NORMALISATION CONTROL IN DEEP INFERENCE
VIA ATOMIC FLOWS[*]

ALESSIO GUGLIELMI AND TOM GUNDERSEN

University of Bath, Bath BA2 7AY, UK
e-mail address: {A.Guglielmi,T.E.Gundersen}@Bath.Ac.UK

ABSTRACT. We introduce 'atomic flows': they are graphs obtained from derivations by tracing atom occurrences and forgetting the logical structure. We study simple manipulations of atomic flows that correspond to complex reductions on derivations. This allows us to prove, for propositional logic, a new and very general normalisation theorem, which contains cut elimination as a special case. We operate in deep inference, which is more general than other syntactic paradigms, and where normalisation is more difficult to control. We argue that atomic flows are a significant technical advance for normalisation theory, because 1) the technique they support is largely independent of syntax; 2) indeed, it is largely independent of logical inference rules; 3) they constitute a powerful geometric formalism, which is more intuitive than syntax.

1. INTRODUCTION

We are interested in normalising derivations in proof systems of propositional logic. As for natural deduction and the sequent calculus [Gen69], we intend normalisation as eliminating cuts, or, more in general, 'detours', from derivations. Normalisation is performed by algorithms that, given a non-normal derivation, produce a normal derivation, free of detours. A typical detour can be depicted, in an abstract representation of a portion of a derivation, as on the left in

$$
\begin{array}{c}
\end{array}
a \overline{|\quad \bar{a}|\quad\quad|a} \quad \rightarrow \quad |a \quad ,
$$

where the atom $\bar{a}$ is created and destroyed, respectively, by an axiom and a cut, represented as two horizontal bars. In many cases, the diagram on the left can be streamlined as on the right, and this is what happens in a typical normalisation step. However, inside concrete proof systems, this abstract, geometric simplicity is almost always severely obscured by syntactic bureaucracy.

Originally stimulated by Girard's [Gir87], we share with several colleagues the research objective of getting rid of bureaucracy in proof systems. We believe that, ultimately, derivations are geometric objects of some sort, for which current syntactic formalisms only offer imperfect, bureaucratic representations. Our hopes rely, in part, on the success of proof nets for multiplicative linear logic. Those proof nets are a geometric, largely bureaucracy-free proof system, where normalisation can be directly performed as depicted above.

In classical propositional logic, the situation is more complicated. There are proof nets for classical propositional logic, but they are not a proof system (as defined by Cook and Reckhow in [CR79]), because checking their correctness requires exponential time in their size. There is not much hope for improvement, because obtaining a proof system for classical propositional logic based on proof nets would imply that coNP is equal to NP. So, proof nets are a source of inspiration, and especially those devised by Lamarche and Straßburger in [LS05b], but they are not a solution to our quest of bureaucracy-free formalisms for logics as expressive as classical logic.

This paper is about a geometric, bureaucracy-free formalism, called *atomic flows*. An atomic flow is a directed graph obtained from a derivation by only retaining information about the creation and destruction of atom occurrences. Atomic flows are, essentially, specialised Buss flow graphs [Bus91]. Notably, the atomic flow of a derivation completely disregards all the logical relations and associated inference steps; so, an atomic flow is not a derivation, but only a very abstract representation of it. Since atomic flows and the techniques they induce are largely independent of syntax, we think that they will help us defining a bureaucracy-free formalism. In fact, they yield a geometric and bureaucracy-free understanding of normalisation, which is the most important proof-theoretic aspect of proof systems.

We show that the information contained in an atomic flow is sufficient to control several normalisation algorithms for its associated derivation. This means that a normalisation algorithm extracts from a derivation its atomic flow, and then decides the normalisation steps only based on the structure of the atomic flow. In particular, atomic flows provide convenient induction measures for termination. The advantage of atomic flows is their simplicity, compared to derivations, as Figure 6 (page 30) eloquently shows. They allow us to prove, in this paper, a new and more general normalisation result than cut elimination. Proving the same without atomic flows is conceivable, but perverse; finding the normalisation algorithm without atomic flows is, in our opinion, inconceivable.

We consider derivations in the proof system SKS of the calculus of structures [BT01], which is a very general formalism based on deep inference [Gug07]. Because of their comparatively larger expressive power, normalisation in pure, unconstrained proof systems in the calculus of structures is much more challenging than in more disciplined formalisms, like the sequent calculus or natural deduction. In fact, deep-inference rules disregard the notion of root connective of a formula, which is a crucial asset for normalisation in non-deep inference formalisms.

