From Deep Inference to Proof Nets via Cut Elimination

Lutz Straßburger

INRIA Saclay–Île-de-France, France

http://www.lix.polytechnique.fr/~lutz

June 24, 2009

Abstract

This paper shows how derivations in the deep inference system SKS for classical propositional logic can be translated into proof nets. Since an SKS derivation contains more information about a proof than the corresponding proof net, we observe a loss of information which can be understood as "eliminating bureaucracy". Technically this is achieved by cut reduction on proof nets. As an intermediate step between the two extremes, SKS derivations and proof nets, we will see proof graphs representing derivations in "Formalism A".

1 Introduction

Through the development of the two concepts of *deep inference* [Gug07] and *proof nets* [Gir87] the quest for the identity of proofs has been revived, and new effort is being put on the fundamental question "When are two proofs the same?"

Proof nets have been conceived by Girard [Gir87] in order to avoid *bureaucracy*: in formal systems like the sequent calculus two proofs that are "morally the same" are distinguished by trivial rule permutations, and proof nets are able to abstract away from these permutations.

Deep inference has been conceived by Guglielmi in order to obtain a deductive system for a non-commutative logic [Gug07]. In a formalism employing deep inference, like the calculus of structures, one can apply inference rules anywhere deep inside formulas as we know it from term rewriting, instead of decomposing formulas along their main connectives as we know it from traditional formalisms, like natural deduction and sequent calculus. From the "we-wish-to-eliminate-bureaucracy" point of view, this is a disaster: The number of possible "trivial rule permutations" explodes, compared to the sequent calculus. However, the finer granularity of inference rules—one inference step in the sequent calculus corresponds to many inference steps in the calculus of structures—allows a finer

analysis of the inner structure of proofs, which in turn can lead to new notions of proof nets (as happened in [SL04] and [LS05b]).

In this paper we will see how proof nets (or, more precisely, *proof graphs*) can be extracted directly from deep inference systems. We will concentrate here only on classical logic, more precisely on system SKS [BT01, Brü03b], a well-studied system for classical logic in the calculus of structures (see also [BG09]). But it should be clear that the exercise of this paper can be carried out in the same way for any other system, in particular also for linear logic as it is presented in [Str02]. The reason is that proof graphs, as they are used in this paper, can be defined accordingly for other systems. However, the relation between these proof graphs and the known proof nets for the corresponding logic is not always evident. For example, our proof graphs used in this paper are in close correspondence to the proof nets introduced by Lamarche and Straßburger in [LS05b], but are very different from the proof nets studied by Robinson in [Rob03]. On the other hand, applying the methods of this paper to linear logic, as presented in [Str02], would for the unit-free multiplicative fragment (MLL) yield exactly the proof nets of Girard [Gir87], but would for the multiplicative additive fragment (MALL) yield proof graphs that are very different from the proof nets studied by Girard [Gir87, Gir96] and Hughes and van Glabbeek [HvG03].

To some extend, one can say that proof graphs make as many identifications between proofs as possible (without ending up in a triviality), and derivations in the calculus of structures make as few identifications as possible. These two extremes span a whole universe of possible proof identifications. And going from the extreme with few identifications to the extreme with many identification means losing information, namely, the "bureaucratic" information that makes the additional distinctions. We will argue, that this process of losing information can be modeled by cut elimination. In each single cut reduction step some bit of information is lost. Depending on the restrictions on cut elimination one can choose which information to lose.

The paper is structured as follows. We will first introduce the concept of deep inference, and show various kinds of bureaucracy that occur in a deep inference system. We will also show how this is related to category theoretical concepts, as it has been indicated in [Hug04]. Afterwards we introduce a notion of proof graphs which is a variant of the proof nets introduced in [LS05b]. Finally, we show how to translate deep inference derivations into proof graphs and how cut elimination corresponds to eliminating bureaucracy. We also explain the relation between our proof graphs and similar concepts as Buss' logical flow-graphs [Bus91] and Guglielmi's and Gundersen's atomic flows [GG08]. Finally we discuss some properties of the induced proof graph categories.

This paper is written from the viewpoint that syntactic derivations, morphisms in certain categories, and proof nets should not be seen as distinct mathematical objects, but only as three different presentations of the same kind of objects: *proofs*.

An early draft of this work is available as [Str05].

2 Deep Inference for Classical Logic

In this section we will acquaint the reader with the basic notions of deep inference and prove a robustness theorem, as it is known for Frege-systems.

Deep inference is a paradigm for proof theoretical formalisms. The most prominent example of a formalism employing deep inference is the calculus of structures. It has successfully been employed to give new presentations for many logics, including classical logic [BT01, Brü03b], minimal logic [Brü03c], intuitionistic logic [Tiu06], modal logics [SS05, Sto07, Brü06a, BS09], linear logic [Str03, Str02], and various non-commutative logics [DG04, Gug07, GS02].

The basic idea of deep inference is that inference rules can apply anywhere deep inside a formula. This is very different from more traditional formalisms like sequent calculus, natural deduction, or tableau systems, where inference rules can attack formulas only at their root connective. A typical deep inference rule has the shape

$$\mathsf{r}\,\frac{F\{A\}}{F\{B\}}$$

which says that the formula A can be rewritten as the formula B inside any positive formula context $F\{\ \}$. This corresponds to the validity of the implication $A \supset B$. A deep inference *proof system* (short: *system*) is as set of deep inference rule schemes. A deep inference *derivation* can then be seen as a rewriting path. It is denoted as

$$\begin{array}{c} A \\ \Delta \,\|\, \mathsf{S} \\ B \end{array}$$

where A is the premise, B is the conclusion, Δ is the name of the derivation, and S is the system in which it is carried out.

In principle, every valid implication can be transformed into a deep inference rule. For example, the implication

$$(A \wedge B) \vee (C \wedge D) \ \supset\ [A \vee C] \wedge [B \vee D]$$

can be transformed into the inference rule

$$\mathsf{m}\,\frac{F\{(A \wedge B) \vee (C \wedge D)\}}{F\{[A \vee C] \wedge [B \vee D]\}} \tag{1}$$

which is called *medial*. More precisely, (1) is a rule scheme, because A, B, C, and D, stand for arbitrary formulas, and $F\{\ \}$ stands for an arbitrary (positive) formula context.

Consequently, any Frege-system can trivially be converted into a deep inference system. Conversely, every deep inference rule scheme can be converted into an axiom of a Frege-system. The details for translating between deep inference and Frege-systems can be found in [BG09], which also discusses the size of the proof translations. It is not surprising that similarly to the robustness theorem

for Frege-systems [CR79] there is also a robustness theorem for deep inference systems, which we will discuss at the end of this section. A consequence of robustness is that from the viewpoint of proof complexity it does not matter which system one chooses. However, from the proof theoretic point of view there can be large differences because of properties, like cut elimination and decomposition, that some proof systems might have and others might not have.

In the following, we will discuss system SKS (in a slight variation of the original presentation in [BT01]) and some of its properties. We consider formulas to be generated from a countable set $\mathcal{A} = \{a, b, c, \ldots\}$ of *propositional variables*, their duals $\bar{\mathcal{A}} = \{\bar{a}, \bar{b}, \bar{c}, \ldots\}$, and the units $\mathbf{t}$ (*truth*) and $\mathbf{f}$ (*falsum*), via the binary connectives $\wedge$ (*and*) and $\vee$ (*or*). The elements of the set $\mathcal{A} \cup \bar{\mathcal{A}} \cup \{\mathbf{t}, \mathbf{f}\}$ will be called *atoms*, and the elements of the set $\mathcal{A} \cup \bar{\mathcal{A}}$ will be called *proper atoms*. For simplicity, we assume formulas to be in negation normal form, i.e., negation is only allowed over propositional variables. Formally, the set $\mathcal{F}$ of formulas is generated by the grammar

$$\mathcal{F} ::= \mathcal{A} \mid \bar{\mathcal{A}} \mid \mathbf{t} \mid \mathbf{f} \mid [\mathcal{F} \vee \mathcal{F}] \mid (\mathcal{F} \wedge \mathcal{F})$$

To ease the readability of long formulas, we use different kinds of parentheses for $\wedge$- and $\vee$-formulas, and we omit outermost parentheses.[1] We use $A, B, C, \ldots$ to denote formulas. The negation $\bar{A}$ of a formula A is defined via De Morgan laws:[2]

$$\bar{\bar{a}} = a \qquad \bar{\mathbf{t}} = \mathbf{f} \qquad \bar{\mathbf{f}} = \mathbf{t} \qquad \overline{A \wedge B} = \bar{B} \vee \bar{A} \qquad \overline{A \vee B} = \bar{B} \wedge \bar{A} \tag{2}$$

Here a ranges over the set $\mathcal{A}$. However, from now on we will use a to denote an arbitrary atom (i.e., a propositional variable or its negation or a unit). Note that it follows that $\bar{\bar{A}} = A$ for every formula A.

Formula contexts are generated by the grammar

$$\mathcal{C} ::= \{\ \} \mid [\mathcal{C} \vee \mathcal{F}] \mid [\mathcal{F} \vee \mathcal{C}] \mid (\mathcal{C} \wedge \mathcal{F}) \mid (\mathcal{F} \wedge \mathcal{C})$$

In other words, a context is a formula with a special place holder $\{\ \}$, the *hole*. Formula contexts are denoted by $F\{\ \}$, and $F\{A\}$ means stands for the formula which is obtained by substituting the hole in $F\{\ \}$ by A.

Figure 1 shows the inference rules that we use in this paper, and the system containing these rules is called SKS. Figure 2 shows two examples of derivations in system SKS.

2.1 Remark In the original presentation [BT01], the system SKS contained only the rules $\mathsf{ai}{\downarrow}$, $\mathsf{ai}{\uparrow}$, s, m, $\mathsf{ac}{\downarrow}$, $\mathsf{ac}{\uparrow}$, $\mathsf{aw}{\downarrow}$, $\mathsf{aw}{\uparrow}$ (i.e., the first three lines in

[1] In early papers on deep inference the formula $a \wedge [b \vee (\bar{a} \wedge c)]$ would have been written $(a, [b, (\bar{a}, c)])$, i.e, without the connectives, whereas without our convention, it would be written as $a \wedge (b \vee (\bar{a} \wedge c))$. We try here to take the best from both notations.

[2] For reasons that will become clear later, we invert the order of the arguments, when taking the negation.

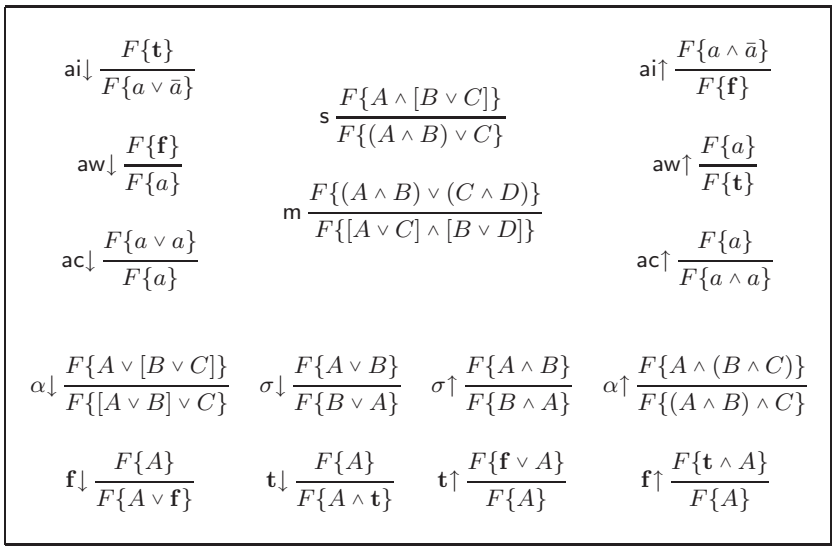


Figure 1: The inference rules of system SKS

Figure 1). The other rules where replaced by an equational theory generated by the equations:

$$\begin{array}{llll} A \wedge (B \wedge C) = (A \wedge B) \wedge C & A \wedge B = B \wedge A & A \wedge \mathbf{t} = A & \mathbf{f} \wedge \mathbf{f} = \mathbf{f} \\ A \vee [B \vee C] = [A \vee B] \vee C & A \vee B = B \vee A & A \vee \mathbf{f} = A & \mathbf{t} \vee \mathbf{t} = \mathbf{t} \end{array} \tag{3}$$

Let us call SKS' the system $\{\mathsf{ai}{\downarrow}, \mathsf{ai}{\uparrow}, \mathsf{s}, \mathsf{m}, \mathsf{ac}{\downarrow}, \mathsf{ac}{\uparrow}, \mathsf{aw}{\downarrow}, \mathsf{aw}{\uparrow}\}$ extended by the rule

$$= \frac{A}{B} \tag{4}$$

where $A = B$ according to (3). Then we have that SKS and SKS' are strongly equivalent in the following sense:

$$\begin{array}{c} A \\ \| \, \mathsf{SKS} \\ B \end{array} \quad \text{iff} \quad \begin{array}{c} A \\ \| \, \mathsf{SKS}' \\ B \end{array} \tag{5}$$

