Breaking Paths in Atomic Flows for Classical Logic

Alessio Guglielmi
INRIA Nancy – Grand Est, LORIA and University of Bath
A.Guglielmi AT bath.ac.uk

Tom Gundersen
INRIA Saclay – Île-de-France and École Polytechnique, LIX
teg AT jklm.no

Lutz Straßburger
INRIA Saclay – Île-de-France and École Polytechnique, LIX
lutz AT lix.polytechnique.fr

Abstract

This work belongs to a wider effort aimed at eliminating syntactic bureaucracy from proof systems. In this paper, we present a novel cut elimination procedure for classical propositional logic. It is based on the recently introduced atomic flows: *they are purely graphical devices that abstract away from much of the typical bureaucracy of proofs. We make crucial use of the* path breaker, *an atomic-flow construction that avoids some nasty termination problems, and that can be used in any proof system with sufficient symmetry. This paper contains an original 2-dimensional-diagram exposition of atomic flows, which helps us to connect atomic flows with other known formalisms.*

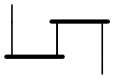
1 Introduction

The investigation of the cut elimination property of logical systems is a central topic in current proof theory, and, as pointed out by Girard [9], the *lack of modularity* is one of its main technical limitations. More precisely, the argument for showing cut elimination is usually based on heavy syntactic arguments and a tedious case analysis depending on the shape of the inference rules. A slight change in design makes the whole proof break down, and if one wishes to add some rules, one usually has to redo the whole cut elimination proof from scratch.

Our work in this paper suggests that the source of this "lack of modularity" might not be in the nature of the cut elimination property, but in the method that is used for proving it. We present here a cut elimination procedure for classical propositional logic that is independent from the shape of the logical rules. It is not based on the permutation of inference rules but on the manipulation of *atomic flows* [10].

Atomic flows capture the structural skeleton of a proof and ignore the logical content. Due to their "graphical nature", atomic flows can be seen as relatives of Girard's *proof nets* [8, 6] and Buss' *logical flow graphs* [5]. Proof nets have originally been proposed only for linear logic, but there have been various proposals for proof nets for classical logic with different purpose and design, e.g., by Laurent [17], by Robinson [19] and by Lamarche and Straßburger [16]. Logical flow graphs have only been defined for classical logic, but their definition for linear logic would be literally the same. In fact, for the multiplicative fragment of linear logic (MLL) the two notions essentially coincide. This, however, is no longer the case for classical logic, which can be obtained from MLL by adding the rules for contraction and weakening. Atomic flows can be seen as a development that takes the best out of both worlds. Like proof nets they simplify proof normalization because they avoid unnecessary bureaucracy due to trivial rule permutations, and like logical flow graphs they precisely capture the information flow inside the proof. In this respect, they are very similar to the variant of proof nets discussed in [23]. Since atomic flows contain for each atom occurrence every contraction and weakening that is applied to it, they can be used for controlling the size of proofs, and thus can also play a role in proof complexity (see [3]).

Atomic flows are also very similar to *string diagrams* for representing morphisms in monoidal categories (see [20] for a survey). However, in (classical) logic one usually finds two dual monoidal structures and not just one. Thus, atomic flows are, in spirit, more closely related to *coherence graphs* in monoidal closed categories [13]. Nonetheless, it should be stressed that atomic flows do not form a monoidal closed category. The following two flows are *not* the same

 and (1)

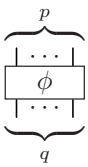
although, during the normalization process, we wish to reduce the atomic flow on the left (a cut connected to an identity) to the atomic flow on the right (a single edge). In linear logic one can simply "pull the edges" and directly reduce the left atomic flow in (1) to the right one, whereas in classical logic this step might involve duplication of large parts of the proof. The main insight coming from atomic flows is that this duplication and the whole normalization process is independent from the logical content of the proof and independent from the design of the logical rules in use, as is discussed in [10].

This paper is structured into three parts:

- In the first part (Sections 2 and 3) we introduce atomic flows. For doing so, we use the language of two-dimensional diagrams [15], originally due to Penrose (see also [11]). The atomic flows as they are defined here are different from those defined in [10], but the differences can be easily reconciled. We chose this slightly different definition for the sake of convenience when composing and manipulating atomic flows independently from their associated derivations.
- In the second part (Sections 4 and 5) we first define local transformations on atomic flows that are similar to the reduction steps in linear logic proof nets or in interaction nets [14], and that have the goal to normalize the proof. However, due to the presence of contractions, the atomic flows can contain cycles that prevent these local reductions from terminating. To solve this problem, we define a global transformation on the atomic flows, called the *path breaker*, that treats the proof as a black box; it simply duplicates the whole proof and combines the copies. Note that this is conceptually different from the cut elimination in standard proof nets [8, 17, 19], where cut reduction steps are mixed local/global: A single step involves a local cut reduction and some duplication of a part of a proof (a box or an empire). In our case the procedure consists of two phases, a purely global one followed by a purely local one. However it remains an important research objective to investigate the computational meaning of these reductions.
- In the third part (Sections 6 and 7) we show how formal proofs in a deductive system are mapped to atomic flows, and how the operations on atomic flows that we defined before can be lifted to the deductive system, and thus can be used to provide a cut elimination procedure. This can be done because the symmetry of the deductive system we use allows to reduce the cut to its atomic form, in the same as it is done for the identity rule in traditional systems.

2 Atomic Flows

We start from a countable set $\mathcal{A}$ of *atomic types*, equipped with an involutive bijection $\bar{(\cdot)} : \mathcal{A} \to \mathcal{A}$, such that for all $a \in \mathcal{A}$, we have $\bar{a} \neq a$ and $\bar{\bar{a}} = a$. A *(flow) type* is a finite list of atomic types, denoted by $p, q, r, \ldots$, and we write $p \mid q$ for the list concatenation of p and q, and we write 0 for the empty list. An *atomic flow* $\phi : p \to q$ is a two-dimensional diagram [15], written as



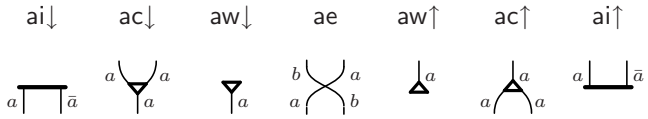
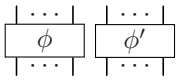