We show that every SKS derivation can be *streamlined*, *i.e.*, we can remove all causal dependencies between axioms and cuts of the kind depicted above on the left. This result generalises cut elimination for two reasons: 1) SKS can faithfully embed derivations in most other proof systems of different formalisms, like the sequent calculus, hypersequents, natural deduction, resolution and others; 2) a cut-free proof is a special case of a streamlined derivation. Intuitively, this result is sort of a Craig's interpolation for derivations instead

of formulae. We note that, contrary to traditional cut elimination, our result is *symmetric* along the axis premiss-conclusion of a derivation (as is traditional in deep inference).

The core idea of our algorithms, as controlled by atomic flows, consists in slightly altering a derivation around a couple of matching axiom and cut, and composing it with itself. In principle, this is very similar to normalisation in natural deduction. This idea is due to Tiu and then used by Brünnler, in [Brü03], to simplify his previous proof of cut elimination in the calculus of structures. In our case, the idea allows us to dispense entirely with the usual case analysis and permutation of rules, which is the standard routine in cut elimination proofs in the sequent calculus and elsewhere.

The observed independence from syntax leads us to believe that successful normalisation stems less than is usually believed from the mutual 'harmony' between logical rules. In fact, in this paper, there is no concern whatsoever for this issue. Rather, the choice in designing logical rules seems to be essentially free, provided that they are linear and that they support atomic structural rules. Our technique relies on substituting formulae for occurrences of atoms, which is only manageable if the structural part of the proof system is atomic. As a matter of fact, complete and analytic proof systems with atomic structural rules and linear logical ones can only be designed in deep inference, for most logics.

Finally, we note that several normalisation algorithms can be designed and several normal forms can be obtained by employing atomic flows. We are seeing a *robust* normalisation phenomenon: variations in the algorithms are possible and interesting, we are not dealing here with a delicate property of derivations that requires extra care and attention to the tiniest syntactic detail. Once the basics of the technique are mastered, derivations can be manipulated with ease. We think that atomic flows get closer than ever to the essence of normalisation in classical propositional logic.

After a brief introduction to deep inference, we introduce atomic flows and their reductions, and then the streamlining algorithms. The two crucial, simple ideas to understand in this paper are shown in Remark 4.16 (page 19) and in Definition 5.1 (page 22).

2. Background on Deep Inference

Deep inference is a relatively recent development in proof theory. It is a methodology according to which several formalisms can be defined with excellent structural properties. The calculus of structures [Gug07] is one of them and is now well developed for classical [Brü03, Brü06a, Brü06d, BT01, Brü06b], intuitionistic [Tiu06a], linear [Str02, Str03b], modal [Brü06c, GT07, Sto07] and commutative/non-commutative logics [Gug07, Tiu06b, Str03a, Bru02, DG04, GS01, GS02, GS07, Kah06, Kah07b]. The basic proof complexity properties of the calculus of structures are known [BG08]. The calculus of structures promoted the discovery of a new class of proof nets for classical and linear logic [LS05a, LS05b, LS06, SL04] (see also [Gui06]). There exist implementations in Maude of deep-inference proof systems [Kah07a]. For a better introduction than this, we refer the reader to [Brü03].

Definition 2.1. *Formulae*, α, β, γ, δ are freely built from: *units*, f (false), t (true); *atoms*, a, b, c, d, e; *disjunction* and *conjunction*, $[\alpha \vee \beta]$ and $(\alpha \wedge \beta)$. On the set of atoms a (non-identical) involution $\bar{\cdot}$ is defined, and dual atom occurrences, as a and $\bar{a}$, can appear in formulae. We denote *contexts*, *i.e.*, formulae with a hole, by $\xi\{\ \}$ and $\zeta\{\ \}$; we also use *multiple* contexts, $\xi\{\ \}\cdots\{\ \}$, *i.e.*, formulae with many holes; for example, if $\xi\{a\}$

is $b \wedge [a \vee c]$, then $\xi\{\ \}$ is $b \wedge [\{\ \} \vee c]$, $\xi\{b\}$ is $b \wedge [b \vee c]$ and $\xi\{a \wedge d\}$ is $b \wedge [(a \wedge d) \vee c]$; if $\xi\{a\}\{b\}\{c\}$ is $b \wedge [(a \wedge d) \vee c]$ then $\xi\{b\}\{c\}\{a\}$ is $c \wedge [(b \wedge d) \vee a]$.