For proving this, observe that whenever there is a derivation

$$\begin{array}{c} A \\ \| \, \{\alpha{\downarrow},\alpha{\uparrow},\sigma{\downarrow},\sigma{\uparrow},\mathbf{f}{\downarrow},\mathbf{f}{\uparrow},\mathbf{t}{\downarrow},\mathbf{t}{\uparrow}\} \\ B \end{array}$$

$$
\begin{array}{rc}
 & \bar{b} \vee a \\ \cline{2-2}
\mathbf{t}\downarrow & \bar{b} \vee (a \wedge \mathbf{t}) \\ \cline{2-2}
\mathsf{ai}\downarrow & \bar{b} \vee (a \wedge [b \vee \bar{b}]) \\ \cline{2-2}
\mathsf{s} & \bar{b} \vee [(a \wedge b) \vee \bar{b}] \\ \cline{2-2}
\sigma\uparrow & \bar{b} \vee [(b \wedge a) \vee \bar{b}] \\ \cline{2-2}
\sigma\downarrow & \bar{b} \vee [\bar{b} \vee (b \wedge a)] \\ \cline{2-2}
\alpha\downarrow & [\bar{b} \vee \bar{b}] \vee (b \wedge a) \\ \cline{2-2}
\mathsf{ac}\downarrow & \bar{b} \vee (b \wedge a) \\ \cline{2-2}
\mathsf{ac}\uparrow & (\bar{b} \wedge \bar{b}) \vee (b \wedge a) \\ \cline{2-2}
\mathbf{t}\downarrow & (\bar{b} \wedge (\bar{b} \wedge \mathbf{t})) \vee (b \wedge a) \\ \cline{2-2}
\mathsf{ai}\downarrow & (\bar{b} \wedge (\bar{b} \wedge [a \vee \bar{a}])) \vee (b \wedge a) \\ \cline{2-2}
\mathsf{s} & (\bar{b} \wedge [(\bar{b} \wedge \bar{a}) \vee a]) \vee (b \wedge a) \\ \cline{2-2}
\sigma\uparrow & (\bar{b} \wedge [(\bar{a} \wedge \bar{b}) \vee a]) \vee (b \wedge a) \\ \cline{2-2}
\sigma\downarrow & (\bar{b} \wedge [a \vee (\bar{a} \wedge \bar{b})]) \vee (b \wedge a) \\ \cline{2-2}
\mathsf{s} & [(\bar{b} \wedge a) \vee (\bar{a} \wedge \bar{b})] \vee (b \wedge a)
\end{array}
\qquad \text{and} \qquad
\begin{array}{rc}
 & \bar{b} \vee a \\ \cline{2-2}
\mathbf{t}\downarrow & \bar{b} \vee (a \wedge \mathbf{t}) \\ \cline{2-2}
\mathsf{ai}\downarrow & \bar{b} \vee (a \wedge [b \vee \bar{b}]) \\ \cline{2-2}
\mathbf{t}\downarrow & \bar{b} \vee (a \wedge [b \vee (\bar{b} \wedge \mathbf{t})]) \\ \cline{2-2}
\mathsf{ai}\downarrow & \bar{b} \vee (a \wedge [b \vee (\bar{b} \wedge [\bar{b} \vee b])]) \\ \cline{2-2}
\mathsf{s} & \bar{b} \vee (a \wedge [b \vee [(\bar{b} \wedge \bar{b}) \vee b]]) \\ \cline{2-2}
\sigma\downarrow & \bar{b} \vee (a \wedge [b \vee [b \vee (\bar{b} \wedge \bar{b})]]) \\ \cline{2-2}
\alpha\downarrow & \bar{b} \vee (a \wedge [[b \vee b] \vee (\bar{b} \wedge \bar{b})]) \\ \cline{2-2}
\mathsf{s} & \bar{b} \vee [(a \wedge [b \vee b]) \vee (\bar{b} \wedge \bar{b})] \\ \cline{2-2}
\sigma\downarrow & \bar{b} \vee [(\bar{b} \wedge \bar{b}) \vee (a \wedge [b \vee b])] \\ \cline{2-2}
\sigma\uparrow & \bar{b} \vee [(\bar{b} \wedge \bar{b}) \vee ([b \vee b] \wedge a)] \\ \cline{2-2}
\alpha\downarrow & [\bar{b} \vee (\bar{b} \wedge \bar{b})] \vee ([b \vee b] \wedge a) \\ \cline{2-2}
\mathsf{ac}\uparrow & [(\bar{b} \wedge \bar{b}) \vee (\bar{b} \wedge \bar{b})] \vee ([b \vee b] \wedge a) \\ \cline{2-2}
\mathsf{ac}\downarrow & [(\bar{b} \wedge \bar{b}) \vee (\bar{b} \wedge \bar{b})] \vee (b \wedge a) \\ \cline{2-2}
\mathsf{m} & ([\bar{b} \vee \bar{b}] \wedge [\bar{b} \vee \bar{b}]) \vee (b \wedge a) \\ \cline{2-2}
\mathsf{ac}\downarrow & (\bar{b} \wedge [\bar{b} \vee \bar{b}]) \vee (b \wedge a) \\ \cline{2-2}
\mathsf{ac}\downarrow & (\bar{b} \wedge \bar{b}) \vee (b \wedge a) \\ \cline{2-2}
\mathbf{t}\downarrow & (\bar{b} \wedge (\bar{b} \wedge \mathbf{t})) \vee (b \wedge a) \\ \cline{2-2}
\mathsf{ai}\downarrow & (\bar{b} \wedge (\bar{b} \wedge [a \vee \bar{a}])) \vee (b \wedge a) \\ \cline{2-2}
\mathsf{s} & (\bar{b} \wedge [(\bar{b} \wedge \bar{a}) \vee a]) \vee (b \wedge a) \\ \cline{2-2}
\sigma\uparrow & (\bar{b} \wedge [(\bar{a} \wedge \bar{b}) \vee a]) \vee (b \wedge a) \\ \cline{2-2}
\sigma\downarrow & (\bar{b} \wedge [a \vee (\bar{a} \wedge \bar{b})]) \vee (b \wedge a) \\ \cline{2-2}
\mathsf{s} & [(\bar{b} \wedge a) \vee (\bar{a} \wedge \bar{b})] \vee (b \wedge a)
\end{array}
$$

Figure 2: Two examples of derivations

then $A = B$ according to (3). Conversely, if we have $A = B$, then we can provide a derivation

$$
\begin{array}{c} A \\ \Vert \{\mathsf{aw}\downarrow,\mathsf{aw}\uparrow,\mathsf{ac}\downarrow,\mathsf{ac}\uparrow,\alpha\downarrow,\alpha\uparrow,\sigma\downarrow,\sigma\uparrow,\mathbf{f}\downarrow,\mathbf{f}\uparrow,\mathbf{t}\downarrow,\mathbf{t}\uparrow\} \\ B \end{array} \qquad (6)
$$

To see this, note that the equations in the last column of (3) can be simulated by the following derivations:

$$
\mathsf{ac}\uparrow \frac{\mathbf{f}}{\mathbf{f} \wedge \mathbf{f}}
\qquad
\begin{array}{rc} & \mathbf{f} \wedge \mathbf{f} \\ \cline{2-2} \mathsf{aw}\uparrow & \mathbf{t} \wedge \mathbf{f} \\ \cline{2-2} \mathbf{f}\uparrow & \mathbf{f} \end{array}
\qquad
\begin{array}{rc} & \mathbf{t} \\ \cline{2-2} \mathbf{f}\downarrow & \mathbf{t} \vee \mathbf{f} \\ \cline{2-2} \mathsf{aw}\downarrow & \mathbf{t} \vee \mathbf{t} \end{array}
\qquad
\mathsf{ac}\downarrow \frac{\mathbf{t}}{\mathbf{t} \vee \mathbf{f}}
$$

(Recall that the rules $\mathsf{aw}\downarrow$, $\mathsf{aw}\uparrow$, $\mathsf{ac}\downarrow$, $\mathsf{ac}\uparrow$ can also be applied to units.)

Note that system SKS is self-dual, i.e., for every rule r in SKS there is a dual rule r' which is obtained from r by negating and exchanging premise and

conclusion. For example, the rules $\mathsf{ac}\downarrow$ and $\mathsf{ac}\uparrow$ are dual to each other, and the rule s (*switch*) is dual to itself. The rules with an $\downarrow$ in the name are called *down-rules*, and the rules with an $\uparrow$ in the name are called *up-rules*. The system consisting of switch, medial, and all down-rules is called $\mathsf{SKS}\downarrow$, and the system consisting of switch, medial, and all up-rules is called $\mathsf{SKS}\uparrow$. The system $\mathsf{SKS}\downarrow$ is also called KS.

2.2 Remark Following Remark 2.1, we can define KS' to be the system $\{\mathsf{ai}\downarrow, \mathsf{aw}\downarrow, \mathsf{ac}\downarrow, \mathsf{s}, \mathsf{m}\}$ extended by the equality rule in (4). This is the way as KS has been presented in [BT01]. However, KS' and our KS are not strongly equivalent as in (5). We only have that

$$\begin{array}{c} A \\ \| \,\mathsf{KS} \\ B \end{array} \quad \text{implies} \quad \begin{array}{c} A \\ \| \,\mathsf{KS}' \\ B \end{array} \tag{7}$$

The other direction does not hold. For example the two derivations

$$\begin{array}{c} \mathbf{f} \\ \| \,\mathsf{KS}' \\ \mathbf{f} \wedge \mathbf{f} \end{array} \quad \text{and} \quad \begin{array}{c} A \wedge (B \wedge C) \\ \| \,\mathsf{KS}' \\ (A \wedge B) \wedge C \end{array} \tag{8}$$

cannot be obtained in our version of KS. However, we have

$$\begin{array}{c} \mathbf{t} \\ \| \,\mathsf{KS} \\ A \end{array} \quad \text{iff} \quad \begin{array}{c} \mathbf{t} \\ \| \,\mathsf{KS}' \\ A \end{array} \tag{9}$$

This follows from the completeness of KS, which is proved below in Theorem 2.6.

2.3 Remark The rules $\mathsf{aw}\downarrow$ (*atomic weakening*), $\mathsf{ac}\downarrow$ (*atomic contraction*), $\mathsf{ai}\downarrow$ (*atomic interaction*) and their duals $\mathsf{aw}\uparrow$ (*atomic coweakening*), $\mathsf{ac}\uparrow$ (*atomic cocontraction*), $\mathsf{ai}\uparrow$ (*atomic cointeraction* or *atomic cut*) have special status because they are applicable only to atoms. However, their general versions

$$\mathsf{w}\downarrow \frac{F\{\mathbf{f}\}}{F\{A\}} \quad , \qquad \mathsf{c}\downarrow \frac{F\{A \vee A\}}{F\{A\}} \quad , \qquad \mathsf{i}\downarrow \frac{F\{\mathbf{t}\}}{F\{A \vee \bar{A}\}} \tag{10}$$

and

$$\mathsf{w}\uparrow \frac{F\{A\}}{F\{\mathbf{t}\}} \quad , \qquad \mathsf{c}\uparrow \frac{F\{A\}}{F\{A \wedge A\}} \quad , \qquad \mathsf{i}\uparrow \frac{F\{A \wedge \bar{A}\}}{F\{\mathbf{f}\}} \tag{11}$$

are derivable in SKS, as Proposition 2.4 below shows.

We use the following conventions. The *size* of a formula A is the number of symbols in A. The *length* of a derivation Δ, denoted by $\mathsf{length}(\Delta)$, is the number of lines in it, and its *width*, denoted by $\mathsf{width}(\Delta)$ is the maximal formula size appearing in Δ.

2.4 Proposition *For every formula A, there are derivations*

$$\begin{array}{c}\mathbf{f}\\ \Delta_{\mathsf{w}\downarrow}\,\|\,\mathsf{SKS}\\ A\end{array}\quad,\quad\begin{array}{c}A\vee A\\ \Delta_{\mathsf{c}\downarrow}\,\|\,\mathsf{SKS}\\ A\end{array}\quad,\quad\begin{array}{c}\mathbf{t}\\ \Delta_{\mathsf{i}\downarrow}\,\|\,\mathsf{SKS}\\ A\vee\bar{A}\end{array}\tag{12}$$

whose lengths and widths are linear in the size of A.

Proof: The proof is a straightforward induction on A, and has already been presented in [BT01]. However, since we are here more restrictive with respect to the equational theory (3), we show here again the inductive cases:

$$\text{for } \mathsf{w}\downarrow\text{:}\quad \dfrac{\dfrac{\dfrac{\mathbf{f}}{\mathbf{f}\vee\mathbf{f}}\,\mathsf{w}\downarrow}{A_1\vee\mathbf{f}}\,\mathsf{w}\downarrow}{A_1\vee A_2}\qquad\qquad \dfrac{\dfrac{\dfrac{\mathbf{f}}{\mathbf{f}\wedge\mathbf{f}}\,\mathsf{w}\downarrow}{A_1\wedge\mathbf{f}}\,\mathsf{w}\downarrow}{A_1\wedge A_2}\,\mathsf{ac}\uparrow$$

$$\text{for } \mathsf{c}\downarrow\text{:}\quad \dfrac{\dfrac{\dfrac{[A_1\vee A_2]\vee[A_1\vee A_2]}{[A_1\vee A_1]\vee[A_2\vee A_2]}\,5{\cdot}\sigma\downarrow,3{\cdot}\alpha\downarrow}{A_1\vee[A_2\vee A_2]}\,\mathsf{c}\downarrow}{A_1\vee A_2}\,\mathsf{c}\downarrow\qquad\qquad \dfrac{\dfrac{\dfrac{(A_1\wedge A_2)\vee(A_1\wedge A_2)}{[A_1\vee A_1]\wedge[A_2\vee A_2]}\,\mathsf{m}}{A_1\wedge[A_2\vee A_2]}\,\mathsf{c}\downarrow}{A_1\wedge A_2}\,\mathsf{c}\downarrow$$

$$\text{for } \mathsf{i}\downarrow\text{:}\quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{t}}{A_2\vee\bar{A}_2}\,\mathsf{i}\downarrow}{(\bar{A}_2\wedge\mathbf{t})\vee A_2}\,\mathbf{t}\downarrow}{(\bar{A}_2\wedge[\bar{A}_1\vee A_1])\vee A_2}\,\mathsf{i}\downarrow}{[(\bar{A}_2\wedge\bar{A}_1)\vee A_1]\vee A_2}\,\mathsf{s}}{[A_1\vee A_2]\vee(\bar{A}_2\wedge\bar{A}_1)}\,3{\cdot}\sigma\downarrow,1{\cdot}\alpha\downarrow\qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{t}}{A_1\vee\bar{A}_1}\,\mathsf{i}\downarrow}{(A_1\wedge\mathbf{t})\vee\bar{A}_1}\,\mathbf{t}\downarrow}{(A_1\wedge[A_2\vee\bar{A}_2])\vee\bar{A}_1}\,\mathsf{i}\downarrow}{[(A_1\wedge A_2)\vee\bar{A}_2]\vee\bar{A}_1}\,\mathsf{s}}{(A_1\wedge A_2)\vee[\bar{A}_2\vee\bar{A}_1]}\,4{\cdot}\sigma\downarrow,1{\cdot}\alpha\downarrow$$

Here the notation $n{\cdot}\sigma\downarrow, m{\cdot}\alpha\downarrow$ means that there is a derivation consisting of n instances of $\sigma\downarrow$ and m instances of $\alpha\downarrow$. It follows immediately that length and width are linear in A. □

The system $\mathsf{SKS}\setminus\{\mathsf{ai}\downarrow,\mathsf{ai}\uparrow,\mathsf{aw}\downarrow,\mathsf{aw}\uparrow,\mathsf{ac}\downarrow,\mathsf{ac}\uparrow,\mathsf{m}\}\cup\{\mathsf{i}\downarrow,\mathsf{i}\uparrow,\mathsf{w}\downarrow,\mathsf{w}\uparrow,\mathsf{c}\downarrow,\mathsf{c}\uparrow\}$ is called SKSg, and $\mathsf{SKSg}\downarrow$, $\mathsf{SKSg}\uparrow$, and KSg can be defined accordingly.