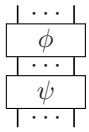
Figure 1: Generators for atomic flows

where p is the *input type* and q is the *output type*. The number of edges corresponds to the lengths of the lists, and each edge is labelled by the corresponding list element. For each type q, we have the *identity flow* id_q:

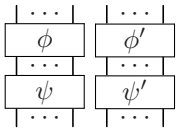
We can compose atomic flows horizontally: for $\phi : p \to q$ and $\phi' : p' \to q'$, we get $\phi \mid \phi' : p \mid p' \to q \mid q'$ of the shape



And we can compose atomic flows vertically: for $\phi : p \to q$ and $\psi : q \to r$, we get $\psi \circ \phi : p \to r$ of the shape



For $\phi : p \to q$ we have $\phi \circ \mathsf{id}_p = \phi = \mathsf{id}_q \circ \phi$ and $\phi \mid \mathsf{id}_0 = \phi = \mathsf{id}_0 \mid \phi$. We also have $(\psi \circ \phi) \mid (\psi' \circ \phi') = (\psi \mid \psi') \circ (\phi \mid \phi')$ which is pictured as



Finally, we have to give a set of generators and relations, which is done in Figures 1 and 2. It is easy to see that atomic flows form a (strict) monoidal category, that we denote by $\mathbf{AF}$.

The generators in Figure 1 are called $\mathsf{ai}{\downarrow}$ (*atomic interaction down*), $\mathsf{ac}{\downarrow}$ (*atomic contraction down*), $\mathsf{aw}{\downarrow}$ (*atomic weakening down*), ae (*atomic exchange*), $\mathsf{aw}{\uparrow}$ (*atomic weakening up*), $\mathsf{ac}{\uparrow}$ (*atomic contraction up*), and $\mathsf{ai}{\uparrow}$ (*atomic interaction up*). Note that some of them have already been studied, e.g., [4, 15, 7, 18]. The typing information in Figure 1 says that

- for $\mathsf{ai}{\downarrow}$ (resp. $\mathsf{ai}{\uparrow}$) the two output edges (resp. input edges) carry opposite atomic types,
- for $\mathsf{ac}{\downarrow}$ (resp. $\mathsf{ac}{\uparrow}$) all input and output edges carry the same atomic type,

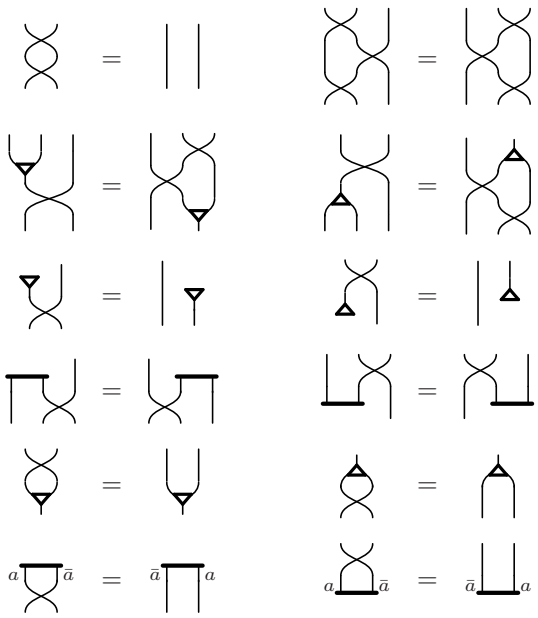


Figure 2: Relations for atomic flows

- for $\mathsf{aw}\downarrow$ (resp. $\mathsf{aw}\uparrow$) there are no typing restrictions, and
- for ae, the left input has to carry the same type as the right output, and vice versa.

When picturing an atomic flow we will omit the typing when this information is irrelevant or clear from context, as done in Figure 2. The typing is needed for two reasons: first, we need to exclude illegal flows like

Note, that for this it would suffice to have only two types, $+$ and $-$, as done in [10]. The second reason for having the types here is the use of the flows as tool for proof transformations later in this paper.

Definition 2.1. For a given atomic flow diagram ϕ, we define its *atomic flow graph* G_ϕ to be the directed acyclic graph whose vertices are the generators $\mathsf{ai}\downarrow$, $\mathsf{ai}\uparrow$, $\mathsf{ac}\downarrow$, $\mathsf{ac}\uparrow$, $\mathsf{aw}\downarrow$, $\mathsf{aw}\uparrow$ (i.e., all except ae) appearing in ϕ, whose incoming (resp. outcoming) edges are the incoming (resp. outcoming) edges of ϕ, and whose inner edges are downwards oriented as indicated by the flow diagram for ϕ. A *path* in ϕ is a path in G_ϕ.

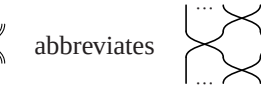
Remark 2.2. If we label the edges in G_ϕ by the corresponding atomic type, then for every path in ϕ, all its edges carry the same label.

Remark 2.3. In [10], atomic flows have been defined as directed graphs, as done in Definition 2.1. Indeed, G_ϕ is the "canonical representative" of a class of flow diagrams wrt. to the equalities in Figure 2. However, with this definition the order of the input/output edges is lost, which makes the vertical composition and the mapping from formal derivations (done in Section 6) more difficult to define.

Remark 2.4. We could add the relations

and their duals to the ones given in Figure 2, and this would equip every object in $\mathbf{AF}$ with a monoid and a comonoid structure. However, the results in this paper do not rely on that, and we decided to keep the set of relations minimal.

