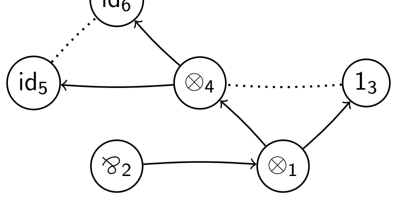
Proof. Let $\mathcal{F} = \langle \mathcal{I}_{\mathcal{F}}, <_{\mathcal{F}} \rangle$ be a conservative full sequentialisation. First we prove that it is a tree because any node has at most one incoming link. Indeed, given three nodes $\chi_1, \chi_2, \chi_3 \in \mathcal{I}_{\mathcal{F}}$, we suppose $\chi_1 <_{\mathcal{F}} \chi_3$ and $\chi_2 <_{\mathcal{F}} \chi_3$, so that a link between χ_1 and χ_2 would not modify their multiplicity. This link must exist because $\mathcal{F}$ is a full sequentialisation, but then one of the links to χ_3 does not exist, since $<_{\mathcal{F}}$ is a transitive reduction, and we have a contradiction. Moreover, the graph is connected, and thus rooted in one node, so that it is actually a tree. Finally, $\mathcal{F}$ is a valid sequent calculus proof because of proposition 2, which is preserved by additional sequential links, and ensures that any node, being a rule instance, has precisely as many outgoing edges in $\mathcal{F}$ as it has premises in any of the proofs corresponding to the initial graph. $\square$

Finally, the main sequentialisation result is achieved by proving that given a proof Π of the sequent calculus, its conservative permutability graph $\mathcal{G}$ can be *sequentialised* into a proof Π', equivalent *modulo* conservative permutations to the initial proof Π, by producing the corresponding full sequentialisation — using what we know of Π' to add sequential links exactly where needed.

Theorem 5 (Sequentialisation) *Given a proof Π, its conservative permutability graph $\mathcal{G}_{\Pi}$ can be sequentialised into any conservative permutation Π' of Π.*

Proof. Let Π be a proof and Π' a conservative permutation of Π, so that we have $\mathcal{G}_{\Pi'} = \mathcal{G}_{\Pi}$. We want to build a full sequentialisation $\mathcal{F}$ of $\mathcal{G}_{\Pi}$, which is a proof Π'' by lemma 4, such that $\Pi'' = \Pi'$. For this we start with $\mathcal{G}_{\Pi}$ and use a sequential relation $\rhd_{\Pi'}$ that we deduce from the structure of Π'. It is defined as follows : for $\chi_1, \chi_2 \in \mathcal{I}_{\Pi'}$, $\chi_1 \rhd_{\Pi'} \chi_2$ if and only if χ_2 is applied on a premise of χ_1, so that it simply mimicks the deductive structure of the proof Π' that we want to reproduce. This is a valid sequential relation, since it cannot merge branches, by contradiction with the existence of Π'. Finally, if $\mathcal{F}$ is built this way, it is precisely the proof Π', because it is organised using the same ordering relation $<_{\Pi'}$ on inference rule instances. $\square$

Example 11 One of the two possible full sequentialisations of the permutability graph given in example 7 is shown below. The corresponding proof, given on the right, is a simple permutation of the original one, where the $\parr_2$ rule instance has been moved under the $\otimes_1$ instance.



$$\longrightarrow \quad \parr_2\dfrac{\otimes_1\dfrac{\otimes_4\dfrac{\mathsf{id}_5\,\overline{\vdash a, a^\perp} \qquad \mathsf{id}_6\,\overline{\vdash b, b^\perp}}{\vdash a, b, a^\perp \otimes b^\perp} \qquad 1_3\,\overline{\vdash 1}}{\vdash a, b, a^\perp \otimes b^\perp \otimes 1}}{\vdash a \parr b, a^\perp \otimes b^\perp \otimes 1}$$

4.2 — General Sequentialisation

In the general case, the sequentialisation is done on each slice of the general permutability graph, and the proof tree is finally built by partially merging sequentialised slices. However, there is a condition on the way different slices can be sequentialised, that is applied on *slicing nodes* — the nodes that induce incompatibility between branches.

Definition 16 *Two graphs $\mathcal{H}_1 = \langle \mathcal{I}_1, <_1 \rangle$ and $\mathcal{H}_2 = \langle \mathcal{I}_2, <_2 \rangle$ are* consistent *with each other if and only if for any $\chi, \chi' \in \mathcal{I}_1 \cap \mathcal{I}_2$, if χ is a slicing node then the two ordering relations $<_1$ and $<_2$ are equal on the subset $\{\chi, \chi'\}$.*

Now, we can define simple and full sequentialisations in the general case, which are imported from the conservative case by reducing the problem to slices, while asking for the consistency condition to be respected.

Definition 17 *Given a general permutability graph $\mathcal{G}^+ = \{\mathcal{H}_1, \cdots, \mathcal{H}_n\}$, a (full) sequentialisation of $\mathcal{G}^+$ is a set $\mathcal{S}^+ = \{\mathcal{S}_1, \cdots, \mathcal{S}_n\}$ such that for all $i, j \in [1, n]$, $\mathcal{S}_i$ is a (full) sequentialisation of $\mathcal{H}_i$, and $\mathcal{S}_i$ is consistent with $\mathcal{S}_j$.*

The sequentialisation result is again achieved by proving that any general permutability graph can be sequentialised into a proof tree that is a permutation of the original proof tree, as done below.

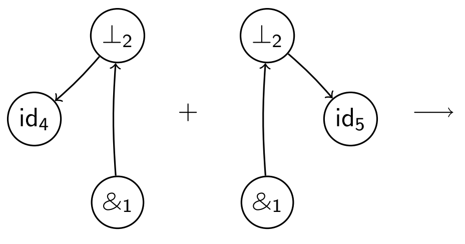
Theorem 6 (Sequentialisation) *Given a proof Π, its general permutability graph $\mathcal{G}^+_\Pi$ can be sequentialised into any general permutation Π' of Π.*

Proof Sketch. Let Π be a proof and Π' a general permutation of Π, so that we have $\mathcal{G}^+_{\Pi'} = \mathcal{G}^+_\Pi$. The idea is to build a specific full sequentialisation $\mathcal{F}$ of $\mathcal{G}_\Pi$, and prove that it is isomorphic to a proof Π'', such that $\Pi'' = \Pi'$.

First, building a full sequentialisation on the model of Π' is done using the same scheme as in the conservative case. Indeed, it is sufficient to follow the deductive structure of Π'. Moreover, the ordering relation of different branches should remain isolated in the different slices they are associated to.

Then, we have to build a valid proof tree by merging the sequentialised slices, which is not trivial since some nodes need to be merged while some other should not. The idea is that identical slicing nodes should be merged until they reach to the correct arity, and other nodes should be merged only under slicing nodes — that can be seen in the example below, and in the second possible full sequentialisation of the same graph. When all slices have be grouped, the resulting tree is actually a proof, because problems of cycles and wrong ordering have been ruled out by the consistency condition and the merging process. $\square$

Example 12 We consider here one of the two possible full sequentialisations of the permutability graph given in example 8. The corresponding proof, given on the right, uses twice the $\bot_2$ rule, as a result of the merging of the two slices.



$$\&_1\,\cfrac{\bot_2^1\,\cfrac{\mathsf{id}_4\,\cfrac{}{\vdash a, a^\perp}}{\vdash a, \bot, a^\perp} \qquad \bot_2^2\,\cfrac{\mathsf{id}_5\,\cfrac{}{\vdash a, a^\perp}}{\vdash a, \bot, a^\perp}}{\vdash a, \bot, a^\perp \parr a^\perp}$$

The sequentialisation result presented here is very important when studying permutability graphs, since it ensures that they provide a correct representation of equivalent proofs. Moreover, it is the key to possible proofs of completeness for transformations such as focusing, for which it is only necessary to refine the sequentialisation theorem, by proving that a full sequentialisation of a certain shape can be built from any permutability graph.

4.3 — Completeness of Focusing for MLL

As an example of the use of permutability graphs in the study of different proof transformations, we discuss here the proof of completeness for focusing. The methodology used here allows for a very simple and modular proof, which can be seen as an extension of the focalization graphs method used in [MS07]. The basic idea is to prove a sequentialisation theorem which ensures that a *focused full sequentialisation* can be produced from a permutability graph — such a sequentialisation being one which respects the focusing conditions.

Defining focusing. The first step here is to explicitly define what a focused proof is, in terms of the ordering relation, and depending on the polarity of the nodes. In the case of MLL, this is fairly easy : basically, any $\parr$ or $\bot$ rule instance, which is negative, should be located under all possible $\otimes$ and 1 instances, which are positive — this is not the case only when a negative rule instance actually depends on a positive rule instance. Therefore, when adding more links to a permutability graph, choosing only links going out of a negative node is a condition to build full sequentialisation that is a focused proof.

Proving completeness. Most of the work for this proof has been done in establishing properties of permutability graphs. Indeed, all of the MLL proofs equivalent up to conservative permutations can be produced by sequentialising the corresponding permutability graph. Thus, it is sufficient to prove that it is always possible to build a particular full sequentialisation, that respects all the focusing conditions. This is easy to prove, because the geometry of MLL proofs is such that if it is possible to add a link from a positive node to a negative node, then it also possible to add the link between the same nodes in the other direction — because they necessarily have a common successor.

5 — Conclusion and Future Work

The study of permutability graphs and related techniques is a work in progress. We only presented here the basic definitions and theorems, with the purpose of emphasizing the simplicity and genericity of these graph structures. This should of course be related to proof-nets, but as noticed before the scope is wider, since permutability properties can be easily studied in many different logics. However, more than a technical artifact inspired by proof-nets and focalization graphs, this work should be seen as an attempt to find new insights on general proof theory by focusing on the notions of permutability and concurrency in the geometrical structure of proofs. There are many possible research directions, and we will now briefly describe the most important ones, considered as natural next steps in this project.

LL, LK and LJ. Because of the intended genericity of our permutability graph method, this paper is based on the very general setting of the sequent calculus instead of one particular logical system in this formalism. The systems for linear logic, and classical and intuitionistic logics should be studied in details, so that general observations could be made on the permutability properties of their inference rule, and thus on the geometrical structure of their proofs.

Focusing for LL. The original inspiration for the definition of permutability graphs came from the focalization graphs used in [MS07] to prove completeness of a focused system for LL. The difficulties due to the construction of a new graph for each *positive trunk* step are solved here by the use of a single graph which holds all the required information. As already noticed, this proof of completeness benefits from the simple setting established here. Moreover, this graph method could be used to abstract the notion of focusing out of its strong origins in the LL sequent calculus, and transport it to other settings.

Deep inference. The *deep inference* methodology, introduced by Guglielmi for the definition of the *calculus of structures* formalism in [Gug07], induces a complex and interesting behaviour with respect to permutability. Since inference rules can be applied deep inside formulas, some of the redex dependencies, that could be considered useless, disappear in this setting. Moreover, permutability graphs could allow for a simple and elegant proof of completeness for focusing, which could be defined in this formalism using this graph abstraction, since the sequentialisation process is here reduced to the usual notion of topological sorting of their nodes.

Proof transformations. Beyond focusing, many proof transformations, and especially those based on inference rule instances permutations, could be studied using permutability graphs. In the calculus of structures, the splitting lemma, used to prove cut-elimination, and the decomposition theorems are interesting examples. By observing the permutability properties of the cut rule, it should be possible to study its interaction with other inference rules and establish or refute the commutation of cut-elimination with proof transformations such as focusing, using a permutability graph rewriting viewpoint on cut-elimination.

Proof-nets. The main research direction suggested by the graph structure we use and recent results on the canonicity of focused proofs is the improvement of proof-nets and the search for a better understanding of their behaviour. The study started here should try to link this geometrical view of proofs with their permutability properties, thus allowing for a different approach on this topic.

References

[CF05] P-L. Curien and C. Faggian. L-nets, strategies and proof-nets. In C.-H. L. Ong, editor, *CSL'05*, volume 3634 of *LNCS*, pages 167–183, 2005.

[GF08] P. Di Giamberardino and C. Faggian. Proof nets sequentialisation in multiplicative linear logic. *Annals of Pure and Applied Logic*, 155(3):173–182, 2008.

[Gir96] J-Y. Girard. Proof-nets : the parallel syntax for proof-theory. In A. Ursini and P. Agliano, editors, *Logic and Algebra*. M. Dekker, New York, 1996.

[Gug07] A. Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1):1–64, January 2007.

[MS07] D. Miller and A. Saurin. From proofs to focused proofs : a modular proof of focalization in linear logic. In J. Duparc and T. A. Henzinger, editors, *CSL'07*, volume 4646 of *LNCS*, pages 405–419, 2007.

[Win86] G. Winskel. Event structures. In *Advances in Petri Nets*, volume 255 of *LNCS*, pages 352–392, 1986.

Categorical Semantics and Non-Free Categories [*]

A.El Khoury, S. Soloviev [†]L. Mehats [‡]M. Spivakovsky[§]

1 Introduction.

Usually when categorical semantics of proofs is considered it is based on the structure of the free category of a certain type defined on a certain logical system, for example the structure of free Cartesian Closed Category on the $(\wedge, \rightarrow)$-fragment of the Intuitionistic Propositional Calculus or the structure of free Symmetric Monoidal Closed Category on the $(\otimes, \multimap)$-fragment of the Intuitionistic Multiplicative Linear Logic, etc. One may ask why non-free categories are usually not considered.

A possible answer is that while it seems natural that there may be close connections between free categories with structure and logical calculi (after all both are essentially syntactical systems) one does not expect much of the use of proof-theoretic methods in the study of non-free categories because they mostly belong to the concrete mathematical domains with their own methods and problems that are very different from problems typically studied by proof theory (take, e.g. commutative algebra).

Another possible answer may be that it is expected (again on the intuitive level, before any serious study) that if there is some interesting non-free category in a known class of categories (e.g., closed categories) and this category can be studied by logical methods then eventually one will find some additional axioms and describe a new interesting subclass such that the category in question will be free with respect to this subclass. So, again there is no need to consider non-free categories.

In our presentation we consider several cases relatively little known to the community (proof-theorists and specialists in categorical logics) that, in our opinion, show why non-free categories may be interesting to proof-theorists and also show the efficiency of proof-theoretic methods in the study of non-free categories.

[*]This work was partially supported by PEPS ST2I CNRS “Vérification de la commutativité des diagrammes catégoriques en calcul formel”.

[†]IRIT, UMR 5505 CNRS, University of Toulouse, 118 route de Narbonne, 31062 Toulouse, France {elkhoury, soloviev}@irit.fr

[‡]LaBRI, University of Bordeaux I, 351 Cours de la Libération 33405 Talence, France, mehats@labri.fr.

[§]Institute of Mathematics, UMR 5219 CNRS, University of Toulouse, 118 route de Narbonne, 31062 Toulouse, France, spivakov@math.ups-tlse.fr.

Some of the results considered below were published before, some are very recent and not yet published. The main new results can be found in section 4.

We will consider the following problems:

- Equivalences on derivations generated by interpretations in non-free categories. Different types of equivalences, critical pairs and a canonical axiomatization.
- Triple-dual conjecture and the problem of full coherence in closed categories.
- Varieties of categories with structure.
- Commutativity and dependency of diagrams in non-free categories. Proof-theoretic methods of verification of commutativity.
- Arbitrary natural transformations and proof-theoretic methods in their study.

We shall mostly consider Symmetric Monoidal Closed Categories (SMCC) and the related logical system - Intuionistic Multiplicative Linear Logic (IMLL), because they were one of the central subjects of our studies. Our interest in them is explained by the fact that in this case the algebraic and logical aspects are "in equilibrium", while in other cases usually one or another side prevails (for example, logical side in case of Cartesian Closed Categories.) It should be noticed, though, that many of the definitions and some of the results have larger domain of application. We shall comment on this when appropriate.

2 Equivalences on derivations.

The calculus $\mathbf{L}(\mathbf{A})$ is defined as follows:

Axioms

$A \to A \quad (1_A) \qquad \to I \quad (\text{unit})$

Structural Rules

$$\frac{\Gamma \to A \quad A, \Delta \to B}{\Gamma, \Delta \to B}(\text{cut}) \qquad \frac{\Delta \to I \quad \Sigma \to A}{\Delta, \Sigma \to A}(\text{wkn}) \qquad \frac{\Gamma \to A}{\Gamma' \to A}(\text{perm})$$

Logical rules

$$\frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \otimes B}(\to\otimes) \qquad \frac{A, B, \Gamma \to C}{A \otimes B, \Gamma \to C}(\otimes\to)$$

$$\frac{A, \Gamma \to B}{\Gamma \to A \multimap B}(\to\multimap) \qquad \frac{\Gamma \to A \quad B, \Delta \to C}{\Gamma, A \multimap B, \Delta \to C}(\multimap\to)$$

Here Γ, Δ, Σ are lists of formulas. A list of formulas $\Gamma = A_1, ..., A_n$ may be viewed as an abbreviation of $\overline{\Gamma} = (...(A_1 \otimes ...) \otimes A_n) \otimes I$.

It is well known that on $\mathbf{L}(\mathbf{A})$ a structure of a free SMCC over the set of atoms $\mathbf{A}$ can be defined. In this structure the equivalence classes of derivations of the sequent $A \to B$ will play the role of morphisms from A to B. The equivalence of free SMCC on derivations will be denoted by $\equiv$. (It is defined as the smallest equivalence relation satisfying certain axioms.)

As for every free category, any valuation $v{:}\mathbf{A} \to Ob(K)$ where K is some SMCC defines a unique structure-preserving functor $|-|_v{:}\mathbf{L}(\mathbf{A}) \to K$.

Via this functor a new equivalence relation $\sim_v$ on derivations can be defined: $d \sim_v d' \Leftrightarrow |d|_v = |d'|_v$. The relation $\sim_v$ also defines a certain structure of SMCC on $\mathbf{L}(\mathbf{A})$; $\equiv \subseteq \sim_v$.

One may notice that $\sim_v$ is a congruence with respect to the rules of $\mathbf{L}(\mathbf{A})$, but in general it is not closed with respect to substitution of formulas for variables.

Relations $\sim$ that define the structure of SMCC on $\mathbf{L}(\mathbf{A})$ have many interesting properties (for example, they form a lower semilattice) but we shall mostly consider two types defined below. Let us fix some SMCC K:

(a) The relation $\sim_\forall$ is defined by $d \sim_\forall d' \Leftrightarrow \forall v{:}\mathbf{A} \to Ob(K).(d \sim_v d')$.

(b) Let $NAT(K)$ be the SMCC category of functors and natural transformations over K (standard definition). Let v_1 denote the interpretation of $\mathbf{L}(\mathbf{A})$ in $NAT(K)$ defined by $a \mapsto 1{:}K \to K$ (every $a \in \mathbf{A}$ has the identity functor $1{:}K \to K$ as its value). The relation $\sim_{nat}$ is defined by $d \sim_{nat} d' \Leftrightarrow .d \sim_{v_1} d'$.

The relations $\sim_{nat}$ and $\sim_\forall$ are both congruences with respect to the rules of $\mathbf{L}(\mathbf{A})$. Both are closed with respect to substitution of formulas for atoms.

Due to these properties, there are very efficient proof-theoretic methods to study these two types of relations.

The relation $\equiv$ is defined on the derivations of $\mathbf{L}(\mathbf{A})$ by the axioms of SMCC. Clearly, all the relations $\sim$ that define some SMCC structure on $\mathbf{L}(\mathbf{A})$ are defined by adding some new axioms (equivalences between derivations) to the axioms of SMCC. The same relations can be defined via $|-|_v$ for some K and v (if necessary, one may take the factor category of $\mathbf{L}(\mathbf{A})$ by $\sim$ as K).

A non-trivial problem here is to find some canonical axiomatization of these relations if possible.

Let us mention some general results (see [10]) concerning these categorical relations on derivations of $\mathbf{L}(\mathbf{A})$.

Theorem 2.1 *Let K be an SMCC category with biproduct. Then the relations $\sim_{nat}$ and $\sim_\forall$ (defined with respect to K) coincide.*

Definition 2.2 *A sequent S is called balanced if every atom has exactly two occurrences with opposite signs in S. It is pure if there are no subformulas of the form $I \otimes A, A \otimes I, I \multimap A$.*

Theorem 2.3 *A relation of type $\sim_{nat}$ can always be axiomatized using the axioms of the form $d \sim d'{:}\Gamma \to A$ where $\Gamma \to A$ is balanced and pure.*

For the relations of the type $\sim_{nat}$ there exists a nice canonical axiomatization.

Let us describe first a very useful equivalence-preserving transformation of derivations: reduction to 2-sequents (cf. [13, 14, 15, 10].

Definition 2.4 *The sequent $\Gamma \to A$ is called a 2-sequent if A contains no more than one connective and each member of Γ no more than two connectives.*

Some formulas may be replaced by isomorphic ones (reducing further the number of possibilities).

Definition 2.5 *$\Gamma \to A$ is called a pure 2-sequent if A has one of the forms $x, a \otimes b, a \multimap x$ and each member of Γ has one of the forms $x, a \multimap x, a \multimap (b \otimes c), (a \otimes b) \multimap x, (a \multimap x) \multimap y$. Here x, y stand for I or atoms, a, b are atoms.*

Every derivation can be transformed into a derivation of some 2-sequent using two operations (followed by isomorphisms to obtain a *pure* 2-sequent):

$$\Gamma \overset{d}{\to} B \mapsto \frac{\Gamma \overset{d}{\to} B \quad p \overset{id}{\to} p}{\Gamma, B \multimap p \to p}(p\, fresh)$$

and *cut* with left premises of the form $p \multimap C, A[p] \to A[C]$ or $C \multimap p, A[p] \to A[C]$ (p fresh) [1]. There also exists the inverse transformation using substitutions $[C/p]$ and *cuts* with $\to C \multimap C$ (*cut* can be eliminated afterwards.)

Theorem 2.6 *(Reduction to 2-sequents.) Let an equivalence relation $\sim$ on derivations contain $\equiv$ and be a substitutive congruence. Let d_1, d_2 be two derivations of the same sequent S. Then there exist two derivations d'_1, d'_2 of the same pure 2-sequent S' (balanced if S was balanced) such that d_1, d_2 are $\sim$-equivalent iff d'_1, d'_2 are $\sim$-equivalent.*

Definition 2.7 *A pair of derivations of the same balanced pure 2-sequent S is critical if*

$$(1)\ \ d_1 \equiv \frac{\Gamma, A' \multimap I \overset{d'_1}{\to} A \quad I \to I}{\Gamma, A' \multimap I, A \multimap I \to I}\multimap\to,\quad d_2 \equiv \frac{\Gamma, A \multimap I \overset{d'_2}{\to} A' \quad I \to I}{\Gamma, A' \multimap I, A \multimap I \to I}\multimap\to, perm;$$

(2) a cut-free derivation of S can end only by some application of $\multimap\to$;

(3) the derivations d'_1, d'_2 are not $\equiv$-equivalent to derivations ending by $\multimap\to$.

The pair is minimal if Γ does not contain members being single atoms.

[1] Here a single occurrence of C is replaced by p. The form depends on the variance (sign) of this occurrence of C in A. One takes a standard derivation of these sequents, which always exists in "symmetric" calculi, i.e., logical systems for CCC, SMCC, SCC categories. The theorem that follows holds in each of these systems.

Let α be some substitution of I for variables. In [15] the "substitutions with purification" were defined. Let $d{:}\Gamma \to A$ be a derivation of some 2-sequent. Then $\alpha * d$ is the derivation obtained from d by α and *cuts* with isomorphisms that will make its final sequent pure. The derivation $\alpha * d$ is defined up to $\equiv$, but its final sequent is defined without ambiguity.

Theorem 2.8 *(Cf. [15].) Let d_1, d_2 be derivations of a balanced sequent $\Gamma \to A$ and d_1', d_2' the corresponding derivations of a balanced pure 2-sequent. Then $d_1 \equiv d_2$ iff there exists a substitution α of I for variables such that $\alpha * d_1', \alpha * d_2'$ is a minimal critical pair*[2].

The following theorem shows that every relation of the type $\sim_{nat}$ is generated by minimal critical pairs.

Theorem 2.9 *[10] For every relation $\sim_{nat}$ there exists some set M of minimal critical pairs such that $\sim_{nat}$ is the smallest equivalence relation that is a congruence with respect to the rules of $\mathbf{L(A)}$, closed with respect to substitution, contains $\equiv$ and all the pairs $(d, d') \in M$.*

3 The problem of full coherence in closed categories and the triple-dual conjecture.

Below we shall call diagrams not only pairs $f, g{:}A \to B$ but also pairs of $\mathbf{L(A)}$-derivations of arbitrary sequent $\Gamma \to A$.

A sequent S is called proper if it does not contain occurrences of subformulas of the form $A \multimap B$ where B is constant (contains only I) and A is not constant.

Theorem 3.1 *(The Kelly-Mac Lane coherence theorem reformulated for $\mathbf{L(A)}$, cf. [5].) Let $f, g{:}\Gamma \to A$ and assume that the sequent $\Gamma \to A$ is proper. If f and g have the same graph*[3] *then $f \equiv g$.*

Example 3.2 *If the sequent is not proper, f may be non-equivalent to g. The sequent $((a \multimap I) \multimap I) \multimap I, (a \multimap I) \multimap I \to I$ has two non-equivalent derivations (with respect to $\equiv$), for example, f with $((a \multimap I) \multimap I) \multimap I$ and g with $(a \multimap I) \multimap I$*

[2]The conditions on the left premises that require verification of equivalence are applied to the (finite number of) derivations with smaller final sequent. This theorem may be used recursively to obtain deciding algorithms for $\equiv$. In [17] an algorithm of low polynomial complexity was described.

[3]In particular, if the sequent is balanced.

as the main formula of the last rule. The so called "triple-dual" diagram (cf. [5])

$$\begin{tikzcd}
((a \multimap I) \multimap I) \multimap I \arrow[rr, "1"] \arrow[dr, "k_a \multimap 1"'] & & ((a \multimap I) \multimap I) \multimap I \\
 & a \multimap I \arrow[ur, "k_{a \multimap I}"'] &
\end{tikzcd} \tag{1}$$

where a is a variable and $k_a = (1 \multimap e_{aI}) \circ d_{a(a \multimap I)}{:}a \to (a \multimap I) \multimap I$ is the standard "embedding of a into its second dual" corresponds to the pair of derivations of the sequent $((a \multimap I) \multimap I) \multimap I \to ((a \multimap I) \multimap I) \multimap I$ obtained by $\to \multimap$ from the previous one[4].

Non-commutativity of this diagram may be checked formally (the equivalence relation $\equiv$ is decidable). It is non-commutative also in certain models such as the SMCC of vector spaces (including infinitely dimensional) or the SMCC of modules over a commutative ring with unit.

The "Triple-Dual" Conjecture.

Conjecture 3.3 *Commutativity of the triple-dual diagram (equivalence of corresponding derivations) implies commutativity of all the diagrams of canonical maps $f, g{:}\Gamma \to B$ with balanced $\Gamma \to B$. More precisely: let $\sim$ be the smallest equivalence relation that satisfies all axioms of SMCC, is substitutive and the triple-dual diagram is commutative with respect to $\sim$. Then for all $f, g{:}\Gamma \to B$ with balanced $\Gamma \to B$ in $\mathbf{L(A)}$ we have $f \sim g$.*

An argument in favor of this conjecture is that the following theorem holds.

Theorem 3.4 *[14]. If $\sim$ is the smallest equivalence relation that satisfies all the axioms of SMCC, is substitutive, the triple-dual diagram is commutative with respect to $\sim$, and for all f, g and any atom a*

$(*)$ $[a \multimap I/a]f \sim [a \multimap I/a]f \Rightarrow f \sim g$,

then $f \sim g$ for all $f, g{:}\Gamma \to B$ with the same graph[5].

Antoine El Khoury recently checked that (without the assumption (*)) the commutativity of the triple-dual diagram implies the commutativity of all diagrams $f, g{:}A \to B$ with balanced $A \to B$ containing no more than 3 variables.

4 Varieties of SMCC

The main results presented in this section are new.

[4]Applying the algorithm of reduction to 2-sequents one may obtain another non-commutative diagram $f', g'{:}(a \multimap I) \multimap I, (b \multimap I) \multimap I, a \otimes b \multimap I \to I$. Notice that all these diagrams are related to critical pairs.

[5]Equivalently: with balanced $A \to B$.

We consider equivalence relations $\sim$ on derivations of $\mathbf{L(A)}$, $\equiv\subseteq\sim$, generated by the equations of the form $f \sim g$; here $f, g{:}\Gamma \to A$ with $\Gamma \to A$ not necessarily balanced but f, g have the same graph[6].

Obviously, any set of axioms of this form (plus the axioms of SMCC) defines a variety of SMCC in the sense of universal algebra.

Commutativity of the diagrams considered below **does not imply** the commutativity of the triple-dual diagram (or the diagrams equivalent to the triple-dual), so the equivalence relation generated by these identities are strictly between $\equiv$ (relation of the free SMCC) and the relation generated by commutativity of the triple-dual.

First non-trivial "intermediate" equation was obtained due to a suggestion of M. Spivakovsky, developed later by L. Mehats and S. Soloviev [10].

The diagram (3)

$$f', g'{:}(((a \multimap I) \otimes (b \multimap I)) \multimap I) \multimap I, ((b \multimap I) \multimap I), ((a \multimap I) \multimap I) \rightrightarrows I$$

studied in [10] was obtained from

$$(2)\ \ f, g{:}\ (a \otimes b \multimap I), ((b \multimap I) \multimap I), ((a \multimap I) \multimap I) \rightrightarrows I$$

by *cut* with (unique) $h{:}(((a \multimap I) \otimes (b \multimap I)) \multimap I) \multimap I \to (a \otimes b \multimap I)$.