Remark 2.2. Negation is only defined for atoms, which is not a limitation thanks to De Morgan laws.

Note that when we write $\xi\{a\}$, we mean that an occurrence of a exists in the formula, we singled it out and we refer specifically to that occurrence. It is important to distinguish between an atom a and a set of occurrences of atom a inside a formula or a derivation. In the following, we mark in various ways occurrences of atoms, and we perform several substitutions of formulae in the place of atom occurrences.

Definition 2.3. *Inference rules*, ρ, have one *premiss* and one *conclusion*, and their *instances* are used in *inference steps* to rewrite inside formulae. A *derivation*, Φ, from α (*premiss*) to β (*conclusion*) is a chain of inference steps with α at the top and β at the bottom, and is

usually indicated by $\Phi \left\Vert \begin{array}{c} \alpha \\ \mathcal{S} \\ \beta \end{array}\right.$, where $\mathcal{S}$ is the name of the deductive system or a set of inference

rules; a *proof* is a derivation from t; besides Φ, we denote derivations with Ψ. We denote with $\xi\{\Phi\}$ the result of including every formula of Φ into the context $\xi\{\ \}$: since we adopt deep inference, $\xi\{\Phi\}$ is a valid derivation. We denote with $\Phi\{a \leftarrow \alpha\}$, $\Phi\{\mathsf{f} \leftarrow \alpha\}$ and $\Phi\{\mathsf{t} \leftarrow \alpha\}$ the operation of substituting α into a set of *occurrences* of an atom a or unit in Φ; the result is not necessarily a valid derivation, because some instances of rules might break; which occurrences to replace is always made clear by suitable decorations of a, f and t, like $a^{\mathbf{1}}$ and $\mathsf{f}^{\bullet}$.

Now we define the two standard deductive systems for classical propositional logic in deep inference that are used throughout the paper. KS is analytic, in the sense that premisses only contain subformulae of conclusions, and SKS is not [Brü03, Brü06a, Brü06d, BT01].

Definition 2.4. System SKS in the calculus of structures is defined by the following *structural* rules:

$$\mathsf{ai}\!\downarrow \frac{\mathsf{t}}{a \vee \bar{a}} \qquad \mathsf{aw}\!\downarrow \frac{\mathsf{f}}{a} \qquad \mathsf{ac}\!\downarrow \frac{a \vee a}{a}$$
$$\textit{interaction} \qquad \textit{weakening} \qquad \textit{contraction}$$

$$\mathsf{ai}\!\uparrow \frac{a \wedge \bar{a}}{\mathsf{f}} \qquad \mathsf{aw}\!\uparrow \frac{a}{\mathsf{t}} \qquad \mathsf{ac}\!\uparrow \frac{a}{a \wedge a}$$
$$\textit{cointeraction} \qquad \textit{coweakening} \qquad \textit{cocontraction}$$

and by the two *logical* rules:

$$\mathsf{s} \frac{\alpha \wedge [\beta \vee \gamma]}{(\alpha \wedge \beta) \vee \gamma} \qquad \mathsf{m} \frac{(\alpha \wedge \beta) \vee (\gamma \wedge \delta)}{[\alpha \vee \gamma] \wedge [\beta \vee \delta]} \quad .$$
$$\textit{switch} \qquad \textit{medial}$$

The rule cointeraction is also called an (*atomic*) *cut.* In addition to the rules shown, there is a rule $=\dfrac{\gamma}{\delta}$, such that γ and δ are opposite sides in one of the following equations:

$$\alpha \vee \beta = \beta \vee \alpha \quad , \qquad\qquad\qquad \alpha \vee \mathsf{f} = \alpha \quad ,$$
$$\alpha \wedge \beta = \beta \wedge \alpha \quad , \qquad\qquad\qquad \alpha \wedge \mathsf{t} = \alpha \quad ,$$
$$[\alpha \vee \beta] \vee \gamma = \alpha \vee [\beta \vee \gamma] \quad , \qquad\qquad \mathsf{t} \vee \mathsf{t} = \mathsf{t} \quad ,$$
$$(\alpha \wedge \beta) \wedge \gamma = \alpha \wedge (\beta \wedge \gamma) \quad , \qquad\qquad \mathsf{f} \wedge \mathsf{f} = \mathsf{f} \quad .$$

We do not always show the instances of rule $=$, and when we do show them, we gather several contiguous instances into one. System KS is the same as SKS, but without the rules $\mathsf{ai}{\uparrow}$, $\mathsf{aw}{\uparrow}$ and $\mathsf{ac}{\uparrow}$. A *cut-free* derivation is a derivation where $\mathsf{ai}{\uparrow}$ is not used. All derivations in this paper are in SKS, unless indicated otherwise.