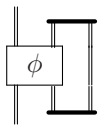
Notation 2.5. For making large atomic flows easier to read, we introduce the following notation:



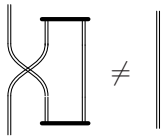
This can be extended to all generators:

And similarly for $\mathsf{aw}\uparrow$, $\mathsf{ac}\uparrow$, and $\mathsf{ai}\uparrow$. In each case we allow the number of edges to be 0, which then yields the empty flow. Moreover, if we label an abbreviation with atomic type a, we mean that each edge being abbreviated has type a. For instance:

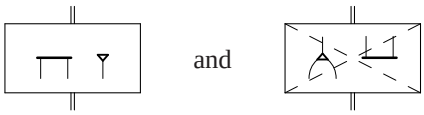
Remark 2.6. The category $\mathbf{AF}$ of atomic flows is strictly monoidal, but it is not compact closed, basically because we do not have an equality between the two atomic flows shown in (1). More precisely, although we can for a given atomic flow $\phi\colon p \,|\, x \to q \,|\, x$ define the atomic flow $\mathrm{Tr}^x(\phi)\colon p \to q$ as



the category $\mathbf{AF}$ is not traced [12], because it does not obey yanking:

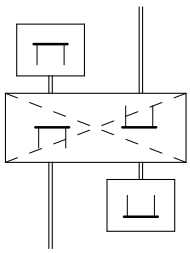


Notation 2.7. A box containing some generators stands for an atomic flow generated only from these generators, and a box containing some generators crossed out stands for an atomic flow that does not contain any of these generators. For example, the two diagrams



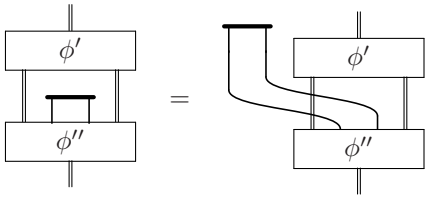
stand for a flow that contains only $\mathsf{ai}{\downarrow}$ and $\mathsf{aw}{\downarrow}$ generators and a flow that does not contain any $\mathsf{ac}{\uparrow}$ and $\mathsf{ai}{\uparrow}$ generators, respectively.

Proposition 2.8. *Every atomic flow ϕ can be written in the following form:*



(2)

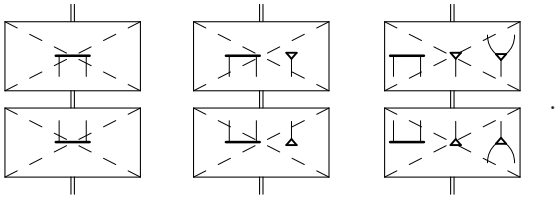
Proof. Let ϕ be given and pick an arbitrary occurrence of $\mathsf{ai}{\downarrow}$ inside ϕ. Then ϕ can be written as shown on the left below.



(3)

The equality follows by induction on the number of vertical edges to cross, For $\mathsf{ai}{\uparrow}$ we proceed dually. □

Definition 2.9. An atomic flow is *weakly streamlined* (resp., *streamlined* and *strongly streamlined*) if it can be represented as the flow on the left (resp., in the middle and on the right):



Proposition 2.10. *An atomic flow ϕ is weakly streamlined if and only if in G_ϕ there is no path from an $\mathsf{ai}{\downarrow}$-vertex to an $\mathsf{ai}{\uparrow}$-vertex.*

Proof. Immediate from (3), read from right to left. □

Definition 2.11. An atomic flow ϕ is *weakly streamlined with respect to an atomic type a* if in G_ϕ there is no path from an $\mathsf{ai}{\downarrow}$-vertex to an $\mathsf{ai}{\uparrow}$-vertex, whose edges are labelled by a or $\bar{a}$.

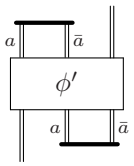
3 Properties of Atomic Flows

In this section we show some properties of atomic flows. Apart from Proposition 3.3 they are not needed in later sections of this paper, but they lead to an interesting normal form for atomic flows (Theorem 3.8).

Remark 3.1. Lafont [15] has shown that the generator ae together with the first two relations in Figure 2 defines the category of permutations.

Definition 3.2. Let a be an atomic type. An atomic flow ϕ is ai-*free with respect to a* if ϕ does not contain any $\mathsf{ai}{\downarrow}$ generators whose outputs are typed by a and $\bar{a}$, and ϕ does not contain any $\mathsf{ai}{\uparrow}$ generators whose inputs are typed by a and $\bar{a}$.

Proposition 3.3. *Let a be an atomic type. Then every atomic flow ϕ can be written as*

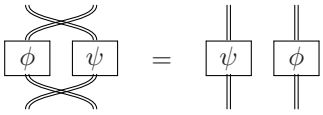


(4)

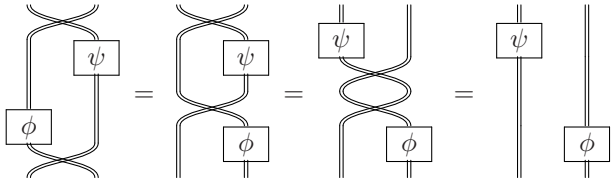
where ϕ' is ai-free with respect to a.

Proof. We apply the construction of the proof of Proposition 2.8 together with Remark 3.1 and the relations in the last line of Figure 2. □

Proposition 3.4. *For any two atomic flows ϕ and ψ, we have*



Proof. We have

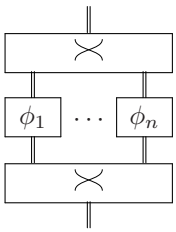


Where the first equality follows by induction on the size of ϕ, the second by induction on the size of ψ, and the third from Remark 3.1. □

Definition 3.5. An atomic flow ϕ is called *pure* if all edges carry the same atomic type. It is called *semi-pure* if only two different atomic types a and b occur in ϕ with $b = \bar{a}$.

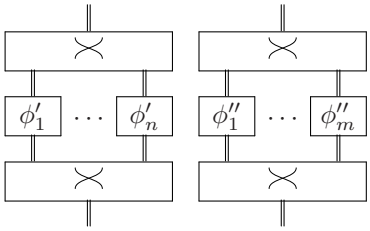
Note that a pure atomic flow cannot contain any $\mathsf{ai}{\downarrow}$ nor $\mathsf{ai}{\uparrow}$ generators.

Proposition 3.6. *Every atomic flow can be written as*

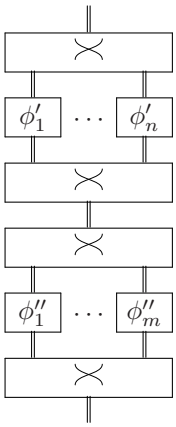


where $\phi_1, \ldots, \phi_n$ are all semi-pure.