It was shown in [10] that in a certain category of modules over a ring (3) is commutative while (2) and (1) are not (lemma 5.8). So, if we add to the axioms of SMCC the equation corresponding to (3), the diagrams (2) and (1) will remain non-commutative.

In this paper we describe a sequence $D_2, ..., D_k, ..., D_m, ...$ of diagrams and certain models K_k such that in K_k the diagrams $D_2, ..., D_k$ are non-commutative and there exists $m > k$ such that $D_m, ...$ are commutative (we do not know whether $D_k, ..., D_{m-1}$ are commutative).

Below we shall write A^* instead of $A \multimap I$. Let A^n denote the n-th "tensor power" of an object A, $A^n = (A \otimes ...) \otimes A$, and f^n the n-th "tensor power" of a morphism f, $f^n = (f \otimes ...) \otimes f$.

To obtain the diagrams $D_2, ..., D_k, ...$ we notice that there exists a canonical morphism

$$h_k{:}(((a \multimap I)^k \multimap I) \multimap I) \to (a^k \multimap I).$$

The diagram

$$(D_2^0)\ f_2^0, g_2^0{:}(a^2)^*, a^{**}, a^{**} \rightrightarrows I$$

is obtained from the diagram (2) by substitution of a for b (in other words, by identification of variables a and b). The diagram D_2

$$(D_2)\ f_2, g_2{:}((a^*)^2)^{**}, a^{**}, a^{**} \to I$$

[6]For the derivations in $\mathbf{L(A)}$ it means that f, g are obtained from some derivations f_0, g_0 of the same balanced sequent. We know that $f \equiv g \Leftrightarrow f_0 \equiv g_0$. It is not necessarily true for $\sim$.

is obtained from D_2^0 by *cut* with h_2.

We define[7] the diagram D_m^0, $m \geq 2$, as the result of the substitution of a^{m-1} for b into diagram (2). The morphisms (derivations) obtained from f, g by this substitution are denoted by f_m^0, g_m^0.

The diagram D_m is obtained from D_m^0 by *cut* with h_m (f_m, g_m are the resulting derivations):

$$(D_m)\ f_m, g_m{:}((a^*)^m)^{**}, a^{**}, (a^{m-1})^{**} \overset{\rightarrow}{\rightarrow} I.$$

In order to obtain the models K_k we consider certain SMCCs of commutative semimodules over commutative semirings.

For all basic definitions concerning semirings and semimodules see [4]. Below we shall denote the "addition" of the semiring I by $+$ and "multiplication" by $*$. In case of a semimodule M we shall denote by $+_M$ its additive operation and $*_M$ the action of I on M; the index M will often be omitted.

Proposition 4.1 *I-semimodiles over a commutative semiring I and their homomorphisms form a SMCC with tensor product $\otimes$ and internal hom-functor $\multimap$ defined in usual way.*

We consider categories of semimodules over the semiring $I_n = \{0, ..., n\}$ with *max* as addition and with "bounded multiplication" $*$ as multiplication:

$p * q = p \cdot q$ if $p \cdot q < n$ and $p * q = n$ otherwise.

Obviously I_n is a commutative semiring.

We consider the semimodules M over I that have some additional properties.

Definition 4.2 (Top) *There is a "top" element $T_M \in M, T_M \neq 0_M$ such that for all $x \in M$, $x + T_M = T_M + x = T_M$, if $x \in M, x \neq 0_M$ then $n * x = T_M$ and if $0 \neq k \in I$ then $k * T_M = T_M$*

Obviously, I itself does satisfy these consditions if we take $T_I = n$. For I_s to be a semimodule over I, s must be not greater than n.

Definition 4.3 *Let us call an $I-$semimodule r-reducible for some $r \in I, 1 < r < n$ if for every $x \in M$ $r * x = T_M$.*

Example 4.4 *Let $M = I_2 = \{0, 1, 2\}$ considered as a semi-module over the semi-ring $I_4 = \{0, 1, 2, 3, 4\}$ (with the ordinary multiplication "bounded by 2" as the action). It satisfies* **(Top)** *with $T_M = 2$ and is 2-reducible. Of course M of this example is also 3- and 4-reducible.*

[7]The index m will correspond to the number of factors in tensor products.

We shall consider semimodules M such that

(r-red) M is $r-$reducible for some $r \in I, 1 < r < n$.

Theorem 4.5 *(1) All semi-modules over I satisfying* **(top)** *form an SMCC.*

(2)Let r be fixed, $1 < r < n$. All semi-modules M over I satisfying **(top)** *and* **(r-red)** *form an SMCC.*

Due to this theorem one may consider the SMCC generated by $\otimes$ and $\multimap$ from I and some given semimodule, for example $I = \{0, 1, 2, 3, 4\}$ and $M = \{0, 1, 2\}$ and be sure that all the objects of this category have a top element and are r-reducible.

Lemma 4.6 *Let M be an r-reducible semimodule over I and f:$M \to I$ (we may say also that $f \in M \multimap I$). Then for all $x \in M$ $f(x) \geq n/k$.*

Let h_m^- denote $a^m \otimes (a^*)^m \xrightarrow{\xi} (a^* \otimes a)^m \xrightarrow{e_{aI}^m} I^m \xrightarrow{b_I^{m-1}} I$. Consider the SMCC K of semimodules satisfying **(top)** and **(r-red)** over I. Now we can easily prove the following lemma.

Lemma 4.7 *Let n, r be as above, m such that $(n/r)^m \geq n$ and v an interpretation defined by $v(a) = M \in Ob(K)$. Then the morphism $|h_m^-|$:$(M^m) \otimes (M^*)^m \to I$ takes the value 0 if its argument is 0 and $T_I = n$ otherwise.*

Corollary 4.8 *Under the same conditions, the morphism $|h_m|$ takes only two values: 0 when its argument is 0 and $T_{(M^m)^*}$ otherwise (for every $M \in Ob(K)$).*

Lemma 4.9 *For all n, r, m as in lemma 4.7 and every interpretation v in the SMCC K of I-semimodules satisying* **top** *and* **r-red**, *the diagram $|D_m|_v$ is commutative. So it is commutative with respect to the relation $\sim_\forall$ (with respect to K).*

Let $2 \leq k$, $n = 3^k + 1, l = n/2$. Let $I = I_n$, $M = \{0, 1, ..., l\}$. Notice that M is $n/2$-reducible. Consider the SMCC K_k of all semimodules satisfying **top** and **l-red** generated by I and M.

Lemma 4.10 *Let the interpretation v be defined by $v(a) = M \in Ob(K_k)$. The diagrams $|D_2|_v, ..., |D_k|_v$ are non-commutative.*

Theorem 4.11 *There exist infinitely many different varieties of SMCC. Each of these varieties is defined by taking some (single) diagram D_m of the sequence above as a new axiom.*

To prove this theorem we use the fact that for any $k \geq 2$ there exists m (it is enough to take $m \geq log_2(3^k + 1)$) such that D_k cannot belong to the smallest equivalence relation generated by D_m. In other words, the commutativity of D_m does not imply the commutativity of D_k (by lemmas 4.9, 4.10).

5 Commutativity and dependency of diagrams

To verify the commutativity of a diagram in a model can be very difficult. Instead of verifying commutativity of diagrams case by case one may hope that if one diagram is commutative then the commutativity of another will follow.

Definition 5.1 *We shall say that the diagram $f', g' \colon \Gamma' \to B'$ depends on $f, g \colon \Gamma \to B$ if for every SMCC K the equivalence $f \sim_\forall g$ implies $f' \sim_\forall g'$.*

Theorem 5.2 *Let $\sim$ be the smallest substitutive equivalence relation on the derivations of $\mathbf{L(A)}$ such that $f \sim g$, $\sim$ contains $\equiv$, and is a congruence with respect to the rules of $\mathbf{L(A)}$. The diagram $f', g' \colon \Gamma' \to B'$ depends on $f, g \colon \Gamma \to B$ iff $f' \sim g'$.*

This theorem shows that syntactic methods may be used for verification of dependency of diagrams, since the standard construction for the smallest equivalence relation $\sim$ uses certain syntactic calculus with pairs of derivations as derivable objects.

By theorem 4.11 there exist infinitely many distinct equivalence relations $\sim$ on derivations of $\mathbf{L(A)}$. This fact shows the importance of the study of dependency of diagrams in SMCC and closed categories with weaker structure.

Remark 5.3 *The situation is different in the case of Cartesian Closed Categories because of the maximality theorem [1]: for any non-trivial axiom $f \sim g \colon \Gamma \to B$ the smallest equivalence relation defined by this axiom as in the theorem 5.2 above identifies all $f', g' \colon \Gamma' \to B'$. This is why the notion of dependency is useless in the case of CCC.*

The study of diagrams D in the previous section was done using model-theoretic methods.

We believe that the development of syntactical methods of verification of dependency of diagrams (e.g., via some auxilliary calculus with good properties) is an interesting and important problem. We use the opportunity to attract the attention of proof-theoretic and category-theoretical community to this interesting direction of research. At the moment we are able to suggest only some heuristics and partial solutions.

Taking into account the existence of efficient deciding algorithms for the commutativity of diagrams in free closed categories, the first step would be to verify whether the diagram $f', g' \colon \Gamma' \to B'$ is commutative in the free case. If it is non-commutative, the study of dependency may follow. In particular, one may find some "key" diagrams whose commutativity will imply the commutativity of others (cf. the axiomatization of equivalence relations by critical pairs mentioned above).

Other syntactical methods, for example, reduction to 2-sequents, that permit to establish that one diagram is commutative iff another diagram is, may be useful, probably, in combination with substitutions and congruence properties.

Example 5.4 *One may check that the diagram*

$$(a \multimap b) \multimap I, a \multimap b \otimes c, c \otimes d \multimap I, a' \multimap d \otimes b', (a' \multimap b') \multimap I \to I$$

is commutative iff the diagram

$$(a \multimap I) \multimap I, (b \multimap I) \multimap I, a \otimes b \multimap I \to I$$

is commutative. (Verification is left to the reader.)

6 Arbitrary natural transformations and proof-theoretic methods of their study.

In this section we shall consider one more step beyond purely syntactic free categories.

It turns out that in certain cases arbitrary natural transformations can be studied by proof-theoretic methods.

The results we discuss in this section were published in [13, 16].

Let us consider a sequent $\Gamma \to A$, $\Gamma = A_1, ..., A_n$. As in section 2, Γ may be seen as an abbreviation for $(...(A_1 \otimes ...) \otimes A_n) \otimes I$. Given a SMCC K, there is standard interpretation $\mathbf{L}(\mathbf{A}) \to NAT(K)$, $a \mapsto 1{:}K \to K$. Let $|\Gamma|, |A|$ be functors over K corresponding to Γ, A via this interpretation. So, given a sequent $\Gamma \to A$ and a category K the class of all natural transformations $|\Gamma| \to |A|$ can be studied.

The interpretations of derivations of $\Gamma \to A$ form a subset in this class.

Among possible problems one may mention, for example, the description of arbitrary natural transformations $|\Gamma| \to |A|$ in terms of canonical natural transformations (interpretations of derivations), or the problem of existence of non-trivial natural transformations.

Similar problems may be studied not only in the case of closed categories of various types, but also for other types of categories with structure (it may be necessary to consider other types of deductive systems to represent canonical natural transformations).

Before we consider proof-theoretic methods that can be used to study arbitrary natural transformation, let us outline main algebraic results, concerning arbitrary natural transformations.

Main assumption considered in [13, 16] was that the “tensor unit” I must be a generator in the category K.

Main algebraic results. In [13] was obtained a complete description of arbitrary natural transformations $|\Gamma| \to |A|$ in the case when K is *compact closed.* The description is very simple: every natural transformation $\phi{:}|\Gamma| \to |A|$ can be represented in the form $\phi = \theta * \phi_0$ where $\phi_0{:}|\Gamma| \to |A|$ is a canonical natural transformation, $\theta{:}I \to I$ in K, and $\theta * \phi_0$ means the following composition

$$|\Gamma| \xrightarrow{\phi_0} |A| \xrightarrow{b^{-1}} I \otimes |A| \xrightarrow{\theta\otimes 1_{|A|}} I \otimes |A| \xrightarrow{b} |A|.$$

In [16] similar ideas were applied to the case of symmetric monoidal category (without internal hom-functor $\multimap$) extended by biproduct and diagonal functor.(Diagonal functor permits identification of arguments in natural transformations, so the variables are not necessarily distinct, and the general case cannot be reduced to the case of balanced sequent.) Here again a complete description of natural transformations was obtained. In general, due to the presence of biproducts and diagonal functor they are obtained from canonical natural transformations using matrices of parameters $\theta{:}I \to I$.

Main technical lemma. An important role was played by the following lemma. (Cf. [13].)

Lemma 6.1 *Let K be an SMCC, and I be a generator in K. Let $\phi{:}|(A \multimap p) \otimes (p \multimap B)| \to |C|$ be a natural transformation in K (the variable p has only two occurrences shown explicitely). Then ϕ is equal to the following composition:*

$$|(A \multimap p) \otimes (p \multimap B)| \xrightarrow{|Comp|} |A \multimap B| \xrightarrow{\phi_0} |C|$$

where $Comp{:}(A \multimap p) \otimes (p \multimap B) \to A \multimap B$ is a standard derivation representing composition, and ϕ_0 is some natural transformation in K that has one argument less than ϕ.

Remark 6.2 *Obviously, using this lemma, commutativity of $\otimes$ and adjunctions in K one may decompose into a canonical part and a simpler natural transformation ϕ_0 any natural transformation $\phi{:}|\Gamma| \to |C|$ when $|\Gamma|$ contains members of the form $(A \multimap p), (p \multimap B)$ in any position.*

Combining proof-theoretic and algebraic methods. How these results can be used in combination with proof-theoretic methods can be illustrated by the use of reduction to 2-sequents.

Theorem 2.7 can be applied to arbitrary natural transformations as well (this version of the theorem can be found in [13]). That is, given any SMCC K and two natural transformations $\phi_1, \phi_2{:}|\Gamma| \to |C|$ there exist natural transformations $\phi_1', \phi_2'{:}|\Gamma'| \to |p|$ where $|\Gamma'| \to |p|$ is a pure 2-sequent such that $\phi_1 = \phi_2 \Leftrightarrow \phi_1' = \phi_2'$ Moreover, ϕ_1', ϕ_2' can be obtained from ϕ_1, ϕ_2 by composition with canonical natural transformations, and ϕ_1, ϕ_2 can be obtained from ϕ_1', ϕ_2' by substitutions (composition with functors) and composition with canonical natural transformations.

Using this fact and lemma 6.1 we will see that the lemma will permit a full description of natural transformations if $\Gamma \to p$ is a balanced 2-sequent and Γ contains only members of the form $a \multimap b$ or single variables. (All the variables can be eliminated one by one.) In general the process will be blocked if, for example, we will apply the lemma to the members like $(a \multimap b) \multimap p, p \multimap c \otimes d$.

As we see, transformations of diagrams and the notion of dependency of diagrams can be useful even in the case of arbitrary natural transformations.

7 Conclusion

In our presentation we tried to explain why non-free categories may be interesting to proof-theorists and also show the efficiency of proof-theoretic methods in the study of non-free categories.

The examples considered above demonstrate that in general the study of the structure of a free category is in no way easy (cf. the triple-dual conjecture and the problem of full coherence).

The existence of infinitely many different varieties of closed categories shows that to describe each time a free categorical structure may be impractical.

On the other hand, proof-theoretic methods will allow to study interesting concrete dependencies (dependent diagrams) "locally".

The description of arbitrary natural transformations considered in the previous section may be viewed also as a way to represent natural transformations in a complex category using natural transformations in simpler categories (for example, using natural transformations of free category and endomorphisms of I). This permits to obtain new coherence theorems.

We believe that this direction of research will provide new and promising applications of proof theory to categorical algebra, and also be a source of new insights and more flexible categorical semantics for proof theory.

We would like to express our thanks to Kosta Dosen and Zoran Petric, who helped to clarify many points and improve the presentation of our work, and to Nikolai Vasilyev for fruitful discussions concerning its algebraic aspects.

References

[1] K. Dosen and Z. Petric. The maximality of the typed lambda calculus and of cartesian closed catgeories. Belgrade, *Publications de l'Institut Mathématique*, Nouvelle Série, tome 68(82) (2000),pp.1-19.

[2] S. Eilenberg and G. M. Kelly. A generalization of the functorial calculus.- *J.of Algebra*, 1966.

[3] G.-Y. Girard, Y. Lafont. Linear logic and lazy computation. In: Proc.TAPSOFT 87 (Pisa), v.2, p.52-66, LNCS v.250 , 1987.

[4] J. Golan. Semirings and their applications. Kluwer Acad. Publishers, Dordrecht, 1999.

[5] G.M. Kelly and S. Mac Lane. Coherence in Closed Categories. *Journal of Pure and Applied Algebra*, 1(1):97–140, 1971.

[6] G.M.Kelly. A cut-elimination theorem. *Lecture Notes in Mathematics*, 281 (1972), pp 196-213.

[7] J. Lambek. Deductive Systems and Categories. I. *Math. Systems Theory*, 2, 287-318, 1968.

[8] J. Lambek. Deductive Systems and Categories II. Lect. Notes in Math., v.86, Springer, 1969, pp. 76-122.

[9] J. Lambek. Deductive Systems and Categories III. Lect. Notes in Math., v.274, Springer, 1972, pp. 57-82.

[10] L. Mehats, S. Soloviev. Coherence in SMCCs and equivalences on derivations in IMLL with unit. *Annals of Pure and Applied Logic,* v.147, 3, p. 127-179, august 2007.

[11] G.E. Mints. Closed categories and Proof Theory. *Journal of Soviet Mathematics*, 15, 45–62, 1981.

[12] G. E. Mints. Category theory and proof theory (in Russian), in: *Aktualnye voprosy logiki i metodologii nauki*, Naukova Dumka, Kiev, 1980, 252-278. (English translation, with permuted title, in: G.E. Mints. Selected Papers in Proof Theory, Bibliopolis, Naples, 1992.)

[13] S. Soloviev. On natural transformations of distinguished functors and their superpositions in certain closed categories. *Journal of Pure and Applied Algebra*, 47:181-204, 1987.

[14] S. Soloviev. On the conditions of full coherence in closed categories. *Journal of Pure and Applied Algebra*, 69:301-329, 1990.

[15] S. Soloviev. Proof of a conjecture of S. Mac Lane. *Annals of Pure and Applied Logic,* 90 (1997), pp.101-162.

[16] R. Cockett, M. Hyland, S. Soloviev. Natural transformations between tensor powers in the presence of direct sums. *Rapport de Recherche*, 01-12-R, IRIT, Jul. 2001.

[17] S. Soloviev, V. Orevkov. On categorical equivalence of Gentzen-style derivations in IMLL. *Theoretical Comp. Science*, 303 (2003), pp. 245-260.

[18] R. Voreadou. Coherence and non-commutative diagrams in closed categories. *Memoirs of the AMS*, v. 9, issue 1, N 182, Jan. 1977.

Investigations into Forest Proofs
An ongoing research project

Willem Heijltjes

LFCS, Edinburgh

A direction in classical logic that has of late received much attention—and has given rise to many new ideas—is the search for invariants, or semantics, of classical proofs; be it to reduce dependence on syntax, to characterise a notion of proof identity or to capture computational content. Approaches include, among others, proof nets [3, 12, 9], general graphs [8], 'atomic flows' [4] and 'proof profiles' [7].

This note reports on the progress of a programme investigating one such abstraction, proof forests [5], for classical proofs of prenex formulae. Such forests were originally introduced by Miller [11] as a structurally minimal representation of classical proofs. They can be seen as a graphical presentation of Buss' 'Herbrand expansion proofs' [1] and are closely related to proof nets. These forests also provide a natural representation of two-player backtracking games in the style of Coquand [2].

The minimal structure of forest proofs is characterised by the absence of syntactic features, such as permutable rules, that force inessential choices to be made in the design of a proof—informally known as 'bureaucracy'. A second distinctive feature of the forests is a form of implicit sharing unavailable in sequent calculus. Unlike the latter, where each subproof is entirely self-contained, sub-forests of a proof forest may overlap; in fact, this is the default setting, and the layout of the forest specifies only where this is not the case.

Because they form a natural, minimal representation of classical proofs, the possibility of composition of proof forests, by performing cut-elimination, is a natural direction for investigation. This route has recently been taken, independently, by Richard McKinley and the author.

From the game-theoretic perspective, cut-elimination amounts to the composition of strategies for asymmetric backtracking games. In contrast with Coquand's games, the strategies represented by proof forests do not describe a linear sequence of moves. Instead, a strategy gives a range of possible moves at each position, introducing an element of non-determinism. This in particular is a defining factor in the design of the reduction steps.

In the graphical representation cut-elimination takes the form of graph-rewriting steps that make use of the available structure in a natural way and, superficially, have much in common with their sequent calculus counterparts. However, there is an interesting interplay between the reduction steps and the implicit sharing, and reductions in the two formalisms display intriguingly different behaviour. This is most clearly witnessed by the fact that there exist reductions that take forests from inside to outside the image of sequent calculus translations—and also in the other direction.

The aim of the current project is to understand and characterise classical proof forests and their dynamics in the context of their minimal structure and game-theoretic reading. The intention of the author is to present at the workshop a summary of the programme, from the basic ideas to the state of the art at the time of presentation. The present situation of the project is as follows:

- The notion of a 'classical proof forest with cut' has been rigourously defined and its semantic soundness and completeness proved. The precise formulation generalises that by Miller in [11], who considered only cut-free forests. This generalisation is needed to accommodate the intermediate steps in a reduction. The connections with Herbrand proofs [1] and strategies for backtracking games have been established, and translations to and from sequent proofs are available. This will appear in [6].
- A natural procedure for cut-reduction on proof forests was developed, independently, by McKinley and the author, in the belief that it would be strongly normalising. However, an example forest exhibiting an infinite reduction trace was discovered by the author and presented in [5]. It remains an open question whether the reduction relation is weakly normalising.
- A modification of the original reduction procedure has been shown to be weakly normalising. The modification hinges on a notion of 'conflict', motivated by concurrency theory and naturally applicable to the game-theoretic interpretation of forests, which allows 'self-conflicting' nodes to be pruned from the graph. This will appear in [6].
- It is conjectured that the modified reduction relation is in fact strongly normalising. Again, this is currently an open question.
- Neither of the reduction relations is confluent. (It is known that confluence is not easily obtained for classical proof.)
- A generalised notion of 'strong' conflict has been introduced, using a controlled form of transitivity, to give rise to stronger versions of the correctness and validity criteria for forest proofs with cut. It has been shown that forests translated from sequent proofs obey these stronger criteria. Moreover, this characterisation seems robust, no matter which particular formulation of sequent calculus is taken (e.g. additive versus multiplicative).

The project outlined above has proceeded in parallel with similar investigations by McKinley, who independently discovered a syntactic version of forest proofs. McKinley has formulated reductions as a process calculus and explored the possibility of adding axiom links to the forests, much like those found in proof nets, to obtain a closer correspondence to sequent calculus proofs and reductions, for a subclass of proof forests.

Work by the author, in contrast, has focused on the combinatorics of proof reduction in forest proofs themselves, irrespective of sequentializability. This is motivated by the idea that the reading of normalisation in proof forests as the composition of non-deterministic strategies for backtracking games is a much more natural interpretation than the correspondence with sequent calculus can provide.

Acknowledgements: many thanks go out to Alex Simpson for encouragement and guidance on this project, and to Richard McKinley for helpful discussions.

References

1. Buss, S.R.: On Herbrand's Theorem. LNCS, vol. 960, pp. 195–209. Springer-Verlag (1995)
2. Coquand, T.: A Semantics of Evidence for Classical Arithmetic. JSL, vol. 60 (1), pp. 325–337 (1995)
3. Jean-Yves Girard. A New Constructive Logic: Classical Logic. Mathematical Structures in Computer Science, vol. 1 (3), pp. 255–296 (1991)
4. Guglielmi, A. and Gundersen, T.: Normalisation Control in Deep Inference via Atomic Flows. Logical Methods in Computer Science, vol 4(1:9), pp. 1–36 (2008)
5. Heijltjes, W.B.: Proof Forests with Cut Based on Herbrand's Theorem. Presented at CL&C'08, Reykjavik, and available at http://homepages.inf.ed.ac.uk/s0792892/research.html (2008)
6. Heijltjes, W.B.: Classical Proof Forests. In preparation, to appear at http://homepages.inf.ed.ac.uk/s0792892/research.html (2009)
7. Hetzl, S. and Leitsch, A.: Proof Transformations and Structural Invariance. LNCS, vol. 4460, pp. 201–230. Springer (2007)
8. Hughes, D.J.D.: Proofs without syntax. Annals of Mathematics, vol 164(3), pp. 1065–1076 (2006)
9. Lamarche, F. and Straßburger, L.: Naming Proofs in Classical Propositional Logic. LNCS, vol. 3461, pp. 246–261. Springer (2005)
10. McKinley, R.: On Herbrand's Theorem and Cut-Elimination. In preparation (2009)
11. Miller, D.A.: A Compact Representation of Proofs. Studia Logica, vol. 46(4), pp. 347–370 (1987)
12. Robinson, E.P.: Proof Nets for Classical Logic. Journal of Logic and Computation, vol. 13 (5), pp. 777–797 (2003)

The alpha-epsilon calculus

Richard McKinley[*]

Laboratoire PPS
Université Paris Diderot – Paris 7
75205 PARIS Cedex 31
richard.mckinley@pps.jussieu.fr

Abstract. This paper is a brief introduction to the $\alpha\varepsilon$-calculus – a calculus of communication and duplication inspired by the structure of the *classical* quantifiers. We will summarize the results of a paper in preparation on connections between extensions of the calculus, sequent systems/proof nets for classical logic, and Herbrand's theorem.

1 Introduction

A striking feature of the connection between lambda calculus and intuitionistic logic is that the same binder represents the both binding of assumptions (in an implication) and the binding of individuals (a free variable in a universal quantification), and that furthermore, within second-order logic the remaining connectives and the units can be captured using a combination of of those two binding mechanisms. This has particular appeal for the connectives ($\exists$, $\vee$) whose natural deduction rules are not well behaved. For this reason attention is typically focused only on the $\implies, \forall_2$ fragment of the logic, with the researcher being justly confident that if everything works out for those two connectives, the rest will easily follow. This minimality of presentation also means that computational interpretations of intuitionistic logic are parsimonious, requiring just abstraction and application as primitives. In this paper I will describe some work towards finding a similar minimal treatment of classical logic.

We start from the observation that presenting classical logic via implication forces us to model certain highly symmetric features of the logic in an asymmetric fashion. The propositional connectives of classical logic can be presented in two (essentially different) ways: positively or negatively (the negative versions are those with invertible rules on the right-hand side of the turnstile, with the positive versions being invertible on the left). In the presence of primitives for classical negation lambda calculus is an effectful language, and as such we must restrict the reduction strategy to maintain confluence; this restriction typically forces either the positive or the negative interpretation of the propositional connectives, but not both. Of course, we might instead consider the full

[*] Work supported by the Swiss National Science Foundation grant "Algebraic and Logical Aspects of Knowledge Processing." and the ANR project "INFER: Theory and Application of Deep Inference

(non-confluent) reduction system, but this is rather alien to the spirit of lambda calculus and the functional paradigm. Instead, we choose to focus on the structure of the classical quantifiers, whose polarity is fixed (existential is positive, universal is negative), and on the paradigm of *communicating processes.*

The intuition of our approach is not new: to interpret proofs in a multiple succedent calculus not as functions but as processes, with cut being a form of communication between processes. This appeared first in a series of presentation by Abramsky's [1], which proposed to interpret proofs in linear logic as processes in the π-calculus. Details appear in Bellin and Scott [3]; like many nice ideas in linear logic, it works well for unit-free multiplicative linear logic, but not so well for additives; there is a mismatch between the non-deterministic $+$ of the π-calculus and the corresponding non-determinism in a proof with a conclusion $A\&B$. Our alternative approach in this paper is to derive a process-like language from proof theory; at the expense of producing something that perhaps looks strange to a process theorist, we hope give a calculus which corresponds well to logic. Communication in this calculus will be closely modelled on cut-elimination in the sequent calculus (a related game-semantic approach to cut-elimination has been studied by Coquand [4]) The test of this approach (yet to be performed) will be the computational expressiveness of the resulting calculus.

The $\alpha\varepsilon$ calculus arose from studies of Herbrand's theorem, and in particular a sequent systems related to Herbrand's theorem in which contraction is restricted to formulae whose leading connective is an existential quantifier. Contraction can be limited to existentials because the rule for the universal rule is *invertible.* This system can be seen in Figure 1. The natural calculus of proof nets for this sequent calculus (which we will see later) can be written as a language involving three combinators, which we use to decorate the sequent tree.

$$\frac{}{\vdash P_1, \ldots, P_n}\ Taut$$

$$\frac{\vdash \Gamma, A[x := a]}{\vdash \Gamma, \forall x.A}\ \forall\mathrm{R} \qquad \frac{\vdash \Gamma, A[x := M]}{\vdash \Gamma, \exists x.A}\ \exists\mathrm{R}$$

$$\frac{\vdash \Gamma, \exists x.A, \exists x.A}{\vdash \Gamma, \exists x.A}\ C^{\exists}$$

Fig. 1. A sequent calculus for (prenex) Herbrand's theorem

The calculus contains one binder

$$\alpha[x]$$

whose role is to receive a witness (and bind it to x). This binder represents the binding of a variable in a universal quantification. The calculus also contains an *instantiator*

$$\varepsilon[M]$$

which can supply a term M (a piece of data) to an alpha, and represents a single witness to an existential quantification. These constructors may be used as prefixes (corresponding to quantifier prefixes) so for example the string

$$\alpha[x].\varepsilon[M]$$

receives a piece of data into x and then outputs a piece of data $[M]$. These prefixes can be thought of as specifying protocols.