Note that all the structural rules only apply to atoms. As shown later, equivalent structural rules applying to formulae instead of atoms can be derived from the atomic ones together with the logical rules. The fact that we can work only with atomic structural rules is essential later on.

Instead of the term 'axiom' we use 'interaction'; the reason is that, in deep inference, axioms do not close derivation branches. However, it is not misleading to think of interaction instances as axiom instances in the sequent calculus. In several papers, including [Brü03], the reader can find explanations of how reducing a proof in SKS to a proof in KS is a cut-elimination process in the traditional sense. In other words, the rules $\mathsf{ai}{\uparrow}$, $\mathsf{aw}{\uparrow}$ and $\mathsf{ac}{\uparrow}$ are, together, morally equivalent to a cut in the sequent calculus.

There are many SKS derivations in this paper, providing examples for the above definitions. Mastering the following, standard constructions of the calculus of structures is crucial: 1) moving a formula outside of a context, as in $\begin{array}{c}\xi\{\alpha\}\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\xi\{\mathsf{t}\}\wedge\alpha\end{array}$; 2) bringing a formula inside a context, as in $\begin{array}{c}\xi\{\mathsf{f}\}\vee\alpha\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\xi\{\alpha\}\end{array}$. In the next remark, we appeal to both. The constructions in the rest of this subsection are needed later in the paper.

Remark 2.5. For any $\xi\{\ \}$, $\zeta\{\ \}$ and α, by working inductively on the contexts $\xi\{\ \}$ and $\zeta\{\ \}$, we can build

$$\begin{array}{c}\xi\{\mathsf{t}\}\wedge\zeta\{\alpha\}\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\xi\{\alpha\}\vee\zeta\{\mathsf{f}\}\end{array}\quad .$$

We can do this according to the following two schemes:

$$\begin{array}{c}\xi\{\mathsf{t}\}\wedge\zeta\{\alpha\}\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\zeta\{\xi\{\mathsf{t}\}\wedge\alpha\}\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\zeta\{\xi\{\alpha\}\}\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\xi\{\alpha\}\vee\zeta\{\mathsf{f}\}\end{array}\qquad\text{and}\qquad\begin{array}{c}\xi\{\mathsf{t}\}\wedge\zeta\{\alpha\}\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\xi\{\zeta\{\alpha\}\}\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\xi\{\alpha\vee\zeta\{\mathsf{f}\}\}\\[2pt]\big\Vert\,{\scriptstyle\{\mathsf{s}\}}\\[2pt]\xi\{\alpha\}\vee\zeta\{\mathsf{f}\}\end{array}\quad .$$

For example, for $\xi\{\ \} = [\{\ \} \vee b] \wedge c$ and $\zeta\{\ \} = (d \wedge \{\ \}) \vee e$, consider

$$
\begin{array}{c}
= \dfrac{([\mathsf{t} \vee b] \wedge c) \wedge [(d \wedge \alpha) \vee e]}{([\mathsf{t} \vee b] \wedge c) \wedge [(\alpha \wedge d) \vee e]} \\
\mathsf{s}\ \dfrac{([\mathsf{t} \vee b] \wedge c) \wedge [(\alpha \wedge d) \vee e]}{(([\mathsf{t} \vee b] \wedge c) \wedge (\alpha \wedge d)) \vee e} \\
= \dfrac{(([\mathsf{t} \vee b] \wedge c) \wedge (\alpha \wedge d)) \vee e}{(((\alpha \wedge [\mathsf{t} \vee b]) \wedge c) \wedge d) \vee e} \\
\mathsf{s}\ \dfrac{(((\alpha \wedge [\mathsf{t} \vee b]) \wedge c) \wedge d) \vee e}{(([(\alpha \wedge \mathsf{t}) \vee b] \wedge c) \wedge d) \vee e} \\
= \dfrac{(([(\alpha \wedge \mathsf{t}) \vee b] \wedge c) \wedge d) \vee e}{(d \wedge [\mathsf{f} \vee ([\alpha \vee b] \wedge c)]) \vee e} \\
\mathsf{s}\ \dfrac{(d \wedge [\mathsf{f} \vee ([\alpha \vee b] \wedge c)]) \vee e}{[(d \wedge \mathsf{f}) \vee ([\alpha \vee b] \wedge c)] \vee e} \\
= \dfrac{[(d \wedge \mathsf{f}) \vee ([\alpha \vee b] \wedge c)] \vee e}{([\alpha \vee b] \wedge c) \vee [(d \wedge \mathsf{f}) \vee e]}
\end{array}\quad .
$$