Proof. We proceed by induction on the size of ϕ. If ϕ is a generator or id_a for some atomic type a, then the result is trivial. If $\phi = \phi' \mid \phi''$, then by induction hypothesis we have that ϕ is equal to

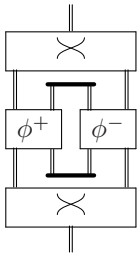


and the result follows from Proposition 3.4. If $\phi = \phi' \circ \phi''$, then by induction hypothesis we have that ϕ is equal to



Because of Proposition 3.4, we can assume that the edges in ϕ'_i and ϕ''_i carry the same atomic types, and by allowing the empty flow, we can assume that $n = m$. Then, the two exchange boxes in the middle must compose to the identity (see Remark 3.1). □

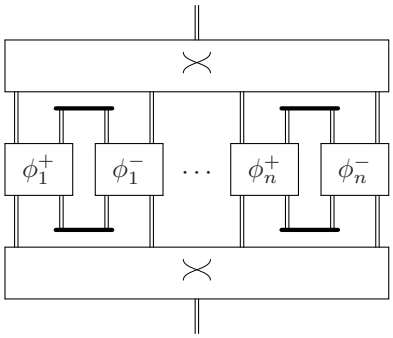
Proposition 3.7. *Every semi-pure atomic flow ϕ can be written as*



where ϕ^+ and ϕ^- are both pure.

Proof. First apply Proposition 3.3 to get a flow of shape (4). Then apply the construction of the previous proof to ϕ'. The result then follows by Remark 3.1. □

Theorem 3.8. *Every atomic flow can be written as*



where $\phi_1^+, \phi_1^-, \ldots, \phi_n^+, \phi_n^-$ are all pure.

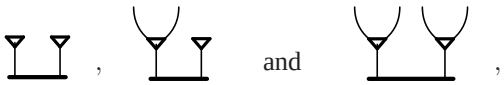
Proof. Immediate from Propositions 3.6 and 3.7. □

4 Local Flow Transformations

We denote by $\overset{\mathsf{cw}}{\to}$ the rewrite relation on atomic flows generated by the rules shown in Figure 3.

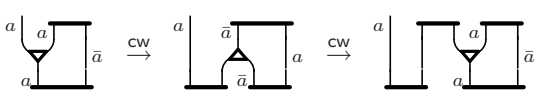
Proposition 4.1. *The rewrite relation $\overset{\mathsf{cw}}{\to}$ is locally confluent.*

Proof. The result follows from a case analysis on the critical peaks, which are:



and their duals. □

However, in general the reduction $\overset{\mathsf{cw}}{\to}$ is not terminating. This can easily be seen by the following example:



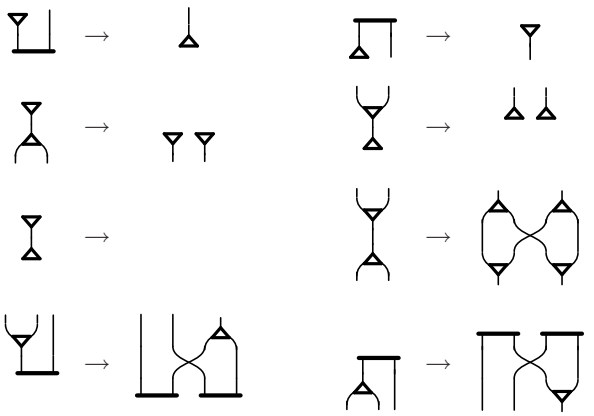


Figure 3: Local rewrite rules

The reason is that there can be cycles composed of paths connecting instances of the $\mathsf{ai}{\downarrow}$ and $\mathsf{ai}{\uparrow}$ generators. The purpose of the notion "weakly streamlined" (Definition 2.9) is precisely to avoid such a situation.

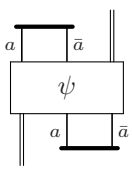
Theorem 4.2. *Every weakly streamlined atomic flow has a unique normal form with respect to* $\overset{\mathsf{cw}}{\rightarrow}$*, and this normal form is strongly streamlined.*

Proof. We do not show the proof of termination here since it can be found in [10]. We only note that the crucial point is Proposition 2.10. Then, by Proposition 4.1, we have uniqueness of the normal form. Since $\overset{\mathsf{cw}}{\rightarrow}$ preserves the property of being weakly streamlined, and in the normal form there are no more redexes for $\overset{\mathsf{cw}}{\rightarrow}$, there is no generator $\mathsf{ai}{\downarrow}$, $\mathsf{aw}{\downarrow}$, $\mathsf{ac}{\downarrow}$ above a generator $\mathsf{ai}{\uparrow}$, $\mathsf{aw}{\uparrow}$, $\mathsf{ac}{\uparrow}$. □

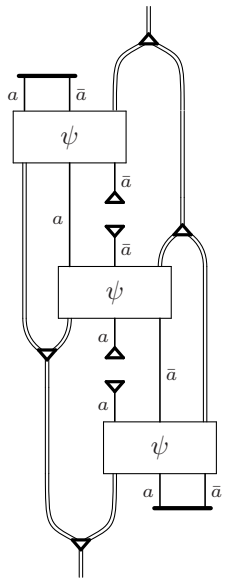
5 Global Flow Transformations

The purpose of this section is to present a method for transforming any atomic flow into a weakly streamlined one. The challenge is acually to find an operation that can be lifted to proofs in a deductive system. Here, we present one that breaks paths in the flow without removing any edge. This construction can be considered to be the heart of this paper.

Definition 5.1. Let ϕ be an atomic flow of the shape

$\phi \quad = \quad$  $\quad , \qquad (5)$

where the edges of the selected $\mathsf{ai}{\downarrow}$ and $\mathsf{ai}{\uparrow}$ generators carry the same atomic types, as indicated in (5), and let ϕ' be the atomic flow

$\phi' \quad = \quad$  $\quad . \qquad (6)$

Then we call ϕ' a *path breaker of* ϕ *with respect to* a, and write $\phi \overset{\mathsf{pb}}{\rightarrow}_a \phi'$.