2.5 Definition A *proof* of a formula A is a derivation

$$\begin{array}{c}\mathbf{t}\\ \|\,\mathsf{S}\\ A\end{array}$$

with truth as premise. A system is *complete*, if it can prove all tautologies.

In [BT01] it has been shown that KS' (see Remark 2.2) is complete. Due to the absence of the equations for commutativity and associativity of conjunction, that proof does not apply for KS. Thus, we give a different proof below.

2.6 Theorem *System* KS *is complete.*

Proof: Given a tautology A, we proceed by induction on the number of $\wedge$-nodes in the formula tree of A to show that there is a derivation

$$\begin{array}{c}\mathbf{t}\\ \Vert\ \mathsf{KS}\\ A\end{array} \tag{13}$$

If A contains no conjunction, then it is a disjunction of atoms, containing either $\mathbf{t}$ or a pair of a propositional variable and its dual (because A is a tautology). In the first case, we repeatedly apply $\alpha\downarrow$ and $\sigma\downarrow$ to move the $\mathbf{t}$ to the first position, and then apply repeatedly $\mathsf{aw}\downarrow$ and $\mathbf{f}\downarrow$ to remove everything else. In the second case, we use $\alpha\downarrow$ and $\sigma\downarrow$ to bring the pair together so that we can apply $\mathsf{ai}\downarrow$ to it. Then we proceed as in the first case.

Let us now assume that A contains at least one occurrence of $\wedge$. Then we can use $\alpha\downarrow$ and $\sigma\downarrow$ to bring A into the form $(B_1 \wedge B_2) \vee C$. Since this formula is a tautology, also the formulas $B_1 \vee C$ and $B_2 \vee C$ are tautologies. Hence, by induction hypothesis we have derivations

$$\begin{array}{c}\mathbf{t}\\ \Delta_1 \Vert\ \mathsf{KS}\\ B_1 \vee C\end{array} \qquad \text{and} \qquad \begin{array}{c}\mathbf{t}\\ \Delta_2 \Vert\ \mathsf{KS}\\ B_2 \vee C\end{array}$$

Wen can now build:

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathbf{t}\\ \Delta_1 \Vert\ \mathsf{KS}\\ B_1 \vee C\end{array}}{(B_1 \wedge \mathbf{t}) \vee C}\,\mathbf{t}\downarrow}{\begin{array}{c}\Delta_2 \Vert\ \mathsf{KS}\\ (B_1 \wedge [B_2 \vee C]) \vee C\end{array}}}{[(B_1 \wedge B_2) \vee C] \vee C}\,\mathsf{s}}{\begin{array}{c}\Delta_3 \Vert\ \{\mathsf{ac}\downarrow,\mathsf{m},\alpha\downarrow,\sigma\downarrow\}\\ (B_1 \wedge B_2) \vee C\end{array}}$$

where the existence of Δ_3 follows from the proof of Proposition 2.4. □

2.7 Definition A deep inference system S is *implicationally complete* if for every valid implication $A \supset B$ there is a derivation

$$\begin{array}{c}A\\ \Vert\ \mathsf{S}\\ B\end{array}$$

The following two theorems and their corollaries have already been presented in [BT01, Brü03b, Brü06b].

2.8 Theorem *System* SKS *is implicationally complete, and system* KS *is not implicationally complete.*

Proof: Let $A \supset B$ be a valid implication. Since KS is complete, we have a proof

$$\begin{array}{c} \mathbf{t} \\ \Delta \,\|\, \mathsf{KS} \\ \bar{A} \vee B \end{array}$$

Since the rule $\mathsf{i}\!\uparrow$ is derivable in SKS (Proposition 2.4), we can construct the derivation

$$\dfrac{\dfrac{\dfrac{\mathbf{t}\!\downarrow \dfrac{A}{A \wedge \mathbf{t}}}{\Delta \,\|\, \mathsf{KS}}}{\mathsf{s}\,\dfrac{A \wedge [\bar{A} \vee B]}{(A \wedge \bar{A}) \vee B}}}{\mathsf{i}\!\uparrow \dfrac{\mathbf{t}\!\uparrow \dfrac{\mathbf{f} \vee B}{B}}{(A \wedge \bar{A}) \vee B}}$$

Hence, SKS is implicationally complete. To see that this is not the case for KS, observe that no rule in KS can introduce a new atom while going up in a derivation. Hence, there can be no derivation

$$\begin{array}{c} a \wedge \bar{a} \\ \| \,\mathsf{KS} \\ \mathbf{f} \end{array}$$

□

2.9 Corollary $\mathsf{KS} + \mathsf{i}\!\uparrow$ *is implicationally complete.*

Proof: Every up-rule $\mathsf{r}\!\uparrow$ in SKS is derivable in the system $\{\mathsf{r}\!\downarrow, \mathsf{i}\!\downarrow, \mathsf{i}\!\uparrow, \mathsf{s}, \mathbf{t}\!\downarrow, \mathbf{t}\!\uparrow\}$:

$$\mathsf{r}\!\uparrow \dfrac{F\{\bar{B}\}}{F\{\bar{A}\}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{\mathbf{t}\!\downarrow \dfrac{F\{\bar{B}\}}{F\{\bar{B} \wedge \mathbf{t}\}}}{\mathsf{i}\!\downarrow \dfrac{F\{\bar{B} \wedge [A \vee \bar{A}]\}}{F\{\bar{B} \wedge [A \vee \bar{A}]\}}}}{\mathsf{r}\!\downarrow \dfrac{F\{\bar{B} \wedge [B \vee \bar{A}]\}}{F\{(\bar{B} \wedge B) \vee \bar{A}\}}}}{\mathsf{s}\,\dfrac{F\{(\bar{B} \wedge B) \vee \bar{A}\}}{F\{\mathbf{f} \vee \bar{A}\}}}}{\mathsf{i}\!\uparrow \dfrac{\mathbf{t}\!\uparrow \dfrac{F\{\mathbf{f} \vee \bar{A}\}}{F\{\bar{A}\}}}{F\{\mathbf{f} \vee \bar{A}\}}} \tag{14}$$

Hence, the corollary follows immediately from the previous theorem (together with the proof of Proposition 2.4). □

2.10 Theorem (Cut Elimination) *The rule* $\mathsf{i}\!\uparrow$ *is admissible for* KS.

Proof: A semantic proof follows immediately from Theorem 2.6. However, there is a syntactic proof of cut elimination for KS' in [Brü03a]. That proof is also applicable in our case even though associativity and commutativity of $\wedge$ are not available. □

2.11 Corollary (Consistency) *There is no formula A, such that* SKS *proves both, A and $\bar{A}$.*

Proof: By way of contradiction, assume we have a formula A and two proofs

$$\begin{array}{c}\mathbf{t}\\ \Delta_1 \,\|\, \mathsf{SKS}\\ A\end{array} \qquad \text{and} \qquad \begin{array}{c}\mathbf{t}\\ \Delta_2 \,\|\, \mathsf{SKS}\\ \bar{A}\end{array}$$

By the self-duality of SKS we can form the dual of Δ_2 and plug it together with Δ_1:

$$\begin{array}{c}\mathbf{t}\\ \Delta_1 \,\|\, \mathsf{SKS}\\ A\\ \bar{\Delta}_2 \,\|\, \mathsf{SKS}\\ \mathbf{f}\end{array}$$

By (14) and cut elimination, we get a proof of $\mathbf{f}$ in KS. By inspecting the rules, one immediately observes that this is impossible. □

For further details on SKS, the reader is referred to [Brü03b]. Let us now come back to robustness. Observe that in the deep inference setting one can have rule schemes that apply to general subformulas, as well as rule schemes that are restricted to atomic subformulas (cf. Remark 2.3). Such a distinction does not exist for Frege-systems, where all axioms and rules are general. Hence, in order to keep things simple, we consider the robustness theorem here only for deep inference systems without atomic restrictions in the inference rules. Then the proof is almost literally the same as for Frege-systems [CR79]. Recall that a system S_2 *p-simulates* a system S_1 if there is a polynomial f such that for every proof Δ_1 in S_1 there is a proof Δ_2 of the same conclusion in S_2 such that $s(\Delta_2) \leq f(s(\Delta_1))$, where $s(\Delta)$ denotes the size of the proof Δ, i.e., the number of symbols in Δ. Note that we have $s(\Delta) \leq \mathsf{length}(\Delta) \cdot \mathsf{width}(\Delta)$.

2.12 Theorem (Robustness) *Let S_1 and S_2 be two implicationally complete deep inference systems (using the same set of connectives), such that S_2 does not contain any rule with an atomic restriction. Then S_2 p-simulates S_1.*

Proof: For every rule scheme in S_1

$$\mathsf{r}\,\frac{F\{A\}}{F\{B\}}$$

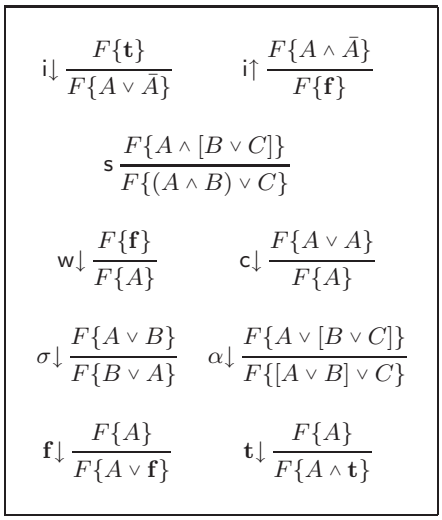


Figure 3: System $\mathsf{KSg} + \mathsf{i}{\uparrow}$

we can provide a derivation scheme

$$\begin{array}{c} F\{A\} \\ \Delta_{\mathsf{r}} \,\big\|\, \mathsf{S}_2 \\ F\{B\} \end{array} \tag{15}$$

because S_2 is implicationally complete. Let c_1 be the maximal length of such a derivation scheme, and let c_2 be the maximal width:

$$c_1 = \max\{\mathsf{length}(\Delta_{\mathsf{r}}) \mid \mathsf{r} \in \mathsf{S}_1\} \quad \text{and} \quad c_2 = \max\{\mathsf{width}(\Delta_{\mathsf{r}}) \mid \mathsf{r} \in \mathsf{S}_1\}$$

When we replace an instance of r by an instance of Δ_{r}, its length does not change, and its width is smaller than or equal to the product of the width of Δ_{r} and the size of the conclusion of the rule instance. Hence, we can transform a derivation Δ_1 in S_1 into a derivation Δ_2 in S_2 such that $\mathsf{length}(\Delta_2) \leq c_1 \cdot \mathsf{length}(\Delta_1)$ and $\mathsf{width}(\Delta_2) \leq c_2 \cdot \mathsf{width}(\Delta_1)$. □

If we allow rules with atomic restrictions in S_2, as it is the case with SKS, we can no longer "instantiate" the derivations Δ_{r} in (15). Thus we have to find another way to ensure that every derivation Δ_{r} in S_2 replacing a rule instance r in S_1 has only polynomial size. Fortunately, for SKS this is ensured by Proposition 2.4.

2.13 Corollary SKS *p-simulates any other implicationally complete deep inference system.*

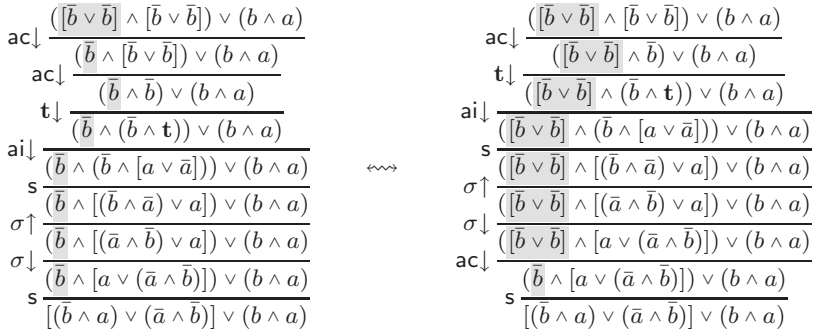


Figure 4: Example for type-A bureaucracy

Proof: From Proposition 2.4 we get immediately that SKS p-simulates the system $\mathsf{KSg} + \mathsf{i}\uparrow$, shown in Figure 3, which in turn p-simulates any other implicationally complete deep inference system by Theorem 2.12. □

It is not obvious how to give a direct proof of Theorem 2.12 if atomic rules are involved. Is there a general method, or do we have to prove an equivalent of Proposition 2.4 for every system individually? We have to leave this question open for future research.