So far, this is reminiscent of the $x()$ and $\bar{x}\langle\rangle$ of Milner's π-calculus [12], but there are crucial differences. First, $\alpha\varepsilon$ is a calculus without mobility: communication between an α and a ε may only happen across a *cut*, which we write as

$$t \bowtie s.$$

Second, while a π-calculus process is a tree, with explicit scoping of the $x()$ binder, an $\alpha\varepsilon$-structure is a linked forest, and the scope of a binder $\alpha[x]$ is implicit. The use of concepts from proof-net theory allow us to recover some sequential structure, and in particular the notion of *kingdom* will be very important. Finally, we will allow data to be non-deterministic, in the sense that data may take the form

$$\{\varepsilon[M_1].t_1, \ldots \varepsilon[M_n].t_n\}$$

corresponding to the contraction of existential formulae. Given the polarity of the quantifiers underlying α and ε, , our slogan will be "Negative is demand, Positive is *non-deterministic* data". In order for a single input (α) to receive these multiple outputs, there is a reduction which *duplicates* the input, along with a substructure containing that input. Thus $\alpha\varepsilon$ is a calculus of *communication and duplication*.

Since we aim, eventually, at a Turing complete computational calculus, we should look beyond logic, just as we consider *untyped* lambda terms. In this setting the equivalent notion is a (proof-)structure; a graph which does not necessarily satisfy the usual proof-net correctness criterion. Clearly reduction on such structures is not strongly normalizing: we will give examples of such pathological structures.

This paper (necessarily) comprises mostly definitions and examples: those lemmata and theorems that do appear will mostly remain unproved (in most cases the proofs are standard). This is an area of research where most of the interesting questions remain open, and this paper serves as an introduction to that area.

2 The $\alpha\varepsilon$ calculus

The principal objects of the $\alpha\varepsilon$ calculus are *structures*, made up of several ports and several cuts, which are built up as follows:

Definition 2.1. *Let $\mathcal{A}$ be a countable set of variables, and let $\mathcal{M}$ be a term language building terms from a set $\mathcal{F}$ of function symbols and members of $\mathcal{A}$. The set of ports $\mathcal{P}$ over $\mathcal{M}$ is given by*

$$\mathcal{P} := \alpha[\mathcal{A}].\mathcal{P} \mid \{\varepsilon[\mathcal{M}_1].\mathcal{P}_1, \dots, \varepsilon[\mathcal{M}_n].\mathcal{P}_n\}$$

where $\{\dots\}$ denotes finite multiset. We require (for now) that these multisets be nonempty. A cut has the form

$$\mathcal{P}_1 \bowtie_S \mathcal{P}_2$$

where S is a finite set of variables from $\mathcal{A}$ (The set S allows a cut to depend on variables bound by an alpha, and are necessary to calculate the scope of an alpha binder).

We will use letters a, b, c, x, y, z to denote variables, M, N to denote terms in the language $\mathcal{M}$ and t, s to denote ports.

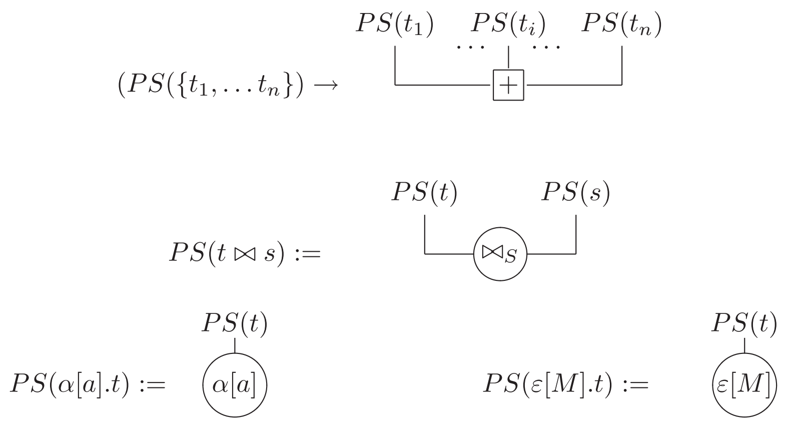


Fig. 2. $\alpha\varepsilon$ ports/cuts as trees

Since α is a binder with non-obvious scope, we will require that each instance of α binds a unique eigenvariable. A multiset of ports and cuts will be said to *satisfy the eigenvariable condition* if this is the case.

Definition 2.2. *Given a multiset of ports F satisfying the eigenvariable condition, consider the underlying forest of F as a directed graph where edges point towards the root. Now add to that graph two new sets of edges, connecting an alpha to its* direct dependents*:*

(a) Edges from each $\varepsilon[M]$ to each $\alpha[a]$ such that a is a variable contained in M.
(b) Edges from $t \bowtie_S s$ to each $\alpha[a]$ such that a is a variable in S.

We call this graph the dependency graph *of F. The dependency successors of a node X are the nodes Y for which there is an edge from Y to X in the dependency graph of F.*

Definition 2.3. *A multiset F of ports satisfying the eigenvariable condition is an $\alpha\varepsilon$* structure *if its dependency is directed acyclic.*

Writing a structure F as a forest (Figure 2 gives a function PS from ports and communications to labelled trees), the connection with proof-nets becomes evident. These are proof nets of a rather curious kind, however: in particular, there is no axiom as such. Instead the edges added to form the dependency graph can be seen as *quantifier jumps* as first described in Girard [6]. Playing the role of axioms will be a special class of such jumps:

Definition 2.4. *Let F be an $\alpha\varepsilon$-structure. An* axiom node *is a node $\alpha[\hat{x}]$ with no child and exactly one direct dependent $\varepsilon[M]$. We will denote the variables bound in an axiom node with a hat, as written above, and refer to the edge between an axiom node and its dependent as an axiom link.*

Continuing the analogy with proof-nets, we can consider *correctness* for our structures. We turn to the well-known notion of *switching*, due to Danos and Regnier, in the form used by Bellin and van de Wiele for first-order MLL. The multi-set and α nodes of a structure are switched, and the ε and $\bowtie$ nodes are unswitched:

Definition 2.5. *A* switching *σ for a structure F is a choice of on dependency-successor for each α-node and one member of each multiset node. The* switching graph *F_σ is an undirected graph obtained from the graph of F by erasing each incoming edge of an α or multiset node except that chosen by the switching, and then forgetting the direction of arrows.*

Remark 2.1. It is here that we see the reason for isolating axiom nodes as a special case of alpha nodes: an axiom node is not really switched, since there is only one choice for the switching.

Definition 2.6. *A structure F is a* net *if, for every switching σ, F_σ is connected and acyclic. It is a* mixnet *if, for every switching, F_σ is acyclic.*

The nets are structures which can be made sequential, in the sense that they can be built from structures containing no cuts, by applying cuts (we will not prove this here, but it follows from standard proof-net theory). A mixnet can similarly be built from cut-free structures using cuts and a *mix* rule which simply puts structures side-by-side.

We will use uppercase Greek letters Γ, Δ to denote structures (presaging the use of structures to annotate sequents). We use the symbol $+$ to denote multiset union. We will work modulo alpha-equivalence (which here is the renaming of α-bound variables).

2.1 Reductions

We divide the reductions (computation steps) of $\alpha\varepsilon$ into three groups. The first, called "Axiom" reductions, is given in Figure 3. These reductions serve the same purpose as "cut against (non-atomic) axiom" reductions in sequent calculus, and concern the axiom nodes. The next two reductions are the communication

$$\begin{array}{rcl}
\Gamma,\ \alpha[\hat{x}] \bowtie_S t,\ \{\varepsilon[\hat{x}]\} \bowtie_S s & \leadsto & \Gamma,\ t \bowtie_S s \\
\Gamma,\ \alpha[\hat{x}] \bowtie_S t,\quad \{\varepsilon[\hat{x}]\} & \leadsto & \Gamma,\ s \\
\Gamma, \{\varepsilon[\hat{x}]\} \bowtie_S t,\quad \alpha[\hat{x}] & \leadsto & \Gamma, t
\end{array}$$

Fig. 3. "Axiom" reductions

of a single piece of data, which in sequent calculus is the logical reduction of a cut involving quantifiers. To define these reductions, we will need to define the substitution of a term for a variable:

Definition 2.7. *Define the operation* $[a := M]$ *(substitution) on structures in which the name* a *is not bound as follows:*

$$\begin{aligned}
(t_1, \dots, t_n)[a := M] &= (t_1[a := M], \dots, t_n[a := M]) \\
(\alpha[d].t)[a := M] &= \alpha[d](t[a := M]) \\
(\varepsilon[d]t)[a := M]) &= \begin{cases} \varepsilon[d](t[a := M]) & d \neq a \\ \varepsilon[b](t[a := M]) & d = a \end{cases} \\
(\{t_1, t_n\})[a := M] &= \{t_1[a := M], \dots, t_n[a := M]\} \\
(t \bowtie_S s)[a := b] &= \begin{cases} (t[a := M] \bowtie_S s[a := M]) & a \notin S \\ (t[a := M] \bowtie_{((S \setminus a) \cup fv(M))} s[a := M]) & a \in S \end{cases}
\end{aligned}$$

The "Communication" reductions can be found in Figure 4.

The final reduction of $\alpha\varepsilon$ we will consider is *duplication.* This reduction is the most distinctive to the alpha-epsilon calculus. In order to reduce certain structures, for example

$$\Gamma, \alpha[x] \bowtie \{\varepsilon[M_1], \varepsilon[M_2]\}$$

we need to duplicate x. Analogous to the cut against contraction in sequent calculus, we will duplicate a substructure containing $\alpha[x]$. That substructure must clearly, at the very least, be closed under dependency and under axiom links. it should also not include the cut itself.

$$\Gamma, \alpha[x] \bowtie_S \{\varepsilon[M]\} \rightsquigarrow \Gamma[x := M]$$

$$\Gamma, \alpha[x].t \bowtie_S \{\varepsilon[M].s\} \rightsquigarrow (\Gamma, t \bowtie_{(S \cup fv(M))} s)[x := M]$$

Fig. 4. "Communication" reductions

Definition 2.8. *A subset G of the nodes of a structure is a* substructure *if it is closed under dependency, and if an axiom node is in G if and only if its unique direct dependent is in G (axiom closed).*

Essentially, the duplication rule of the $\alpha\varepsilon$-calculus duplicates a substructure, mimicking the pushing of a cut above a contraction in the sequent calculus. However, the way structures are built necessitates that only subtrees rooted with a ε or a $\bowtie$ be duplicated In what follows, fix a particular substructure K of t. Write K_ε for largest dependency-closed subset of K in with no dependency-maximal alpha or multiset nodes.

In the process of duplication we will need to use the following renaming functions:

Definition 2.9. *We define two renaming functions T_0 and T_1 which rename certain bound variables, as follows:*

$$T_i(b) = \begin{cases} b & \alpha[b] \notin K_\varepsilon \\ b_X & \alpha[b] \in K_\varepsilon \end{cases}$$

Define T_i pointwise on sequences and subsets of variables. On terms, T_i is defined as follows:

$$T_i(f(\bar{a}) = f(T_i(\bar{a}))$$

and on ports and cuts as follows:

$$\begin{aligned} T_i(t \bowtie_S s) &= T_i(t) \bowtie_{T_i(S)} T_i(s) \\ T_i(\alpha[a].t) &= \alpha[T_i(a)].T_i(t) \\ T_i(\{t_1, \dots t_n\} &= \{T_i(t_1), \dots, T_i(t_n)\} \\ T_i(\varepsilon[M]t) &= \varepsilon[T_i(M)]T_i(t) \end{aligned}$$

Definition 2.10. *Let $F = \Gamma, \alpha[x].t \bowtie_S (s_1 + s_2)$ be a structure, and K a substructure of F not containing the displayed cut. Define a function D_x^K on sub-*

ports of F as follows:

$$D^K_x(t \bowtie_S s) = \begin{cases} D^K_x(t) \bowtie_S D^K_x(s) & t \bowtie_S s \notin K_\varepsilon \\ T_0(t \bowtie_S s), T_1(t \bowtie_S s) & t \bowtie_S s \in K_\varepsilon \end{cases} \quad D^K_x(\alpha[a].t) \quad = \alpha[a].D^K_x(t)$$

$$D^K_x(\{t_1, \dots t_n\} = \{D^K_x(t_1), \dots, \Delta^K_x(t_n)\}$$

$$D^K_x(\varepsilon[M].t) = \begin{cases} \varepsilon[M].D^K_x(t) & \varepsilon[M] \notin K_\varepsilon \\ T_0(\varepsilon[M].t), T_1(\varepsilon[m].t) & \varepsilon[M] \in K_\varepsilon \end{cases}$$

Let $D^K_x(\Gamma)$ be defined pointwise on its members.
Then F duplication-reduces to

$$D^K_x(\Gamma), \alpha[x_0].T_0(t) \bowtie_S D^K_x(s_1), \alpha[x_1].T_1(t) \bowtie_S D^K_x(s_2)$$

2.2 What to duplicate?

The duplication rule introduced above is intentionally very general – this is because it is not entirely clear how much of a structure one should duplicate. In a sequential system this would be clear: it is the smeared out, desequentialized nature of $\alpha\varepsilon$ that causes this problem.

One could imagine a simple scheme of duplication reduction in which we simply duplicate the ports above the minimal dependents of $\alpha[x]$. This was, indeed, the reduction in an early stage of research, but it falls foul of logic: the structures we use to annotate sequent proofs are always nets, but this simpler reduction fails to respect the correctness criterion, in particular it can produce substructures of the form

$$\alpha[x] \bowtie \{\varepsilon[x], \varepsilon[y]\}.$$

Logically, this is a cut over which there is communication: any structure including this cut clearly does not satisfy the correctness criterion. A parallel line of research to this one by Heijltjes [7] (in the typed case) takes the alternative approach of simply removing these "conflicting" branches in the proof, since anything which can be proved with them can be proved without them: there we find a reduction of the form:

$$\alpha[x] \bowtie \{\varepsilon[x], \varepsilon[y]\} \rightsquigarrow \alpha[x] \bowtie \{\varepsilon[y]\}.$$

By contrast, let us examine the reductions of this structure in our system:

$$\begin{aligned} \alpha[x] \bowtie \{\varepsilon[x], \varepsilon[y]\} &\rightsquigarrow \alpha[x_0] \bowtie \{\varepsilon[x_0], \varepsilon[x_1]\}, \qquad \alpha[x_0] \bowtie \{\varepsilon[y]\} \\ &\rightsquigarrow \alpha[x_1] \bowtie \{\varepsilon[x_1], \varepsilon[y]\} \end{aligned}$$

which, of course, is alpha equivalent to the original structure.

The duplication rule preserves correctness if the substructure duplicated is a subnet, in the following sense:

Definition 2.11. *A* subnet *of a net F is a substructure G of F such that, for every switching σ of F, the switching graph F_σ restricted to the members of G is connected.*

This definition is slightly curious (since a subnet is not a net — it might have dependency-maximal epsilon nodes) but it is the correct generalization of subnet to this setting.

As in the usual theory of proof nets, subnets having a a node X as a door are closed under intersection and union, and for each node there is at least one subnet having that node as a door (except in the case where that node is the unique dependent of an axiom node). Because of this, there exists for every node X two special subnets: the *empire* (the largest subnet containing X as a maximal node) and the *kingdom* (the smallest subnet containing X as a maximal node). The kingdom will be of particular importance for us here:

Definition 2.12 (Minimal duplication). *The* minimal duplication *system for $\alpha\varepsilon$ nets consists of the axiom reductions, the communication reductions, and the duplication reduction where K is the kingdom of $\alpha[x]$.*

At first sight, one might suppose that the computation of the kingdom of a formula is unacceptably complicated operation: too complex to be primitive. This is perhaps so, but it is worth noting that, at least complexity-wise, it is no worse than beta-reduction (the calculation of the kingdom of a node is at worst quadratic in the size of the structure, and the duplication itself is of course linear). One could imagine an "explicit duplication" version of $\alpha\varepsilon$, but that is of course some way off.

2.3 Results and Conjectures

We have seen above that reduction in $\alpha\varepsilon$ is not weakly normalizing. Nor is it confluent:

Proposition 2.1. *Reduction in $\alpha\varepsilon$ is non-confluent.*

This result is due to Heijltjes [8]. Note that this non-confluence cannot be a result of a contraction/contraction or weakening/weakening style counterexample, since these are excluded by the calculus. Instead, the critical choice is in the order in which cuts are reduced.

We will see below a sketch of a proof that reduction in a typed $\alpha\varepsilon$ is weakly normalizing. Of course, our goal is a strongly normalizing calculus in the typed case, and indeed, in the case of nets:

Conjecture 2.1. Minimal reduction of $\alpha\varepsilon$ nets is strongly normalizing.

We look also for subsystems of $\alpha\varepsilon$ that are confluent. A possible contender is

Conjecture 2.2 (Call-by-Value $\alpha\varepsilon$?). Let F be a structure. A *value instantiator* in F is a term $\varepsilon[M].t$ where every variable occurring in M is either free in F or is bound by an α which occurs outside a cut. The restriction of $\alpha\varepsilon$ where we may only communication-reduce cuts involving a value instantiator is confluent.

3 Typed $\alpha\varepsilon$ and Herbrand's theorem

As an antidote to the speculation in the previous section, I summarize here some results from [10] on assigning (extensions of) $\alpha\varepsilon$ nets to sequent derivations. The sequent system appearing here is *polarized*: a formula is either positive or negative, and contraction is only allowed on positive formulae (contraction on negative formulae is admissible, since the rules for negative formulae are invertible). This system was the starting point of this project: it concerns a generalization of Herbrand expansions such that they may be considered proof theoretically, with proofs composable via cut and with a cut-elimination theorem.

We consider a system assigning ports to the sequent calculus in Figure 1. The completeness of this system is a statement of Herbrand's theorem: A formula A in first-order classical logic is provable if and only if it has a cut-free proof in this system, if and only if there is a tautology formed by instantiating an expanded form of A.

The philosophy of the port-assignment system is that each existential generalization rule $\exists$R creates a witness to its type, and that the contraction rule $C^{\exists}$ collects together such witnesses.

To assign ports to this system, we will need an assignment of ports to the tautology rules. To do this, we consider an extended calculus of ports in which, for any set S of natural numbers, the figure $[S]$ is a port. Each natural number will then represent an instance of the tautology rule. We consider types from classical predicate logic built over a signature of basic relation symbols $\mathcal{R}$ and a set of basic function symbols $\mathcal{F}$ each of which has a given arity. Given a set $X = x, y, z \ldots$ of variables, each predicate symbol r of arity n defines, for every n-tuple of variables $(x_1, \ldots x_n)$ an atomic predicate $r(x_1, \ldots x_n)$. We will use the symbol M to denote a term over these variables constructed using the symbols from $\mathcal{F}$.

We consider formulae in negation normal form, as we will use a one-sided sequent-calculus. A quantifier free formula (QFF) is an element generated by the following grammar, where p ranges over the atomic predicates:

$$P := \ p \,|\, \bar{p} \,|\, (P \vee P) \,|\, (P \wedge P)$$

Negation on general formulae is a derived operation:

$$\bar{\bar{p}} := p, \quad \overline{(P \vee Q)} := (\bar{P} \wedge \bar{Q}), \quad \overline{P \wedge Q} := \bar{P} \vee \bar{Q}.$$

Prenex formulae are defined by the following grammar, where x ranges over the variables in X:

$$A := P \mid \exists x.A \mid \forall x.A$$

with the usual definitions for negation:

$$\overline{\forall x.A} := \exists x.\bar{A}, \quad \overline{\exists x.A} := \forall x.\bar{A}$$

Ports and cuts are typed according to Figure 5. The assignment of structures to sequent proofs in this fragment of classical predicate logic is given in Figure

6. Without cuts, these typed structures are a presentation of Miller's expansion-tree proofs for first-order prenex classical logic [11], and in this sense our system is an extension of expansion-tree proofs with cut and cut-reduction.

$$\frac{}{[S] : P}$$

$$\frac{t : A[x := a]}{\alpha[a].t : \forall x.A} \qquad \frac{t : A[x := a]}{\{\varepsilon[a].t\} : \exists x.A}$$

$$\frac{t : \exists x.A \quad s : \exists x.A}{t + s : \exists x.A}$$

$$\frac{t : A \quad s : \bar{A}}{(t \bowtie_S s) : (A \mid \bar{A})}\ \mathrm{S} = \mathrm{fv(A)}$$

Fig. 5. Typing in prenex classical logic

$$\frac{}{\vdash [i] : P_1, \ldots, [i] : P_n}\ Taut_i$$

$$\frac{\vdash \Gamma, t : A[x := a]}{\vdash \Gamma, \alpha[a].t : \forall x.A}\ \forall\mathrm{R} \qquad \frac{\vdash \Gamma, t : A[x := a]}{\vdash \Gamma, \{\varepsilon[a].t\} : \exists x.A}\ \exists\mathrm{R}$$

$$\frac{\vdash \Gamma, t : \exists x.A, s : \exists x.A}{\vdash \Gamma, t + s : \exists x.A}\ C^{\exists} \qquad \frac{\vdash \Gamma, [S] : P, [T] : P}{\vdash \Gamma, [S \cup T] : P}\ C^{P}$$

$$\frac{\vdash \Gamma, t : A \quad \Delta, s : \bar{A}}{\vdash \Gamma, \Delta, (t \bowtie s) : (A \mid \bar{A})}$$

Fig. 6. A polarized prenex sequent calculus, plus $\alpha\varepsilon$ assignment

The paper [10] extends the definition of correctness to cover these extended ports. The graph of a structure F contains, in addition to the nodes of the F, a node i for each natural number contained in the tautology nodes $[S]$ of F – we refer to these extra nodes as tautology links. There are directed edges in the graph of F from each tautology link to the tautology nodes containing it, and tautology nodes of the form $[i]$ are considered atomic, with unique dependent i. To the definition of a switching we add a switching for each tautology node $[S]$, which is a member of the set S. Once again, a structure F is a net if and only if the switching graph is acyclic for each switching. It is an *Herbrand net* if it is correct, and if for every i appearing in F, we have that

$$\bigvee\{P \mid [S] : P \text{ a node of F}, i \in S\}$$

is a tautology.

Example 3.1. The cut-free typed structure

$$\{\varepsilon[a]\alpha[b].[1], \quad \varepsilon[b]\alpha[c].[1]\} : \exists x.\forall y(\bar{A}(x) \vee A(y))$$

is an Herbrand net: the annotated derivation is given below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[1] : (\bar{A}(a) \vee A(b)) \quad [1] : (\bar{A}(b) \vee A(c))}\,Taut}{[1] : (\bar{A}(a) \vee A(b)) \quad \alpha[c].[1] : \forall y(\bar{A}(b) \vee A(y))}\,\forall \mathrm{R}}{[1] : (\bar{A}(a) \vee A(b)) \quad \{\varepsilon[b]\alpha[c].[1]\} : \exists x.\forall y(\bar{A}(x) \vee A(y))}\,\exists \mathrm{R}}{\alpha[b].[1] : \forall y(\bar{A}(a) \vee A(y)) \quad \{\varepsilon[b]\alpha[c].[1]\} : \exists x.\forall y(\bar{A}(x) \vee A(y))}\,\forall \mathrm{R}}{\{\varepsilon[a]\alpha[b].[1]\} : \exists x.\forall y(\bar{A}(x) \vee A(y)) \quad \{\varepsilon[b]\alpha[c].[1]\} : \exists x.\forall y(\bar{A}(x) \vee A(y))}\,\exists \mathrm{R}}{\{\varepsilon[a]\alpha[b].[1], \quad \varepsilon[b]\alpha[c].[1]\} : \exists x.\forall y(\bar{A}(x) \vee A(y))}\,\exists C$$

Let A be a two-place predicate. The typed structure (two ports and one cut)

$$\alpha[a].[1] : \forall x.(A(x,x)), \qquad \{\varepsilon[b].[2], \varepsilon[c].[2]\} : \exists z.(\bar{A}(z,z)),$$
$$(\{\varepsilon[a].\{\varepsilon[a].[1]\}\} \bowtie_{\emptyset} \alpha[b].\alpha[c].[2]) : (\exists v \exists w.\bar{A}(v,w) | \forall v.\forall w A(v,w))$$

is an Herbrand net, and can be formed by "cutting together" the cut-free typed structures

$$\alpha[a].[1] : \forall x.(A(x,x)), \quad \{\varepsilon[a].\{\varepsilon[a].[1]\}\} : \exists v \exists w.\bar{A}(v,w)$$

and

$$\{\varepsilon[b].[2], \varepsilon[c].[2]\} : \exists z.(\bar{A}(z,z)), \quad \alpha[b].\alpha[c].[2] : \forall v.\forall w A(v,w))$$

which the reader may easily verify are correct.

By standard means (a "splitting tensor" theorem for cuts), we obtain the following result:

Theorem 3.1. *A structure typed in prenex classical logic is an Herbrand net if and only if it is the annotation of some sequent proof.*

Remark 3.1. The extension of $\alpha\varepsilon$ to include tautology nodes and links can be avoided if we add to the set $\mathcal{F}$ of function symbols an additional n-ary function symbol f_n for each natural number n. The idea is assign to each $[S]$ node an eigenvariable x, to then replace $[S]$ by $\alpha[x]$, and to replace the tautology link i with $\varepsilon[f_n(x_1, \ldots x_n)]$, where the x_i are the eigenvariables of tautology nodes $[S]$ with $i \in S$. This complicates the port assignment system a little, but gives an equivalent switching condition for correctness.

By applying the previous remark, we may deduce that the definition kingdom, and the reductions of $\alpha\varepsilon$, transfer from the untyped case. Making use of the following result, we can prove weak normalization for Herbrand nets.

Lemma 3.1. *The relation $X \ll Y$ iff $X \in k(Y)$ is a strict partial order.*

Theorem 3.2. *By applying communication and minimal duplication reductions, any correct structure typed in prenex classical logic reduces to a structure in which all the cuts are of the form $[S] \bowtie [T]$ (we will call these trivial cuts).*

Proof (Sketch). The proof is essentially the same as Gentzen's original cut-elimination argument, except that the measure is much simpler. We first prove that we can remove one nontrivial cut from a structure, arguing by induction on the complexity of the cut formula, with a sub-induction on the *width of a cut*, defined as

$$w(\alpha[a].t \bowtie \{\varepsilon[M_1].t_1, \ldots \varepsilon[M_n].t_n\}) = n.$$

Given a net of the form $\Gamma, \alpha[a].t \bowtie \{\varepsilon[M_1].t_1, \ldots \varepsilon[M_n].t_n\})$, there is at least one $\varepsilon[M_i]$ which is $\ll$-maximal among the members of the multiset. If we now duplication-reduce to form

$$\Gamma', \alpha[a_0].t \bowtie \{\varepsilon[M_1].t_1, \ldots, \varepsilon[M_{i-1}].t_{i-1}, \varepsilon[M_{i+1}].t_{i+1}, \ldots, \varepsilon[M_n].t_n\}) \quad \alpha[a_1].t \bowtie \{\varepsilon[M_i].t_i\}$$

then the second cut is $\ll$-maximal. We it has width 1, and we may eliminate it within its own kingdom, a correct substructure which does not contain any $\varepsilon[M_j]$ from the first cut. Thus, the result of that elimination is a net

$$\Gamma'', \alpha[a_0].t \bowtie \{\varepsilon[M_1].t'_1, \ldots, \varepsilon[M_{i-1}].t'_{i-1}, \varepsilon[M_{i+1}].t'_{i+1}, \ldots, \varepsilon[M_n].t'_n\})$$

which also falls under the induction hypothesis and can be eliminated.

Now, since we can eliminate one non-trivial cut from a structure, we can eliminate any number of cuts by a "topmost" strategy of eliminating $\ll$-maximal cuts.

Full cut-elimination now follows from the following, which is clear by semantic considerations:

Proposition 3.1. *If an Herbrand net contains only trivial cuts, deleting all those cuts and replacing each $[S]$ by $[1]$ yields a cut-free correct net.*

4 Further Work

The major open problem is to establish strong normalization of minimal reduction in the case of untyped nets. There already exists an extension of $\alpha\varepsilon$ typing propositional classical logic, in a multiplicative formulation: there are ideas about representing the additive rules, but as yet no satisfactory correctness criterion. Crucially, the multiplicative extension of $\alpha\varepsilon$ can be encoded within $\alpha\varepsilon$ itself by allowing pairing and projection of the individuals bound by α and ε. As mentioned before, we seek to establish $\alpha\varepsilon$ as a computational paradigm: to do this, we need to establish the right form of the duplication rules fro structures not satisfying the correctness criterion.