Lemma 5.2. *Let* ϕ *and* ϕ' *be given with* $\phi \overset{\mathsf{pb}}{\rightarrow}_a \phi'$*, and let* b *be any atomic type. If* ϕ *is weakly streamlined with respect to* b*, then so is* ϕ'*.*

Proof. The only edges connecting an output of one copy of ψ to an input of another copy of ψ are typed by a and $\bar{a}$. Thus, the lemma is evident for $b \neq a$ and $b \neq \bar{a}$. Let us now assume $b = a$ and proceed by contradiction. Assume there is an $\mathsf{ai}{\downarrow}$ generator connected to an $\mathsf{ai}{\uparrow}$ generator via a path typed by a. If this is inside a copy of ψ, we have a contradiction; if it passes through the a-edge between the upper and the middle copy of ψ in (6), then this path connects to the $\mathsf{ai}{\downarrow}$ in (5), which also is a contradiction. Similarly for a path typed by $\bar{a}$. □

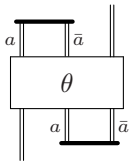
Lemma 5.3. *Let* ϕ*,* ψ*, and* a *be given as in* (5)*, and let* $\phi \overset{\mathsf{pb}}{\rightarrow}_a \phi'$*. If* ψ *is* ai*-free with respect to* a*, then* ϕ' *is weakly streamlined with respect to* a*.*

Proof. For not being weakly streamlined with respect to a, we would need a path connecting the upper $\mathsf{ai}{\downarrow}$ in (6) to the lower $\mathsf{ai}{\uparrow}$. However, such a path must pass through both the evidenced edge of type a and the evidenced edge of type $\bar{a}$, which is impossible (see Remark 2.2). □

Lemmas 5.2 and 5.3 justify the name *path breaker* for the atomic flow in (6). It breaks all paths between the upper $\mathsf{ai}{\downarrow}$ and the lower $\mathsf{ai}{\uparrow}$ in (5), and it does not introduce any new paths. Furthermore, the interior of the flow ψ is not touched.

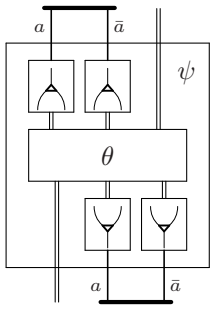
We now have to find a way to convert any atomic flow ϕ into one of shape (5) with ψ being ai-free with respect to a. For this, notice that by Proposition 3.3, we can write ϕ as

$$\phi \quad = \quad \theta \quad , \tag{7}$$



where θ is ai-free with respect to a. This can be transformed into a flow ϕ':

$$\phi' \quad = \quad \psi \quad \theta \quad , \tag{8}$$



which is of the desired shape and fulfills the condition of Lemma 5.3.

Definition 5.4. Let ϕ and ϕ' of shape (7) and (8) be given. If θ is ai-free with respect to a, then we call ϕ' a *taming of ϕ with respect to a*, and write $\phi \overset{\mathsf{tm}}{\to}_a \phi'$.

Lemma 5.5. *Let ϕ and ϕ' be given with $\phi \overset{\mathsf{tm}}{\to}_a \phi'$, and let b be any atomic type. If ϕ is weakly streamlined with respect to b, then so is ϕ'.*

Proof. Immediate from (7) and (8). □

Definition 5.6. On atomic flows, we define the *path breaking relation* $\overset{\mathsf{PB}}{\to}$ as follows. We have $\phi \overset{\mathsf{PB}}{\to} \phi'$ if and only if there is a flow ϕ'' and an atomic type a, such that $\phi \overset{\mathsf{tm}}{\to}_a \phi'' \overset{\mathsf{pb}}{\to}_a \phi'$ and ϕ is not weakly streamlined with respect to a.

Theorem 5.7. *The relation $\overset{\mathsf{PB}}{\to}$ is terminating, and its normal forms are weakly streamlined.*

Proof. Let ϕ be given. We proceed by induction on the number of atomic types occurring in ϕ, with respect to which ϕ is not weakly streamlined. Whenever we have $\phi \overset{\mathsf{PB}}{\to} \phi'$, this number is decreased by one for ϕ' (by Lemmas 5.2, 5.3 and 5.5). By the constructions in (6) and (8), there is always such a ϕ' if ϕ is not weakly streamlined. □

6 From Formal Deductions to Atomic Flows

We consider *formulas* that are generated from the set $\mathcal{A}$ of atomic types via the binary connectives $\wedge$ and $\vee$, and the nullary connectives (constants) t and f. We interpret the duality bijection $\bar{(\cdot)} : \mathcal{A} \to \mathcal{A}$ on atomic types as negation and extend the definition of negation to all formulas via the usual DeMorgan laws. *Sequents* are finite lists of formulas, separated by comma. Formulas are denoted by A, B, C, D, and sequents are denoted by Γ, Δ. We can assign to each formula, sequent, or list of sequents its flow type by forgetting the structural information of $\wedge$, $\vee$, t and f, and simply keeping the list of atomic types as they occur in the formulas. For a formula A, we denote this type by $\mathit{fl}(A)$.

Similarly, we will assign flows to inference rules. Rules like

$$\wedge \frac{\vdash \Gamma, A \qquad \vdash B, \Delta}{\vdash \Gamma, A \wedge B, \Delta} \qquad \text{and} \qquad \vee \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \vee B}$$

will be translated into the identity flows $\mathsf{id}_{\mathit{fl}(\Gamma)} \mid \mathsf{id}_{\mathit{fl}(A)} \mid \mathsf{id}_{\mathit{fl}(B)} \mid \mathsf{id}_{\mathit{fl}(\Delta)}$ and $\mathsf{id}_{\mathit{fl}(\Gamma)} \mid \mathsf{id}_{\mathit{fl}(A)} \mid \mathsf{id}_{\mathit{fl}(B)}$, respectively. And the rules

$$\mathsf{cont} \frac{\vdash \Gamma, A, A}{\vdash \Gamma, A} \qquad \text{and} \qquad \mathsf{weak} \frac{\vdash \Gamma}{\vdash \Gamma, A}$$

will be translated into flows of the shape

 and  .

In this manner, we could translate whole sequent proofs into atomic flows. However, atomic flows carry more symmetries than present in the sequent calculus. In order to be able to mirror the richness of atomic flows inside a sound and complete deductive system for classical logic, we use here a deep inference system.