3 The ABC of Bureaucracy

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

Consider now the right derivation in Figure 2. It is intuitively clear that we would not change the essence of the derivation if we exchanged the two instances of $\mathsf{ac}\downarrow$ marked highlighted between the m and the $\mathsf{t}\downarrow$ in the lower half of the derivation. That the two instances of $\mathsf{ac}\downarrow$ are ordered one above the other can be considered to be an act of "bureaucracy". In fact, following this intuition, we can permute the first $\mathsf{ac}\downarrow$ almost all the way down in the derivation, as shown in Figure 4. This kind of "bureaucracy" has been called *bureaucracy of type A* by Guglielmi [Gug04a]. More generally whenever there is a derivation Δ_1 from A to B and a derivation Δ_2 from C to D, then

$$\begin{array}{c} A \wedge C \\ {\scriptstyle\Delta_1}\,\| \\ B \wedge C \\ {\scriptstyle\Delta_2}\,\| \\ B \wedge D \end{array} \qquad \text{and} \qquad \begin{array}{c} A \wedge C \\ {\scriptstyle\Delta_2}\,\| \\ A \wedge D \\ {\scriptstyle\Delta_1}\,\| \\ B \wedge D \end{array} \tag{16}$$

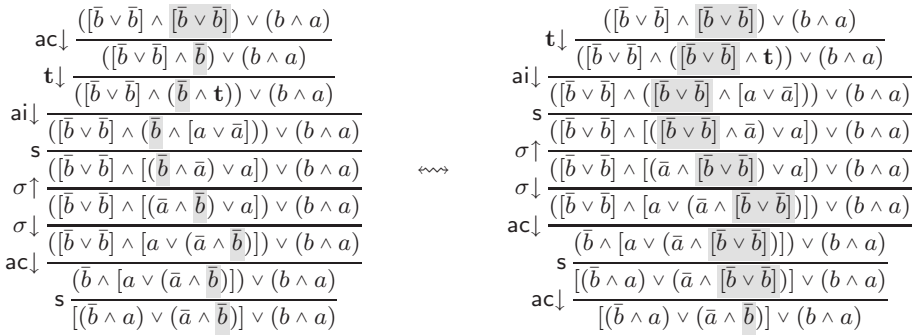


Figure 5: Example for type-B bureaucracy

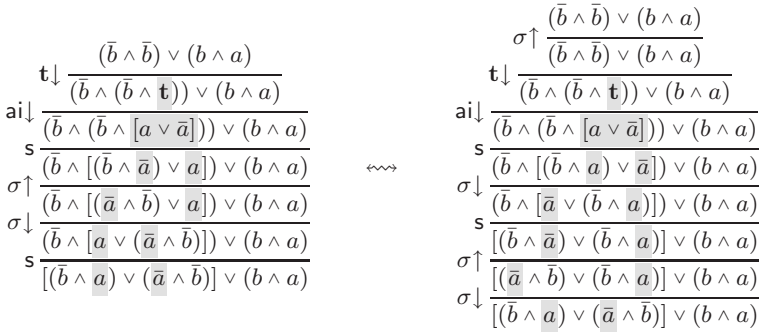


Figure 6: Example for type-C bureaucracy

as well as any other "merge" of Δ_1 and Δ_2 should be considered to be the same. This suggest the more efficient notation

$$\begin{array}{c} A \wedge C \\ {\scriptstyle \Delta_1 \wedge \Delta_2} \,\Big\| \\ B \wedge D \end{array} \tag{17}$$

and similarly for disjunction. Guglielmi calls a formalism which *per se* makes these identifications *Formalism A*. However, Formalism A does not allow the identification of the two derivations in Figure 5. where the $\mathsf{ac}\downarrow$ is not "next to" another derivation but "inside" another derivation. This phenomenon has been called *bureaucracy of type B* [Gug04b]. Then *Formalism B* is a formalism that avoids this kind of bureaucracy. More generally, the following two derivations

deep inference	$\simeq$	working in a syntactic category
calculus of structures	$\simeq$	free syntactic category
Formalism A	$\simeq$	free syntactic category + bifunctoriality of $- \wedge -$ and $- \vee -$ + naturality of $\alpha\downarrow$, $\alpha\uparrow$, $\sigma\downarrow$, $\sigma\uparrow$, $\mathbf{t}\downarrow$, $\mathbf{t}\uparrow$, $\mathbf{f}\downarrow$, $\mathbf{f}\uparrow$ + coherence for associativity, commutativity, and units + equations for $\vee$-monoids and for $\wedge$-comonoids + trivial rule identities
Formalism B	$\simeq$	Formalism A + naturality of s and m
Formalism C	$\simeq$	Formalism B + "full coherence"

Figure 7: The bureaucracy-ABC vs. categorical proof theory

should be identified:

$$\begin{array}{c} F\{A\} \\ {\scriptstyle\Delta_1}\,\|\, \\ F\{B\} \\ {\scriptstyle\Delta_3}\,\|\, \\ F'\{B\} \end{array} \qquad \text{and} \qquad \begin{array}{c} F\{A\} \\ {\scriptstyle\Delta_3}\,\|\, \\ F'\{A\} \\ {\scriptstyle\Delta_1}\,\|\, \\ F'\{B\} \end{array} \tag{18}$$

where Δ_1 is as above and Δ_3 only works on the context.

Besides bureaucracy of type A and B, we can observe another kind of bureaucracy, which we will call here *bureaucracy of type C*. Consider the two derivations in Figure 6. They can considered to be essentially the same, because in both the same a and $\bar{a}$ in the conclusion are "brought together" and disappear in an identity. The difference is that the derivation on the right contains two more applications of commutativity in which the two $\bar{b}$ are exchanged. But neither Formalism A nor Formalism B can identify the two. Let us call *Formalism C* a formalism that is able to avoid this kind of bureaucracy.

In their original presentations [Gug04a, Gug04b], formalisms A and B where presented modulo the equational theory (see Remark 2.1). When the equations are explicit, we see that some more care is needed. For example, the following two derivations should be identified:

$$\alpha\downarrow \dfrac{\sigma\downarrow \dfrac{\alpha\downarrow \dfrac{A \vee [B \vee [D \vee C]]}{[A \vee B] \vee [D \vee C]}}{[A \vee B] \vee [C \vee D]}}{[[A \vee B] \vee C] \vee D} \qquad \leftrightsquigarrow \qquad \sigma\downarrow \dfrac{\alpha\downarrow \dfrac{\alpha\downarrow \dfrac{A \vee [B \vee [D \vee C]]}{A \vee [B \vee [C \vee D]]}}{A \vee [[B \vee C] \vee D]}}{[A \vee [B \vee C]] \vee D}}{[[A \vee B] \vee C] \vee D}$$

$$
\begin{array}{l}
\alpha\downarrow \dfrac{A \vee [B \vee C]}{[A \vee B] \vee C} \\
\sigma\downarrow \dfrac{}{C \vee [A \vee B]} \\
\sigma\downarrow \dfrac{}{C \vee [B \vee A]} \\
\alpha\downarrow \dfrac{}{[C \vee B] \vee A} \\
\sigma\downarrow \dfrac{}{[B \vee C] \vee A} \\
\sigma\downarrow \dfrac{}{A \vee [B \vee C]}
\end{array}
\quad = \quad A \vee [B \vee C]
\qquad\qquad
\begin{array}{l}
\sigma\downarrow \dfrac{[A \vee B] \vee C}{C \vee [A \vee B]} \\
\sigma\downarrow \dfrac{}{C \vee [B \vee A]} \\
\alpha\downarrow \dfrac{}{[C \vee B] \vee A} \\
\sigma\downarrow \dfrac{}{[B \vee C] \vee A} \\
\sigma\downarrow \dfrac{}{A \vee [B \vee C]} \\
\alpha\downarrow \dfrac{}{[A \vee B] \vee C}
\end{array}
\quad = \quad [A \vee B] \vee C
$$

Figure 8: The equations that make $\alpha\downarrow$ an isomorphisms

but, on the other hand, the following two derivations should not be identified:

$$
\begin{array}{l}
\sigma\downarrow \dfrac{A \vee A}{} \\
\sigma\downarrow \dfrac{}{A \vee A}
\end{array}
\qquad \not\leftrightsquigarrow \qquad
\sigma\downarrow \frac{A \vee A}{A \vee A}
$$

In the following, we will make the notions of formalisms A, B, and C precise. We will see that the language of category theory contains all the needed vocabulary for doing so.[3]

Every deep inference proof system defines a category in which the formulas are the objects and the derivations are the morphisms. Composition of morphisms is obtained by plugging together derivations. In this setting, the intuitions about bureaucracy given above, can easily be translated into category theoretical terms, as it is shown in Figure 7 (see also [Hug04] and [McK05] for work relating deep inference and categorical proof theory). The remainder of this section is dedicated to explaining the terminology in Figure 7:

Bifunctoriality of $- \wedge -$ means exactly the identification of the three derivations in (16) and (17), and *bifunctoriality of* $- \vee -$ is defined analogously. *Naturality of* $\sigma\downarrow$ means that the following two equations between derivations hold:

$$
\begin{array}{c}
B \vee A' \\
{\scriptstyle \Delta_1}\,\big\| \\
\sigma\downarrow \dfrac{B \vee A}{A \vee B}
\end{array}
\quad = \quad
\begin{array}{c}
\sigma\downarrow \dfrac{B \vee A'}{A' \vee B} \\
{\scriptstyle \Delta_1}\,\big\| \\
A \vee B
\end{array}
\qquad \text{and} \qquad
\begin{array}{c}
B' \vee A \\
{\scriptstyle \Delta_2}\,\big\| \\
\sigma\downarrow \dfrac{B \vee A}{A \vee B}
\end{array}
\quad = \quad
\begin{array}{c}
\sigma\downarrow \dfrac{B' \vee A}{A \vee B'} \\
{\scriptstyle \Delta_2}\,\big\| \\
A \vee B
\end{array}
\tag{19}
$$

where Δ_1 is an arbitrary derivation from A' to A, and Δ_2 is an arbitrary derivation from B' to B. *Naturality* of the other rules is defined analogously.

The terminology *coherence for associativity, commutativity, and units* refers to the set of equations shown in Figures 8, 9, and 10 for disjunction and the

[3] We refer the reader to the standard texts [Mac71, BW99] for the basic notions of category theory. However, for understanding this paper the reader does not need any knowledge of category theory, since we will explain everything in proof theoretical terms.

$$\sigma\downarrow\dfrac{\dfrac{A\vee B}{B\vee A}}{A\vee B} \;=\; A\vee B \qquad\qquad \begin{matrix}\mathbf{f}\downarrow\\ \mathbf{t}\uparrow\end{matrix}\dfrac{\dfrac{A}{A\vee\mathbf{f}}}{A} \;=\; A \qquad\qquad \begin{matrix}\mathbf{t}\uparrow\\ \mathbf{f}\downarrow\end{matrix}\dfrac{\dfrac{A\vee\mathbf{f}}{A}}{A\vee\mathbf{f}} \;=\; A\vee\mathbf{f}$$

Figure 9: The equations that make $\sigma\downarrow$, $\mathbf{f}\downarrow$, and $\mathbf{t}\uparrow$ isomorphisms

$$\begin{matrix}\alpha\downarrow\\ \alpha\downarrow\end{matrix}\dfrac{\dfrac{A\vee[B\vee[C\vee D]]}{[A\vee B]\vee[C\vee D]}}{[[A\vee B]\vee C]\vee D} \;=\; \begin{matrix}\alpha\downarrow\\ \alpha\downarrow\\ \alpha\downarrow\end{matrix}\dfrac{\dfrac{\dfrac{A\vee[B\vee[C\vee D]]}{A\vee[[B\vee C]\vee D]}}{[A\vee[B\vee C]]\vee D}}{[[A\vee B]\vee C]\vee D}$$

$$\begin{matrix}\sigma\downarrow\\ \alpha\downarrow\\ \sigma\downarrow\end{matrix}\dfrac{\dfrac{\dfrac{A\vee[B\vee C]}{A\vee[C\vee B]}}{[A\vee C]\vee B}}{[C\vee A]\vee B} \;=\; \begin{matrix}\alpha\downarrow\\ \sigma\downarrow\\ \alpha\downarrow\end{matrix}\dfrac{\dfrac{\dfrac{A\vee[B\vee C]}{[A\vee B]\vee C}}{C\vee[A\vee B]}}{[C\vee A]\vee B} \qquad\qquad \begin{matrix}\mathbf{f}\downarrow\\ \alpha\downarrow\end{matrix}\dfrac{\dfrac{A\vee B}{A\vee[B\vee\mathbf{f}]}}{[A\vee B]\vee\mathbf{f}} \;=\; \mathbf{f}\downarrow\dfrac{A\vee B}{[A\vee B]\vee\mathbf{f}}$$

Figure 10: Mac Lane's coherence equations for symmetric monoidal categories

corresponding equations for conjunction (which are not shown). In Figures 8 and 9, the single formula on the right-hand side of an equation stands for a single-line derivation where premise and conclusion coincide. In category theoretical terms this stands for the identity morphism on that formula. The equations in Figures 8 and 9 then say that $\alpha\downarrow$, $\sigma\downarrow$, $\mathbf{f}\downarrow$, and $\mathbf{t}\uparrow$ are treated as isomorphisms in the category theoretical sense, where $\mathbf{f}\downarrow$ and $\mathbf{t}\uparrow$ are inverse to each other, $\sigma\downarrow$ is inverse to itself, and the inverse of $\alpha\downarrow$ is a derivations consisting of four $\sigma\downarrow$ and one $\alpha\downarrow$. The equations in Figure 10 are the coherence equations for symmetric monoidal categories given by Mac Lane [Mac63] (see also [Mac71, Kel64]). That these equations give indeed the proof identifications desired for Formalism A is a consequence of Mac Lane's coherence theorem for symmetric monoidal categories [Mac63]. Translated into our setting, this theorem is stated as follows.

3.1 Theorem *Let two formulas A and B and two derivations*

$$\begin{matrix}A\\ \Delta_1\;\Big\|\;\{\alpha\downarrow,\alpha\uparrow,\sigma\downarrow,\sigma\uparrow,\mathbf{t}\downarrow,\mathbf{t}\uparrow,\mathbf{f}\downarrow,\mathbf{f}\uparrow\}\\ B\end{matrix} \qquad \textit{and} \qquad \begin{matrix}A\\ \Delta_2\;\Big\|\;\{\alpha\downarrow,\alpha\uparrow,\sigma\downarrow,\sigma\uparrow,\mathbf{t}\downarrow,\mathbf{t}\uparrow,\mathbf{f}\downarrow,\mathbf{f}\uparrow\}\\ B\end{matrix}$$

be given. If we assume bifunctoriality of $-\vee-$ and $-\wedge-$, naturality of $\alpha\downarrow$, $\alpha\uparrow$, $\sigma\downarrow$, $\sigma\uparrow$, $\mathbf{t}\downarrow$, $\mathbf{t}\uparrow$, $\mathbf{f}\downarrow$, and $\mathbf{f}\uparrow$, and the equations in Figures 8, 9, and 10, then $\Delta_1=\Delta_2$.