References

1. S. Abramsky. Proofs as processes. In *J. Theoretical Computer Science*, volume 135, pages 5–9, 1994.
2. G. Bellin and J. van de Wiele. Subnets of proof-nets in MLL-. In *Proceedings of the workshop on Advances in linear logic*, pages 249–270, New York, NY, USA, 1995. Cambridge University Press.
3. Gianluigi Bellin and Philip J. Scott. On the pi-calculus and linear logic. *Theoretical Computer Science*, 135(1):11–65, 1994.
4. Thierry Coquand. A semantics of evidence for classical arithmetic. *J. Symb. Logic*, 60(1):325–337, 1995.
5. Jean-Yves Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.
6. Jean-Yves Girard. Proof-nets: The parallel syntax for proof-theory. In *Logic and Algebra*, pages 97–124. Marcel Dekker, 1996.
7. Willem Heijltjes. Classical proof forests. In preparation.
8. Willem Heijltjes. Proof forests with cut-elimination based on Herbrand's theorem. Presented at Classical Logic and Computation, ICALP Workshop, 2008.
9. Richard McKinley. Expansion nets: proof nets for classical propositional logic. In preparation, will appear at `http://www.iam.unibe.ch/~mckinley`.
10. Richard McKinley. On Herbrand's theorem and cut-elimination. In preparation, will appear at `http://www.iam.unibe.ch/~mckinley`.
11. Dale Miller. A compact representation of proofs. *Studia Logica*, 46(4):347–370, 1987.
12. Robin Milner. *Communicating and Mobile Systems: the Pi-Calculus.* Cambridge University Press, 1999.
13. Edmund Robinson. Proof nets for classical logic. *Journal of Logic and Computation*, 13(5):777–797, 2003.

A Expansion nets for multiplicative propositional classical logic

$$\frac{}{\{\varepsilon_{a_1}[x_1], \dots \varepsilon_{a_n}[x_n]\} : p} \qquad \frac{}{\alpha[x] : \bar{p}}$$

$$\frac{}{\emptyset_a : A \wedge B} \qquad \frac{}{\emptyset_a : p} \qquad \frac{}{\emptyset_a : \bar{p}}$$

$$\frac{t : A,\ s : B}{[t, s] : A \vee B} \qquad \frac{t_1 : A, \dots, t_n : A \quad s_1 : B, \dots, s_n : B}{\{(t_1, s_1), \dots (t_n, s_n)\} : A \wedge B} \qquad \frac{t : A \quad s : \bar{A}}{t \bowtie s : \mathrm{Cut}}$$

Fig. 7. Typing for ports in propositional classical logic

$$\frac{}{\vdash \{\varepsilon_a[x]\} : p,\ \bar{p}_x}\ Ax \qquad \frac{\vdash \Gamma,\ \bar{p}_x}{\vdash \Gamma,\ \alpha[x] : \bar{p}}\ \alpha$$

$$\frac{\vdash \Gamma,\ s : A,\ t : B}{\vdash \Gamma,\ [s,t] : A \vee B}\ \vee \qquad \frac{\vdash \Gamma,\ s : A \quad \vdash \Gamma',\ t : B}{\vdash \Gamma,\ \Gamma',\ \{(s,t)\} : A \wedge B}\ \wedge$$

$$\frac{\vdash \Gamma,\ t : A \quad \vdash \Delta,\ s : \bar{A}}{\vdash \Gamma, \Delta, (t \bowtie s)}\ Cut$$

$$\frac{\vdash \Gamma}{\vdash \Gamma,\ \emptyset_a : P}\ W \qquad \frac{\vdash \Gamma}{\vdash \Gamma,\ \emptyset_a : \bar{p}}\ W \quad [a \in \mathcal{L}(\Gamma)]$$

$$\frac{\vdash \Gamma,\ t : P,\ s : P}{\vdash \Gamma,\ t + s : P}\ C \quad [t, s \neq \emptyset] \qquad \frac{\vdash \Gamma,\ \bar{p}_x,\ \bar{p}_x}{\vdash \Gamma,\ \bar{p}_x}\ C$$

Fig. 8. The annotated sequent calculus $\mathbf{LK}_{\alpha\varepsilon}$

The paper [9] introduces a new class of proof nets for propositional classical logic which are inspired by the structure of $\alpha\varepsilon$ calculus. We briefly summarize their definition here.

The nets are called expansion nets, in reference to the idea of an Herbrand expansion. These nets improve on the nets of Robinson [13] by staying closer to the suggestions of Girard in [5]: in particular unlike contraction, expansion is n-ary, and an expansion may not be the "premise" of another expansion. We do not just follow Girard's recipe: in addition, we restrict the system so expansion is only applied to positive formulae (in the sense described below).

Definition A.1. *Let $\mathcal{P}$ be a countable set of positive atom symbols, and let p stand for a member of $\mathcal{P}$. Let $\mathcal{V}$ be a countable set of variables.*

(a) Define positive and negative formulae as follows:

$$P := p \mid A \wedge B \qquad N := \bar{p} \mid A \vee B$$

$$A, B := P \mid N$$

(b) The negation *of an atom p is $\bar{p}$. The negation of an arbitrary formula A is defined by de Morgan duality:*

$$\bar{\bar{p}} := \bar{p} \quad \overline{A \vee B} := \bar{A} \wedge \bar{B} \quad \overline{A \wedge B} := \bar{A} \vee \bar{B}$$

(c) A subatom *is a pair consisting of a negative atom $\bar{p}$ and a variable $x \in \mathcal{V}$; we will write such a subatom as $\bar{p}_x$.*

The subatoms are something rather curious: their purpose is to allow us to represent the flow of information in an axiom using the α and *epsilon*. Essentially, the subatom should be thought of as positive subformulae (we will allow contraction on subatoms) of a negative atom.

To assign ports to propositional classical logic, we must introduce port constructors for the propositional connectives, but also for *weakening*, which will add a *deletion* reduction. Weakening in proof nets is somewhat tricky to handle, and for safety's sake the approach described here takes the most orthodox approach, in which each weakening (whose port will be an empty set of witnesses) is *wired* to an occurrence of ε (which here will occur only at axioms). This allows us to keep the usual Danos-Regnier switching condition for correctness, at the expense that our nets are not canonical – in particular the wiring of a weakening dictates its kingdom, which for us is crucial. We conjecture that a system without anchorings and with the MIX rule has a correctness criterion, but there has not been time to check the details.

The ports we will use to annotate formulae of propositional classical logic are as follows:

$$t, s := \alpha[x] \mid \{\varepsilon_{a_1}[x_1], \ldots, \varepsilon_{a_n}[x_n]\} \mid \emptyset_a \mid [t, s] \mid \{(t_1, s_1), \ldots, (t_n, s_n)\} | t \bowtie s$$

Notice that the ε nodes are now named, in order that they may serve as anchors for weakenings. Typing of ports is given in Figure 7. Notice that disjunction behaves like universal quantification (no contraction/multisets) and the conjunction like existential quantification. This is because we will annotate a variant of the multiplicative formulation of sequent calculus, which can be seen in Figure 8.

The extension of switching and correctness to these structures is derives from the usual switching of multiplicative nets: the constructor (,) is unswitched, and [,] is switched. Just as for the Herbrand system, we have sequentialization and weak normalization. We have the same hopes for strong normalization here that we have in the untyped net and Herbrand net cases.

Do Light Logics allow a unified view of Stratification and Boundedness?

Marco Gaboardi[1], Luca Roversi[1], and Luca Vercelli[2]

[1] Dip. di Informatica - Univ. di Torino
[2] Dip. di Matematica - Univ. di Torino
`http://www.di.unito.it/~{gaboardi | rover | vercelli}`

Abstract. This few pages are meant to illustrate an ongoing work whose goal is to find fine-grained logical principles, in the tradition of structural proof-theory, on which we can base the definition of a deductive system that contains both Light Affine Logic and Soft Linear Logic.

1 Introduction

1.1 Bounding the computational cost inside the structural proof theory

One of the key features of Linear Logic (LL) is that the structural rules, which account for duplication, hence for the complexity of the cut elimination, are only allowed on "modal" formulæ. The study about the cost of the cut elimination in Linear Logic may be done using proof nets, a graphical representation of LL proofs. In the particular version of LL we consider, proof nets introduce modal formulæ by means of special constructs, called "functorial boxes", "digging" and "dereliction", and only the proof nets inside a box may be duplicated during the cut elimination process.

Some constraints can be imposed on LL in order to achive bounds on the number of steps nedded to perform the cut elimination. Forbidding the use of digging and dereliction leads to Elementary Linear Logic (ELL) [Gir98]. ELL enjoys an elementary bound on the number of duplications that can occur during the cut elimination of any of its derivations. Through the Curry-Howard correspondence, ELL characterizes the class of elementary functions. Recall that they are those ones that can be computed by a Turing machine whose run-time is bounded by a tower of exponentials of fixed height. Further constraints on functorial boxes allow to characterize the class of deterministic polytime functions, under the Curry-Howard correspondence; we get Light Linear Logic (LLL) [Gir98], together with its affine version Light Affine Logic (LAL) [AR02].

On the other side, allowing dereliction but forbidding digging, and imposing some more restrictions on the duplications, leads to Soft Linear Logic (SLL) [Laf04].

The derivations of LLL/LAL enjoy a structural invariant, called *stratification*, while for those ones in SLL the invariant will be called *boundedness*.

1.2 Stratification vs. Boundedness, a little bit more technically.

By *stratification* we summarize a logical property of LLL/LAL, which is structurally obtained by dropping dereliction and digging principles. Stratification means that the boundary of the functorial boxes, that enclose well formed sub-derivations, neither can disappear, nor can be created. So, it is natural to say that a node in a proof net of LLL/LAL, or, equivalently, a rule in a derivation of LLL/LAL, is at depth d if it is contained into d nested boxes. At the dynamic level this sums up to have that if a cut elimination step involves nodes at depth d, then only the complexity of the proof nets at depths strictly deeper than d can increase. Consequently, the control over the dimension of every single reduct becomes the control on the overall cut elimination time. By the way, the mechanism is implicit in the structural and combinatorial properties of the proof nets of LLL/LAL, and is independent from the logical soundness.

Concerning *boundedness*, recall that, in ordinary Linear Logic, $!A$ is semantically equivalent to $A^* = \bigcup_{n\in\mathbf{N}} \underbrace{(A \otimes \ldots \otimes A)}_{n}$. *Boundedness* refers to various methodologies that, informally, put $!A$ in correspondence to a *finite* subset of A^*. Computationally, this means that given a proof net, the number of copies of each box coming up during the cut elimination can somehow be statically predicted, i.e. bounded. Usually, the "deep" reason is that the interpretation of $!A$ cannot be equal to the interpretation of $!A \otimes A$ and this rules out the principle underpinning self-copying. For completeness, it is worth concluding by recalling that also Bounded Linear Logic [GSS92], of which SLL seems to catch the spirit, was conceived in accordance with the principle of boundedness just illustrated.

1.3 Goal

We are currently looking for a set of logical principles, which both Stratification and Boundedness become specific subcases of.

The intuition driving us now follows. The structural bound of the polynomial time soundness of the cut elimination for LAL can be viewed as analogous to some structural bounds that can be obtained for systems exploiting the boundedness principle. In particular, the polynomial time soundness bound of LAL can be rephrased saying that there is a constant, called *rank*, that bounds the number of copies of any sub-derivation of LAL that may be duplicated under the effect of the cut elimination.

The argument applies to SLL in the reversed direction. Specifically, the polynomial time soundness of SLL can be proved in some sense as if it was a stratified deductive system. Every level in a derivation of SLL seems to be connected to suitable sets of occurrences of the *multiplexor* rule/node that simultaneously contract many occurrences of a formula, while adding a modality in front of it.

In the coming sections we shall roughly illustrate the ideas and the conjectures that drive on going work about the two intuitions just illustrated on the polytime soundness of LAL and SLL.

$$T^{\Pi}_{\mathcal{D}}(r) = \sum_{n \text{ node of } \Pi} W^{n}_{\mathcal{D}}(r)$$

$$\begin{cases} W^{n}_{\mathcal{D}}(r) = S_{\mathcal{D}}\left(r, T^{\Pi'}_{\mathcal{D}}(R_{\mathcal{D}}(r))\right) & n \text{ root of a !-box with } \Pi' \text{ in it} \\ W^{n}_{\mathcal{D}}(r) = k_n \quad \text{every other } n \end{cases}$$

Fig. 1. The weight of any proof net Π of a deductive system $\mathcal{D}$, depending on a rank r. The constants k_n and the functions $R_{\mathcal{D}}, S_{\mathcal{D}}$ are specific for the deductive system $\mathcal{D}$.

$$S_{\mathcal{D}}(r,t) = r \cdot t + 1$$
$$R_{\mathcal{D}}(r) = r$$

Fig. 2. Size measure and the rank-update functions for SLL.

2 LAL as a bounded system

Saying that LAL is a bounded system means looking for a *rank*, for every proof net Π of LAL. Given any Π, the rank is a constant that only depends on Π itself. The intended use of the rank is twofold. On one side it is the constant that fixes the maximal number of copies of subnets we can produce in the course of the cut elimination at a given depth d. On the other, (a function of) the rank bounds the size of the proof nets at depth deeper than d.

Formally, the idea of *rank* is given in Figure 1. $\mathcal{D}$ is a generic deductive system, as SLL or LAL.

By letting $\mathcal{D}$ be SLL, and defining both the size measure $S_{\mathcal{D}}$ and the rank-update function $R_{\mathcal{D}}$ as in Figure 2, we get the cut elimination bound of any proof net Π of SLL described in [Laf04].

Now we try letting $\mathcal{D}$ more general. Then $T^{\Pi}_{\mathcal{D}}(r)$ is a bound on the cut elimination cost that depends on the rank r that can intuitively be bounded by the initial dimension S_{Π} of Π, namely by the number of nodes in Π. $T^{\Pi}_{\mathcal{D}}(r)$ is the sum of the weights $W^{n}_{\mathcal{D}}$ of the nodes of Π. The weight is a suitable, in fact, essentially irrelevant, constant k_n, if n is not the root of a box in LAL. Otherwise, if n is the root of a box that contains the proof net Π', the weight is a bound on a size measure $S_{\mathcal{D}}$ that depends on the rank, and on the bound on the cut elimination that can operate on Π', namely on $T^{\Pi'}_{\mathcal{D}}(v)$, for a suitable value v. In its turn, the value v must be a function $R_{\mathcal{D}}(r)$ that updates the rank for Π' which is at a deeper depth than Π.

In our case, with $\mathcal{D}$ being LAL, a possible choice for the size measure $S_{\mathcal{D}}$ and the rank-update function $R_{\mathcal{D}}$ is in Figure 3. All this should convince that LAL has a notion of rank which cannot be fixed once for all by looking at specific nodes

$$S_{\mathfrak{D}}(r,t) = r \cdot t$$
$$R_{\mathfrak{D}}(r) = r^2$$

Fig. 3. A possible choice for the size measure and the rank-update functions for LAL.

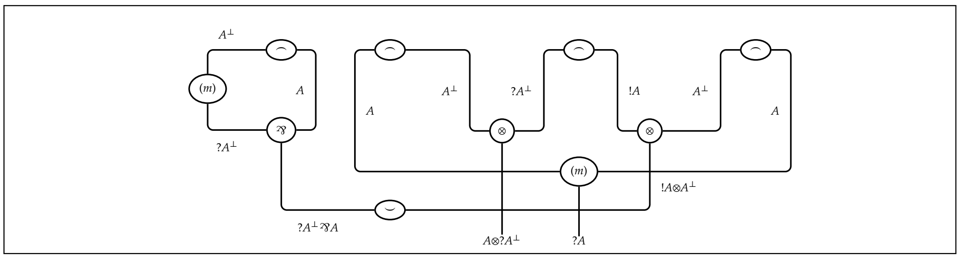


Fig. 4. The proof net Σ of SLL where we shall discover boxes.

of a given proof net, like in SLL, but which must be updated in relation to the depth we are normalizing at.

Once defined the weight, one would use it in order to give a proof of LAL polytime soundness in analogy to the proof given by Lafont in [Laf04] for SLL. In fact, the weight as defined in Figure 1 cannot be used in that way. The reason is that it does not decrease during the cut elimination. To fix this problem very likely it is necessary to extend the definition in Figure 1 with another parameter.

3 SLL as a stratified system

We give an idea about why we conjecture that the proof nets of SLL are stratified, and what we mean by stratification. SLL should be stratified in a sense weaker than LLL/LAL. We use a reasonably simple, but relevant example. Let us look at Figure 4 that contains a proof net Σ of SLL. Our idea is that every multiplexor node (m) of Σ actually corresponds to a *generalized* paragraph box, that in some way generalizes the ones we know from LLL/LAL. We call SLL^4 the hypothetical new system obtained by SLL after adding these new paragraph boxes.

SLL^4 should allow to map Σ into the proof net $\bar{\Sigma}$ in Figure 5. The leftmost multiplexor and paragraph nodes should delineate a paragraph box of which the multiplexor represents the inputs and the paragraph node is the output.

However, the rightmost multiplexor node is not so evidently connected to the remaining two paragraph nodes to form a paragraph box. Asking SLL^4 to have a more general introduction of paragraph boxes than LLL/LAL, comes in help exactly now. The introduction of paragraph boxes in SLL^4 must be so general that we must be able to prove the isomorphisms among logical operators in

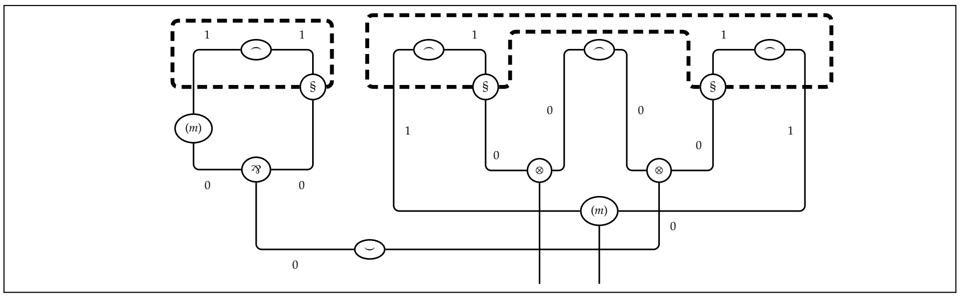


Fig. 5. The proof net $\bar{\Sigma}$ in SLL^4, image of Σ of SLL.

$$
\begin{gathered}
\S(A \otimes B) \simeq \S A \otimes \S B \\
\S(A \parr B) \simeq \S A \parr \S B \\
\S !A \simeq !\S A \\
\S\forall\alpha.A \simeq \forall\alpha.\S A
\end{gathered}
$$

Fig. 6. Isomorphims among the logical operators of SLL^4.

Figure 6. Moreover, assuming that on SLL^4 the η-expansion on axioms holds, we can transform the rightmost multiplexor in Figure 5 as a node that introduces the assumptions of a paragraph box, using the two following steps:

– we η-expand the third axiom, counting from left, and
– we use the isomorphisms in Figure 6 to shift downward all the paragraphs to the right until one paragraph occurs below one of the two $\otimes$ nodes, and another paragraph node occurs below the other $\otimes$ node.

4 SLL, LLL and ML^3

The candidate system where to look for SLL^4 is ML^3 [BM08] since it enjoys η-expansion on axioms and it can prove the isomorphisms in Figure 6.

ML^3 is a proof nets system defined as the subset of LL proof nets enjoying a certain structural property called *indexing*. In [BM08], the authors show that: (i) ML^3 extends ELL, (ii) ML^3 is *weakly elementary time sound*, in the sense that every proof net may be reduced in elementary time, just following a particular reduction strategy, (iii) there exists a subsystem ML^4 of ML^3 strictly larger than LLL, and ML^4 is *weakly polynomial time sound*. One of our targets is to find an analogous system SLL^4 of ML^3 strictly containing SLL, and that will be (at least weakly) polynomial time sound too.

While searching for such a subsystem, we run into the following property.

Theorem 1 ([GRV09]). *There exists a (reasonably simple) translation @ from every proof net of* the propositional fragment of LL *into* ML^3*, that respects the cut elimination procedure.*

In particular, the restriction of @ to propositional SLL identifies a subsystem SLL^4_p of ML^3 that partially solves our research. But of course the *propositional* fragment of SLL is quite less complex than the full SLL.

5 Conclusions

Suppose for a while we shall be able to bring the definition of the embedding of SLL into a suitable SLL^4 to its end. This will not mean we shall have a system where both LAL and SLL embed, but just a system, where stratification and boundedness, used to characterize polynomial sound computations, under the Curry-Howard correspondence, actually coexist.

We conjecture it is also possible to obtain a system where both LAL and SLL embed by generalizing SLL^4, namely, by generalizing the system where we can embed SLL by means of a suitable set of modalities whose goal is to locally distinguish the origins of contracted formulæ. To this aim we plan to exploit the experience developed on Multimodal Stratified framework (MS) [RV09]. MS allows to characterize polynomial time sound systems of proof nets, with, at least in principle, an unlimited set of modal operators. So, its principles should be exploitable to add to SLL^4 the modal operators we need.

References

[AR02] A. Asperti and L. Roversi. Intuitionistic light affine logic. *ACM Transaction on Computational Logic*, 3(1):137–175, 2002.

[BM08] P. Baillot and D. Mazza. Linear logic by levels and bounded time complexity. Technical report, http://arxiv.org/abs/0801.1253v1, January 2008.

[Gir98] J.-Y. Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998.

[GRV09] M. Gaboardi, L. Roversi, and L. Vercelli. Multiplicative Exponential Linear Logic can be levelled. http://www.di.unito.it/~vercelli/works/soli-part-I.pdf, Submitted, 2009.

[GSS92] J.-Y. Girard, A. Scedrov, and P. Scott. Bounded linear logic: A modular approach to polynomial time computability. *Theoretical Computer Science*, 97:1–66, 1992.

[Laf04] Y. Lafont. Soft linear logic and polynomial time. *Theoretical Computer Science*, 318:163–180, 2004.

[RV09] L. Roversi and L. Vercelli. Some Complexity and Expressiveness results on Multimodal and Stratified Proof-nets. In *TYPES*, 2009. http://www.di.unito.it/~vercelli/works/ms-part-I.pdf.

On pseudocategories in a category with a 2-cell structure

N. Martins-Ferreira

Abstract. For a given (fixed) category, we consider the category of all 2-cell structures (over it) and study some naturality properties. A category with a 2-cell structure is a sesquicategory; we use additive notation for the vertical composition of 2-cells; instead of a law for horizontal composition we consider a relation saying which pairs of 2-cells can be horizontally composed; for a 2-cell structure with every 2-cell invertible, we also consider a notion of commutator, measuring the obstruction for horizontal composition. We compare the concept of naturality in an abstract 2-cell structure with the example of internal natural transformations in a category of the form Cat(**B**), of internal categories in some category **B**, and show that they coincide. We provide a general construction of 2-cell structures over an arbitrary category, under some mild assumptions. In particular, the canonical 2-cell structures over groups and crossed-modules, respectively "conjugations" and "derivations", are instances of these general constructions. We define cartesian 2-cell structure and extend the notion of pseudocategory from the context of a 2-category (as in [**6**]) to the more general context of a sesquicategory. As an example of application we consider pseudocategories in the sesquicategory of abelian chain complexes.

1. Introduction

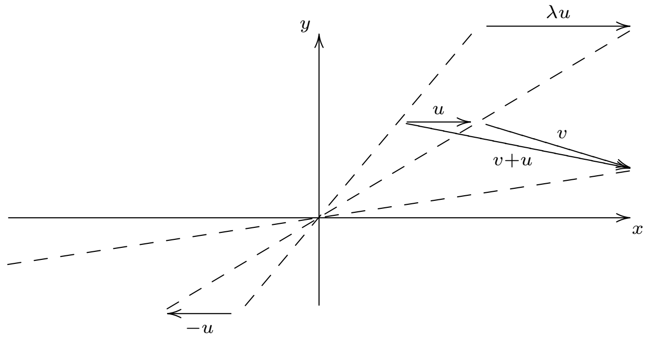
In this article we use a different notation for the vertical composition of 2-cells: instead of the usual dot '·' we use plus '+'. To support this we present the following analogy between geometrical vectors in the plane and 2-cells between morphisms

2000 *Mathematics Subject Classification.* Primary 18D05, 18D35; Secondary 18E05.

Key words and phrases. sesquicategory, 2-cell structure, cartesian 2-cell structure, natural 2-cell structure, pseudocategory.

This is the result of several talks given by the author in Milan (Oct2006), Coimbra (May2007) and Haute-Bodeux (Jun2007). The author thanks to Professor G. Janelidze for much appreciated help of various kinds.

in a category.



Two geometrical vectors in the plane can be added only if the end point of the second (u as in the picture above) is the starting point of the first one (v as in the picture) and in that case the resulting vector (the sum) goes from the starting point of the second to the end point of the first: exactly the same as with 2-cells

$$\begin{tikzcd}
\cdot \arrow[r, bend left=70, "\mathrm{dom}\, u"{name=U}] \arrow[r, ""{name=M}] \arrow[r, bend right=70, "\mathrm{cod}\, v"'{name=D}] \arrow[from=U, to=M, Rightarrow, "u"] \arrow[from=M, to=D, Rightarrow, "v"] & \cdot \arrow[r, phantom, "\longmapsto"] & \cdot \arrow[r, bend left=50, "\mathrm{dom}\, u"{name=A}] \arrow[r, bend right=50, "\mathrm{cod}\, v"'{name=B}] \arrow[from=A, to=B, Rightarrow, "v+u"] & \cdot \; ;
\end{tikzcd}$$

In some sense the analogy still holds for scalar multiplication

$$\begin{tikzcd}
\cdot \arrow[r, "\lambda"] & \cdot \arrow[r, bend left=50, "\mathrm{dom}\, u"{name=A}] \arrow[r, bend right=50, "\mathrm{cod}\, u"'{name=B}] \arrow[from=A, to=B, Rightarrow, "u"] & \cdot \arrow[r, "\rho"] & \cdot \arrow[r, phantom, "\longmapsto"] & \cdot \arrow[r, bend left=50, "\rho\, \mathrm{dom}(u)\lambda"{name=C}] \arrow[r, bend right=50, "\rho\, \mathrm{cod}(u)\lambda"'{name=D}] \arrow[from=C, to=D, Rightarrow, "\rho u \lambda"] & \cdot \; ,
\end{tikzcd}$$

and for inverses (in the case they exist)

$$\begin{tikzcd}
\cdot \arrow[r, bend left=50, "\mathrm{dom}\, u"{name=A}] \arrow[r, bend right=50, "\mathrm{cod}\, u"'{name=B}] \arrow[from=A, to=B, Rightarrow, "u"] & \cdot \arrow[r, phantom, "\longmapsto"] & \cdot \arrow[r, bend left=50, "\mathrm{cod}\, u"{name=C}] \arrow[r, bend right=50, "\mathrm{dom}\, u"'{name=D}] \arrow[from=C, to=D, Rightarrow, "-u"] & \cdot \; .
\end{tikzcd}$$

Concerning horizontal composition, there is still an analogy with some relevance: it is, in some sense, analogous to the cross product of vectors – in the sense that it raises in dimension (see the introduction of [**1**] and its references for further discussion on this). Given 2-cells, u and v

$$\begin{tikzcd}
\cdot \arrow[r, bend left=50, "\mathrm{dom}\, u"{name=A}] \arrow[r, bend right=50, "\mathrm{cod}\, u"'{name=B}] \arrow[from=A, to=B, Rightarrow, "u"] & \cdot \arrow[r, bend left=50, "\mathrm{dom}\, v"{name=C}] \arrow[r, bend right=50, "\mathrm{cod}\, v"'{name=D}] \arrow[from=C, to=D, Rightarrow, "v"] & \cdot
\end{tikzcd}$$

the horizontal composition $v \circ u$ should be a 3-cell, from the 2-cell

(1.1) $$\mathrm{cod}\,(v)\, u + v\, \mathrm{dom}\,(u)$$

to the 2-cell

$$v \operatorname{cod}(u) + \operatorname{dom}(v)\, u. \tag{1.2}$$

In some cases (1.1) and (1.2) coincide (as it happens in a 2-category) and this is the reason why one may think of a horizontal composition, but it is an illusion; to overcome this we better consider a relation $v \circ u$ saying that the 2-cell v is natural with respect to u, defined as

$$v \circ u \Longleftrightarrow (1.1) = (1.2)\,,$$

in this sense, the horizontal composition is only defined for those pairs (v, u) that are in relation $v \circ u$, with the composite being then given by either (1.1) or (1.2).

This is a geometrical intuition. An algebraic intuition is also provided in Proposition 1.

This article is organized as follows.

For a fixed category, $\mathbf{C}$, we define a 2-cell structure (over $\mathbf{C}$, as to make it a sesquicategory) and give a characterization of such a structure as a family of sets, together with maps and actions, satisfying some conditions. It generalizes the characterization of 2-Ab-categories as a family of abelian groups, together with group homomorphisms and laws of composition as given in [**5**] and [**7**] where the (strong) condition

$$D(x)\, y = xD(y)$$

is no longer required. A useful consequence is that the example of chain complexes, say of order 2, can be considered in this more general setting. Of course, this condition is equivalent to the naturality condition, and the results obtained in [**5**] and [**7**] heavily rest on this assumption, so one must be careful in removing it. For this we introduce and study the concept of a 2-cell being natural with respect to another 2-cell, and the concept of natural 2-cell, as one being natural with respect to all. Next we compare this notions when $\mathbf{C}$ is a category of the form $\mathrm{Cat}(\mathbf{B})$, of internal categories in some category $\mathbf{B}$, and conclude that if the 2-cell structure is the canonical one (internal transformations, not necessarily natural) then a natural 2-cell corresponds to a natural transformation, and furthermore, it is sufficient to check if a given transformation is natural with respect to a particular 2-cell (from the "category of arrows"), to determine if it is natural.

We give a general process for constructing 2-cell structures in arbitrary categories, and for the purposes of latter discussions we will restrict our study to the 2-cell structures obtained this way. In order to argue that we are not restricting too much, we show that the canonical 2-cell structures over groups and crossed-modules, that are respectively "conjugations" and "derivations", are captured by this construction.