More precisely, in this paper, we concentrate on system SKS [2], whose rules are given in Figure 4. Recall that these rules can, like rewrite rules, be applied inside arbitrary contexts. For example,

$$\mathsf{ai}{\uparrow} \frac{[a \vee (c \wedge [(b \wedge \bar{b}) \vee \bar{a}])] \vee (b \wedge \bar{c})}{[a \vee (c \wedge [\mathsf{f} \vee \bar{a}])] \vee (b \wedge \bar{c})}$$

is a correct application of the rule $\mathsf{ai}{\uparrow}$ inside the context $[a \vee (c \wedge [\{\cdot\} \vee \bar{a}])] \vee (b \wedge \bar{c})$. Note that here we allow negation only on atoms, and therefore all contexts are positive. A *derivation* $\Phi \colon A \to B$ in SKS is a rewrite path from A to B using the rules in Figure 4. We call A the *premise* and B the *conclusion* of Φ. A derivation is also denoted as

$$\begin{array}{c} A \\ \Phi \big\| \mathcal{S} \\ B \end{array} .$$

where $\mathcal{S}$ is the set of inference rules used in Φ.

$$\mathsf{ai}{\downarrow}\,\frac{K\{\mathsf{t}\}}{K\{a\vee\bar{a}\}} \qquad \mathsf{s}\,\frac{K\{A\wedge[B\vee C]\}}{K\{(A\wedge B)\vee C\}} \qquad \mathsf{ai}{\uparrow}\,\frac{K\{a\wedge\bar{a}\}}{K\{\mathsf{f}\}}$$

$$\mathsf{aw}{\downarrow}\,\frac{K\{\mathsf{f}\}}{K\{a\}} \qquad \mathsf{m}\,\frac{K\{(A\wedge B)\vee(C\wedge D)\}}{K\{[A\vee C]\wedge[B\vee D]\}} \qquad \mathsf{aw}{\uparrow}\,\frac{K\{a\}}{K\{\mathsf{t}\}}$$

$$\mathsf{ac}{\downarrow}\,\frac{K\{a\vee a\}}{K\{a\}} \qquad \mathsf{nm_f}\,\frac{K\{\mathsf{f}\}}{K\{\mathsf{f}\wedge\mathsf{f}\}} \qquad \mathsf{nm_t}\,\frac{K\{\mathsf{t}\vee\mathsf{t}\}}{K\{\mathsf{t}\}} \qquad \mathsf{ac}{\uparrow}\,\frac{K\{a\}}{K\{a\wedge a\}}$$

$$=_{\alpha\vee}\,\frac{K\{A\vee[B\vee C]\}}{K\{[A\vee B]\vee C\}} \qquad =_{\sigma\vee}\,\frac{K\{A\vee B\}}{K\{B\vee A\}} \qquad =_{\sigma\wedge}\,\frac{K\{A\wedge B\}}{K\{B\wedge A\}} \qquad =_{\alpha\wedge}\,\frac{K\{A\wedge(B\wedge C)\}}{K\{(A\wedge B)\wedge C\}}$$

$$=_{\hat{\mathsf{f}}}\,\frac{K\{A\}}{K\{A\vee\mathsf{f}\}} \qquad =_{\mathsf{f}}\,\frac{K\{A\vee\mathsf{f}\}}{K\{A\}} \qquad =_{\hat{\mathsf{t}}}\,\frac{K\{\mathsf{t}\wedge A\}}{K\{A\}} \qquad =_{\mathsf{t}}\,\frac{K\{A\}}{K\{\mathsf{t}\wedge A\}}$$

Figure 4: The inference rules of system SKS

Proposition 6.1. *The inference rules*

$$\mathsf{c}{\downarrow}\,\frac{K\{A\vee A\}}{K\{A\}} \quad \textit{and} \quad \mathsf{c}{\uparrow}\,\frac{K\{A\}}{K\{A\wedge A\}}$$

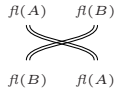
are derivable in system SKS. [2]

Definition 6.2. A *proof* of a formula A in SKS is a derivation $\Pi\colon \mathsf{t}\to A$. A proof is *cut-free* if it does not contain any instances of the rules ai↑, aw↑, or ac↑.

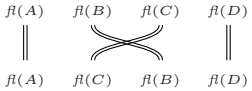
We refer the reader to [2, 1] for the precise correspondence between cut-freeness in SKS and cut-freeness in the sequent calculus.

We use **SKS** to denote the category whose objects are the formulas and whose arrows are the derivations of SKS. This system has the additional advantage that the rules for weakening, contraction, and identity and cut are already in atomic form. Thus, it is straightforward to translate SKS derivations into atomic flows. Formally, we assign to each context $K\{\cdot\}$ a left type and a right type denoted by $l(K\{\cdot\})$ and $r(K\{\cdot\})$, containing the lists of atomic types appearing in $K\{\cdot\}$ on the left, respectively on the right of the hole $\{\cdot\}$. For example, for $K\{\cdot\}=[a\vee(c\wedge[\{\cdot\}\vee\bar{a}])]\vee(b\wedge\bar{c})$ we have $l(K\{\cdot\})=\langle a,c\rangle$ and $r(K\{\cdot\})=\langle\bar{a},b,\bar{c}\rangle$. Then, for each rule r of SKS we define the *rule flow* $\mathit{fl}(\mathsf{r})$ as follows: we map the rules ai↓, ai↑, ac↓, ac↑, aw↓, and aw↑ to the corresponding generator (with the appropriate typing), and we map the rules $=_{\sigma\vee}$, $=_{\sigma\wedge}$, and m to the permutation flows shown below:

$=_{\sigma\vee}, =_{\sigma\wedge}$:



m :



All remaining rules are mapped to the identity flow.

Then an inference step

$$\mathsf{r}\,\frac{K\{A\}}{K\{B\}} \quad \text{is mapped to} \quad \mathsf{id}_{l(K\{\cdot\})}\mid \mathit{fl}(\mathsf{r})\mid \mathsf{id}_{r(K\{\cdot\})}$$

A derivation Φ is mapped to the atomic flow $\phi=\mathit{fl}(\Phi)$, which is the vertical composition of the atomic flows obtained from the inference steps in Φ. This translation defines a forgetful functor $\mathit{fl}\colon \mathbf{SKS}\to\mathbf{AF}$. Note that this functor is independent from the fact whether the binary connectives $\wedge$ and $\vee$ are bifunctors in **SKS** (with or without monoidal structure), whether the inference rules s and m are natural transformation, and whether $=_{\alpha\vee}$, $=_{\alpha\wedge}$, etc., are isomorphisms in **SKS** or not (see [22] for details on this issue). This functor has, however, an interesting property: for every atomic flow there is a derivation that maps to it:

Theorem 6.3. *For every flow $\phi\colon p\to q$ there is a derivation $\Phi\colon A\to B$ with $\mathit{fl}(A)=p$ and $\mathit{fl}(B)=q$ and $\mathit{fl}(\Phi)=\phi$.* [10]

Theorem 6.3 only works because the flows forget the structural information about $\wedge$, $\vee$, t and f. If we fix $\phi\colon p\to q$ together with A and B with $\mathit{fl}(A)=p$ and $\mathit{fl}(B)=q$, we can in general not provide a derivation $\Phi\colon A\to B$ with $\mathit{fl}(\Phi)=\phi$. We are thus interested in properties of atomic flows that can be lifted to derivations, in the following sense:

Definition 6.4. We say that a binary relation R on atomic flows *can be lifted to* SKS, if $R(\phi,\phi')$ implies that for every derivation $\Phi\colon A\to B$ with $\mathit{fl}(\Phi)=\phi$ there is a derivation $\Phi'\colon A\to B$ with $\mathit{fl}(\Phi')=\phi'$.