Proof: This immediately follows from Mac Lane's coherence theorem for symmetric monoidal categories [Mac63] and the fact that the two monoidal struc-

$$\alpha\downarrow\dfrac{\mathsf{ac}\downarrow\dfrac{a\vee[a\vee a]}{[a\vee a]\vee a}}{\mathsf{ac}\downarrow\dfrac{a\vee a}{a}} = \mathsf{ac}\downarrow\dfrac{a\vee[a\vee a]}{\mathsf{ac}\downarrow\dfrac{a\vee a}{a}} \qquad \sigma\downarrow\dfrac{a\vee a}{\mathsf{ac}\downarrow\dfrac{a\vee a}{a}} = \mathsf{ac}\downarrow\dfrac{a\vee a}{a} \qquad \mathsf{aw}\downarrow\dfrac{a\vee\mathbf{f}}{\mathsf{ac}\downarrow\dfrac{a\vee a}{a}} = \mathbf{t}\uparrow\dfrac{a\vee\mathbf{f}}{a}$$

Figure 11: Equations for $\vee$-monoids

$$\mathsf{ai}\downarrow\frac{\mathbf{t}}{\mathbf{t}\vee\mathbf{f}} = \mathbf{f}\downarrow\frac{\mathbf{t}}{\mathbf{t}\vee\mathbf{f}} \qquad \mathsf{aw}\downarrow\frac{\mathbf{t}}{\mathbf{f}} = \mathsf{aw}\uparrow\frac{\mathbf{t}}{\mathbf{f}} \qquad \mathsf{ac}\uparrow\frac{\mathbf{t}}{\mathbf{t}\wedge\mathbf{t}} = \mathbf{t}\downarrow\frac{\mathbf{t}}{\mathbf{t}\wedge\mathbf{t}}$$

$$\mathsf{ai}\uparrow\frac{\mathbf{t}\wedge\mathbf{f}}{\mathbf{f}} = \mathbf{f}\uparrow\frac{\mathbf{t}\wedge\mathbf{f}}{\mathbf{f}} \qquad \mathsf{ac}\downarrow\frac{\mathbf{f}\vee\mathbf{f}}{\mathbf{f}} = \mathbf{t}\uparrow\frac{\mathbf{f}\vee\mathbf{f}}{\mathbf{f}}$$

Figure 12: Trivial rule identities

tures for $\wedge$ and $\vee$ do not interact. □

Due to the presence of (atomic) contraction and weakening we should also add the *equations for $\vee$-monoids* (shown in Figure 11) and for *$\wedge$-comonoids* (which are the up-down inverses of those in Figure 11). These equations say that if we apply the contraction rule to many copies of the same atom then it does not matter in which order we do that, and if we have a weakening followed by a contraction on the same atom, then this is the same as doing nothing.

With *trivial rule identities* we mean the equations in Figure 12. For going from $\mathbf{t}$ to $\mathbf{t}\vee\mathbf{f}$, we could use the rule $\mathsf{ai}\downarrow$ as well as the rule $\mathbf{f}\downarrow$. The first equation in Figure 12 says that we do not make a difference between the two. The other equations in that figure are similar.

Unfortunately, the terminology *full coherence* in Figure 7 does not (yet) have such a precise meaning as formulated in Theorem 3.1 for associativity commutativity and units. In fact, it is an important research problem to unveil the necessary equations for obtaining a coherence theorem. An example of a desired equation is the following:

$$\mathsf{s}\dfrac{(A\wedge C)\vee(B\wedge[D\vee E])}{\alpha\downarrow\dfrac{(A\wedge C)\vee[(B\wedge D)\vee E]}{\mathsf{m}\dfrac{[(A\wedge C)\vee(B\wedge D)]\vee E}{([A\vee B]\wedge[C\vee D])\vee E}}} = \mathsf{m}\dfrac{(A\wedge C)\vee(B\wedge[D\vee E])}{\alpha\downarrow\dfrac{[A\vee B]\wedge[C\vee[D\vee E]]}{\mathsf{s}\dfrac{[A\vee B]\wedge[[C\vee D]\vee E]}{([A\vee B]\wedge[C\vee D])\vee E}}} \tag{20}$$

For further details, the reader is referred to [LS05a, Str07b, Lam07]. Even though the algebra for making the intuition behind Formalism C is not yet fully developed, we can use the notion of *proof graphs* to give a precise meaning to Formalism C in a combinatorial sense. This will be done in the following sections.

An alternative approach towards bureaucracy from a category theoretical viewpoint is based on n-categories and n-dimensional rewriting [Gui06], where proofs are three-dimensional objects, and bureaucracy is eliminated by isotopy.

4 Proof Nets and Proof Graphs for Classical Logic

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

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

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

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

This additional graph structure is supposed to capture the *essence* of the proof. To our knowledge the first notion of proof net in this more modern setting were [HvG03] for unit-free multiplicative additive linear logic (MALL) and [SL04, Hug05] for multiplicative linear logic (MLL) with units.[4] Then in [LS05b] proof nets for classical logic obeying Slogan 2 followed.

The notion of proof net usually comes with a so called *correctness criterion* which allows to decide whether a given "net" does indeed represent a proof. If this criterion provides a polynomial decision procedure (polynomial in the size of the proof net) then one can speak of a *proof system* in the sense of Cook and Reckhow [CR79]. For the proof nets in [Rob03] for classical logic and the ones in [Gir96] and [HvG03] for MALL we have such a polynomial criterion. But for the proof nets for classical logic in [LS05b] there is only an exponential criterion. In order to avoid confusion, we will in such a case speak of *proof graphs*, in order to reserve the term *proof net* for objects with a polynomial correctness criterion.

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

In the remainder of this section we will recall the proof graphs for classical logic presented in [LS05b]. More precisely, we will concentrate on the version of proof graphs which have been called $\mathbb{N}$-*nets* in [LS05b].

As before, the set of *formulas* is generated via the binary connectives $\wedge$ (*conjunction*) and $\vee$ (*disjunction*) from the set $\mathcal{A} \cup \bar{\mathcal{A}} \cup \{\mathbf{t}, \mathbf{f}\}$, whose elements are called *atoms*. A finite list $\Gamma = A_1, A_2, \ldots, A_n$ of formulas is called a *sequent*. We will consider formulas as binary trees (and sequents as forests), whose leaves are decorated by atoms, and whose inner nodes are decorated by the connectives.

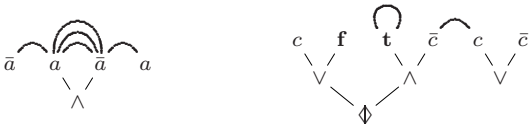
There is a special kind of auxiliary formula, called *cut*, which is of the shape $B \Diamond \bar{B}$, where B is an arbitrary formula and where $\Diamond$ is called the *cut connective*. It is important to note that $\Diamond$ is allowed only at the root of a formula tree. A *cut sequent* is a finite list $\Sigma = B_1 \Diamond \bar{B}_1, \ldots, B_n \Diamond \bar{B}_n$ of cuts.

4.1 Definition A *proof graph* $P, \Sigma \rhd \Gamma$ consists of a sequent Γ, a cut sequent Σ, and an undirected multi-graph P whose set of vertices is the set of leaves of Γ and Σ and whose set of edges obeys the following conditions:

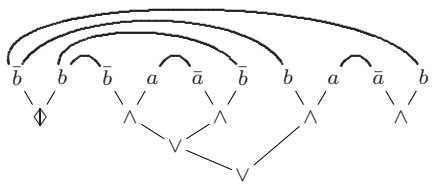
1. whenever there is an edge between two leaves, then one is decorated by a propositional variable a and the other by its dual $\bar{a}$,
2. whenever there is an edge connecting a leaf to itself, then this leaf is decorated by $\mathbf{t}$, and for each $\mathbf{t}$ there is *exactly* one such edge.

4.2 Remark There are two differences to the $\mathbb{N}$-nets of [LS05b]: first, we disallow here edges connecting $\mathbf{f}$ and $\mathbf{t}$, and second, we disallow multiple edges connecting a $\mathbf{t}$ to itself. This gives us a better algebraic behavior of proofs with respect to the units. This is explained in more detail in [LS05a, Str07b]. However, we *do allow* multiple edges connecting a pair of a propositional variable and its dual, in order to keep track of the size of the proof.

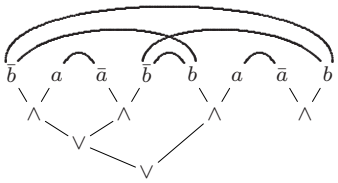
One can think of P also as an undirected graph whose edges are labeled by natural numbers (hence the name $\mathbb{N}$-net in [LS05b]), but in this paper we draw it as multi-graph. Here are two examples:



In the following, we will consider only proof graphs $P, \Sigma \rhd \Gamma$ where Γ contains exactly two formulas (but there is no restriction on the number of cuts in Σ). Here are two examples, one with and one without cuts:

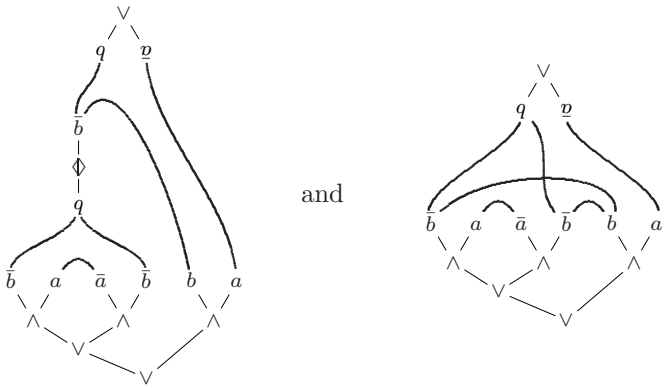


(21)



(22)

The two proof graphs in (21) and (22) can also be drawn as follows where the two formulas in Γ are at the outer ends of the picture:



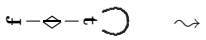
(23)

This will be the preferred way from now on.

4.3 Cut Reduction The cut reduction procedure for proof graphs is defined as follows. For cuts on compound formulas we have:

(24)

For saving space, the picture is put on the side. Cuts on the units can simply be removed:



(25)

Finally, atomic cuts are reduced as follows:

(26)

This means that for every pair of edges, where one is connected to one side of the atomic cut and the other is connected to the other side of the cut, there is in the reduced proof graph an edge connecting the two outer ends of the pair. If there happens to be an edge connecting the two leaves of the cut, then this

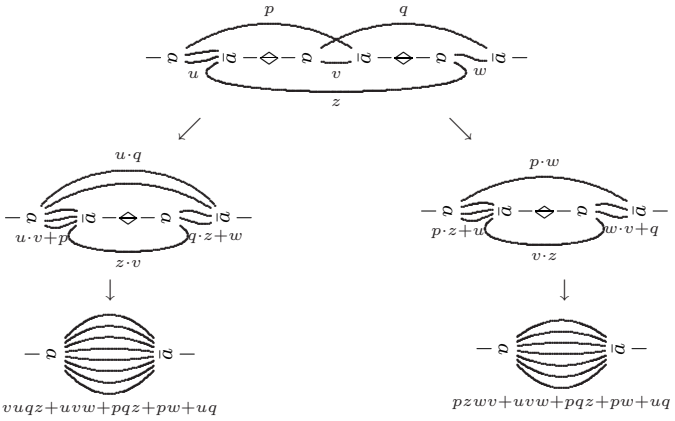
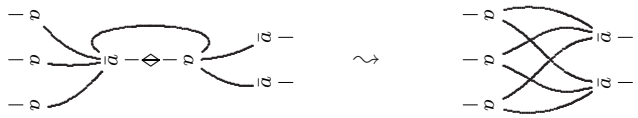


Figure 13: Cut reduction is not confluent

edge disappears with the cut, as in the following example:



(27)

If there is more than one edge between two dual atoms, the number of edges is multiplied, as in the following example:

(28)

This causes an exponential increase of the number of edges during the cut reduction process. In [LS05b] this cut reduction is defined formally by using the semiring structure on the set of natural numbers.

4.4 Theorem *The cut reduction on proof graphs is terminating.*

Proof: Every reduction step reduced the size of the cut sequent Σ in the proof graph. □

Note that the cut reduction is in general not confluent. A simple counterexample is shown in Figure 13, where the labels of the edges indicate how the number of edges between two atoms is computed (for example $u \cdot v + p = 2 \cdot 1 + 1 = 3$). It is interesting to observe that the reasons for this nonconfluence is different from the nonconfluence of cut-elimination in the sequent calculus for classical logic. An instructive example is the one in (21), (22), and (23). This example has been used by Girard in the appendix of [Gir91] to exhibit the inherent nonconfluence of normalization in the sequent calculus (and because of Slogan 1 above he concluded that the same problem must occur with any kind of proof

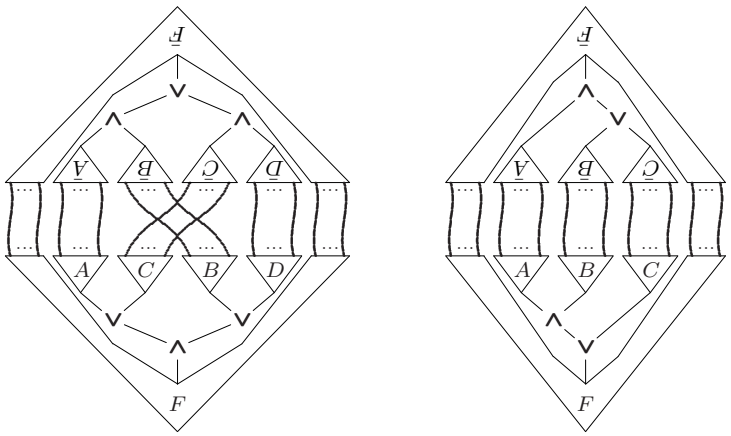


Figure 14: The shape of m-rule graphs and s-rule graphs

nets for classical logic). However, reducing the cut from (21) yields (22) in one step, i.e., there is no room for nonconfluence.

We always have confluence if the number of edges between two atoms is restricted to one, i.e, the multi-graph is just a graph. These objects where called $\mathbb{B}$-*nets* in [LS05b], and a correctness criterion was given for them. However, this criterion is not adequate for the general case where P is a multi-graph.

4.5 Remark There are two very different notions of "cut" and the two should not be mixed up, although both are direct translations of the cut in sequent calculus. On the one side we have the cut as a rule, and cut elimination (Theorem 2.10) means that this rule is admissible, such that the remaining system is *analytic*. For SKS it means that the whole up-fragment of the system (i.e., all rules with the $\uparrow$ in the name) is admissible. On the other side we have the cut $\Diamond\!\!\!\!\!\!\Diamond$, and cut elimination means composition of derivations (or proofs, or arrows in a category), and this will be a central point of interest in the remainder of this paper. It is a particular feature of classical logic that these two notions of cut elimination are so different. For MLL on the other hand, they do coincide.