We introduce the notion of cartesian 2-cell structure, in order to consider 2-cells of the form $u \times_w v$ that are used in the coherence conditions involved in a pseudocategory.

At the end we extend the notion of pseudocategory from the context of a 2-category to the more general context of a category with a 2-cell structure (sesquicategory). As an example of application we consider the sesquicategory of abelian chain complexes with homotopies as 2-cells and study pseudocategories in there.

All the notions defined in [**6**]: pseudofunctor, natural and pseudo-natural transformation, modification, may also be extended in this way. However some careful is needed when dealing with coherence issues. For example MacLane's Coherence

Theorem, saying that it suffices to consider the coherence for the *pentagon* and *middle triangle* is no longer true in general, since it uses the fact that α, λ, ρ are natural. One way to overcome this difficulty is to impose the naturality for α, λ, ρ in the definition, so that in [**6**] (introduction, definition of pseudocategory in a 2-category) instead of saying

$$"...\alpha, \lambda, \rho \text{ are 2-cells (which are isomorphisms)...}"$$

we have to say

$$"...\alpha, \lambda, \rho \text{ are natural and invertible 2-cells ...}"$$

We will not study deeply all the consequences of this. Instead we will restrict ourselves to the study of 2-cell structures such that all 2-cells are invertible (since the main examples are groups, abelian groups, 1-chain complexes and crossed modules) and hence the question of α, λ, ρ being invertible becomes intrinsic to the 2-cell structure. The issue of naturality is more delicate. To prove the results in [**5**], [**7**] and [**9**], we will only need λ and ρ to be natural with respect to each other, that is

$$\lambda \circ \lambda, \lambda \circ \rho, \rho \circ \lambda, \rho \circ \rho.$$

If interested in the Coherence Theorem, we can always use the reflection

$$\text{2-cellstruct}(\mathbf{C}) \xrightarrow{I} \text{nat-2-cellstruct}(\mathbf{C})$$

of the category of 2-cell structures over $\mathbf{C}$ (sesquicategories "with base $\mathbf{C}$"), into the subcategory of natural 2-cell structures over $\mathbf{C}$ (2-categories "with base $\mathbf{C}$"), sending each 2-cell structure to its "naturalization"; which, if $\mathbf{C} = \mathbf{1}$, becomes the familiar reflection of monoids into commutative monoids

$$\text{Mon} \xrightarrow{I} \text{CommMon}$$

and if restricting further to invertible 2-cells gives the reflection

$$\text{Grp} \xrightarrow{I} \text{Ab}$$

of groups into abelian groups.

All these considerations are to be developed in some future work. This paper is the starting point for a systematic study of internal categorical structures in a category with a given 2-cell structure, and also to investigate how these categorical structures are changed when the given 2-cell structure over the (fixed) base category also changes. For example, a pseudo category, in a category with the discrete 2-cell structure is an internal category, while if the 2-cell structure is the codiscrete one, it becomes simply a precategory.

2. 2-cell structures and sesquicategories

Let $\mathbf{C}$ be a fixed category.

Definition 1 (2-cell structure). *A 2-cell structure over* $\mathbf{C}$ *is a system*

$$\mathbf{H} = (H, \text{dom}, \text{cod}, 0, +)$$

where

$$H : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Set$$

is a functor and

$$\begin{tikzcd} H \times_{\mathrm{hom}} H \arrow[r, "+"] & H \arrow[r, shift left=2, "\mathrm{dom}"] \arrow[r, shift right=2, "\mathrm{cod}"'] & \mathrm{hom}_{\mathbf{C}} \arrow[l, "0" description] \end{tikzcd}$$

are natural transformations, such that

$$(\mathrm{hom}_{\mathbf{C}}, H, \mathrm{dom}, \mathrm{cod}, 0, +)$$

is a category object in the functor category $Set^{C^{op}\times C}$ *or, in other words, an object in* $Cat\left(Set^{C^{op}\times C}\right)$*.*

Proposition 1. *Giving a 2-cell structure over a category* $\mathbf{C}$*, is to give, for every pair* (A, B) *of objects in* $\mathbf{C}$*, a set* $H(A,B)$*, together with maps*

$$\begin{tikzcd} H(A,B) \times_{\mathrm{hom}(A,b)} H(A,B) \arrow[r, "+"] & H(A,B) \arrow[r, shift left=2, "\mathrm{dom}"] \arrow[r, shift right=2, "\mathrm{cod}"'] & \mathrm{hom}(A,B) \arrow[l, "0" description] \end{tikzcd},$$

and actions

$$\begin{array}{ccc} H(B,C) \times \mathrm{hom}(A,B) & \longrightarrow & H(A,C) \\ (x,f) & \longmapsto & xf \\ \mathrm{hom}(B,C) \times H(A,B) & \longrightarrow & H(A,C) \\ (g,y) & \longmapsto & gy \end{array}$$

satisfying the following conditions

$$\begin{gathered} \mathrm{dom}(gy) = g\,\mathrm{dom}(y) \ , \ \mathrm{dom}(xf) = \mathrm{dom}(x)\,f \\ \mathrm{cod}(gy) = g\,\mathrm{cod}(y) \ , \ \mathrm{cod}(xf) = \mathrm{cod}(x)\,f \\ g0_f = 0_{gf} = 0_g f \\ (x+x')\,f = xf + x'f \ , \ g(y+y') = gy + gy' \end{gathered} \tag{2.1}$$

$$\begin{aligned} g'(gy) &= (g'g)\,y \ , \ (xf)\,f' = x\,(ff') \\ g'(xf) &= (g'x)\,f \\ 1_C x &= x = x1_B \end{aligned} \tag{2.2}$$

$$\begin{aligned} \mathrm{dom}(0_f) &= f = \mathrm{cod}(0_f) \\ \mathrm{dom}(x+x') &= x' \ , \ \mathrm{cod}(x+x') = x \\ 0_{\mathrm{cod}\,x} + x &= x = x + 0_{\mathrm{dom}\,x} \\ x + (x'+x'') &= (x+x') + x''. \end{aligned} \tag{2.3}$$

Proof. For every $f : A' \longrightarrow A$, $g : B \longrightarrow B'$ and $x \in H(A,B)$, write

$$H(f,g)(x) = gxf$$

and it is clear that the set of conditions (2.1) asserts the naturality of $\mathrm{dom}, \mathrm{cod}, 0, +$; the set of conditions (2.2) asserts the functoriality of H and the set of conditions (2.3) asserts the axioms for a category. □

Definition 2 (sesquicategory). *A* sesquicategory *is a pair* $(\mathbf{C}, \mathbf{H})$ *where* $\mathbf{C}$ *is a category and* $\mathbf{H}$ *a 2-cell structure over it.*

Observation: A sesquicategory, as defined, is the same as a sesquicategory in the sense of Ross Street ([**10**],[**11**],[**12**]), that is, a category $\mathbf{C}$ together with a functor $\mathbf{H}$ into Cat, such that the restriction to Set gives $\hom_{\mathbf{C}}$, as displayed in the following picture

$$\begin{tikzcd} & Cat \arrow[d] \\ \mathbf{C}^{op} \times \mathbf{C} \arrow[ur, "\mathbf{H}"] \arrow[r, "\hom"'] & Set \end{tikzcd} .$$

Proposition 2. *A category $\mathbf{C}$ with a 2-cell structure*

$$\mathbf{H} = (H, \mathrm{dom}, \mathrm{cod}, 0, +),$$

is a 2-category if and only if the naturality condition

(naturality condition) $\qquad \mathrm{cod}\,(x)\,y + x\,\mathrm{dom}\,(y) = x\,\mathrm{cod}\,(y) + \mathrm{dom}\,(x)\,y$

holds for every $x \in H\,(B,C)$, $y \in H\,(A,B)$, and every triple of objects (A,B,C) in $\mathbf{C}$, as displayed in the diagram below

$$\begin{tikzcd} A \arrow[r, bend left=50, "\mathrm{dom}\, y"] \arrow[r, bend right=50, "\mathrm{cod}\, y"'] \arrow[r, phantom, "\Downarrow y"] & B \arrow[r, bend left=50, "\mathrm{dom}\, x"] \arrow[r, bend right=50, "\mathrm{cod}\, x"'] \arrow[r, phantom, "\Downarrow x"] & C \end{tikzcd} .$$

Proof. If $\mathbf{C}$ is a 2-category, the naturality condition follows from the horizontal composition of 2-cells and, conversely, given a 2-cell structure over $\mathbf{C}$, in order to make it a 2-category one has to define a horizontal composition and it is defined as

$$x \circ y = \mathrm{cod}\,(x)\,y + x\,\mathrm{dom}\,(y)$$

or

$$x \circ y = x\,\mathrm{cod}\,(y) + \mathrm{dom}\,(x)\,y$$

provided the naturality condition is satisfied for every appropriate x, y. The middle interchange law also follows from the naturality condition. □

It may happen that the naturality condition does not hold for all possible x and y, but only for a few; thus the following definitions.

Let $\mathbf{C}$ be a category and $(H, \mathrm{dom}, \mathrm{cod}, 0, +)$ a 2-cell structure over it.

Definition 3. *A 2-cell $\delta \in H\,(A,B)$ is* natural with respect to *a 2-cell $z \in H\,(X,A)$, when*

$$\mathrm{cod}\,(\delta)\,z + \delta\,\mathrm{dom}\,(z) = \delta\,\mathrm{cod}\,(z) + \mathrm{dom}\,(\delta)\,z;$$

in that case one writes $\delta \circ z$.

Definition 4. *A 2-cell $\delta \in H\,(A,B)$ is* natural *when it is natural with respect to all possible $z \in H\,(X,A)$ for all $X \in \mathbf{C}$, i.e., δ is a natural 2-cell if and only if $\delta \circ z$ for all possible z.*

3. Examples of application

We shall now see how the above notions of naturality are related, in the case where $\mathbf{C} = Cat\,(\mathbf{B})$ for some category $\mathbf{B}$, with the 2-cell structure given by the internal (natural) transformations.

3.1. The canonical 2-cell structure of Cat(B). Consider $\mathbf{C} = Cat\,(\mathbf{B})$ the category of internal categories in some category $\mathbf{B}$. The objects are[1] (see also [**4**], p.267)

$$A = (A_0, A_1, d, c, e, m)\,,\ B = (B_0, B_1, d, c, e, m)\,,\ ...$$

and morphisms

$$f = (f_1, f_0) : A \longrightarrow B, ...$$

In this case, we have a canonical 2-cell structure, given by the internal transformations (not necessarily natural) and it is as follows:

$$\begin{aligned}
H\,(A, B) &= \{(k, t, h) \mid t : A_0 \longrightarrow B_1;\ h, k \in \hom_{\mathbf{C}}(A, B)\,;\ dt = h_0, ct = k_0\} \\
H\,(f, g)\,(k, t, h) &= (gkf, g_1 t f_0, ghf) \\
\operatorname{dom}(k, t, h) &= h \\
\operatorname{cod}(k, t, h) &= k \\
0_h &= (h, eh_0, h) \\
(k, t, h) + (h, s, l) &= (k, m\,\langle t, s\rangle\,, l)
\end{aligned}$$

where $f : A' \longrightarrow A, g : B \longrightarrow B', h, k, l : A \longrightarrow B$ are morphisms in $Cat\,(\mathbf{B})$ and $t, s : A_0 \longrightarrow B_1$ are morphisms in $\mathbf{B}$.
Observe that, in particular, for every $A = (A_0, A_1, d, c, e, m)$ there is $A^{\rightarrow} = (A_1, A_1, 1, 1, 1, 1)$ and the two morphisms

$$d^{\rightarrow} = (ed, d) : A^{\rightarrow} \longrightarrow A$$

and

$$c^{\rightarrow} = (ec, c) : A^{\rightarrow} \longrightarrow A.$$

PROPOSITION 3. *In the context of the above setting, a 2-cell* $\mathbf{t} = (k, t, h) \in H\,(A, B)$ *is an internal natural transformation* $\mathbf{t} : h \longrightarrow k$ *if and only if it is natural with respect to the 2-cell*

$$(c^{\rightarrow}, 1_{A_1}, d^{\rightarrow}) \in H\,(A^{\rightarrow}, A)\,.$$

PROOF. Consider $\mathbf{t} = (k, t, h) \in H\,(A, B)$ and $\mathbf{z} = (g, z, f) \in H\,(X, A)\,,$

$$\begin{tikzcd}
\cdots & X_1 \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[d, shift right, "f_1"'] \arrow[d, shift left, "g_1"] & X_0 \arrow[l] \arrow[d, shift right, "f_0"'] \arrow[d, shift left, "g_0"] \arrow[dl, "z"'] \\
\cdots & A_1 \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[d, shift right, "h_1"'] \arrow[d, shift left, "k_1"] & A_0 \arrow[l] \arrow[d, shift right, "h_0"'] \arrow[d, shift left, "k_0"] \arrow[dl, "t"'] \\
\cdots & B_1 \arrow[r, shift left=2] \arrow[r, shift right=2] & B_0 \arrow[l]
\end{tikzcd}$$

by definition

$$\begin{aligned}
\mathbf{t} \circ \mathbf{z} \ &\Leftrightarrow\ (kg, k_1 z, kf) + (kf, tf_0, hf) = (kg, tg_0, hg) + (hg, h_1 z, hf) \\
&\Leftrightarrow\ (kg, m\,\langle k_1 z, tf_0\rangle\,, hf) = (kg, m\,\langle tg_0, h_1 z\rangle\,, hf) \\
&\Leftrightarrow\ m\,\langle k_1 z, tf_0\rangle = m\,\langle tg_0, h_1 z\rangle \qquad (3.1)
\end{aligned}$$

and also by definition t is an internal natural transformation when

$$m\,\langle k_1, td\rangle = m\,\langle tc, h_1\rangle \qquad (3.2)$$

[1]An internal category is a system (C_0, C_1, d, c, e, m) where $d, c : C_1 \longrightarrow C_0, e : C_0 \longrightarrow C_1, m : C_1 \times_{C_0} C_1 \longrightarrow C_1$ satisfying $de = 1_{C_0} = ec$, plus the usual axioms for preservation of domain, d, codomain, c, identity (e) for composition (m) and associativity.

which is equivalent to saying that (k,t,h) is natural relative to $(c^{\rightarrow},1_{A_1},d^{\rightarrow})$, as displayed below

$$\begin{tikzcd}
\cdots & A_1 \arrow[r, equal] \arrow[d, shift right, "ed"'] \arrow[d, shift left, "ec"] & A_1 \arrow[dl, "1"'] \arrow[d, shift right, "d"'] \arrow[d, shift left, "c"] & . \\
\cdots & A_1 \arrow[r, shift left=2] \arrow[r, shift right=2] & A_0 \arrow[l]
\end{tikzcd}$$

□

Corollary 1. *Every internal natural transformation is a natural 2-cell.*

Proof. Simply observe that

$$(3.2) \Longrightarrow (3.1)$$

since

$$\begin{aligned}
m\langle k_1,td\rangle z &= m\langle tc,h_1\rangle z\\
m\langle k_1z,tdz\rangle &= m\langle tcz,h_1z\rangle\\
m\langle k_1z,tf_0\rangle &= m\langle tg_0,h_1z\rangle .
\end{aligned}$$

□

The notion of a category with a 2-cell structure, besides giving a simple characterization of a 2-category as

"2-category" = "sesquicategory" + "naturality condition";

it also provides a powerful tool to construct examples in arbitrary situations.

3.2. Abstract 2-cells, and conjugations. Consider $\mathbf{C}$ a category and

$$H:\mathbf{C}^{op}\times\mathbf{C}\longrightarrow Mon$$

a functor into Mon, the category of monoids, together with a natural transformation

$$D:UH\times\hom_{\mathbf{C}}\longrightarrow\hom_{\mathbf{C}}$$

(where $U:Mon\longrightarrow Set$ denotes the forgetful functor) satisfying

$$\begin{aligned}
D(0,f) &= f\\
D(x'+x,f) &= D(x',D(x,f))
\end{aligned}$$

for all $f:A\longrightarrow B$ in $\mathbf{C}$ and $x',x\in H(A,B)$, with 0 the zero of the monoid $H(A,B)$ considered in additive notation.
A 2-cell structure in $\mathbf{C}$ is now given as

$$\begin{tikzcd}
A \arrow[r, bend left=50, "f"{name=U}] \arrow[r, bend right=50, "D(x,f)"'{name=L}] & B \arrow[from=U, to=L, Rightarrow, "(x,f)"]
\end{tikzcd}$$

with vertical composition

$$\begin{tikzcd}
A \arrow[rr, bend left=60, "f"{name=U}] \arrow[rr, "D(x,f)" description, ""{name=M}] \arrow[rr, bend right=60, "D(x',D(x,f))"'{name=L}] && B
\arrow[Rightarrow, from=U, to=M, "(x,f)"] \arrow[Rightarrow, from=M, to=L, "(x',D(x,f))"]
\end{tikzcd}$$

$$(x', D(x,f)) + (x,f) = (x'+x, f)$$

(well defined because $D(x'+x,f) = D(x', D(x,f))$), with identity 2-cells

$$\begin{tikzcd}
A \arrow[r, bend left=50, "f"{name=U}] \arrow[r, bend right=50, "f"'{name=L}] & B
\arrow[Rightarrow, from=U, to=L, "(0,f)"]
\end{tikzcd}$$

well defined because $D(0,f) = f$, and the left and right actions of morphisms in 2-cells,

$$\begin{tikzcd}
A' \arrow[r, "h"] & A \arrow[r, bend left=50, "f"{name=U}] \arrow[r, bend right=50, "D(x,f)"'{name=L}] & B \arrow[r, "g"] & B'
\arrow[Rightarrow, from=U, to=L, "(x,f)"]
\end{tikzcd}$$

$$g(x,f)h = (gxf, gfh) = (H(h,g)(x), gfh).$$

If in addition,

$$D(y,g)x + yf = yD(x,f) + gx \tag{3.3}$$

for all x, y, f, g pictured as

$$\begin{tikzcd}
A \arrow[r, bend left=50, "f"{name=U1}] \arrow[r, bend right=50, "D(x,f)"'{name=L1}] & B \arrow[r, bend left=50, "g"{name=U2}] \arrow[r, bend right=50, "D(y,g)"'{name=L2}] & C ,
\arrow[Rightarrow, from=U1, to=L1, "(x,f)"] \arrow[Rightarrow, from=U2, to=L2, "(y,g)"]
\end{tikzcd}$$

then the result is a 2-category.

3.2.1. *The case of Groups.* In the case of $\mathbf{C} = Grp$ the category of groups and group homomorphisms, the construction above is so general that it includes the canonical 2-cells that are obtained if considering each group as a one object groupoid and each group homomorphism as a functor. In that case, as it is well known, a 2-cell

$$t : f \longrightarrow g$$

from the homomorphism f to the homomorphism g, both from the group A to the group B, is an element $t \in B$ such that

$$tf(x) = g(x)t \quad , \quad \text{for all } x \in A.$$

Now, given t and f, the homomorphism g is uniquely determined as

$$g(x) = tf(x)t^{-1} = {}^{t}f(x),$$

and hence, this particular 2-cell structure over Grp is an instance of Example 3.2 with Grp instead of Mon.

To see this just consider H the functor that projects the second argument

$$H : Grp^{op} \times Grp \longrightarrow Grp$$
$$(A, B) \longmapsto B$$

and

$$D : B \times \hom(A, B) \longrightarrow \hom(A, B)$$
$$(t, f) \longmapsto {}^{t}f$$

and it is a straightforward calculation to check that

$$\begin{aligned} D(0, f) &= f \\ D(t + t', f) &= D(t, D(t', f)) \end{aligned}$$

and also, since condition (3.3) is satisfied, the 2-cell structure is natural.

In some cases, the above construction may even be pushed further.

3.3. Abstract 2-cells, and derivations. Suppose the functor

$$\hom_{\mathbf{C}} : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Set$$

may be extended to Mon, that is, there is a functor (denote it by *map*, and think of the underlying map of a homomorphism)

$$map : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Mon \xrightarrow{U} Set$$

with $\hom \subseteq Umap$, in the sense that $\hom(A, B) \subseteq Umap(A, B)$ naturally for every $A, B \in \mathbf{C}$;
Now, given any functor

$$K : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Mon$$

and any natural transformation

$$D : K \longrightarrow map,$$

define

$$\begin{aligned} H(A, B) &= \{(x, f) \in UK(A, B) \times \hom(A, B) \mid D(x) + f \in \hom(A, B)\} \\ H(h, g)(x, f) &= (gxh, gfh) \end{aligned}$$

and obtain a functor $H : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Set$. With obvious $\mathrm{dom}, \mathrm{cod}, 0, +$, a 2-cell structure in $\mathbf{C}$ is obtained as follows

$$\begin{tikzcd} A \arrow[r, bend left=50, "f"{name=U}] \arrow[r, bend right=50, "D(x)+f"'{name=L}] & B \arrow[Rightarrow, from=U, to=L, "{(x,f)}"] \end{tikzcd}$$

where $(x, f) \in H(A, B)$,
vertical composition: $(x', D(x) + f) + (x, f) = (x' + x, f)$

identity: $(0, f)$
left and right actions: $g(x, f)h = (gxh, gfh)$.
If in addition the property

$$D(y)x + gx + yf = yD(x) + yf + gx \tag{3.4}$$

is satisfied for all $(x, f) \in H_1(A, B)$ and $(y, g) \in H_1(A, C)$, then the resulting structure is a 2-category.

3.3.1. *The case of crossed modules.* In the case $\mathbf{C}$=X-Mod, the category of crossed modules, we have the canonical 2-cell structure given by derivations, and it is an instance of the above construction with Grp instead of Mon:

The objects in X-Mod are of the form

$$A = \left(X \xrightarrow{d} B, \varphi : B \longrightarrow Aut(X)\right)$$

where $d : X \longrightarrow B$ is a group homomorphism, together with a group action of B in X denoted by $b \cdot x$ satisfying

$$\begin{aligned} d(b \cdot x) &= bd(x)b^{-1} \\ d(x) \cdot x' &= x + x' - x; \end{aligned}$$

a morphism $f : A \longrightarrow A'$ in X-Mod is of the form

$$f = (f_1, f_0)$$

where $f_1 : X \longrightarrow X'$ and $f_0 : B \longrightarrow B'$ are group homomorphisms such that

$$f_0 d = d' f_1$$

and

$$f_1(b \cdot x) = f_0(b) \cdot f_1(x).$$

Clearly there are functors

$$map : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Grp$$

sending (A, A') to the group of pairs (f_1, f_0) of maps (not necessarily homomorphisms) $f_1 : UX \longrightarrow UX'$ and $f_0 : UB \longrightarrow UB'$ such that

$$f_0 d = d' f_1,$$

with the group operation defined componentwise

$$(f_1, f_0) + (g_1, g_0) = (f_1 + g_1, f_0 + g_0).$$

Also there is a functor

$$M : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Grp$$

sending (A, A') to the group $M(A, A') = \{t \mid t : UB \longrightarrow UX' \text{ is a map}\}$, and a natural transformation

$$D : M \longrightarrow map$$

defined by

$$D(A, A')(t) = (td, dt).$$

Now, define $H(A, A')$ as

$$\{(t, f) \mid t \in M(A, A') \ , \ f = (f_1, f_0) : A \longrightarrow A' \ , \ (td + f_1, dt + f_0) \in \hom(A, A')\}.$$

It is well known that the map $t : B \longrightarrow X'$ is such that

$$t(bb') = t(b) + f_0(b) \cdot t(b') \quad , \quad \text{for all } b, b' \in B,$$

while $(td+f_1, dt+f_0) \in \hom(A, A')$ asserts that the pair $(td+f_1, dt+f_0)$ is a morphism of crossed modules

(3.5)
$$\begin{tikzcd}
X \arrow[r, "d"] \arrow[d, "td+f_1"'] & B \arrow[d, "dt+f_0"] \\
X' \arrow[r, "d"] & B'
\end{tikzcd}$$

and it is equivalent to

- $dt+f_0$ is a homomorphism of groups
$$dt\,(bb') = d\,(t\,(b) + f_0\,(b) \cdot t\,(b'))$$
- $td+f_1$ is a homomorphism of groups
$$t\,(d\,(x)\,d\,(x')) = t\,(dx) + f_0 d\,(x) \cdot td\,(x')$$
- the square (3.5) commutes, which is trivial because $(f_1, f_0) \in \hom(A, A')$
- $(td+f_1)$ preserves the action of $(dt+f_0)$
$$t\left(bd\,(x)\,b^{-1}\right) = t\,(b) + f_0\,(b) \cdot t\,(d\,(x)) + f_0\left(bd\,(x)\,b^{-1}\right) \cdot (-t\,(b))\,.$$

3.4. Abstract 2-cells and homotopies. In particular, if $\mathbf{C}$ is an Ab-category, a 2-Ab-category as defined in [**5**] and [**7**] is obtained in this way; in that case the functor hom is in fact a functor

$$\hom : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Ab.$$

Giving a 2-cell structure is then to give a functor (usually required to be an Ab-functor) $H : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Ab$, and a natural transformation $D : H \longrightarrow \hom$. This 2-cell structure makes $\mathbf{C}$ a 2-category (in fact a 2-Ab-category) if in addition the condition (3.4) is satisfied, which in the abelian context simplifies to $D\,(y)\,x = yD\,(x)$. Furthermore, as proved in [**5**], every 2-cell structure (if enriched in Ab) is obtained in this way.

Of course, these considerations are also valid for any monoidal category $\mathbf{V}$, except that in general not all 2-cell structures are obtained in this way.

3.4.1. *The case of Abelian Chain Complexes.* The example of abelian chain complexes (say of order 2 to simplify notation) is self explanatory (see also [**13**],[**14**] and references there). We have objects, morphisms and 2-cells (homotopies) as displayed

(3.6)
$$\begin{tikzcd}
A_2 \arrow[r, "d"] \arrow[d, shift right, "f_2"'] \arrow[d, shift left, "g_2"] & A_1 \arrow[r, "d"] \arrow[d, shift right, "f_1"'] \arrow[d, shift left, "g_1"] \arrow[dl, "t_2"'] & A_0 \arrow[d, shift right, "f_0"'] \arrow[d, shift left, "g_0"] \arrow[dl, "t_1"'] \\
A'_2 \arrow[r, "d"] & A'_1 \arrow[r, "d"] & A'_0
\end{tikzcd}$$

with

$$\begin{aligned}
dt_1 &= g_0 - f_0 \\
t_1 d + dt_2 &= g_1 - f_1 \\
t_2 d &= g_2 - f_2
\end{aligned}$$

or equivalently

$$\begin{aligned} g_0 &= dt_1 + f_0 \\ g_1 &= t_1 d + dt_2 + f_1 \\ g_2 &= t_2 d + f_2 \end{aligned}$$

and hence we have, for $\mathbf{C}$ =2-Ch(Ab), the functor

$$H : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Ab$$

sending the pair of objects (A, A') to the abelian group of pairs (t_2, t_1); and the natural transformation

$$D : H \longrightarrow \text{hom}$$

sending a pair (t_2, t_1) as above to the triple $(t_2 d, t_1 d + dt_2, dt_1)$ displayed as follows

$$\begin{tikzcd} A_2 \arrow[r, "d"] \arrow[d, "t_2 d"'] & A_1 \arrow[r, "d"] \arrow[d, "t_1 d + dt_2"] & A_2 \arrow[d, "dt_1"'] \\ A'_2 \arrow[r, "d"] & A'_1 \arrow[r, "d"] & A'_0 \end{tikzcd} .$$

This is in fact an instance of the above construction, however, condition

$$D(x)y = xD(y)$$

is not satisfied in general, and it becomes, for $x = (t_2, t_1)$, $y = (s_2, s_1)$

$$(t_2 ds_2, dt_2 s_1 + t_1 ds_1) = (t_2 ds_2 + t_2 s_1 d, t_1 ds_1)$$

which holds if $t_2 s_1 = 0$, but not in general.

The commutator (see below) in this case is

$$[x, y] = (-t_2 s_1 d, dt_2 s_1) .$$

3.5. Abstract 2-cells and commutators. Previous examples apply to arbitrary (even large) categories, provided they admit the functors and the natural transformations as specified. Interesting examples also appear if one tries to particularize the category $\mathbf{C}$. For example if $\mathbf{C}$ has only one object, or if it is a preorder; the first case gives something that particularizes to a (strict) monoidal category (with fixed set of objects) in the presence of the naturality condition; while the second case gives something that particularizes to an enriched category over monoids. The simplest case, when $\mathbf{C}$=**1**, gives Monoids and Commutative Monoids under the naturality condition; so in particular, if considering only invertible 2-cell structures the result is Groups and Abelian Groups, respectively.

The well known reflection

$$Gr \xrightarrow{I} Ab,$$

accordingly to G. Janelidze, generalizes to a reflection

$$\text{2-cellstruct}(\mathbf{C}) \xrightarrow{I} \text{nat-2-cellstruct}(\mathbf{C})$$

from the category of 2-cell structures over $\mathbf{C}$, into the subcategory of natural 2-cell structures over $\mathbf{C}$, sending each 2-cell structure to its "naturalization"; and, under the assumption that all the 2-cells are invertible, one may consider for each

$$\begin{tikzcd} A \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Downarrow y"] & B \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Downarrow x"] & C \end{tikzcd}$$

the commutator

$$\begin{aligned} [x,y] &= (c_1+d_2-d_1-c_2)(x,y) \\ &= c_1(x,y)+d_2(x,y)-d_1(x,y)-c_2(x,y) \end{aligned}$$

where

$$\begin{aligned} c_1(x,y) &= \operatorname{cod}(x)\, y\ ,\ c_2(x,y)=x\operatorname{cod}(y) \\ d_1(x,y) &= \operatorname{dom}(x)\, y\ ,\ d_2(x,y)=x\operatorname{dom}(y)\,, \end{aligned}$$

and the comparison with $0_{\operatorname{cod}(x)\operatorname{cod}(y)}$ tell us the obstruction that x and y offer to be horizontally composed.