We define the following 'macro' rule ss, called *super switch*, to be a shorthand for any derivation of the above form:

$$
\mathsf{ss}\ \dfrac{\xi\{\mathsf{t}\} \wedge \zeta\{\alpha\}}{\xi\{\alpha\} \vee \zeta\{\mathsf{f}\}}\quad .
$$

Remark 2.6. $\mathsf{c}{\downarrow}\ \dfrac{\alpha \vee \alpha}{\alpha}$ and $\mathsf{c}{\uparrow}\ \dfrac{\alpha}{\alpha \wedge \alpha}$ are two other 'macro' rules: they are called, respectively, *contraction* and *cocontraction*, and they apply to generic formulae instead of atoms. They can be derived, respectively, from $\{\mathsf{ac}{\downarrow}, \mathsf{m}\}$ and $\{\mathsf{ac}{\uparrow}, \mathsf{m}\}$. For an example of the latter, see Figure 1 (page 10).
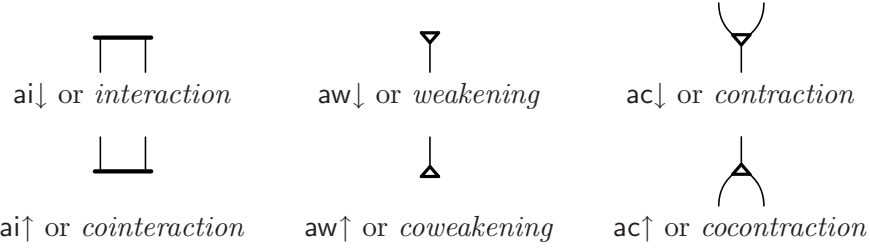
3. Atomic Flows

In this section, we define atomic flows and ai-cycles inside them. These special cycles are circular dependencies between interactions and cointeractions, and they are particularly complex to deal with in the syntax. They turn out to be natural concepts in atomic flows, and understanding them is the key to our normalisation algorithm. The section closes with the definition of streamlined atomic flows and associated derivations.

3.1. **Atomic Flows and Derivations.** Atomic flows are somewhat similar to proof nets. However, we prove that, no matter how we freely build an atomic flow (as opposed to a proof net structure), the flow is associated with some derivation. So, atomic flows are always 'sequentialisable', in proof-net parlance. In fact, atomic flows carry much less information than derivations do, because they do not keep track of the logical relations between the atoms they trace, only their structural information is retained (in the sense of structural rules, as opposed to logical ones).

We can think of atomic flows as composite diagrams that are freely generated from a set of six elementary diagrams. Technically, atomic flows are special kinds of labelled directed acyclic graphs, and the properties of their vertices are dictated by their labels, which we define as follows.

**Definition 3.1.** We call the following six diagrams (*atomic-flow*) *labels*:

ai↓ or *interaction*   aw↓ or *weakening*   ac↓ or *contraction*

ai↑ or *cointeraction*   aw↑ or *coweakening*   ac↑ or *cocontraction*

Cointeraction is also called *cut*.

Definition 3.2. An (*atomic*) *flow* is a tuple (V, E, η, up, lo) such that:

(1) V is a finite set of *vertices*, denoted by ν;

(2) E is a finite set of *edges*, denoted by ϵ;

(3) $\eta \colon V \to \{\mathsf{ai}{\downarrow}, \mathsf{ai}{\uparrow}, \mathsf{aw}{\downarrow}, \mathsf{aw}{\uparrow}, \mathsf{ac}{\downarrow}, \mathsf{ac}{\uparrow}\}$ maps vertices to their *labels*;

(4) $up \colon E \to V \cup \{\top\}$ and $lo \colon E \to V \cup \{\bot\}$ are, respectively, the *upper* and *lower* maps, and $\top$ and $\bot$ are special vertices not belonging to V; we define, for every $\nu \in V \cup \{\top, \bot\}$, the set $L_\nu = \{\, \epsilon \mid up(\epsilon) = \nu \,\}$ of *lower edges of ν*, the set $U_\nu = \{\, \epsilon \mid lo(\epsilon) = \nu \,\}$ of *upper edges of ν*, and the set $E_\nu = L_\nu \cup U_\nu$ of *edges of ν*;