5 From Derivations to Proof Graphs

In this section we will see how derivations are translated into proof graphs. This is done by assigning to each inference rule a *rule graph*:

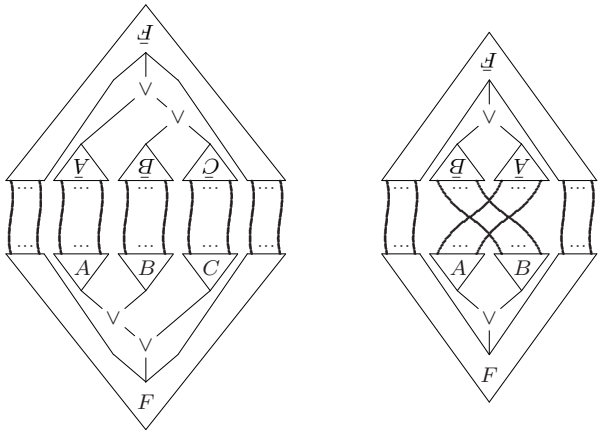


Figure 15: The shape of $\alpha\downarrow$-rule graphs and $\sigma\downarrow$-rule graphs

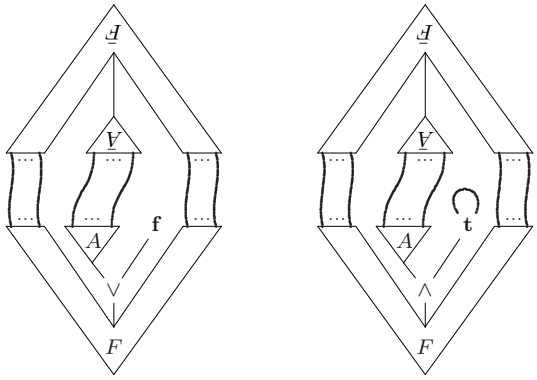


Figure 16: The shape of $\mathbf{f}\downarrow$-rule graphs and $\mathbf{t}\downarrow$-rule graphs

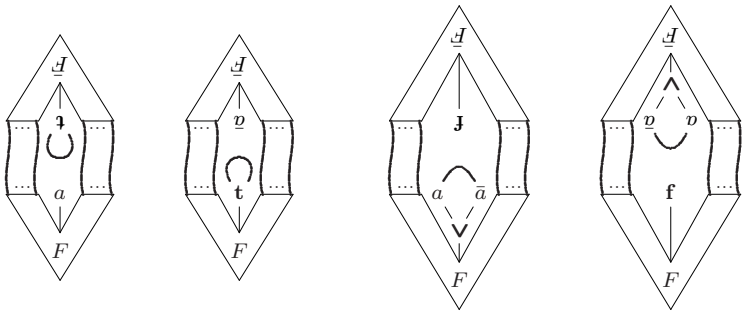


Figure 17: The shape of rule graphs for $\mathsf{aw}\downarrow$, $\mathsf{aw}\uparrow$, $\mathsf{ai}\downarrow$, and $\mathsf{ai}\uparrow$

where the linking is subject to certain side conditions which depend on the rule r.

For the occurrences of $\mathbf{t}$ and $\mathbf{f}$ in the premise and conclusion of r there is no choice: There can never be an edge coming out of an $\mathbf{f}$, and there is always

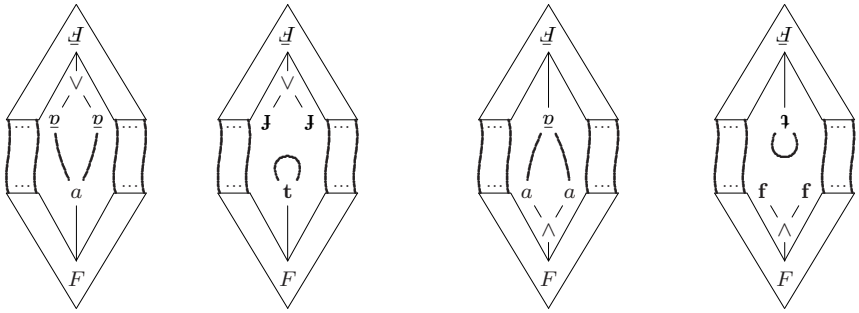


Figure 18: The shape of rule graphs for $\mathsf{ac}{\downarrow}$ and $\mathsf{ac}{\uparrow}$

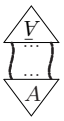
exactly one edge connecting a $\mathbf{t}$ to itself. But we have to explain how to connect propositional variables.

Figures 14, 15, 16, 17 and 18 show the rule graphs for the rules of system SKS, as they are given in Figure 1. For the rules s, m, $\alpha{\downarrow}$, $\alpha{\uparrow}$, $\sigma{\downarrow}$, and $\sigma{\uparrow}$, it is intuitively clear what should happen: every leaf in the premise tree is connected to its counterpart in the conclusion via an edge in in the linking; and there are no other edges. Note that the proof graphs for $\alpha{\downarrow}$ and $\alpha{\uparrow}$ are the same; one written as the upside-down version of the other. The same holds for all other pairs of dual rules. For $\alpha{\downarrow}$, $\sigma{\downarrow}$, $\mathbf{f}{\downarrow}$, and $\mathbf{t}{\downarrow}$ only one picture is shown (Figures 15 and 16), but for the atomic rules down- and up-version are given (Figures 17 and 18) because it is instructive to see them next to each other. In Figure 18 we show the proof graphs for $\mathsf{ac}{\downarrow}$ applied to a propositional variable and to $\mathbf{t}$, and dually for $\mathsf{ac}{\uparrow}$.

We can use cuts to plug rule graphs together to get *derivation graphs*, as it is shown in the upper left of Figure 19. Note that in derivation graphs the "duality" between derivations

$$\dfrac{\dfrac{A}{\Delta}\,\mathsf{s}}{B} \qquad \text{and} \qquad \dfrac{\dfrac{\bar{B}}{\bar{\Delta}}\,\mathsf{s}}{\bar{A}}$$

disappears because both are represented by *the same* graph. A derivation which contains no rules, i.e., premise and conclusion coincide, is represented by the *identity proof graph*:



(29)

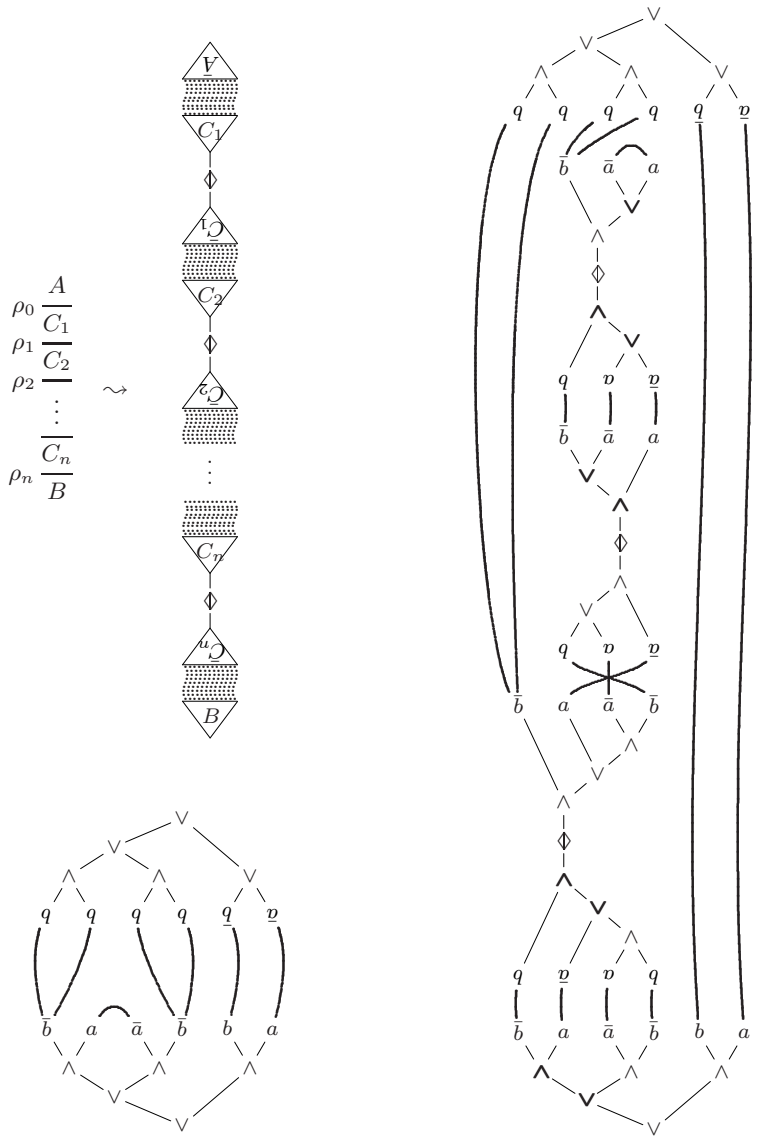


Figure 19: **Upper left:** From derivations to derivation nets. **Right:** Example of an A-reduced proof graph. **Lower left:** Result of applying level-C cut elimination to it.

6 Cut Elimination is Losing Information

In this section we will see how cut elimination removes information, and that this can be bureaucratic as well as non-bureaucratic information.

We introduce three levels of cut reduction, that are called here *level-A reduction*, *level-C reduction*, and *full reduction*. The goal is to establish a connection between level-A cut reduction and Formalism A, and between level-C cut reduction and Formalism C. With full cut reduction we mean the procedure defined in Section 4. For defining level-A reduction, we need the distinction between *heavy* and *light* connectives in the proof graph: An instance of a binary connective is called *heavy* if it is active in a medial, switch, or interaction rule. The heavy instances of connectives are in boldface in Figures 14 and 17. All other instances of connectives are called *light*, i.e., all connectives appearing in the contexts and in rules for associativity, commutativity, contraction, and units (Figures 15, 16 and 18).

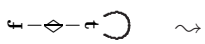
6.1 Level-A Cut Reduction For *level-A cut reduction* we allow the following four reductions:

1. Cuts on binary connectives

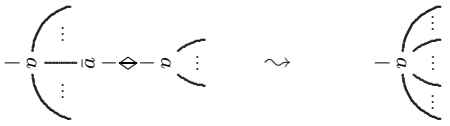
(30)

   can only be reduced if both connectives are light. This means that the following configurations *cannot* be reduced:

2. We allow all reductions between **f** and **t**:



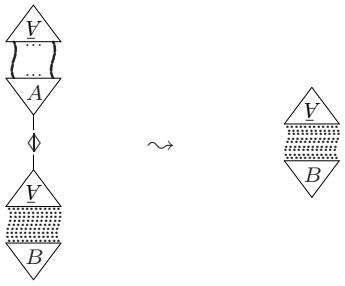
3. An atomic cut can only be reduced if at least one of the two cut-atoms has exactly one adjacent egde in the linking by which it is connected to another atom (its dual). I.e, we have:



(31)

4. Finally, we allow to eliminate the identity proof graph in a situation as

follows:



(32)

We will call a proof graph *A-reduced* if no further level-A cut reduction step is possible.

To give an example, consider the proof graph on the right of Figure 19, which represents the two derivations in Figure 4. It is easy to see that this proof graph cannot be obtained from the right derivation in Figure 5. This means that A-reduced proof graphs are not able to identify the derivations that are identified by Formalism B.

6.2 Remark The reduction (32) in 6.1 is needed because Formalism A requires the identification of the two proof graphs in (32), but the heavy connectives can block ordinary cut reduction (30).

6.3 Theorem *Level-A reduction is terminating and confluent.*

Proof: Termination of level-A cut reduction follows for the same reason as in the proof of Theorem 4.4. Thus, for confluence, it suffices to show local confluence. For this, note that there are 2 critical pairs. The first involves two atomic cuts and is the reason for nonconfluence in the general case (see Figure 13). But due to the restriction (31), this can be resolved immediately, as indicated in Figure 20. The second critical pair does not occur in the general case because it involves a reduction according to (32) against a reduction according to (30). But this can also easily be resolved because the decomposition of an identity proof graph yields two smaller identity proof graphs. □

6.4 Theorem *Two* SKS*-derivations yield the same A-reduced proof graph if and only if they are identified by Formalism A.*

Proof: First note that all equations that generate the identifications of Formalism A, as given in Section 3, yield identical A-reduced proof graphs for the left-hand side and for the right-hand side. Thus, the if-direction of the statement follows from confluence of level-A reduction.

For the only-if-direction we will construct an SKS-derivation from an A-reduced proof graph and show that it is unique up to Formalism A equivalence.