Definition 6.5. A derivation $\Phi\colon A \to B$ is *weakly streamlined (resp. streamlined, resp. strongly streamlined)* if $\mathit{fl}(\Phi)$ is weakly streamlined (resp. streamlined, resp. strongly streamlined).

The property *strongly streamlined* can indeed be seen as the up-down symmetric generalization being *cut-free*:

Proposition 6.6. *Every strongly streamlined proof in* SKS *is cut-free.*

Proof. If the premise of a derivation is t, then the upper box of its flow, as given in Definition 2.9, must be empty. □

7 Normalizing Derivations via Atomic Flows

In this section we show that the relations defined in Sections 4 and 5 on atomic flows can be lifted to SKS. This gives us a procedure to transform an arbitrary SKS derivation first into a weakly streamlined one, and then into a strongly streamlined one, without changing premise and conclusion during the process. In other words we are going to show the following:

Theorem 7.1. *For every* SKS *derivation from A to B, there is a* SKS*-derivation from A to B that is strongly streamlined.*

From which we get immediately the cut elimination theorem for SKS:

Corollary 7.2. *For every* SKS*-proof of A, there is a cut-free* SKS*-proof of A.*

Proof. By Theorem 7.1 and Proposition 6.6. □

It has already been shown in [10] that the local transformation discussed in Section 4 can be lifted to SKS:

Theorem 7.3. *The relation $\overset{\mathsf{cw}}{\to}$ can be lifted to* SKS*.* [10]

For the remainder of this section, we use the following convention: For saving space we use a new inference rule $=$, which stands for any derivation using only the $=_{\cdot\cdot}$ rules in Figure 4. More precisely,

$$= \frac{A}{B} \quad \text{abbreviates} \quad \begin{array}{c} A \\ \| \{=_{\alpha\vee},=_{\alpha\wedge},=_{\sigma\vee},=_{\sigma\wedge},=_{\mathsf{t}},=_{\bar{\mathsf{t}}},=_{\bar{\mathsf{f}}},=_{\mathsf{f}}\} \\ B \end{array} .$$

Lemma 7.4. *Given a context $K\{\cdot\}$ and a formula A, there exist derivations*

$$\begin{array}{c} A \wedge K\{\mathsf{t}\} \\ \| \{\mathsf{s},=\} \\ K\{A\} \end{array} \quad \textit{and} \quad \begin{array}{c} K\{A\} \\ \| \{\mathsf{s},=\} \\ K\{\mathsf{f}\} \vee A \end{array} .$$

Proof. By structural induction on $K\{\cdot\}$. [21, 1]. □

We use this lemma to show that the transformation in the proof of Proposition 2.8, which does nothing to the flows can "be lifted" to SKS in the following sense. Let Φ be a derivation. Then for every instance of the rule $\mathsf{ai}{\downarrow}$, we can do the transformation:

$$\mathsf{ai}{\downarrow}\,\dfrac{\begin{array}{c} A \\ \Phi' \| \\ K\{\mathsf{t}\} \end{array}}{\begin{array}{c} K\{a \vee \bar{a}\} \\ \Phi'' \| \\ B \end{array}} \quad \to \quad \mathsf{ai}{\downarrow}\,\dfrac{=_{\mathsf{t}}\,\dfrac{A}{\mathsf{t} \wedge A}}{[a \vee \bar{a}] \wedge A} \quad \begin{array}{c} \Phi' \| \\ [a \vee \bar{a}] \wedge K\{\mathsf{t}\} \\ \| \{\mathsf{s},=\} \\ K\{a \vee \bar{a}\} \\ \Phi'' \| \\ B \end{array} , \tag{9}$$

which does not change the atomic flow, and dually for $\mathsf{ai}{\uparrow}$.

Lemma 7.5. *The relation $\overset{\mathsf{tm}}{\to}_a$ can be lifted to* SKS*.*

Proof. Let $\Phi\colon A \to B$ with $\mathit{fl}(\Phi) = \phi$ and a be given. By repeatedly applying (9) we get the derivation

$$\begin{array}{c} A \\ \| \{\mathsf{ai}{\downarrow},=\} \\ [a \vee \bar{a}] \wedge \cdots \wedge [a \vee \bar{a}] \wedge A \\ \Theta \| \\ B \vee (a \wedge \bar{a}) \vee \cdots \vee (a \wedge \bar{a}) \\ \| \{\mathsf{ai}{\uparrow},=\} \\ B \end{array} ,$$

with $\mathit{fl}(\Theta) = \theta$, from which we can obtain a derivation

$$\begin{array}{c} A \\ =_{\mathsf{t}} \dfrac{}{\mathsf{t} \wedge A} \\ \mathsf{ai}{\downarrow},=_{\mathsf{t}} \dfrac{}{[a \vee \bar{a}] \wedge A} \\ \| \{\mathsf{ac}{\uparrow},=\} \\ [(a \wedge \cdots \wedge a) \vee (\bar{a} \wedge \cdots \wedge \bar{a})] \wedge A \\ \| \{\mathsf{m},=\} \\ [a \vee \bar{a}] \wedge \cdots \wedge [a \vee \bar{a}] \wedge A \\ \Theta \| \\ B \vee (a \wedge \bar{a}) \vee \cdots \vee (a \wedge \bar{a}) \\ \| \{\mathsf{m},=\} \\ B \vee ([a \vee \cdots \vee a] \wedge [\bar{a} \vee \cdots \vee \bar{a}]) \\ \| \{\mathsf{ac}{\downarrow},=\} \\ \mathsf{ai}{\uparrow}\,\dfrac{B \vee (a \wedge \bar{a})}{B \vee \mathsf{f}} \\ =_{\mathsf{f}} \dfrac{}{B} \end{array} ,$$