We will not developed this concept further, at the moment we are only observing that in the case of **C** being an Ab-category (see [**5**],[**7**] and Section 3.4 above) the notion of commutator reduces to

$$[x,y]=D(x)\,y-xD(y)\,.$$

In fact the notion of 2-Ab-category (as introduced in [**5**]) may be pushed further in the direction of a sesquicategory enriched in any category **A** with a "forgetful" functor into Sets.

It is a simple generalization of Section 3.3 above and it is as follows.

For a category **A** with a "forgetful" functor into Sets, $U:\mathbf{A}\longrightarrow$Sets, assume the existence of a functor

$$map:\mathbf{C}^{op}\times\mathbf{C}\longrightarrow\mathbf{A}$$

such that

$$\hom_C(A,B)\subseteq U map(A,B)$$

(as in Section 3.3).

If **A** were monoidal and **C** a category enriched in **A** then we would always be in the above conditions, simply by choosing $map=\hom$. It is then reasonably to say that, in this more general context, the category **C** is weakly enriched in **A** (for example, in this sense, Groups are weakly enriched in Groups, and every algebraic structure is weakly enriched in itself). In this conditions, we may be interested in considering only 2-cell structures over **C** that are "weakly enriched" in **A** in the same way as **C** is. This concept is obtained if considering only the 2-cell structures that are given by

$$H(A,B)=\{x\in UM(A,B)\mid U\operatorname{dom}x, U\operatorname{cod}x\in\hom(A,B)\}$$

for some $M,\operatorname{dom},\operatorname{cod}$ being part of an internal category object in $\mathbf{A}^{\mathbf{C}^{op}\times\mathbf{C}}$, of the form

$$M\times_{map}M\xrightarrow{+}M\ \overset{\overset{\operatorname{dom}}{\longrightarrow}}{\underset{\underset{\operatorname{cod}}{\longrightarrow}}{\overset{0}{\longleftarrow}}}\ map,$$

with the obvious restrictions after applying U.

It is interesting now to observe that in the case of $\mathbf{A} = Grp$ the result of this is precisely the construction of Section 3.3. If $\mathbf{A}$ =Ab and also requiring M to be an Ab-functor, then the result is a 2-Ab-category if also adding the condition

$$D\left(x\right)y = xD\left(y\right)$$

for all appropriate x and y.

3.5.1. *The one-object case.* In the case of a one object category, we may identify it with a monoid, say M and hence giving a 2-cell structure (let us say in the context of Section 3.2 with Ab instead of Mon, for simplicity) over it is to give (see also Proposition 1) an abelian group H (where M acts on the left and on the right) and a map

$$D : H \times M \longrightarrow M$$

satisfying

$$\begin{aligned} D\left(0,f\right) &= f \\ D\left(x' + x,f\right) &= D\left(x', D\left(x,f\right)\right) \end{aligned}$$

which is simply an action of the group H on the monoid M. Note that by naturality of D we also have $gD\left(x,f\right)h = D\left(gxf, gfh\right)$. The commutator in this case is given by

$$\left[x,y\right] = D\left(y,g\right)x + yf - yD\left(x,f\right) - gx.$$

In the case D is the trivial action we obtain the familiar notion of M semimodule.

4. Morphisms between 2-cell structures and cartesian 2-cell structures

For a fixed category $\mathbf{C}$, there is the category 2-cellstruct($\mathbf{C}$) of all possible 2-cell structures over $\mathbf{C}$, as well as the subcategory nat-2-cellstruct($\mathbf{C}$) of natural 2-cell structures over C and inv-2-cellstruct(C) of all the invertible 2-cell structures over C. The category 2-cellstruct($\mathbf{C}$) has a initial object (the discrete 2-cell structure) and a terminal object (the codiscrete 2-cell structure). If C is of the form Cat(B) for some category B, it also has the canonical 2-cell structure of internal transformations and the canonical natural 2-cell structure of internal natural transformations.

4.1. The category of 2-cell structures over a fixed category C.

The objects of 2-cellstruct($\mathbf{C}$) are of the form

$$\mathbf{H} = \left(H, \mathrm{dom}, \mathrm{cod}, 0, +\right)$$

where

$$H : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Set$$

is a functor and

$$\begin{tikzcd} H \times_{\mathrm{hom}} H \arrow[r, "+"] & H \arrow[r, shift left=2, "\mathrm{dom}"] \arrow[r, shift right=2, "\mathrm{cod}"'] & \hom_{\mathbf{C}} \arrow[l, "0"'] \end{tikzcd}$$

are natural transformations, such that

$$\left(\hom_{\mathbf{C}}, H, \mathrm{dom}, \mathrm{cod}, 0, +\right)$$

is a category object in the functor category $\mathrm{Set}^{C^{op}\times C}$,or , in other words, is an object in $\mathrm{Cat}\big(Set^{C^{op}\times C}\big)$.
A morphism $\varphi : \mathbf{H} \longrightarrow \mathbf{H}'$ is a natural transformation

$$\varphi : H \longrightarrow H'$$

such that

$$\begin{aligned} \mathrm{dom}'\,\varphi &= \mathrm{dom} \\ \mathrm{cod}'\,\varphi &= \mathrm{cod} \\ \varphi 0 &= 0' \\ \varphi + &= +'\left(\varphi \times \varphi\right). \end{aligned}$$

We will often write simply H to refer to a 2-cell structure, whenever confusion is unlikely to appear.

The purpose of describing 2-cellstruct($\mathbf{C}$), the category of all 2-cell structures over a given category $\mathbf{C}$, is the study of pseudocategories in $\mathbf{C}$. The notion of pseudocategory in a category $\mathbf{C}$ depends of the 2-cell structure considered over $\mathbf{C}$. For example, a pseudocategory in $\mathbf{C}$ with the codiscrete 2-cell structure is a precategory, while, if considering the discrete 2-cell structure, it is an internal category. It seems to be interesting to study, for a given category $\mathbf{C}$, how the notion of pseudocategory changes from a precategory to an internal category by changing the 2-cell structure considered over $\mathbf{C}$. This topic was studied in [**9**] for the case of weakly Mal'cev sesquicategories.

Also, every morphism

$$\varphi : H \longrightarrow H' \tag{4.1}$$

in 2-cellstruct($\mathbf{C}$) induces a functor

$$PsCat\left(\mathbf{C}, H\right) \longrightarrow PsCat\left(\mathbf{C}, H'\right) \tag{4.2}$$

from pseudocategories in $\mathbf{C}$ relative to the 2-cell structure H to pseudocategories in $\mathbf{C}$ relative to the 2-cell structure H'.

In some future work we plan to investigate the notion of equivalent 2-cell structures, saying that (4.1) is an equivalence whenever (4.2) is, and relate it with homotopy theory.

The notion of a pseudocategory ([**6**],[**5**]) rests in the construction of the induced 2-cells between pullback objects, thus the following definition.

4.2. Cartesian 2-cell structure. It will be useful to consider 2-cell structures such that the functor $H\left(D, \_\right) : \mathbf{C} \longrightarrow Set$ preserves pullbacks for every object D in $\mathbf{C}$, that is: the functor

$$H : \mathbf{C}^{op} \times \mathbf{C} \longrightarrow Set,$$

giving a 2-cell structure to a category $\mathbf{C}$, has the following property

$$H\left(D, A \times_{\{f,g\}} B\right) \overset{\varphi}{\cong} \left\{(x, y) \in H\left(D, A\right) \times H\left(D, B\right) \mid fx = gy\right\}$$

for every object D in $\mathbf{C}$ and pullback diagram

$$\begin{tikzcd} A \times_C B \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & B \arrow[d, "f"] \\ A \arrow[r, "g"] & C \end{tikzcd} ,$$

where φ is required to be a natural isomorphism, that is, for every $h : D \longrightarrow D'$, the following square commutes

$$\begin{tikzcd}
H(D, A \times_C B) \arrow[r, "\cong\varphi"] \arrow[d, "H(h,1)"'] & \{(x,y) \mid fx = gy\} \arrow[d] \\
H(D', A \times_C B) \arrow[r, "\cong\varphi"] & \{(x',y') \mid fx' = gy'\}
\end{tikzcd}$$

or in other words, that

$$\langle x, y \rangle h = \langle xh, yh \rangle$$

as displayed in the diagram below

$$\begin{tikzcd}
D' \arrow[r, "h"] & D \arrow[dr, Rightarrow, "<x,y>"'] \arrow[drr, Rightarrow, bend left, "y"] \arrow[ddr, Rightarrow, bend right, "x"'] & & \\
& & A \times_C B \arrow[r] \arrow[d] & B \arrow[d, "g"] \\
& & A \arrow[r, "f"'] & C
\end{tikzcd}$$

.

In particular, for $D = A' \times_{C'} B'$, and appropriate x, y, z as in

$$\begin{tikzcd}
A' \arrow[d, Rightarrow, "x"'] & A' \times'_C B' \arrow[l, "\pi'_1"'] \arrow[r, "\pi'_2"] \arrow[d, Rightarrow, "x \times_z y"] & B' \arrow[d, Rightarrow, "y"] & , & C' \arrow[d, Rightarrow, "z"] & , \\
A & A \times_C B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B & , & C &
\end{tikzcd}$$

it follows that $x \times_z y$ is the unique element (2-cell) in $H\,(A' \times_{C'} B', A \times_C B)$ satisfying

$$\begin{aligned}
\pi_2\,(x \times_z y) &= y\pi'_2 \\
\pi_1\,(x \times_z y) &= y\pi'_1 .
\end{aligned}$$

Let $\mathbf{C}$ be a category.

Definition 5 (cartesian 2-cell structure). *A 2-cell structure* $(H, \mathrm{dom}, \mathrm{cod}, 0, +)$ *over the category* $\mathbf{C}$ *is said to be* Cartesian *if the functor* $H\,(D, \_) : \mathbf{C} \longrightarrow Set$ *preserves pullbacks for every object* D *in* $\mathbf{C}$.

5. Pseudocategories

The notion of pseudocategory (as introduced in [**6**]) is only defined internally to a 2-category. Here we extend it to the more general context of a category with a 2-cell structure (or sesquicategory).

First consider three leading examples.

In any category $\mathbf{C}$, it is always possible to consider two different 2-cell structures, namely the discrete one, obtained when $H = \mathrm{hom}$ and $\mathrm{dom}, \mathrm{cod}, 0, +$ are all identities, and the codiscrete one, obtained when $H = \mathrm{hom} \times \mathrm{hom}$, dom is second projection, cod is first projection, 0 is diagonal and $+$ is uniquely determined. A pseudocategory in the first case is an internal category in $\mathbf{C}$, while in the second case is simply a precategory in $\mathbf{C}$.

In the case of $\mathbf{C} = Cat$, and choosing the natural transformations to be the 2-cell structure, a pseudocategory becomes a pseudo-double-category (see [**6**] and references there), which is at the same time a generalization of a double-category and a bicategory.

At this level of generality, it becomes clear that there is no particular reason why to prefer a specific 2-cell structure in a category instead of another.

For instance, in Top it is usually considered the 2-cell structure obtained from the homotopy classes of homotopies, but others may be consider as well.

Let $\mathbf{C}$ be a category with pullbacks and a cartesian 2-cell structure $(H, \mathrm{dom}, \mathrm{cod}, 0, +)$ defined over it.

Definition 6. *A pseudocategory in* $\mathbf{C}$*, with respect to the 2-cell structure* $(H, \mathrm{dom}, \mathrm{cod}, 0, +)$*, is a system*

$$(C_0, C_1, d, c, e, m, \alpha, \lambda, \rho)$$

where C_0, C_1 *are objects,* d, c, e, m *are morphisms, displayed as*

$$\begin{tikzcd} C_2 \arrow[r, "m"] & C_1 \arrow[r, shift left=2, "d"] \arrow[r, shift right=2, "d"'] & C_0 \arrow[l, "e" description] \end{tikzcd}, \; de = 1 = ce, dm = d\pi_2, cm = c\pi_1$$

and α, λ, ρ *are natural and invertible 2-cells, in the sense that*

$$\alpha \in H(C_3, C_1) \text{ and } \lambda, \rho \in H(C_1, C_1)$$

with

$$\begin{aligned} \mathrm{dom}(\alpha) &= mm_1 \ , \ \mathrm{cod}(\alpha) = mm_2 \\ \mathrm{dom}(\lambda) &= me_2 \ , \ \mathrm{dom}(\rho) = me_1 \ , \ \mathrm{cod}(\lambda) = 1_{C_1} = \mathrm{cod}(\rho) \end{aligned}$$

satisfying the following conditions

$$\begin{aligned} d\lambda &= 0_d = d\rho \\ c\lambda &= 0_c = c\rho \\ d\alpha &= 0_{d\pi_2 p_2} \ , \ c\alpha = 0_{c\pi_1 p_1} \end{aligned}$$

$$\lambda e = \rho e$$

$$m(\alpha \times 0_1) + \alpha(1 \times m \times 1) + m(0_1 \times \alpha) = \alpha(m \times 1 \times 1) + \alpha(1 \times 1 \times m) \tag{5.1}$$

$$m(\rho \times 0_1) + \alpha i_0 = m(0_1 \times \lambda), \tag{5.2}$$

with $C_2, C_3, \pi_1, \pi_2, p_1, p_2$ *obtained by the pullback squares*

$$\begin{tikzcd} C_2 \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & C_1 \arrow[d, "c"] \\ C_1 \arrow[r, "d"] & C_0 \end{tikzcd} \qquad \begin{tikzcd} C_3 \arrow[r, "p_2"] \arrow[d, "p_1"'] & C_2 \arrow[d, "\pi_1"] \\ C_2 \arrow[r, "\pi_2"] & C_1 \end{tikzcd}$$

and e_1, e_2, m_1, m_2, i_0 the following induced morphisms

$$\begin{aligned}
e_1 &= \langle 1, ed\rangle : C_1 \longrightarrow C_2\\
e_2 &= \langle ec, 1\rangle : C_1 \longrightarrow C_2\\
m_1 &= 1\times m : C_3 \longrightarrow C_2\\
m_2 &= m\times 1 : C_3 \longrightarrow C_2\\
i_0 &= \langle e_1, e_2\rangle : C_2 \longrightarrow C_3.
\end{aligned}$$

Some remarks:

A 2-cell $x \in H(A,B)$ is invertible when there is a (necessarily unique) element

$$-x \in H(A,B)$$

such that $\operatorname{dom}(x) = \operatorname{cod}(-x)$, $\operatorname{cod}(x) = \operatorname{dom}(-x)$ and

$$x + (-x) = 0_{\operatorname{cod}(x)} \ , \ (-x) + x = 0_{\operatorname{dom}(x)};$$

A 2-cell $x \in H(A,B)$ is natural when

$$\operatorname{cod}(x)\, y + x \operatorname{dom}(y) = x \operatorname{cod}(y) + \operatorname{dom}(x)\, y \tag{5.3}$$

for every element $y \in H(X,A)$ for every object X in $\mathbf{C}$.

The 2-cells α, λ, ρ may also be presented as

$$\begin{tikzcd}
C_3 \arrow[r, bend left, "mm_1", ""{name=U, below}] \arrow[r, bend right, "mm_2"', ""{name=D}] \arrow[Rightarrow, from=U, to=D, "\alpha"] & C_1
\end{tikzcd} \ , \ \begin{tikzcd}
C_1 \arrow[r, bend left, "me_2", ""{name=U, below}] \arrow[r, bend right, "1"', ""{name=D}] \arrow[Rightarrow, from=U, to=D, "\lambda"] & C_1
\end{tikzcd} \ , \ \begin{tikzcd}
C_1 \arrow[r, bend left, "me_1", ""{name=U, below}] \arrow[r, bend right, "1"', ""{name=D}] \arrow[Rightarrow, from=U, to=D, "\rho"] & C_1
\end{tikzcd} \ .$$

Equations (5.1) and (5.2) correspond to the internal versions of the famous MacLane's coherence pentagon and triangle, presented diagrammatically as follows

(5.4)
$$\begin{tikzcd}
 & \bullet \arrow[rr, "m(0_{C_1}\times_{C_0}\alpha)"] \arrow[dl, "\alpha(1_{C_1}\times_{C_0}1_{C_1}\times_{C_0}m)"'] & & \bullet \arrow[dr, "\alpha(1_{C_1}\times_{C_0}m\times_{C_0}1_{C_1})"] & \\
\bullet \arrow[drr, "\alpha(m\times_{C_0}1_{C_1}\times_{C_0}1_{C_1})"'] & & & & \bullet \arrow[dll, "m(\alpha\times_{C_0}0_{C_1})"] \\
 & & \bullet & &
\end{tikzcd}$$

(5.5)
$$\begin{tikzcd}
\bullet \arrow[rr, "\alpha i_0"] \arrow[dr, "m(0_{C_1}\times_{C_0}\lambda)"'] & & \bullet \arrow[dl, "m(\rho\times_{C_0}0_{C_1})"] \\
 & \bullet &
\end{tikzcd}$$

and restated in terms of generalized elements as

(pentagon)
$$\begin{tikzcd}
 & f(g(hk)) \arrow[r, "f\alpha_{g,h,k}"] \arrow[dl, "\alpha_{f,g,hk}"'] & f((gh)k) \arrow[dr, "\alpha_{f,gh,k}"] & \\
(fg)(hk) \arrow[drr, "\alpha_{fg,h,k}"'] & & & (f(gh))k \arrow[dl, "\alpha_{f,g,h}k"] \\
 & & ((fg)h)k &
\end{tikzcd}$$

(midle triangle)
$$\begin{tikzcd}
f(1g) \arrow[rr, "\alpha_{f,1,g}"] \arrow[dr, "f\lambda_g"'] & & (f1)g \arrow[dl, "\rho_f g"] \\
 & fg &
\end{tikzcd}$$

where $m\langle f,g\rangle = fg$.

To check that a given 2-cell $x \in H(A,B)$ is natural is, in general, a complicated task: we have to analyze equation (5.3) for every possible y. On the other hand, removing naturality conditions for α, λ, ρ, we loose the Coherence Theorem [**4**] and there is no longer guaranteed that for example the following diagrams are commutative

(5.6)
$$\begin{tikzcd}
1(fg) \arrow[rr, "\alpha_{1,f,g}"] \arrow[dr, "\lambda_{fg}"'] & & (1f)g \arrow[dl, "\lambda_f g"] \\
 & fg &
\end{tikzcd}$$

(5.7)
$$\begin{tikzcd}
f(g1) \arrow[rr, "\alpha_{f,g,1}"] \arrow[dr, "f\rho_g"'] & & (fg)1\ . \arrow[dl, "\rho_{fg}"] \\
 & fg &
\end{tikzcd}$$

This diagrams, when internalized, correspond, respectively, to the following equations

$$\begin{aligned}
m(\lambda \times 0_{C_1}) + \alpha i_2 &= \lambda m, \\
\rho m + \alpha i_1 &= m(0_{C_1} \times \rho)
\end{aligned}$$

and since the 2-cells are invertible, the above set of equations may be presented as

$$\begin{aligned}
\alpha i_2 &= -m(\lambda \times 0_{C_1}) + \lambda m, \\
\alpha i_1 &= -\rho m + m(0_{C_1} \times \rho).
\end{aligned}$$

Now, in the context of a weakly Mal'cev sesquicategory, as it is proved in [**9**], the 2-cell α is uniquely determined by αi_2 and αi_1 and hence it only depends on λ, ρ and m.

This suggests to consider a (non natural) version of a pseudocategory in a sesquicategory.

Let $\mathbf{C}$ be a category with pullbacks and a cartesian 2-cell structure $(H, \mathrm{dom}, \mathrm{cod}, 0, +)$ defined over it.

A (non natural) pseudo category internal to $\mathbf{C}$ and relative to the 2-cell structure $(H, \mathrm{dom}, \mathrm{cod}, 0, +)$ is a system

$$(C_0, C_1, d, c, e, m, \alpha, \lambda, \rho)$$

where C_0, C_1 are objects, d, c, e, m are morphisms as in the definition above, and, α, λ, ρ are invertible 2-cells $\alpha \in H(C_3, C_1)$ and $\lambda, \rho \in H(C_1, C_1)$ satisfying the following conditions (with $C_2, C_3, \pi_1, \pi_2, p_1, p_2, e_1, e_2, m_1, m_2, i_0$ defined as above)

$$\begin{aligned} \mathrm{dom}(\alpha) &= mm_1\ ,\ \mathrm{cod}(\alpha) = mm_2 \\ \mathrm{dom}(\lambda) &= me_2\ ,\ \mathrm{dom}(\rho) = me_1\ ,\ \mathrm{cod}(\lambda) = 1_{C_1} = \mathrm{cod}(\rho) \end{aligned}$$

$$\begin{aligned} d\lambda &= 0_d = d\rho \\ c\lambda &= 0_c = c\rho \\ d\alpha &= 0_{d\pi_2 p_2}\ ,\ c\alpha = 0_{c\pi_1 p_1} \\ &\lambda e = \rho e \end{aligned}$$

$$m(\alpha \times 0_1) + \alpha(1 \times m \times 1) + m(0_1 \times \alpha) = \alpha(m \times 1 \times 1) + \alpha(1 \times 1 \times m)$$

(5.8)

(5.9) $$\alpha i_0 = -m(\rho \times 0_1) + m(0_1 \times \lambda)$$

(5.10) $$\alpha i_2 = -m(\lambda \times 0_{C_1}) + \lambda m$$

(5.11) $$\alpha i_1 = -\rho m + m(0_{C_1} \times \rho),$$

(5.12) $$\lambda \circ \lambda, \lambda \circ \rho, \rho \circ \rho, \rho \circ \lambda.$$

Of course that in the case α, λ, ρ are natural 2-cells then the last three conditions are redundant and we obtain Definition 6 above.

6. Conclusion

We conclude this note by giving three results of application.

The first example is an instance of section 3.1, the second example is an application of section 3.2, while the third one is from 3.4.

In the setting of section 3.1, let $\mathbf{B}$ be a weakly Mal'cev category (see [**8**], or simply assume that $\mathbf{B}$ is a Mal'cev category) and consider $\mathbf{C}$ =Cat($\mathbf{B}$) and $(H, \mathrm{dom}, \mathrm{cod}, 0, +)$ as in section 3.1.

In particular, $\mathbf{C}$ as above with internal transformations (not necessarily natural), is a weakly Mal'cev sesquicategory and the following result is proved in [**9**].

Theorem 1. *A (non natural) pseudo category internal to* $\mathbf{C}$ *and relative to the specific 2-cell structure as above, satisfying the additional condition*

$$\lambda e = 0_e = \rho e$$

is completely determined by a reflexive graph in $\mathbf{C}$

$$\begin{tikzcd} C_1 \arrow[r, shift left=2, "d"] \arrow[r, shift right=2, "c"'] & C_0 \arrow[l, "e" description] \end{tikzcd} \quad , \quad de = 1_{C_0} = ce$$

together with 2-cells

$$\lambda, \rho \in H(C_1, C_1)$$

satisfying the following conditions,

$$\mathrm{cod}(\lambda) = 1_{C_1} = \mathrm{cod}(\rho)$$

$$\begin{aligned} d\lambda &= 0_d = d\rho \\ c\lambda &= 0_c = c\rho \\ \lambda e &= 0_e = \rho e \end{aligned}$$

and furthermore it is equipped with a morphism

$$m : C_2 \longrightarrow C_1$$

uniquely determined by

$$me_1 = u \ , \ me_2 = v$$

and a 2-cell $\alpha \in H(C_3, C_1)$, uniquely determined by

$$\alpha i_1 e_1 = -\rho u \ , \ \alpha i_2 e_1 = -u\lambda + \lambda u \ , \ \alpha i_2 e_2 = \lambda v$$

where $v = \operatorname{dom}(\lambda)$ and $u = \operatorname{dom}(\rho)$.

Note that this result reflects the striking fact that, in this case,

$$5.10 + 5.11 + 5.12 \Rightarrow 5.8 + 5.9,$$

and if α, λ, ρ are natural then the description above is in fact a pseudocategory.

The second example describes pseudocategories in Groups, and it gives the notion of crossed module with the freedom to choose an element δ in the centre of X.

If considering the category of Groups with the 2-cells structure given by derivations as in 3.2 then an internal pseudocategory is completely determined by a group homomorphism

$$X \xrightarrow{\partial} B,$$

an action of B in X (denoted by $b \cdot x$) and a distinguished element in X, δ satisfying the following conditions

$$\begin{aligned} \partial\delta &= 0 \\ x &= \delta + x - \delta \\ \partial(b \cdot x) &= b\partial(x)b^{-1} \\ \partial(x) \cdot \bar{x} &= x + \bar{x} - x. \end{aligned}$$

Where the objects are the elements of B, the arrows are pairs $(x, b) : b \longrightarrow \partial x + b$ and the composition of $(x', \partial x + b) : \partial x + b \longrightarrow \partial x' + \partial x + b$ with $(x, b) : b \longrightarrow \partial x + b$ is $(x' + x - \lambda + b \cdot \lambda, b) : b \longrightarrow \partial x' + \partial x + b$. The isomorphism between $(0, \partial x + b) \circ (x, b) = (x, b) \circ (0, b)$ and (x, b) is the element $(\delta, 0) \in X \rtimes B$. Associativity is satisfied, since $(x'', \partial x' + \partial x + b) \circ ((x', \partial x + b) \circ (x, b)) = ((x'', \partial x' + \partial x + b) \circ (x', \partial x + b)) \circ (x, b)$.

In the case of an additive category $\mathbf{A}$ with kernels and a 2-cell structure given by an Ab-functor

$$H : \mathbf{A}^{op} \times \mathbf{A} \longrightarrow Ab$$

and a natural transformation

$$D : H \longrightarrow \hom_A$$

and using the notation

$$[x, y] = D(x)y - xD(y)$$

a pseudo category is completely determined by

$$A \xrightarrow{h} B$$

$$\begin{aligned} \lambda, \rho &\in H(A,A) \\ \eta &\in H(B,A) \end{aligned}$$

$$\begin{aligned} h\lambda &= 0 \\ h\rho &= 0 \\ h\eta &= 0 \end{aligned}$$

where α is given by $(5.9), (5.10)$ and (5.11) and (5.8) translates to

$$(1 - D\rho)[\rho, \rho] = 0$$
$$D\lambda[\rho, \rho] = D\rho[\lambda, \rho] + (1 - D\rho)[\rho, \lambda]$$
$$(1 - D\lambda)[\lambda, \rho] + D\lambda[\rho, \lambda] = D\rho[\lambda, \lambda] + (1 - D\rho)[\rho, \eta]\, h$$
$$(1 - D\lambda)[\lambda, \lambda] = (1 - D\lambda)[\lambda, \eta]$$
$$(1 - D\rho - D\lambda)[\lambda, \eta] = (1 - D\rho - D\lambda)[\rho, \eta]$$

which is trivial as soon as we introduce (5.12).

The pseudocategory thus determined is of the form (see [**5**])

$$\begin{tikzcd} A \oplus A \oplus B \arrow[r, "m"] & A \oplus B \arrow[r, shift left=2, "(0\ 1)"] \arrow[r, shift right=2, "(\delta\ 1)"'] & B \arrow[l, "\binom{0}{1}"'] \end{tikzcd}$$

$$m = \begin{pmatrix} f & g & h \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{aligned} g &= 1 - D(\lambda) \\ f &= 1 - D(\rho) \\ h &= -D(\eta) \end{aligned}$$

$$\alpha = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \alpha_1 &= -f\rho \\ \alpha_2 &= \lambda + g\rho - \rho - f\lambda \\ \alpha_3 &= g\lambda - f\eta\delta \\ \alpha_0 &= g\eta - f\eta \end{aligned}$$

$$\lambda = \begin{pmatrix} \lambda & \eta \\ 0 & 0 \end{pmatrix} \qquad \rho = \begin{pmatrix} \rho & \eta \\ 0 & 0 \end{pmatrix}.$$

In particular the category of abelian chain complexes is of this form.

Another example of this form is the case of TAG, Topological Abelian Groups.

Let $\mathbf{C} = TAG$ the category of topological abelian groups with the 2-cell structure given by

$$H(X,Y) = \{\alpha : I \times X \longrightarrow Y \mid \alpha \text{ is continuous, } \alpha(0, \_) = 0 \text{ and } \alpha(t, \_) \text{ is a homomorphism}\} / \sim$$

where I is the unit interval and the equivalence $\sim$ is defined by

$$\alpha \sim \beta$$

if and only if $\alpha(1,\_)=\beta(1,\_)=h$, there is $\Phi : I\times I\times X\longrightarrow Y$, continuous and such that

$$\begin{aligned}\Phi(0,\_,\_) &= \alpha\\ \Phi(1,\_,\_) &= \beta\\ \Phi(s,0,\_) &= 0\\ \Phi(s,1,\_) &= h\end{aligned}$$

and $\Phi(s,t,\_)$ is a homomorphism.