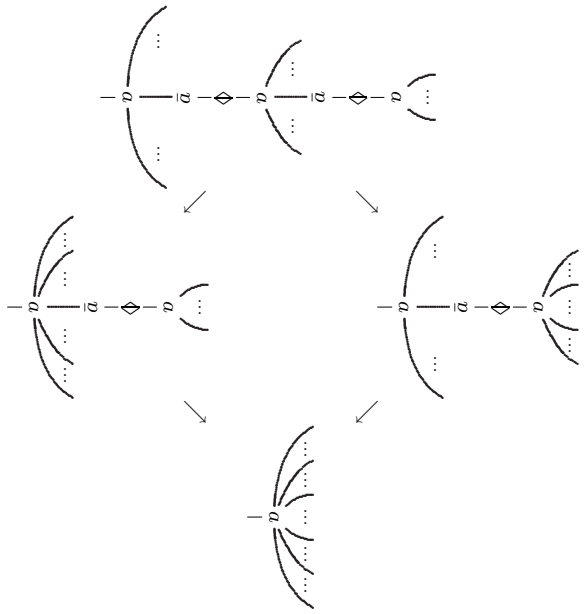
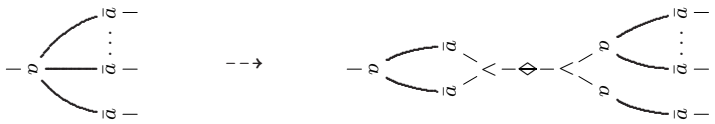


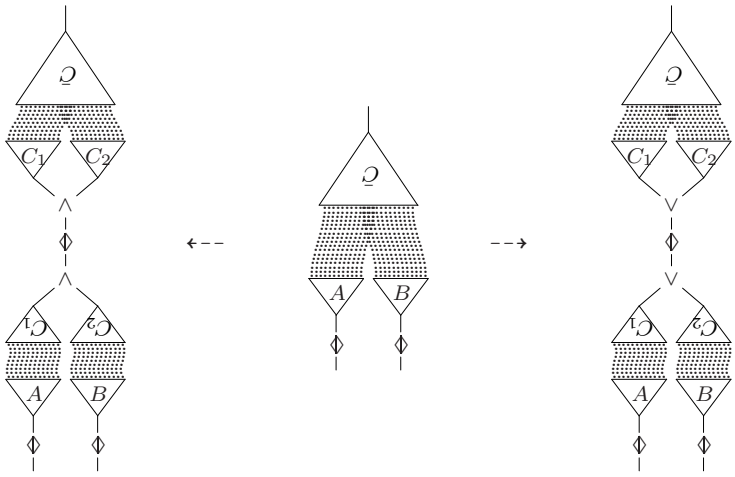
Figure 20: Critical pair for level-A cut reduction

We proceed in two steps. First we "unfold" the proof graph by introducing cuts, and then we extract the SKS-derivation. We begin by "unfolding" the contractions, i.e., whenever an atom is incident to more than one edge, we add new cuts and ac↓-rule graphs (or ac↑-rule graphs, see Figure 18) as follows:



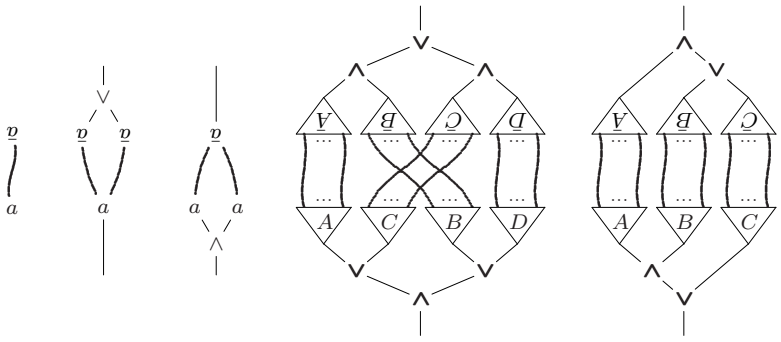
Because of the equations for ∨-monoids (see Figure 11) and ∧-comonoids, the order in which we proceed does not matter. We also need to unfold the following

situation:



where the proof graph in the middle is expanded to either the proof graph on the left or the one on the right, depending on the root connective of C. The formula tree C_1 (resp. C_2) is obtained from C by removing all subtrees that are not connected to a leaf in A (resp. B).

Now we can proceed by induction on the size of the proof graph to construct a corresponding derivation. The base cases are:



with the corresponding derivations:

$$a \qquad \mathsf{ac}{\downarrow}\,\frac{a \vee a}{a} \qquad \mathsf{ac}{\uparrow}\,\frac{a}{a \wedge a} \qquad \mathsf{m}\,\frac{(A \wedge B) \vee (C \wedge D)}{[A \vee C] \wedge [B \vee D]} \qquad \mathsf{s}\,\frac{A \wedge [B \vee C]}{(A \wedge B) \vee C}$$

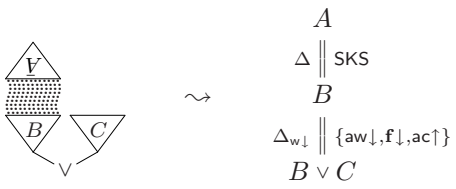
We have the following inductive cases:

1. If the proof graph is the composition of two proof graphs via a cut, we can simply compose the two derivations that we have by induction hypothesis.

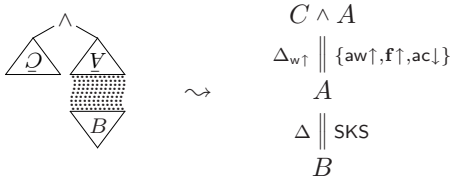
2. If the proof graph is a composition of smaller proof graphs via (light) connectives $\wedge$, $\vee$, and via the units $\mathbf{f}$ and $\mathbf{t}$ (with a loop attached to it, see Figure 16), then we can construct our derivation by composing the smaller

derivations (which exist by induction hypothesis) via the rules $\alpha\downarrow$, $\alpha\uparrow$, $\sigma\downarrow$, $\sigma\uparrow$, $\mathbf{f}\downarrow$, $\mathbf{f}\uparrow$, $\mathbf{t}\downarrow$, and $\mathbf{t}\uparrow$. This is unique up to Formalism A equivalence because of Theorem 3.1.

3. If the proof graph is obtained from a smaller proof graph by attaching a formula tree without incident edges in the linking (except for edges connecting $\mathbf{t}$ to itself), we obtain our derivation as follows

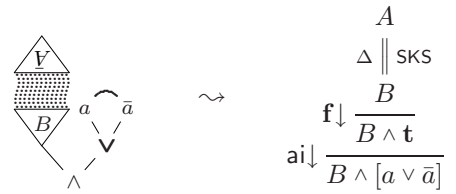


or dually



where Δ exists by induction hypothesis. The uniqueness of $\Delta_{\mathsf{w}\downarrow}$ (respectively, $\Delta_{\mathsf{w}\uparrow}$) is ensured by the comonoid laws for $\mathsf{ac}\uparrow$ (respectively, the monoid laws for $\mathsf{ac}\downarrow$).

4. Finally, if the proof graph is obtained from a smaller proof graph by attaching a pair of linked atoms, the situation is similar:

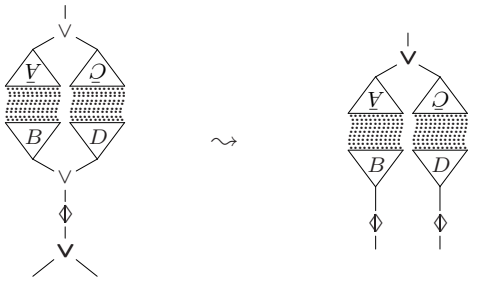


(We do not show the dual case.) Note that for uniqueness in the last two cases, we make crucial use of the equations for the units in Figures 9 and 10.

If none of the cases above applies, then the proof graph cannot be the result of translating an SKS-derivation into a proof graph. □

Note that this proof gives a polynomial algorithm for deciding whether an A-reduced proof graph is obtained from an SKS derivation.

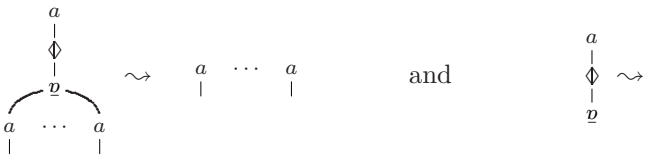
6.5 Remark Unfortunately, there is no local rule replacing (32) for reducing the identity proof graph. The rule



(33)

would reduce too much. In fact, one can consider it as level B-reduction. But unfortunately, with (33) we would lose confluence again. This is the reason why there is no canonical representant for a "B-reduced" proof graph (see also [BL05]).

6.6 Level-C Cut Reduction For *level-C cut elimination* we allow the reduction of all cuts on binary connectives, i.e., we forget the distinction between heavy and light instances of connectives, and thus there is no more need for (32). The restriction on atomic cuts is slightly relaxed. We additionally allow the reductions:



(34)

A proof graph is called *C-reduced* if no further cut reduction steps according to these restrictions are possible.

6.7 Theorem *Level-C reduction is terminating and confluent.*

Proof: Termination follows for the same reason as before. For showing confluence, it suffices to note that Figure 20 shows the only critical pair. □

Examples for C-reduced proof graphs are the two proof graphs in (23). As the reader might verify, they are obtained from the two derivations in Figure 2, which is the reason why these two SKS-derivations are explicitly given. Another example is in the lower left of Figure 19 (which is obtained from the proof graph on the right of that figure).

Finally, we speak of *full cut elimination*, when we have cut reduction without any restrictions. A proof graph is *fully reduced* if it contains no cuts. Hence, the fully reduced proof graphs are exactly the cut-free $\mathbb{N}$-prenets of [LS05b]. For example, the right proof graph in (23) can be obtained by reducing the cut from the left one. This means that fully reduced proof graphs would identify the two derivations in Figure 2. We can safely assume that this goes beyond mere

"bureaucracy elimination". With full cut elimination we not only eliminate "essential" information, we also leave the realm of confluence, as explained in Section 4.

Let us finish this section by stating two open problems related to C-reduced proof graphs. First, we can now make precise we mean by *full coherence* in Figure 7:

6.8 Open Problem *Find a (possibly finite) set of equations between derivations for defining Formalism C such that two derivations are identified if and only if they yield the same C-reduced proof graph.*

In other words, we are looking for a minimal set of equations to be enforced on derivations such that the equivalence classes are in bijection with the C-reduced proof graphs. These would be the axioms that freely generate the category of C-reduced proof graphs. The work in [Str07b, Lam07] gives a list of equations which must hold but which do not suffice.

The second open problem is the question whether the C-reduced proof *nets* form a proof system:

6.9 Open Problem *Let π be a given C-reduced proof graph. Can we decide in polynomial time (polynomial in the size of π) whether π is the translation of an* SKS*-derivation?*

In other words, we are asking for a polynomial correctness criterion that would distinguish the C-reduced proof nets from the "nonproof graphs". It is easy to see that Problem 6.9 can be reduced to Problem 6.4 of [Str07a] which concerns only the switch and medial rules. This simplification is possible because of the decomposition theorems in [Brü03b].

Note that the proof of Theorem 6.4 gives a positive answer to Problem 6.9 for A-reduced proof graphs. Thus, A-reduced proof graphs can be called a *proof system* in the sense of Cook and Reckhow [CR79]. Furthermore, note that for fully reduced proof graphs there is no such polynomial decision procedure (unless P=NP) because the problem can be reduced to the tautology problem, which is CoNP-complete (see also [LS05b]). This is another indication that full cut reduction removes too much information from the proof.

A positive answer to the question of Problem 6.9 could also have some impact in the area of proof complexity. One possible way to tackle the problem is to study combinatorial properties of derivations consisting of the rules s and m, as it has been done in [Str07a].

7 Relation to Logical Flow Graphs and Atomic Flows

We discussed the fully reduced proof graphs here because they are essentially the same as Buss' *logical flow graphs* [Bus91, Car97]. The notion of logical flow graph has recently been improved to *atomic flow* by Guglielmi and Gundersen

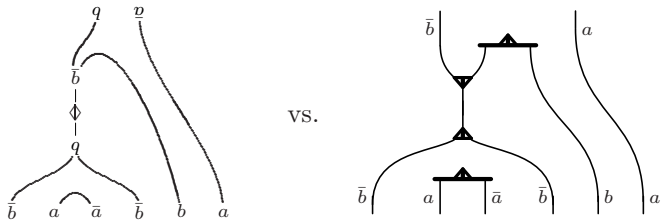
in [GG08] for studying streamlining of derivations. Atomic flows carry more information with respect to the contraction of atoms than logical flow graphs.

To be precise, atomic flows contain almost exactly the same information as C-reduced proof graphs. There are only two differences: First atomic flows *do not* keep the information about premise and conclusion of the derivation, and second, they *do* keep the information about the order of the applications of contraction. We can simulate this with our proof graphs by introducing another form of restricted cut reduction. Let us define *level-G reduction* by allowing the reduction of all $\wedge$-$\vee$-cuts as in (24) and all cuts on units as in (25) and atomic cuts only in the situation

(35)

provided that the disappearing linking edge in the middle is not a bend edge coming from a $\mathsf{ai}{\downarrow}$-rule or $\mathsf{ai}{\uparrow}$-rule (see Figure 17). Let us call a proof graph *G-reduced* if no further reductions according to these restrictions are possible. Note that in G-reduced proof graphs it can never happen that more than two linking edges are coming out of one atom.

We can now define an *atomic flow* to be a G-reduced proof graph in which premise and conclusion are not shown. The notation in [GG08] is mildly different from the one presented here. Here we show the example on the left in (23) written in both ways:



In other words, atomic flows are a special kind of proof graphs with a slightly poorer structure because they do not keep the information about what has been proved. However, the chosen level of abstraction in [GG08] allows to give a simple syntactic proof of *streamlining*, which is an up-down symmetric generalization of Theorem 2.10, independently of the proof system.

We can formulate the Open Problem 6.9 also for atomic flows: Can we decide in polynomial time whether for a given flow graph, a given premise, and a given conclusion there is a matching derivation?[5]

8 Some remarks on the induced categories

We can define the following categories: Let **PreA** (resp. **PreC**, **PreG**) be the category whose objects are the formulas and whose arrows $A \to B$ between formulas A and B are the A-reduced (resp. C-reduced, G-reduced) proof

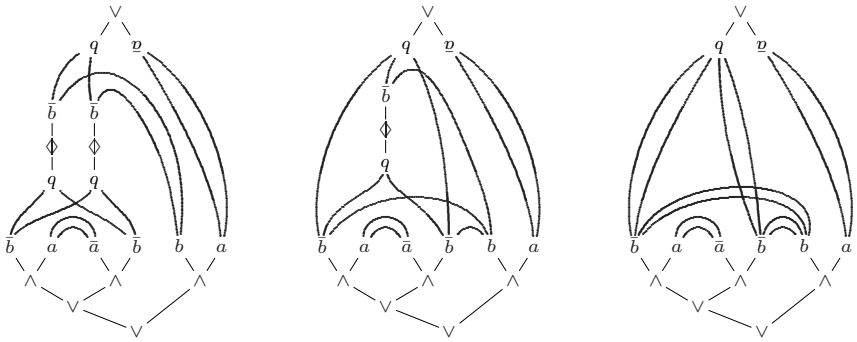
[5] In [GG08] it has only been shown that for any atomic flow there is such a derivation if premise and conclusion are *not* fixed.

graphs with $\Gamma = \bar{A}, B$. The categories **DeriA**, **DeriC**, and **DeriG** are the wide subcategories of **PreA**, **PreC**, and **PreG**, respectively, in which the Hom-sets contain only those proof graphs that are obtained from an SKS-derivation as described above. The fully reduced proof graphs do not form a category, because composition cannot be defined via cut elimination which it is not confluent.