whose flow is as shown in (8). □

Lemma 7.6. *The relation $\overset{\mathsf{pb}}{\to}_a$ can be lifted to* SKS*.*

Proof. Let $\Phi\colon A \to B$ with $\mathit{fl}(\Phi) = \phi$ and a be given. By applying (9) we have a derivation

$$\begin{array}{c} \mathsf{ai}{\downarrow},{=_{\mathsf{t}}} \dfrac{A}{[a \vee \bar{a}] \wedge A} \\ \Psi \Big\| \\ {=_{\mathsf{f}}},\mathsf{ai}{\uparrow} \dfrac{B \vee (a \wedge \bar{a})}{B} \end{array} ,$$

with $\mathit{fl}(\Psi) = \psi$. We also have the derivations

$$\begin{array}{c} \mathsf{aw}{\uparrow} \dfrac{[B \vee (a \wedge \bar{a})] \wedge A}{[B \vee (a \wedge \mathsf{t})] \wedge A} \\ = \dfrac{}{[B \vee [a \vee \mathsf{f}]] \wedge A} \\ \mathsf{aw}{\downarrow} \dfrac{}{[B \vee [a \vee \bar{a}]] \wedge A} \\ \mathsf{s} \dfrac{}{B \vee ([a \vee \bar{a}] \wedge A)} \end{array} \quad \text{and} \quad \begin{array}{c} \mathsf{aw}{\uparrow} \dfrac{[B \vee (a \wedge \bar{a})] \wedge A}{[B \vee (\mathsf{t} \wedge \bar{a})] \wedge A} \\ = \dfrac{}{[B \vee [\mathsf{f} \vee \bar{a}]] \wedge A} \\ \mathsf{aw}{\downarrow} \dfrac{}{[B \vee [a \vee \bar{a}]] \wedge A} \\ \mathsf{s} \dfrac{}{B \vee ([a \vee \bar{a}] \wedge A)} \end{array}$$

that we call Φ_a and $\Phi_{\bar{a}}$, respectively. We can now build

$$\begin{array}{c} A \\ \Big\|\{\mathsf{c}{\uparrow},\mathsf{ai}{\downarrow},=\} \\ (([a \vee \bar{a}] \wedge A) \wedge A) \wedge A \\ (\Psi \wedge A) \wedge A \Big\| \\ ([B \vee (a \wedge \bar{a})] \wedge A) \wedge A \\ \Phi_a \wedge A \Big\| \\ [B \vee ([a \vee \bar{a}] \wedge A)] \wedge A \\ [B \vee \Psi] \wedge A \Big\| \\ B \vee ([B \vee (a \wedge \bar{a})] \wedge A) \\ B \vee \Phi_{\bar{a}} \Big\| \\ B \vee [B \vee ([a \vee \bar{a}] \wedge A)] \\ B \vee [B \vee \Psi] \Big\| \\ B \vee [B \vee [B \vee (a \wedge \bar{a})]] \\ \Big\|\{\mathsf{c}{\downarrow},\mathsf{ai}{\uparrow},=\} \\ B \end{array} ,$$

whose atomic flow is as shown in (6). □

Theorem 7.7. *The relation* $\overset{\mathsf{PB}}{\to}$ *can be lifted to* SKS.

Proof. Immediate from Lemmas 7.5 and 7.6. □

Proof of Theorem 7.1. For every SKS-derivation $\Phi\colon A \to B$ there exists a weakly-streamlined SKS-derivation $\Phi'\colon A \to B$ by Theorem 5.7 and Theorem 7.7; for every weakly-streamlined SKS-derivation $\Phi'\colon A \to B$ there exists a strongly streamlined SKS-derivation $\Phi''\colon A \to B$ by Theorem 4.2 and Theorem 7.3. □

References

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

[2] K. Brünnler and A. F. 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.

[3] P. Bruscoli, A. Guglielmi, T. Gundersen, and M. Parigot. A quasipolynomial cut-elimination procedure in deep inference via atomic flows and threshold formulae. In *LPAR-16*, 2010.

[4] A. Burroni. Higher dimensional word problem. In *Proceedings of the 4th International Conference on Category Theory and Computer Science*, pages 94–105, London, UK, 1991.

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

[6] V. Danos and L. Regnier. The structure of multiplicatives. *Annals of Mathematical Logic*, 28:181–203, 1989.

[7] T. Ehrhard and L. Regnier. Differential interaction nets. *Theor. Comput. Sci.*, 364(2):166–195, 2006.

[8] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[9] J.-Y. Girard. *Proof Theory and Logical Complexity, Volume I*, volume 1 of *Studies in Proof Theory*. Bibliopolis, edizioni di filosofia e scienze, 1987.

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

[11] A. Joyal and R. Street. The geometry of tensor calculus. *Advances in Mathematics*, 88:55–112, 1991.

[12] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 3:447–468, 1996.

[13] G. M. Kelly and S. Mac Lane. Coherence in closed categories. *Jour. of Pure and Applied Algebra*, 1:97–140, 1971.

[14] Y. Lafont. Interaction nets. In *POPL*, pages 95–108, 1990.

[15] Y. Lafont. *Equational reasoning with 2-dimensional diagrams*, volume 909 of *LNCS*, pages 170–195. 1995.

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

[17] O. Laurent. Polarized proof-nets: proof-nets for LC (extended abstract). In J.-Y. Girard, editor, *Typed Lambda Calculi and Applications (TLCA 1999)*, volume 1581 of *LNCS*, pages 213–227. Springer, 1999.

[18] P.-A. Melliès and N. Tabareau. Resource modalities in tensor logic. *Annals of Pure and Applied Logic*, 161(5):632 – 653, 2010. The Third workshop on Games for Logic and Programming Languages (GaLoP), Galop 2008.

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

[20] P. Selinger. A survey of graphical languages for monoidal categories. In B. Coecke, editor, *New Structures for Physics*, Lecture Notes in Physics. Springer, 2009. to appear.

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

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

[23] L. Straßburger. From deep inference to proof nets via cut elimination. *Jour. of Logic and Comp.*, 2009. to appear.