The natural transformation $D : H\longrightarrow \hom$ is given by

$$D([\alpha])=\alpha(1,\_)$$

with

$$(g[\alpha]f)(t,x)=g(\alpha(t,f(x))).$$

We have

$$[\alpha]D([\beta])=D([\alpha])[\beta]$$

because

$$\alpha D(\beta)\sim D(\alpha)\beta\Leftrightarrow\alpha(t,\beta(1,x))\sim\alpha(1,\beta(t,x))$$

and there is

$$\Phi(s,t,x)=\alpha\left(t^{(1-s)},\beta(t^{s},x)\right).$$

¿From [**5**] we now conclude that a pseudocategory in TAG (with the 2-cell structure as above) is given by a morphism in TAG

$$k : A\longrightarrow B,$$

together with

$$\lambda,\rho : I\times A\longrightarrow A$$

in $H(A,A)$ and also

$$\eta : I\times B\longrightarrow A$$

in $H(B,A)$ satisfying

$$k\rho(t,\_)=0, k\lambda(t,\_)=0, k\eta(t,\_)=0.$$

The objects in the pseudocategory are the points in B while the pseudomorphisms are pairs (a,b) with domain b and codomain $k(a)+b$; the composition of

$$b\xrightarrow{(a,b)}b'\xrightarrow{(a',b')}b''$$

is given by the following formula

$$(a-\rho(1,a)+a'-\lambda(1,a')-\eta(1,b),b).$$

In particular, if we consider that A is the space of paths in B starting at zero

$$A=\{x : I\longrightarrow B\mid x\text{ continuous and }x(0)=0\}$$

with

$$k(x)=x(1)$$

and choose representatives of λ, ρ and η as

$$\begin{aligned}
\lambda\left(s,x\right)\left(t\right) &= \begin{cases} x\left(st\right)-x\left(2st\right) & \text{if} \quad t \leqslant \frac{1}{2} \\ x\left(st\right)-x\left(s\right) & \text{if} \quad t \geqslant \frac{1}{2} \end{cases} \\
\rho\left(s,x\right)\left(t\right) &= \begin{cases} x\left(st\right) & \text{if} \quad t \leqslant \frac{1}{2} \\ x\left(st\right)-x\left(2st-s\right) & \text{if} \quad t \geqslant \frac{1}{2} \end{cases} \\
\eta &= 0
\end{aligned}$$

then we obtain the usual composition of paths

$$y+x=\begin{cases} x\left(2t\right) & \text{if} \quad t \leqslant \frac{1}{2} \\ y\left(2t-1\right)+x\left(1\right) & \text{if} \quad t \geqslant \frac{1}{2} \end{cases}.$$

References

[1] Crans, S.E: A tensor product for Gray-categories, Theory and Applications of Categories, Vol.5, N^{o}.2, 1999, pp.12-69.

[2] Gray, J.W.: *Formal Category Theory: Adjointness for 2-categories*, Lecture Notes in Mathematics, Springer-Verlag, 1974

[3] Leinster, T.: *Higher Operads, Higher Categories*, London Mathematical Society Lecture Notes Series, Cambridge University Press, 2003 (electronic version).

[4] MacLane, S.: *Categories for the working Mathematician*, Springer-Verlag, 1998, 2nd edition.

[5] Martins-Ferreira, N.: Internal Weak Categories in Additive 2-Categories with Kernels, Fields Institute Communications, Volume 43, p.387-410, 2004.

[6] Martins-Ferreira, N: Pseudo-categories, *JHRS*, Vol.1(1), 2006, pp.47-78.

[7] Martins-Ferreira, N.: The (tetra)category of pseudocategories in additive 2-categories with kernels, Applied Categorical Structures (published in Online First, and to appear in print), 2008.

[8] Martins-Ferreira, N.: Weakly Mal'cev categories, Theory and Applications of Categories, Vol. 21, No. 6, pp 91-117, 2008.

[9] Martins-Ferreira, N.: Low-dimensional internal categorical structures in weakly Mal'cev sesquicategories, PhD Thesis, 2008.

[10] Street, R.H.: Cosmoi of internal categories, Trans. Amer. Math. Soc. 258, 1980, 271-318

[11] Street, R.H.: Fibrations in Bicategories, Cahiers Topologie et Géométrie Diferéntielle Catégoriques, 21:111-120, 1980.

[12] Street, R.H.: *Handbook of Algebra*, chapter Categorical Structures, pages 529–577. Elsevier Science, 1996.

[13] Bourn, D., "Another denormalization theorem for the abelian chain complexes", J. Pure and Applied Algebra 66, 1990, 229-249.

[14] Brown, R. and Higgins, Ph.J., "Cubical Abelian Groups with Connections are Equivalent to Chain Complexes", Homology, Homotopy and Applications vol. 5(1), 2003, 49-52.

Polytechnic Institute of Leiria
E-mail address: `nelsonmf@estg.ipleiria.pt`
URL: `http://www.estg.ipleiria.pt/~nelsonmf`

Two Denotational Interpretations of Proofs in Classical Logic

François Lamarche and Novak Novaković

LORIA and INRIA Nancy – Grand Est

Abstract. In this paper we present and compare two interpretations of classical logic proofs in a category of posets and relations, and relate their behavior to proof nets, extracting meaningful invariants of proofs. We show that at least one of these interpretation cannot have anything to do with the Curry-Howard correspondence.

1 Introduction

In this paper we will interpret a minor variation of the Gentzen sequent calculus for classical logic (obtained by adding the Mix rule of linear logic to LK) into a certain category whose objects are posets and whose morphisms relations between them. Although they are very similar these two interpretations have quite different good and bad points, that we will discuss. We will show that the interpretations of proofs they allow contain meaningful information about these proofs, and that this information is closely related to the kind of proof net which is presented in [LS05b,LS05a]. Some computations we will make will alow us to conclude that these interpretations cannot follow the Curry-Howard correspondence.

We have taken pains to make our presentation very elementary; for instance, although some terminology of category theory is used (and the whole of the paper is based on a category-theoretical methodology), someone who has only the barest knowledge of what a category is should be able to read this.

2 The general framework

The starting point for our work is an observation made by several people independently, that the well-known interpretation of linear logic in sets and relations could be extended to one where the sets are generalized to posets; thus there exists a notion of "relation between posets", that Lambek once called *comparisons* [Lam94].

Let $(M,\leq),(N,\leq)$ be posets. A *comparison* $f : M \to N$ is a subset $f \subseteq M \times N$ which is down-closed to the left, and up-closed to the right, i.e.,

$$\begin{array}{ll} m \; f \; n, m' \leq m & \text{implies} \;\; m' \; f \; n \\ m \; f \; n, m \leq n' & \text{implies} \;\; m \; f \; n'. \end{array}$$

Composition of these maps is the ordinary composition of relations: given F : $M \to N$ and $g : N \to P$

$$m \; {}^{gf} \; p, \text{ if } (\exists n \in N)\; m \; {}^{f} \; n, \; n \; {}^{g} \; p'.$$

The reader should check that this defines a category, that we will denote **Cmp**. In other words that composition of comparisons is associative and that every poset M is equipped with an identity comparison, that acts as a left- and right-unit, namely $\mathrm{Id}_M = \{\,(m, m') \mid m \leq m'\,\}$—this definition is surprising at first, but then the "ordinary" identity relation (diagonal) is not a comparison! Or the reader can look at [Lam07]. In the present paper, we restrict our attention to the multiplicative fragment of linear logic, since it suffices for our purposes.

We will follow the convention of using the roman typeface for syntactical objects, say $\mathrm{a}, \mathrm{A}, \Gamma \ldots$ and ordinary math italics for their semantical counterparts (here: posets), like $a, A, \mathit{\Gamma} \ldots$ As for the connectives/operations, we do not make distinctions between syntax and semantics in notation. This convention on notation will suffice for purposes of distinguishing between syntactical and semantical objects, since the only types we will be dealing with are obtained from primitive types for variables and the operations for connectives.

So the interpretation of formulas goes as:

- as usual, units $\mathbf{1}$ and $\perp$ are interpreted as a ("the") one-element set $\{*\}$
- each atomic a is interpreted as a chosen poset a;
- for two formulas A, B, tensoring is taking the cartesian product, i.e., the interpretation of $\mathrm{A} \otimes \mathrm{B}$ is $A \times B$, which naturally we will also sometimes write $A \otimes B$. Notice that this operation is (bi)functorial, in other words, given comparisons $f\colon A \to A'$ and $g\colon B \to B'$ there is a well-defined $f \otimes g\colon A \otimes B \to A' \otimes B'$ obtained by taking the cartesian product $f \times g$ (and permuting the order of the posets a little in the product).
- If A interprets the formula A then the interpretation of $\mathrm{A}^\perp$ is inverting the order, i.e. $A^\perp = A^{op}$. This is also functorial, but contravariantly so this time: given $f\colon A \to B$ there is $f^\perp\colon B^\perp \to A^\perp$
- par is, therefore, computed as follows: the interpretation of $A \parr B$ is the same as that of $(\mathrm{A}^\perp \otimes \mathrm{B}^\perp)^\perp$, which is

  $$(A^{op} \times B^{op})^{op} = A \times B = A \otimes B.$$

  In other words, both tensor and par are cartesian product. There is some degeneracy, but we are interested in the interpretation of *proofs,* not *provability.*
- the reader should check that we do indeed have the standard (natural) bijection

  $$\frac{A \otimes B \to C}{\overline{A \to B^\perp \parr C.}}$$

which defines an adjunction. Going down this "invertible rule" is called *currying*, and going up *uncurrying.* Readers who are knowledgeable about this know we have shown that **Cmp** is a $*$-*autonomous category* [Bar79].

Remark 1. Naturally a proof of a formula A is interpreted by a map $\mathbf{1} \to A$, which is a certain subset of $\{*\} \times A$. But it is only natural to drop the first factor, and simply think of the denotation of a proof as an up-closed subset of A. *We will do this all the time.* Notice that in this view, a proof of $\mathrm{A}^\perp \parr \mathrm{B}$ is just an up-close subset of $A^{op} \times B$, which is the same as a map $A \to B$ according to our definition.

Let us use these definitions to interpret the one-sided sequent calculus for multiplicative linear logic (as a matter of fact we also need the Mix rule). Everything in what follows is *dictated* by the definitions we just gave, provided that we think of a map $A \to B$ as a proof of $\vdash \mathrm{A}^\perp, \mathrm{B}$.

$$\frac{}{\vdash \mathrm{a}^\perp, \mathrm{a}} \quad \rightsquigarrow \quad \mathrm{Id}_a = \{(x,y) \in a \times a \mid x \leq y\}$$

$$\frac{\vdash \Gamma, \mathrm{A}, \mathrm{B}}{\vdash \Gamma, \mathrm{A} \parr \mathrm{B}}\parr \quad \rightsquigarrow \quad \text{do nothing}$$

$$\frac{\vdash \Gamma, \mathrm{A} \quad \vdash \mathrm{B}, \Sigma}{\vdash \Gamma, \mathrm{A} \otimes \mathrm{B}, \Sigma}\otimes \quad \rightsquigarrow \quad \begin{array}{l}\text{given } f \text{ for } \mathit{\Gamma} \times A \text{ and } g \text{ for } B \times \mathit{\Sigma},\ \text{take } f \times g \\ \text{for } \mathit{\Gamma} \times A \times B \times \mathit{\Sigma}\end{array}$$

$$\frac{\vdash \Gamma, \mathrm{A} \quad \vdash \mathrm{A}^\perp, \Sigma}{\vdash \Gamma, \Sigma}\mathit{Cut} \rightsquigarrow \quad \begin{array}{l}\text{given } f \text{ for } \mathit{\Gamma} \times A \text{ and } g \text{ for } A^\perp \times \mathit{\Sigma},\ \text{take} \\ \{(\gamma,\delta) \mid \exists x \in A : (\gamma,x) \in f, (x,\delta) \in g\} \text{ for } \mathit{\Gamma} \times \mathit{\Sigma}\end{array}$$

$$\frac{\vdash \Gamma \quad \vdash \Sigma}{\vdash \Gamma, \Sigma}\mathit{Mix} \quad \rightsquigarrow \quad \begin{array}{l}\text{given } f \text{ for } \Gamma \text{ and } g \text{ for } \Sigma,\ \text{take } f \times g \\ \text{for } \mathit{\Gamma} \times \mathit{\Sigma}.\end{array}$$

Notice that in the definition for Cut, the variable x is seen as belonging both to the poset a and its opposite a^{op}, and the same goes for the Axiom rule. Also notice that the Mix rule would be presented in the style of the previous section as a map $A \otimes B \to A \parr B$. Naturally since here the tensor coincides with the par, it is just identity for us.

Some remarks: We will work under the implicit assumption of associativity of cartesian product, i.e., we will not distinguish between things like $(x,(y,z))$ and $((x,y),z)$, even though the corresponding types would actually differ. Consequently, the associativity of the connectives and of the comma in sequents is implicit. Also, the permutations of the formulas within a sequent, sometimes made explicit through the Exchange rule of the calculus will not be subject of our concern. The reason is the fact that we always have at our disposal the obvious natural isomorphism that permutes formulas within a sequent and commutes the arguments of a connective. As is usual in semantics, Cuts are automatically eliminated.

Some notation: we will be using $\mathcal{A}t(\mathrm{X})$ and $\mathcal{A}t(\Gamma)$ to denote set of occurrences of atoms and negatoms in a formula or a sequent, respectively. When we need to denote different occurrences of a same atom or formula, we use superscripts, e.g. $\mathrm{a}^1, \mathrm{a}^2, \ldots$ while subscripts will be used in the usual way.

The following is easy and will be left to the reader:

Proposition 1. *The interpretation of any sequent* Γ *is* $(\prod a)_{\mathrm{a}\in\mathcal{A}t(\Gamma)}$.

Let us now turn to classical logic. The calculus LC we will use is the same as in [LS05b]; it can be seen either as the one-sided version of Gentzen's LK with Mix added, or as MLL + Mix + Weakening + Contraction.

$$\frac{\vdash \Gamma, \mathrm{A}, \mathrm{A}}{\vdash \Gamma, \mathrm{A}}\ \mathit{Contr} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \mathrm{A}}\ \mathit{Weak}$$

Having Contraction in the calculus clearly requires the presence of a map $\nabla_A : A \parr A \rightarrow A$, with the interpretation of the derivation above just post-composition with that map. As for Weakening a map $\amalg_A : \perp \rightarrow A$ will clearly do the job. We add the standard condition that these maps have to obey the same kind of of associativity, commutativity and identity that we associate to commutative monoids. To illustrate what this means, look at the two derivations:

$$\frac{\dfrac{\vdash \Gamma, \mathrm{a}^1, \mathrm{a}^2, \mathrm{a}^3}{\vdash \Gamma, \mathrm{a}^4, \mathrm{a}^3}\ \mathit{Contr}_{\mathrm{a}^1,\mathrm{a}^2}}{\vdash \Gamma, \mathrm{a}^5}\ \mathit{Contr}_{\mathrm{a}^4,\mathrm{a}^3} \qquad \frac{\dfrac{\vdash \Gamma, \mathrm{a}^1, \mathrm{a}^2, \mathrm{a}^3}{\vdash \Gamma, \mathrm{a}^3, \mathrm{a}^4}\ \mathit{Contr}_{\mathrm{a}^2,\mathrm{a}^3}}{\vdash \Gamma, \mathrm{a}^5}\ \mathit{Contr}_{\mathrm{a}^3,\mathrm{a}^4}$$

where the $\mathrm{a}^1, \mathrm{a}^2, \mathrm{a}^3$ are different occurrences of the same atom. The two derivations differ in the order the contraction rules have been applied, and it is natural to want these two proofs to have the same interpretation—what's the point of semantics if we can't get rid of that ugly bureaucracy? This means, we want to have $\nabla \circ (\nabla \parr \mathrm{Id}) = \nabla \circ (\mathrm{Id} \parr \nabla)$ for every type. Notice that strictly speaking the types do not agree exactly here; we are applying our rule of not bothering with the issues related to the not-really-strict associativity of the cartesian product.

Suppose that M is an ordinary set and that $m : M \times M \rightarrow M$ is an ordinary binary operation. Instead of the usual equation in x, y and z we could always write the associativity of m as $m \circ (m \times \mathrm{Id}) = m \circ (\mathrm{Id} \times m)$. The advantage of doing so is that *we can define a monoid using greatly generalized notions of Cartesian product, to suit our needs,* for example replacing ordinary Cartesian product by $\parr$ (in the present case it seems that $\parr$ is still the Cartesian product, but actually this is not strictly true because the *maps* are different in **Cmp**). Thus we can define a notion of commutative $\parr$-monoid, using well-chosen equations between maps. They are usually presented as commutative diagrams, and the reader can consult [FP05,FP04,LS05a] and countless other papers for a full display of these diagrams.

Thus in our model of classical logic every interpretation of a type will come equipped with a commutative $\parr$-monoid structure. Since proofs are defined by induction, it will suffice to define the monoid structure for the atomics $\mathrm{a}, \mathrm{b} \ldots$ and negatomics (which, to signal the fact that we are entering classical logic, will be denoted $\bar{\mathrm{a}}, \bar{\mathrm{b}}, \ldots$) and to give a way to construct a new monoid structure after applying the connectives. Because of the linear rules we already have, the poset for $A \wedge B$ will obviously be $A \otimes B = A \times B$ and the one for $A \vee B$ will be $A \parr B = A \times B$.

Given the $⅋$-monoid structure on A and B, we define $\nabla_{A\vee B} : (A ⅋ B) ⅋ (A ⅋ B) \to A ⅋ B$ as the composition:

$$(A ⅋ B) ⅋ (A ⅋ B) \xrightarrow{\sim} (A ⅋ A) ⅋ (B ⅋ B) \xrightarrow{\nabla_A ⅋ \nabla_A} A ⅋ B.$$

The isomorphism here is the obvious associativity/permutation isomorphism of Cartesian products. As for $\amalg_{A\vee B} : \bot \to A \vee B$, we take:

$$\bot \xrightarrow{\sim} \bot ⅋ \bot \xrightarrow{\amalg_A ⅋ \amalg_B} A ⅋ B.$$

We do the same for conjunction, since here both tensor and par are interpreted by the Cartesian product. But naturally in general this is not the case, and then there is no ready-made way to go from $(A\otimes B) ⅋ (A\otimes B)$ to $A\otimes B$. This is solved by requiring the presence of a natural map $(A\otimes B) ⅋ (A\otimes B) \to (A ⅋ A)\otimes(B ⅋ B)$, which is called *Medial.* It is natural to make it explicitly part of the logic, and it actually first appeared as a deduction rule for Deep Inference calculi in classical logic [Brü03]. For the category-theoretical treatment of the map, independently of classical logic, the reader can look at [Lam07].

Our interpretation already has negation for atomics, by definition. For a composite formula the $⅋$-monoid structure on its negation is obtained recursively, using de Morgan duality: $\overline{A \wedge B} = \overline{A} \vee \overline{B}$ and $\overline{A \vee B} = \overline{A} \wedge \overline{B}$.

There is some advantage to considering the combined monoid structures on $A, \overline{A}$ as a *single* algebraic structure on the object A. Notice that a map $\nabla \colon \overline{A} ⅋ \overline{A} \to \overline{A}$ can also be seen, by duality, as a map $\Delta \colon A \to A \otimes A$, and the same goes for $\amalg \colon \bot \to \overline{A}$, which becomes $\Pi \colon A \to \mathbf{1}$. The associativity, commutativity and unit laws can be translated in this "reversed" contexts by duality; for decades algebraists have called these laws *coassociativity, cocoummutativity and co-unit,* while the resulting structure (A, Δ, Π) is called a *(cocommutative) comonoid.*, and the full thing $(A, \Delta, \Pi, \nabla, \amalg)$ is called a *(commutative, cocommutative) bimonoid* with Δ being the *diagonal*, ∇ the *co-diagonal*, Π the *co-unit* or the *projection*, $\amalg$ the *unit* or the *co-projection.*

Thus, given an object X in an interpretation of linear logic, turning it into an interpretation of a classical formula amounts to finding a bimonoid structure on it.

We can finally define the interpretation of Weakening and Contraction (the reader should keep in mind our convention to identify the proofs with up-closed subsets):

$\dfrac{\vdash \Gamma}{\vdash \Gamma, \mathrm{A}}$ *Weak* $\rightsquigarrow$ given f for $\mathit{\Gamma}$ take
$\{(\gamma_1, \ldots, \gamma_n, \epsilon) \mid (\gamma_1, \ldots, \gamma_n) \in f \text{ and } \epsilon \in \amalg_A\}$ for $\mathit{\Gamma} \times A$

$\dfrac{\vdash \Gamma, \mathrm{A}, \mathrm{A}}{\vdash \Gamma, \mathrm{A}}$ *Contr* $\rightsquigarrow$ given f for $\mathit{\Gamma} \times A \times A$, take
$\{(\gamma, x) \mid \exists x_1, x_2 \in A : (\gamma, x_1, x_2) \in f \text{ and } (x_1, x_2) \nabla_A\, x\}$
for $\mathit{\Gamma} \times A$

Diagrammatically, the defined maps for Contraction and Weakening can be seen as compositions

$$\mathbf{1} \xrightarrow{f} \Gamma \xrightarrow{\sim} \Gamma \times \perp \xrightarrow{\mathrm{Id}_\Gamma \times \amalg_A} \Gamma \times A \quad \text{and} \quad \mathbf{1} \xrightarrow{f} \Gamma \times A \times A \xrightarrow{\mathrm{Id}_\Gamma \times \nabla_A} \Gamma \times A.$$

Before we state the following, easy to prove proposition, notice that the interpretation for the Cut rule we gave before corresponds to the composite

$$\mathbf{1} \xrightarrow{f \times g} \Gamma \times A \times A^\perp \times \Sigma \xrightarrow{\mathrm{Id}_\Gamma \times C_A \times \mathrm{Id}_\Sigma} \Gamma \times \Sigma$$

where $C_A : A \otimes A^\perp \to \perp$ is the map $\{(x, y, *) \mid x \leq y\}$ which is dual to the identity $\mathbf{1} \to A^\perp \invamp A$.

Proposition 2. *Let X be a formula and $\{a_1, \dots, a_n\} = \mathcal{A}t(X)$. Then the following diagrams commute:*

$$\begin{tikzcd}
X^1 \times X^2 \arrow[r, "\nabla_X"] \arrow[d, equal] & (a_1^3 \times \cdots \times a_n^3) = X \\
(a_1^1 \times \cdots \times a_n^1) \times (a_1^2 \times \cdots \times a_n^2) \arrow[r, "\sim"] & (a_1^1 \times a_1^2) \times \cdots \times (a_n^1 \times a_n^2). \arrow[u, "\nabla_{a_1} \times \cdots \times \nabla_{a_n}"']
\end{tikzcd}$$

$$\begin{tikzcd}
\Gamma \times X \times \overline{X} \times \Sigma \arrow[r, "C_X"] \arrow[d, equal] & \Gamma \times \Sigma \\
\Gamma \times (a_1 \times \cdots \times a_n) \times (\overline{a}_1 \times \cdots \times \overline{a}_n) \times \Sigma \arrow[r, "\sim"] & \Gamma \times (a_1 \times \overline{a}_1) \times \cdots \times (a_n \times \overline{a}_n) \times \Sigma. \arrow[u, "C_{a_1} \times \cdots \times C_{a_n}"']
\end{tikzcd}$$

Notice that the last diagram commutes modulo obvious isomorphisms that we choose to omit.

Look at the following derivation:

$$\dfrac{\dfrac{\vdash \overline{\mathrm{a}}, \mathrm{a} \quad \vdash \overline{\mathrm{a}}, \mathrm{a}}{\vdash \overline{\mathrm{a}}, \mathrm{a}, \overline{\mathrm{a}}, \mathrm{a}}\, Mix}{\vdash \overline{\mathrm{a}}, \mathrm{a}}\, 2 \times Cont.$$

Seen as a map, and using the equality of tensor and par, this derivation is the composite $\nabla_a \circ \Delta_a$. It is rather easy to construct semantics for which this map is identity, but much more difficult to get ones for which it is not. The first partially successful results on this are found in [Lam07]. The point is that we want to keep track of resources, since this derivation can be seen as the superposition of two axiom links, not just a single axiom link. In the aformentioned paper, for any number n a bimonoid is constructed such that superpositions of $1, 2, \dots, n$ axiom links (by iteration of the derivation just above) can be distinguished, but since the bimonoids are finite, there is a ceiling where the count saturates. One aim of this paper is to construct interpretations where there is no saturation.

Recall that a partition on a given set Q can be seen either as an equivalence relation on Q or as a set $\mathcal{R} \subseteq \mathcal{P}(X)$ of subsets of Q, whose elements are called classes. Let Γ be a sequent, along with a proof q of it. By induction on the proof, we define below a partition $\mathrm{Prt}_q(\Gamma)$ on the set $\mathcal{A}t(\Gamma)$. In that definition we do not bother with keeping track explicitly of q:

– if Γ is $\vdash \bar{\mathrm{a}}, \mathrm{a}$, then $\mathrm{Prt}(\Gamma) = \{\{\bar{\mathrm{a}}, \mathrm{a}\}\}$
– if Γ has been obtained by application of the tensor or the Mix rule, $\mathrm{Prt}(\Gamma)$ is the obvious "sum" partition on the disjoint union $\mathcal{A}t(\Sigma_1) \uplus \mathcal{A}t(\Sigma_2)$, where Σ_1, Σ_2 are the premiss sequents of the rule application.
– supposing that A has been added to $\vdash \Gamma$ by Weakening, $\mathrm{Prt}(\Gamma, \mathrm{A}) = \mathrm{Prt}(\Gamma) \uplus \big\{\{\mathrm{a}\} \mid \mathrm{a} \in \mathcal{A}t(\mathrm{A})\big\}$, i.e., we add every (neg)atom of A as a singleton class.[1]
– supposing that $\vdash \Gamma, \mathrm{A}$ has been obtained by contraction on $\vdash \Gamma, \mathrm{A}^1, \mathrm{A}^2$, let

$$p \colon \mathcal{A}t(\Gamma, \mathrm{A}^1, \mathrm{A}^2) \longrightarrow \mathcal{A}t(\Gamma, \mathrm{A})$$

be the obvious surjection that identifies pairs of corresponding atoms in the two occurences of A. Take $\mathrm{Prt}(\Gamma, \mathrm{A})$ to be the smallest partition generated by the direct image sets $\{\, p(U) \subseteq \mathcal{A}t(\Gamma, \mathrm{A}) \mid U \in \mathrm{Prt}(\Gamma, \mathrm{A}^1, \mathrm{A}^2)\,\}$. It is easy to see that if we define the binary relation $p(U) \frown p(U')$ as $p(U) \cap p(U') \neq \emptyset$ then the classes in $\mathrm{Prt}(\Gamma, \mathrm{A})$ are in bijective correspondence with the connected components of the graph of $\frown$ and every class is the union of the sets in its corresponding component.
– if $\vdash \Gamma, \Sigma$ has been obtained by applying Cut on $\vdash \Gamma, \mathrm{A}$ and $\vdash \overline{\mathrm{A}}, \Sigma$, we define $\mathrm{Prt}(\Gamma, \Sigma)$ as follows. First define an equivalence relation $\sim$ on $\mathcal{A}t(\Gamma, \mathrm{A}) \uplus \mathcal{A}t(\overline{A}, \Sigma)$ as the symmetric-transitive closure of the union of three binary relations R, S, T, where
  - R, S are the equivalence relations associated with the partitions $\mathrm{Prt}(\Gamma, \mathrm{A}), \mathrm{Prt}(\overline{\mathrm{A}}, \Sigma)$ respectively,
  - xTy when $x = \mathrm{a}$ is a (neg)atom in A and $y = \bar{\mathrm{a}}$ its corresponding negation in $\overline{A}$.

  $\mathrm{Prt}(\Gamma, \Sigma)$ is then defined as the restriction of the partition determined by $\sim$ on the subset $\mathcal{A}t(\Gamma, \Sigma) \subseteq \mathcal{A}t(\Gamma, \mathrm{A}) \uplus \mathcal{A}t(\overline{A}, \Sigma)$

This definition can be explained in terms of the proof nets given in [LS05b], where axiom links are *superposed* when Contraction is applied, giving rise to a relation on $\mathcal{A}t(\Gamma)$ which is not the usual coupling of ordinary linear proof nets, but a much more general kind of relation. Supposing first that neither Weakening nor Cut has been used in a proof of Γ, then the partition $\mathrm{Prt}(\Gamma)$ corresponds exactly to the set of *connected components* of the proof net graph associated to that proof. If a proof contains Weakenings, in our case the atoms are added as singletons in the graph, while they would simply be ignored in a proof net. It is simpler for us to define $\mathrm{Prt}(\Gamma)$ that way, instead of as a subpartition (partition of a subset) of $\mathcal{A}t(\Gamma)$.

[1] Notice the slight abuse of notation where a is used to denote negatomics as well as atomics.

Proposition 3. *Let q be a proof of a sequent $\vdash \Gamma$. Then if $f \subseteq \Gamma = \prod_{\mathrm{a} \in \mathcal{A}t(\Gamma)}$ is the interpretation of q, there is a family $(f_U)_{U \in \mathrm{Prt}(\Gamma)}$ such that f decomposes as a product of factors*

$$f = \prod_{U \in \mathrm{Prt}(\Gamma)} f_U \quad \text{where} \quad f_U \subseteq \prod_{\mathrm{a} \in U} a\,.$$

The proof is a quite straightforward induction, making use of Proposition 2.

The meaning of this is that the decomposition associated with the connected components for a proof net in CL has a simple semantical counterpart, as a product decomposition instead of a sum decomposition. This is syntactical information which is retained by the semantics.

The following well-known result (a proof is in [Lam07]) will turn out to be useful for constructing classes of bimonoids.