The categories **DeriA**, **DeriC**, and **DeriG** can be called "Boolean" in the sense of [LS05a]. The forgetful functor from the category of (small) categories to the category of posets maps them to Boolean algebras. For the categories **PreA**, **PreC**, and **PreG** this is not the case because they contain morphisms that do not correspond to implications in Boolean logic.

Theorem 6.4 implies that **DeriA** is the free category generated by the set $\mathcal{A} \cup \bar{\mathcal{A}} \cup \{\mathbf{f}, \mathbf{t}\}$, the bifunctors $- \vee -$ and $- \wedge -$, and the basic morphisms induced by the rules in SKS, obeying all the equations demanded by Formalism A in Section 3.

It is out of scope of this paper to give a full characterization of the other categories mentioned above. In fact, for **DeriC** (and also for **DeriG**) this is an open problem, as indicated in Problem 6.8. Nonetheless, we can compare our proof graph categories to the different axiomatisations given in [FP04], [DP04], [LS05a], and [Str07b]. All of them have in common that the Hom-sets are equipped with a semigroup structure, which is idempotent in [FP04], [DP04], and [LS05a]. This semigroup structure is also present for **PreC** and **DeriC**, but it is *not* idempotent (see [Str07b]). In the case of **PreC** the sum of two proof graphs is given by their union. This is best understood by seeing an example. Let f be the proof graph on the left in (23) and g the one on the right. Then we can form $f + f$, $f + g$, and $g + g$ as follows:



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

$$\mathsf{c}{\downarrow}\,\frac{F\{A \vee A\}}{F\{A\}} \qquad \text{and} \qquad \mathsf{c}{\uparrow}\,\frac{F\{A\}}{F\{A \wedge A\}} \quad ,$$

which are both derivable in SKS (see Proposition 2.4), and the rule

$$\mathsf{mix}\,\frac{F\{A \wedge B\}}{F\{A \vee B\}} \quad ,$$

which is also derivable in SKS, for example via

$$\dfrac{\dfrac{\dfrac{\dfrac{F\{A \wedge B\}}{F\{[A \vee \mathbf{f}] \wedge B\}}\,\mathbf{f}\downarrow}{F\{[A \vee \mathbf{t}] \wedge B\}}\,\mathsf{aw}\downarrow}{F\{A \vee (\mathbf{t} \wedge B)\}}\,\mathsf{s}}{F\{A \vee B\}}\,\mathbf{f}\uparrow \quad .$$

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

$$\dfrac{\dfrac{\dfrac{\dfrac{A}{A \wedge A}\,\mathsf{c}\uparrow}{B \wedge B}\,{\scriptstyle \Delta_1 \wedge \Delta_2 \,\Vert\, \mathsf{SKS}}}{B \vee B}\,\mathsf{mix}}{B}\,\mathsf{c}\downarrow \quad ,$$

where Δ_1, Δ_2 is some "merge" of Δ_1 and Δ_2; compare (16). Note, that this sum of derivations could also be obtained in a different way, for example by first mixing $A \wedge A$ and then taking $\Delta_1 \vee \Delta_2$, or by using a different derivation for mix. But no matter which one we choose, the translation into a C-reduced proof graph yields the same result for all of them; and we obtain the same result if we first translate the derivations Δ_1 and Δ_2 into C-reduced proof graphs, and then taking their sum as proof graphs (as described above). Hence, the semigroup structure on the Hom-sets is the same for **PreC** and **DeriC**. Note that **PreG** and **DeriG** do not carry this semigroup structure because the addition is not associative. The reason is that in G-reduced proof graphs the associativity equations for $\vee$-monoids and $\wedge$-comonoids (see Figure 11) do not hold.

Furthermore, we can equip the Hom-sets of our categories with a partial order, defined by cut elimination: We say $f \leq g$ if g is obtained from f by eliminating some of the remaining cuts[6], as it is the case in our example above for f and g. Then we also have $f + f \leq f + g \leq g + g$. The important observation about the semigroup structure and this partial order structure is, that *they are independent*. Although this seems to be natural from the viewpoint of our proof graphs, it is not the case in the *classical categories* of [FP04] (called LK-*categories* in [Str07b]) which are based on the proof nets in [Rob03] and the sequent calculus LK [Gen34]. In an LK-category the sum-of-proofs-semigroup structure and the cut-elimination-partial-order structure on the Hom-sets determine each other uniquely via $f \leq g$ iff $f + g = g$. In [DP04] and [LS05a] there is also a partial order structure on the Hom-sets, simply because the semigroup structure is idempotent. But this partial order structure has nothing to do with cut elimination, simply because everything is *a priory* cut-free.

The category **PreC** follows the axiomatisation given in [LS05a, Str07b]: it is *-autonomous, it has monoids and comonoids, and it is "graphical". But it

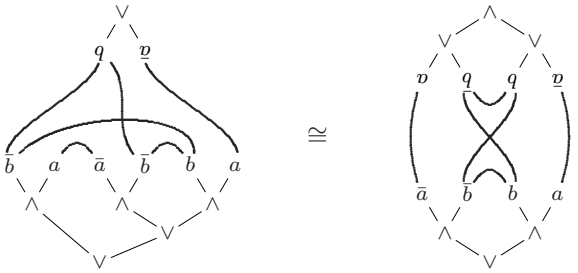
[6] Even if this process is not confluent, we stay in the realm of C-reduced proof graphs.

does not obey the equation

$$\begin{array}{ccc} A \vee A & \xrightarrow{\Delta_{A\vee A}} & (A \vee A) \wedge (A \vee A) \\ {\scriptstyle \nabla_A}\downarrow & & \downarrow{\scriptstyle \nabla_A \wedge \nabla_A} \\ A & \xrightarrow[\Delta_A]{} & A \wedge A \end{array} \tag{36}$$

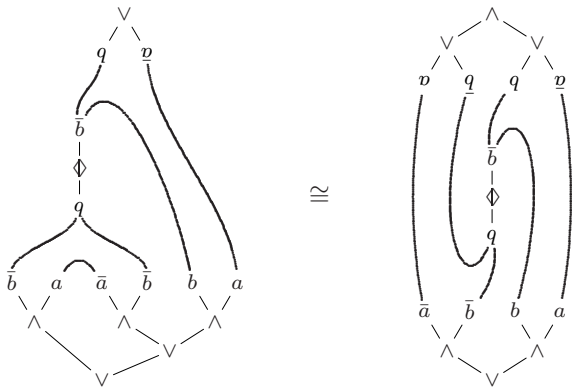
However, the two maps $A \vee A \to A \wedge A$ in (36) are ordered according to the cut-elimination-partial-order defined above, as it is the case in [FP04].

For the category **DeriC** the situation is similar. But **DeriC** is not *-autonomous, because we do in the general case not have a natural bijection between the proofs from $A \wedge B$ to C and the proofs from B to $A \supset C$. To see this consider again the example on the right in (23), which corresponds to the derivation on the right in Figure 2. By applying the rules for associativity and commutativity, we can provide a derivation that corresponds to the proof graph on the left below.



(37)

The proof graph on the right above is the corresponding proof graph provided by the *-autonomous structure of **PreC**. But there is no SKS-derivation that correspond to this proof graph. On the other hand, if we reintroduce the cut as in the example on the left in (23), we can do the same transformation as above, without leaving the category **DeriC**:



(38)

Both proof graphs in (38) are direct translations of SKS-derivations, as the reader can easily verify. This raises the question whether we can find a deep in-

ference proof system S such that its C-reduced proof graphs form a *-autonomous subcategory of **PreC**.

Acknowledgments

I thank Paola Bruscoli an Kai Brünnler for helpful comments improving the readability of this paper.

References

[BG09] Paola Bruscoli and Alessio Guglielmi. On the proof complexity of deep inference. *ACM Transactions on Computational Logic*, 10(2):1–34, 2009. Article 14.

[BL05] Kai Brünnler and Stéphane Lengrand. On two forms of bureaucracy in derivations. In Paola Bruscoli and François Lamarche, editors, *Structures and Deduction 2005 (Satellite Workshop of ICALP'05)*, 2005.

[Brü03a] Kai Brünnler. Atomic cut elimination for classical logic. In M. Baaz and J. A. Makowsky, editors, *CSL 2003*, volume 2803 of *Lecture Notes in Computer Science*, pages 86–97. Springer-Verlag, 2003.

[Brü03b] Kai Brünnler. *Deep Inference and Symmetry for Classical Proofs*. PhD thesis, Technische Universität Dresden, 2003.

[Brü03c] Kai Brünnler. Minimal logic in the calculus of structures. note, 2003.

[Brü06a] Kai Brünnler. Deep sequent systems for modal logic. In Guido Governatori, Ian Hodkinson, and Yde Venema, editors, *Advances in Modal Logic*, volume 6, pages 107–119. College Publications, 2006.

[Brü06b] Kai Brünnler. Locality for classical logic. *Notre Dame Journal of Formal Logic*, 47(4):557–580, 2006.

[BS09] Kai Brünnler and Lutz Straßburger. Modular sequent systems for modal logic. In *Proceedings of Tableaux'09*, 2009.

[BT01] Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2250 of *LNAI*, pages 347–361. Springer, 2001.

[Bus91] Samuel R. Buss. The undecidability of k-provability. *Annals of Pure and Applied Logic*, 53:72–102, 1991.

[BW99] Michael Barr and Charles Wells. *Category Theory for Computing Science*. Les Publications CRM, Montréal, third edition, 1999.

[Car97] Alessandra Carbone. Interpolants, cut elimination and flow graphs for the propositional calculus. *Annals of Pure and Applied Logic*, 83:249–299, 1997.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[DG04] Pietro Di Gianantonio. Structures for multiplicative cyclic linear logic: Deepness vs cyclicity. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, CSL 2004*, volume 3210 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2004.

[DP04] Kosta Došen and Zoran Petrić. *Proof-Theoretical Coherence*. KCL Publications, London, 2004.

[FP04] Carsten Führmann and David Pym. Order-enriched categorical models of the classical sequent calculus. To appear in *Journal of Pure and Applied Algebra*, 2004.

[Gen34] Gerhard Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39:176–210, 1934.

[GG08] Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1:9):1–36, 2008.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir91] Jean-Yves Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1:255–296, 1991.

[Gir96] Jean-Yves Girard. Proof-nets : the parallel syntax for proof-theory. In Aldo Ursini and Paolo Agliano, editors, *Logic and Algebra*. Marcel Dekker, New York, 1996.

[GS02] Alessio Guglielmi and Lutz Straßburger. A non-commutative extension of MELL. In Matthias Baaz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2002*, volume 2514 of *LNAI*, pages 231–246. Springer-Verlag, 2002.

[Gug04a] Alessio Guglielmi. Formalism A. note, April 2004.

[Gug04b] Alessio Guglielmi. Formalism B. note, December 2004.

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

[Gui06] Yves Guiraud. The three dimensions of proofs. *Annals of Pure and Applied Logic*, 141(1-2):266–295, 2006.

[Hug04] Dominic Hughes. Deep inderence proof theory equals categorical proof theory minus coherence. preprint, 2004.

[Hug05] Dominic Hughes. Simple multiplicative proof nets with units. Preprint, 2005.

[HvG03] Dominic Hughes and Rob van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 1–10, 2003.

[Kel64] Gregory Maxwell Kelly. On MacLane's conditions for coherence of natural associativities, commutativities, etc. *Journal of Algebra*, 4:397–402, 1964.

[KM71] Gregory Maxwell Kelly and Saunders Mac Lane. Coherence in closed categories. *Journal of Pure and Applied Algebra*, 1:97–140, 1971.

[Lam07] François Lamarche. Exploring the gap between linear and classical logic. *Theory and Applications of Categories*, 18(18):473–535, 2007.

[LS05a] François Lamarche and Lutz Straßburger. Constructing free Boolean categories. In *Proceedings of the Twentieth Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 209–218, 2005.

[LS05b] François Lamarche and Lutz Straßburger. Naming proofs in classical propositional logic. In Paweł Urzyczyn, editor, *Typed Lambda Calculi and Applications, TLCA 2005*, volume 3461 of *LNCS*, pages 246–261. Springer, 2005.

[Mac63] Saunders Mac Lane. Natural associativity and commutativity. *Rice University Studies*, 49:28–46, 1963.

[Mac71] Saunders Mac Lane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer, 1971.

[McK05] Richard McKinley. Classical categories and deep inference. In *Structures and Deduction 2005 (Satellite Workshop of ICALP'05)*, 2005.

[Rob03] Edmund P. Robinson. Proof nets for classical logic. *Journal of Logic and Computation*, 13:777–797, 2003.

[SL04] Lutz Straßburger and François Lamarche. On proof nets for multiplicative linear logic with units. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, CSL 2004*, volume 3210 of *LNCS*, pages 145–159. Springer-Verlag, 2004.

[SS05] Charles Stewart and Phiniki Stouppa. A systematic proof theory for several modal logics. In R. A. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic, Volume 5*, pages 309–333. King's College Publications, 2005.

[Sto07] Phiniki Stouppa. A deep inference system for the modal logic S5. *Studia Logica*, 85(2):199–214, 2007.

[Str02] Lutz Straßburger. A local system for linear logic. In Matthias Baaz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2002*, volume 2514 of *LNAI*, pages 388–402. Springer-Verlag, 2002.

[Str03] Lutz Straßburger. *Linear Logic and Noncommutativity in the Calculus of Structures*. PhD thesis, Technische Universität Dresden, 2003.

[Str05] Lutz Straßburger. From deep inference to proof nets. In Paola Bruscoli and François Lamarche, editors, *Structures and Deduction 2005 (Satellite Workshop of ICALP'05)*, 2005.

[Str07a] Lutz Straßburger. A characterisation of medial as rewriting rule. In Franz Baader, editor, *Term Rewriting and Applications, RTA'07*, volume 4533 of *LNCS*, pages 344–358. Springer, 2007.

[Str07b] Lutz Straßburger. On the axiomatisation of Boolean categories with and without medial. *Theory and Applications of Categories*, 18(18):536–601, 2007.

[Tiu06] Alwen Tiu. A local system for intuitionistic logic. In M. Hermann and A. Voronkov, editors, *LPAR 2006*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 242–256. Springer-Verlag, 2006.