Proposition 4. *Let $(M, \leq, \cdot, e)$ be a poset which is equipped with a commutative monoid structure $(\cdot, e)$ such that binary $\cdot$ is $\leq$-monotone. (A category theorist would say: let M be a monoid in* **Poset***.) Then if we define $\nabla : M \times M \to M$ and $\amalg : \{*\} \to M$ in* **Cmp** *as*

$$(m, n) \overset{\nabla}{} p \text{ if } m \cdot n \leq p, \qquad * \overset{\amalg}{} m \text{ if } e \leq m$$

we get a commutative $\times$-monoid (which is the same, remember, as a $\invamp$-monoid). Dually, if we define $\Delta : M \to M \times M$ and $\Pi : M \to \{*\}$ as

$$m \overset{\Delta}{} (n, p) \text{ if } m \leq n \cdot p, \qquad m \overset{\Pi}{} * \text{ if } m \leq e$$

we get a cocommutative $\times$-comonoid, i.e., a cocommutative $\otimes$-comonoid.

As a conclusion, a way to construct bimonoids in **Cmp** is to find posets equipped with two monotone monoid structures.

3 An interpretation based on $\mathbb{Z}$

The set $\mathbb{Z}$ of integers comes equipped with the two structures we are interested in: poset $(\mathbb{Z}, \leq)$ and commutative monoid $(\mathbb{Z}, +, 0)$.

So

we assign the poset $\mathbb{Z}$ to every atomic type a and thus $a = \mathbb{Z}$.

We have to look for other monoid structures. Can we modify standard addition as little as possible, so as to keep calculations simple? Yes, since it is well known that for any number c the operation $(x, y) \mapsto x + y - c$ defines a monoid structure on $\mathbb{Z}$, whose unit is c. As a matter of fact the choice of that unit is all which is needed to define the rest: look at the order-isomorphic translation

$x \mapsto x + c$, which has inverse $x \mapsto x - c$. We have only defined our new operation (let us call it $+_c$) by transporting addition along the first iso, i.e.

$$x +_c y = \big((x - c) + (y - c)\big) + c\,. \tag{1}$$

But this shows that our new monoid $(\mathbb{Z}, +_c, c)$ is isomorphic to the one we started with. It seems we haven't progressed very much. But remember, we are looking for *a new* monoid structure, as we already had one!

Let us define a *poset-bimonoid* to be just that: a sextuple $(M, \leq, +_1, e_1, +_2, e_2)$ where $(M, \leq)$ is a poset and $(+_i, e_i)$ two monotone monoid structures for it. It is obvious what an isomorphism of poset-bimonoids should be: an isomorphism of posets which is also a monoid isomorphism for both $(+_1, e_1)$ and $(+_2, e_2)$. Now let us look at poset-bimonoids of the form $(\mathbb{Z}, \leq, +_{e_1}, e_1, +_{e_2}, e_2)$, as in Equation (1) for any choice of e_1, e_2. We will call these *translation bimonoids.* The following is very easy to show:

Fact 1 *Given two translation bimonoids* $(\mathbb{Z}, \leq, +_{a_1}, a_1, +_{a_2}, a_2)$ *and* $(\mathbb{Z}, \leq, +_{b_1}, b_1, +_{b_2}, b_2)$*, then they are isomorphic iff* $a_2 - a_1 = b_2 - b_1$*.*

Now that we have this information, we can decide to look only at poset-bimonoids of the form $(\mathbb{Z}, \leq, +, 0, +_c, c)$ since it gives us all the isomorphism classes of translation bimonoids by varying c.

What we were interested in in the first place was bimonoids in **Cmp**.

Proposition 5. *If a is a translation bimonoid, then $\bar{a}$ is isomorphic to it.*

Proof. Suppose for simplicity that c is positive. Then it is used for the co-unit of the comonoid structure in the negation $\overline{\mathbb{Z}}$, which has reverse order $\mathbb{Z}^{op}$, which makes c the lesser of the two units in the poset-bimonoid, and if we map it to zero by a translation the whole of the poset-bimonoid structure will be respected.

Thus we end up with an interpretation where atomics and negatomics will be isomorphic (as in the relational model). But, interestingly, that isomorphism cannot be an identity unless $c = 0$.

Let us recapitulate. Choose c, and define

$$\begin{array}{llll} (j,k) \mathrel{\nabla_a} i \ \text{ if } \ j + k \leq i + c; & * \mathrel{\amalg_a} i \ \text{ if } \ c \leq i. \\ i \mathrel{\Delta_a} (j,k) \ \text{ if } \ i \leq j + k; & i \mathrel{\amalg_a} * \ \text{ if } \ i \leq 0; \end{array}$$

And now $\mathcal{D}_a = \nabla \circ \Delta = \{(x,y) \mid \exists i, j : x \leq i + j,\ i + j \leq y + c\} = \{(x,y) \mid x \leq y + c\}$. Then it is easy to see that $\mathcal{D}_a^n = \{(x,y) \mid x \leq y + n \cdot c\}$, and therefore the composition of the doubling map with itself will never stabilize.

3.1 More computations

So computing proofs in this interpretation is rather easy. Again, choose one atomic a along with an interpretation $a = (\mathbb{Z}, \leq, +, 0, +_c, c)$.

We will make use of the fact that the relation $x \leq y$ in a is equivalent to $(-x) + y \geq 0$ and that $x \leq y$ in a^{op} is equivalent to $(-x) \leq (-y)$ in a. This allows us to work with a single order structure, the standard one in $\mathbb{Z}$, and not have to deal with its opposite, which is required in general, and makes things very confusing when, as here, the ordering and its opposite are isomorphic.

Theorem 1. *Let f be a proof of a sequent $\vdash \Gamma$. Then every factor f_j of the decomposition*

$$f = f_1 \times f_2 \times \ldots \times f_N,$$

of Proposition 3 is of the form

$$f_j = \{(x_1, \ldots, x_n, y_1, \ldots, y_m) \in a^{n+m} \mid (-x_1)+\ldots+(-x_n)+y_1+\ldots+y_m \geq M_j c\}$$

where $\mathrm{a}^1, \bar{\mathrm{a}}^2, \ldots, \bar{\mathrm{a}}^n, \mathrm{a}^1, \ldots, \mathrm{a}^m$ *is an enumeration of the (neg)atomics of the class* $U_j \in \mathrm{Prt}(\Gamma)$ *and* M_j *an integer.*

That integer M has to do with the number of contractions that were performed in the proof q and the number of atoms that came into existence through Weakening. A more detailed discussion of the relationship between M and q will be provided in the journal version of the paper; right now we will content ourselves with examples. But let us mention that singleton elements of $\mathrm{Prt}(\Gamma)$ are either of the form $\{y \geq c\}$ for positive atoms or of the form $\{-x \geq 0\}$ for negative ones.

Let us now try our hand at Church numerals. First, write the type $(\mathrm{a} \Rightarrow \mathrm{a}) \Rightarrow (\mathrm{a} \Rightarrow \mathrm{a})$ as the sequent $\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}$, where there cannot be any ambiguity about the way the atomics are reordered.

The proofs we are dealing with are of the form

$$n \times \wedge \left\{ \begin{array}{c}
\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \bar{\mathrm{a}}, \mathrm{a}}\,Ax \quad \dfrac{}{\vdash \bar{\mathrm{a}}, \mathrm{a}}\,Ax}{\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}}\,\wedge \quad \dfrac{}{\vdash \bar{\mathrm{a}}, \mathrm{a}}\,Ax}{\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}}\,\wedge}{\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}}\,Contr \\
\\
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}}\,Contr \quad \dfrac{}{\vdash \bar{\mathrm{a}}, \mathrm{a}}\,Ax}{\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}}\,\wedge}{\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}}\,Contr \quad \dfrac{}{\vdash \bar{\mathrm{a}}, \mathrm{a}}\,Ax}{\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}}\,\wedge}{\vdash \bar{\mathrm{a}}, \mathrm{a} \wedge \bar{\mathrm{a}}, \mathrm{a}}\,Contr
\end{array} \right.$$

for encoding the numeral n.

Let us see what this gives in our interpretation.

Starting with a concrete numeral, say, $n = 3$

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\vdash \overline{\mathrm{a}}^0, \mathrm{a}^0$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\vdash \overline{\mathrm{a}}^1, \mathrm{a}^1$}
\RightLabel{$\wedge$}
\BinaryInfC{$\vdash \overline{\mathrm{a}}^0, \mathrm{a}^0 \wedge \overline{\mathrm{a}}^1, \mathrm{a}^1$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\vdash \overline{\mathrm{a}}^2, \mathrm{a}^2$}
\RightLabel{$\wedge$}
\BinaryInfC{$\vdash \overline{\mathrm{a}}^0, \mathrm{a}^0 \wedge \overline{\mathrm{a}}^1, \mathrm{a}^1 \wedge \overline{\mathrm{a}}^2, \mathrm{a}^2$}
\RightLabel{$Contr$}
\UnaryInfC{$\vdash \overline{\mathrm{a}}^0, \mathrm{a}^3 \wedge \overline{\mathrm{a}}^3, \mathrm{a}^2$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\vdash \overline{\mathrm{a}}^4, \mathrm{a}^4$}
\RightLabel{$\wedge$}
\BinaryInfC{$\vdash \overline{\mathrm{a}}^0, \mathrm{a}^3 \wedge \overline{\mathrm{a}}^3, \mathrm{a}^2 \wedge \overline{\mathrm{a}}^4, \mathrm{a}^4$}
\RightLabel{$Contr$}
\UnaryInfC{$\vdash \overline{\mathrm{a}}^0, \mathrm{a}^5 \wedge \overline{\mathrm{a}}^5, \mathrm{a}^4$}
\end{prooftree}
$$

its proof is computed as follows, following the leftmost branch in the proof tree:

$$
\begin{gathered}
\{-x^0 + y^0 \geq 0\} \\
\blacktriangledown \\
\{-x^0 + y^0 \geq 0\} \times \{-x^1 + y^1 \geq 0\} \\
\blacktriangledown \\
\{-x^0 + y^0 \geq 0\} \times \{-x^1 + y^1 \geq 0\} \times \{-x^2 + y^2 \geq 0\} \\
\blacktriangledown \\
\{-x^0 - x^3 + y^2 + y^3 \geq -c\} \\
\blacktriangledown \\
\{-x^0 - x^3 + y^2 + y^3 \geq -c\} \times \{-x^4 + y^4 \ \geq 0\} \\
\blacktriangledown \\
\{-x^0 - x^5 + y^4 + y^5 \geq -2c\}
\end{gathered}
$$

The third and the fifth transition require some comments. When contracting $\mathrm{a}^0 \wedge \overline{\mathrm{a}}^1, \mathrm{a}^1 \wedge \overline{\mathrm{a}}^2$ two atomic contractions are conducted, one on the positive atoms $\mathrm{a}^0, \mathrm{a}^1$, which is the one that adds c, and the other one on the negatoms $\overline{\mathrm{a}}^1, \overline{\mathrm{a}}^2$ whose contraction monoid is plain addition in $\mathbb{Z}$, and thus does not modify the constants M.

It is not difficult to conclude now that in general, given a *positive* n its Church numeral is (where the use of superscripts naturally follows from above):

$$\{\, -x^0 - x^{2n-1} + y^{2n-2} + y^{2n-1} \ \geq \ -(n-1)c \,\}.$$

Recall that if two Church numerals are seen as maps n, m: $(\mathrm{a} \Rightarrow \mathrm{a}) \to (\mathrm{a} \Rightarrow \mathrm{a})$ then multiplying them amounts to *composing* these two maps, i.e., is obtained through a Cut. A simple computation will show that this operation will give the numeral $n + m - 1$.

This is definite evidence that the interpretation we have built has nothing to do with the Curry-Howard correspondence. A multiplication which acts additively as it does here shows that the interpretation of terms cannot correspond to functional programs, since the Curry-Howard interpretation of a Church numeral n is the iterator functional "compose an endofunction n times with itself", and composing the iterators for n and m can only give the iterator for nm.

3.2 Discussion

We had seen that in general an interpretation of a classical proof in **Cmp** could be decomposed as a Cartesian product sets, each of which corresponds to a connected component U_j of the graph of axiom links of the associated proof net. We now see that the information contained in our $\mathbb{Z}$-interpretation consists in that decomposition, along with a natural number M_j which is associated to every connected component U_j, related to the Contractions and Weakenings that were effected, *and nothing else.* This suggests a class of proof nets where the binary relation determined by axiom links would be replaced by an undifferentiated "blob", just a set of atomics and negatomics of the same type, with a number associated to it. There is no guarantee that such proof nets would have a correctness criterion, and more research has to be done about this. Another research problem which arises from this is to see if this numerical invariant associated to *blobs* is a direct consequence of the fact that the bimonoid we have constructed is actually a Frobenius algebra [Hyl04] (Frobenius algebras are a specific class of bimonoids that usually live in categories of vector spaces, but nothing prevents them from being defined in **Cmp**, in the same way we did for bimonoids).

It is also interesting to notice the incontrovertible evidence that there are valid, normalizing semantics of proofs that have nothing to do with the Curry-Howard correspondence.

3.3 Interpretations and Semantics

In what precedes we have used the word "semantics" a few times, but always in the generic sense, as in "semantics of proofs". But the actual constructions were called "interpretations". There is a reason for that. We ask of a semantics to be something stronger than an interpretation. Although it is hard to pinpoint exactly when an interpretation actually becomes a semantics, it should obey a general principle to make the grade:

True semantical objects should exist independently of syntax.

Let us illustrate this principle by supposing that we have constructed a category $\mathcal{C}$ and want to use it as a semantics for logic $\mathcal{L}$. Then an immediate consequence of the general principle is that

Every morphism $f\colon A \to B$ in $\mathcal{C}$ should be a proof of a theorem $A \vdash B$.

Naturally this means that $\mathcal{C}$ will not in general be a *pure* model of $\mathcal{L}$, in other words that it will contain "non-logical" axioms. We have already seen a quite strong example of this, since we have been accepting $A \otimes B \vdash\dashv A \parr B$ in our models of linear logic. But despite the presence of such axioms (they are not wanted by purists, but they can be a source of inspiration for creating logics) this requirement puts strong constraints on the category $\mathcal{C}$.

Keeping this general principle in mind, let us emphasize that there is a very important difference between semantics and ordinary syntax. A syntactic object exists to be written down, and the result of this is that in a formula like A ⊗ A, we can distinguish between the first A and the second one, simply because we write linearly and (westerners at least) read from left to right. But there is no such information in $A \otimes A$, if it is seen as an object of a category. It is equipped with a non-trivial involution $\rho: A \otimes A \to A \otimes A$, $\rho \circ \rho = \mathrm{Id}$ corresponding to exchanging both factors of the tensor. This means that there is no way to distinguish between "left factor" and "right factor" when we look at the result of a commutative operation like tensor in a semantical category. Some additional, extra-semantical information is required. This creates potential problems: suppose in general that $\alpha: B \to B$ is an isomorphism of an object onto itself—a permutation—in a model of classical logic of the sort we have been discussing. This is a very general definition: it is just saying that there is an inverse α^{-1} with $\alpha \circ \alpha^{-1} = \alpha^{-1} \circ \alpha = \mathrm{Id}_B$. Now look at the contraction (codiagonal) $\nabla: B \invamp B \to B$ on B. Using α *we can create a new contraction* by using once again "transport of structure", i.e., defining ∇' as $\alpha \circ \nabla \circ (\alpha^{-1} \otimes \alpha^{-1})$. If we also do this for weakening (i.e., $\amalg' = \alpha \circ \amalg$) we will see that the new structure $(B, \nabla', \amalg')$ will obey all the equations that hold for $(B, \nabla, \amalg)$, and in particular that it will be a commutative bimonoid. Thus the "choice" problem pops up again, but now it is much more serious: which of $(\nabla, \amalg), (\nabla', \amalg')$ do we choose when we look at B and want to interpret contraction on it? *There is no way to distinguish them in our category.* Going back to our original example $A \otimes A$ with ρ as the permutation, some easy calculations show that in this particular case the original $(\nabla, \amalg)$ and its version transported with ρ will be identical. Thus this is not a problem for this example, which can be easily generalized to all composite formulas. But unfortunately there is a real problem with the atomics in our interpretation. The maps of the category are the comparisons, and it is easy to see that an order iso is a comparison—more correctly, can be seen as a comparison through the same down-closure to the left, up-closure to the right process that we used for identities. It is easy to see that the order-automorphisms (monotone permutations) on $\mathbb{Z}$ are exactly the translations, as defined above. And we know that given a translation bimonoid of the form $(\mathbb{Z}, \leq, +, 0, +_c, c)$, which is used to interpret atomics, and an arbitrary translation $x \mapsto x + d$, then transporting the former by the latter will give us the translation bimonoid $(\mathbb{Z}, \leq, +_d, c + d)$, which will be an isomorphic *but not identical* bimonoid.

This means that there is a little something missing to allow us to declare that the category of translation bimonoids, with comparisons as maps, is a semantics. The reader may say "why not instead take a category of bimonoids, along with some additional information? Say the category whose objects are all translation bimonoids $(\mathbb{Z}, \leq, +_{a_1}, a_1, +_{a_2}, a_2$, the additional information being the tagging of every such bimonoid with the pair (a_1, a_2)?" (we know this pair is enough to uniquely define all the necessary equipment for contraction, weakening, etc.).

We admit this is a potential solution, but not a very satisfying one. It is completely ad hoc, and will not help us for any other model we come up with.

4 A semantics based on $\mathbb{N}$

The previous discussion tells us that one way to get that ideal of semantics instead is to to ensure that all the bimonoids we construct are invariant with respect to their order-automorphisms. One promising way to achieve this to try to model the atomics with posets that simply have *no* trivial automorphim. We also need to work with infinite posets if we want to ensure that the doubling map $\mathcal{D}$ can be iterated at will.

Let us consider the order on non-negative integers, $\mathbb{N}$.

It is clear that the only order-authomorphism of it is the identity.

So first assign $\mathbb{N}$ to a fixed atomic type a, restricting our attention to a single atomic variable.

Choose two natural numbers e_1, e_2 and let $\circ$ and $*$ be two binary operations defined on $\mathbb{N}$ as follows:

$$i \circ j = \begin{cases} i + j - e_1, & i, j \geq e_1 \\ \max(i + j - e_1, 0), & i, j < e_1 \\ \min(i, j), & i < e_1 \leq j \text{ or } j < e_1 \leq i. \end{cases}$$

and

$$i * j = \begin{cases} i + j - e_2, & i, j \geq e_2 \\ \max(i + j - e_2, 0), & i, j < e_2 \\ \max(i, j), & i < e_2 \leq j \text{ or } j < e_2 \leq i. \end{cases}$$

Claim. Operations $\circ$ and $*$ are monotone, and define commutative monoids on $\mathbb{N}$, with e_1 and e_2 as units, respectively.

Proof. Commutativity of the operations is trivial to check. The fact that e_1 and e_2 are the units also.

Since we have commutativity, the monotonicity of the operations can be checked by showing that $(-) \circ j$ and $(-) * j$ are monotone functions, i.e., by realizing that the following maps are obviously monotone:

$$\text{for } \circ, j \geq e_1 : i \mapsto \begin{cases} i + j - e_1, & i \geq e_1 \\ i & i < e_1 \end{cases}$$

$$\text{for } \circ, j < e_1 : i \mapsto \begin{cases} \max(i + j - e_1, 0), & i < e_1 \\ j & i \geq e_1 \end{cases}$$

$$\text{for } *, j \geq e_2 : i \mapsto \begin{cases} i + j - e_2, & i \geq e_1 \\ j & i < e_1 \end{cases}$$

$$\text{for } *, j < e_2 : i \mapsto \begin{cases} \max(i + j - e_2, 0), & i < e_2 \\ i & i \geq e_2. \end{cases}$$

Associativity has to be checked by cases. For $\circ$: $(i \circ j) \circ k = i \circ (j \circ k)$. If all of i, j, k are not smaller than e_1, both sides of the equality are equal to $i + j + k - 2e_1$. If one of them is smaller than e, both sides are equal to it; if exactly two of them are smaller than e_1, both sides of the equality are equal to the maximum of 0 and their sum minus e_1. Finally, if all three of i, j, k are smaller than e_1, both sides of the equality are equal to $\max(i + j + k - 2e_1, 0)$. As for $*$: $(i * j) * k = i * (j * k)$, if all of i, j, k are greater than e_2, both sides of the equality are equal to $i + j + k - 2e_2$. If exactly one of them is larger than e_2, both sides are equal to it; if exactly two of them are greater than e_2, both sides of the equality are equal to their sum minus e_2. Finally, if all three of i, j, k are not greater than e_2, both sides of the equality are equal to $\max(i + j + k - 2e_1, 0)$.

From this and Proposition 4 follow the monoid structure (Δ_a, Π_a) and the comonoid structure (∇_a, Π_a):

$$x^{\Delta_a}(y, z) \text{ iff } x \leq y \circ y; \quad x^{\Pi_a}* \text{ iff } x \leq e_1;$$

$$(y, z)^{\nabla_a}x \text{ iff } y * z \leq x; \quad *^{\Pi_a}x \text{ iff } e_2 \leq x.$$

Let us now compute the doubling map, $\mathcal{D}_a$. So far, we made no assumptions on the units e_1 and e_2 of $\circ$ and $*$. In order to simplify the computations, we will require from now on that $0 < e_1 < e_2 < 2e_1$. We will denote $e_2 - e_1$ by δ. Notice that $0 < \delta < e_1$.

As usual, $\mathcal{D}_a = \nabla_a \circ \Delta_a = \{(x, y) \mid \exists i, j : x^{\Delta_a}(i, j),\ (i, j)^{\nabla_a}y\}$ and this means $\mathcal{D}_a = \{(x, y) \mid \exists i, j : x \leq i \circ j,\ i * j \leq y\}$. We compute all of the pairs (x, y) by distinguishing cases depending on i, j.

1. $i, j < e_1$. Then: $x \leq \max(i + j - e_1, 0)$ and $y \geq \max(i + j - e_2, 0)$. By varying i and j within the range, one obtains $\{(x, y) \mid 0 \leq x \leq e_1 - 2; y \geq \max(0, x - \delta)\}$.
2. $i < e_1 \leq j \leq e_2$. This time: $x \leq i < e_1$ and $y \geq \max(i + j - e_2, 0)$. Again, by varying i and j within the range, one obtains $\{(x, y) \mid 0 \leq x \leq e_1 - 1; y \geq max(0, x - \delta)\}$.
3. $i < e_1 < e_2 < j$. In this case, $x \leq i < e_1$ and $y \geq j > e_1$. This time, the resulting relation is $\{(x, y) \mid 0 \leq x \leq e_1 - 1; y \geq e_2 + 1\}$.
4. $e_1 \leq i \leq j \leq e_2$. Then, $x \leq i + j + e_1 < e_1$ and $y \geq \max(i + j - e_2, 0)$. The relation is $\{(x, y) \mid 0 \leq x \leq e_2 + \delta; y \geq \max(e_1 - \delta, x - \delta\}$.
5. $e_1 \leq i \leq e_2 < j$. One has, $x \leq i + j + e_1$ and $y \geq j > e_2$. The relation is $\{(x, y) \mid 0 \leq x; y \geq \max(e_2 + 1, x - \delta\}$.
6. $e_1 < e_2 < i \leq j$. Finally, $x \leq i + j + e_1$ and $y \geq i + j + e_2$, and the relation is $\{(x, y) \mid 0 \leq x; y \geq \max(e_2 + 1, x - \delta\}$.

The map $\mathcal{D}_a$ is union of the sets obtained in 1-6, thus $\mathcal{D}_a = \{(x, y) \mid 0 \leq x; y \geq max(0, x - \delta)\}$. Computing $\mathcal{D}_a^n$ is easy, $\mathcal{D}_a^n$ will be $\{(x, y) \mid 0 \leq x; y \geq max(0, x - n \cdot \delta)\}$. Looking at the coordinate system, $\mathcal{D}_a$ corresponds to the portion of $\mathbb{N}^2$ above $y = x - \delta$, while composing the doubling map with itself can bee seen as the shifting of the $y = x - \delta$ to the right. It is clear that $\mathcal{D}_a^n$ never stabilizes. The previous discussion considered only a single atomic type. We will now show how to get an infinity of non-isomorphic atomic types.

To that purpose, let us assume that our positive atoms are enumerated: $(\mathrm{a}_k)_{k \in \mathbb{N}}$; we can we assign the poset $\mathbb{N}^{+k} = \{0 < 1 < 2 < \ldots < \omega + 1 < \ldots < \omega + k\}$, with the obvious abuse of the ordinal notation. In other words a_k is the ordinal sum $\omega + k$.

Notice that this choice for posets fulfills the abovementioned requirements, i.e. it is clear that the only order isomorphisms between two of these posets

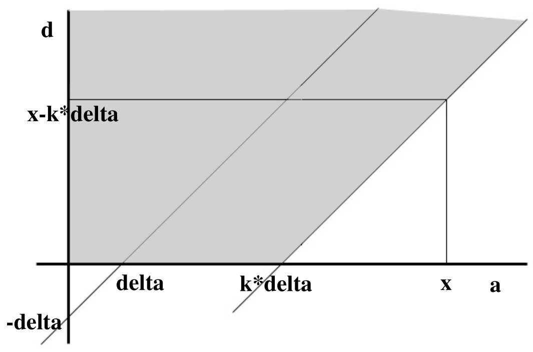


Fig. 1. Graph of the doubling map iterated k times - $\mathcal{D}_a^k$.

are identities, since they are canonical representatives of well-founded orders for different countable ordinals.

The work done so far makes easy now to define the corresponding monoid and comonoid maps.

Let

$$i \circ_k j = \begin{cases} \max(i,j), & \text{if } i \geq \omega+1 \text{ or } j \geq \omega+1, \\ i \circ j, & \text{if } i,j < \omega+1, \end{cases}$$

and

$$i *_k j = \begin{cases} \max(i,j), & \text{if } i \geq \omega+1 \text{ or } j \geq \omega+1, \\ i * j, & \text{if } i,j < \omega+1, \end{cases}$$

with

$$x^{\Delta_{a_k}}(y,z) \text{ iff } x \leq y \circ_k z; \quad x^{\Pi_{a_k}} * \text{ iff } x \leq e;$$

$$(y,z)^{\nabla_{a_k}} x \text{ iff } y * z \leq x; \quad *^{\amalg}_{a_k} x \quad \text{iff } e_2 \leq x.$$

The reader can check that $\circ_k$ and $*_k$ are commutative, associative, monotone with e_1 and e_2 as units. This ensures that the monoid and comonoid diagrams commute.

The doubling map for each of the newly defined bimonoids is not that different from the previously analyzed case, i.e. we have

$$\mathcal{D}^n_{a_k} = \{(x,y) \mid 0 \leq x < \omega+1; y \geq \max(0, x - n \cdot \delta)\} \cup \{(x,y) \mid \omega+1 \leq x; y \geq x\}.$$

In particular, the $\mathcal{D}^n_{a_k}$ map never stabilizes.

It is obvious that a type like $a_k \wedge a_k$ will have a nontrivial order-automorphism, since we can exchange the left and right side of the Cartesian product. But we claim these are the *only* possible automorphisms of these product orders in general. Since it is well-known that products of commutative-cocommutative bimonoids are invariant under these permutations of factors, we will have achieved our goal of constructing a semantics and not just an interpretation.

Theorem 2. *Suppose there is an order isomorphism $\alpha: X_1 \times \cdots \times X_n \to Y_1 \times \cdots \times Y_m$ where every X_i, Y_j is either an order of the type $\omega + k$ as above or the opposite of such an order. Then $n = m$, and there exists a permutation π of $\{1, \ldots, n\}$ such that α decomposes as a product of isomorphisms $X_i \to X_{\pi i}$.*

The proof will be given in the final version.

5 Conclusion

In addition to being a real semantics and not just an interpretation, the previous construction has the advantage that it is much "roomier", since it contains many nonisomorphic types. It is also is a finer semantics from the point of view of the decomposition in connected components, since the stucture it puts on these components amount to more than just numbers. But the computations are much more complicated than for the $\mathbb{Z}$-based interpretation, and we are far from knowing if meaningful proof invariants can be extracted.

In systems or semantics that have the Medial rule, since the Contractions and Weakenings can be "pushed up" to the atomics, it makes a lot of sense to consider a proof as a set of disjoint "blob with structure", i.e. a collection of sets of atomics and negatomics with some additional mathematical structure on each such set. We intend to pursue that direction of enquiry.

References

[Bar79] Michael Barr. *$*$-Autonomous Categories*, volume 752 of *Lecture Notes in Mathematics*. Springer-Verlag, 1979.

[Brü03] Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. PhD thesis, Technische Universität Dresden, 2003.

[FP04] Carsten Führmann and David Pym. On the geometry of interaction for classical logic (extended abstract). In *19th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 211–220, 2004.

[FP05] Carsten Führmann and David Pym. Order-enriched categorical models of the classical sequent calculus. *J. Pure App. Algebra*, 204(1):21–68, 2005.

[Hyl04] J. Martin E. Hyland. Abstract interpretation of proofs: Classical propositional calculus. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, CSL 2004*, volume 3210 of *LNCS*, pages 6–21. Springer-Verlag, 2004.

[Lam94] J. Lambek. Bilinear logic in algebra and linguistics. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Math. Soc. Lec. Notes*, pages 43–59. Cambridge University Press, 1994.

[Lam07] F. Lamarche. Exploring the gap between linear and classical logic. *submitted to Theory and Applications of Categories*, 2007. available at http://www.loria.fr/~lamarche.

[LS05a] F. Lamarche and L. Straßburger. Constructing free boolean categories. In *LICS Proceedings*. IEEE Press, 2005.

[LS05b] F. Lamarche and L. Straßburger. Naming proofs in classical logic. In P. Urzyczyn, editor, *TLCA Proceedings*, volume 3461 of *LNCS*, pages 246–261. Springer, 2005.