(5) if $|S|$ denotes the cardinality of set S, we have that

$$\text{if } \eta(\nu) = \mathsf{ai}{\downarrow} \text{ then } |L_\nu| = 2 \text{ and } |U_\nu| = 0,$$
$$\text{if } \eta(\nu) = \mathsf{ai}{\uparrow} \text{ then } |L_\nu| = 0 \text{ and } |U_\nu| = 2,$$
$$\text{if } \eta(\nu) = \mathsf{aw}{\downarrow} \text{ then } |L_\nu| = 1 \text{ and } |U_\nu| = 0,$$
$$\text{if } \eta(\nu) = \mathsf{aw}{\uparrow} \text{ then } |L_\nu| = 0 \text{ and } |U_\nu| = 1,$$
$$\text{if } \eta(\nu) = \mathsf{ac}{\downarrow} \text{ then } |L_\nu| = 1 \text{ and } |U_\nu| = 2,$$
$$\text{if } \eta(\nu) = \mathsf{ac}{\uparrow} \text{ then } |L_\nu| = 2 \text{ and } |U_\nu| = 1;$$

(6) there is no sequence $\epsilon_1, \dots, \epsilon_h$ of edges of V such that $up(\epsilon_i) = lo(\epsilon_{i+1 \ (\mathrm{mod}\ h)})$, for $1 \le i \le h$;

(7) there is a *polarity assignment* $\pi \colon E \to \{-, +\}$ such that, for every $\nu \in V$,
  (a) if $\eta(\nu) \in \{\mathsf{ac}{\downarrow}, \mathsf{ac}{\uparrow}\}$ then $\pi(E_\nu) = \{-\}$ or $\pi(E_\nu) = \{+\}$;
  (b) if $\eta(\nu) \in \{\mathsf{ai}{\downarrow}, \mathsf{ai}{\uparrow}\}$ then $\pi(E_\nu) = \{-, +\}$.

Besides ϵ, we use small numerals $\mathbf{1}$, $\mathbf{2}$, ... and colours to denote edges. Atomic flows are denoted with A, B, C and D. Given an atomic flow A, we say that the sets $L_\top = \{\epsilon_1, \dots, \epsilon_h\}$ and $U_\bot = \{\epsilon'_1, \dots, \epsilon'_k\}$ contain, respectively, the *upper* and *lower edges of A*; in such a case, we can represent A as

$$\epsilon_1 \boxed{\cdots} \epsilon_h \quad \boxed{A} \quad \epsilon'_1 \boxed{\cdots} \epsilon'_k \quad .$$
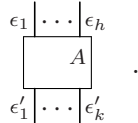
In general, we represent atomic flows as directed-graph diagrams, except that the special vertices $\top$ and $\bot$ are not shown, and the labels of the vertices are explicitly shown as graphical elements. When we refer to the vertices of an atomic flow, we do not include $\top$ and $\bot$. Sometimes we identify vertices with their labels.

An atomic flow is a directed graph, whose edges are associated to atom occurrences in derivations, and the direction of the edges corresponds to the up-down direction in a

derivation. Vertices are associated to points in the derivation where atom occurrences are created or destroyed, and the nature of each vertex is described by its label. Naturally, these graphs are acyclic (condition 6). The two special vertices $\top$ and $\bot$ represent the top and bottom of a derivation: we can consider $\top$ the vertex that creates all the atom occurrences in the premiss and $\bot$ the vertex that destroys all atom occurrences in the conclusion.

The polarity assignment condition (7) ensures that atoms in (co)contractions have the same polarity, and those in (co)interactions have dual polarities (as happens in derivations). Every atomic flow has 2^n polarity assignments, where n is the number of connected components in the graph. We should not be worried about the apparent complexity of the polarity assignment condition: in fact, we could equivalently consider two sorts of (co)contraction and (co)weakening labels, the negative and the positive ones, and ask for vertices to be joined by respecting their polarities. This is clearly a locally checkable property, much simpler than, for example, some global correctness criterion for proof nets.

Example 3.3. Consider the atomic flow

$$A = (\{\, \nu_1 \,,\, \nu_2 \,,\, \nu_3 \,\},$$
$$\{\, 1 \,,\, 2 \,,\, 3 \,,\, 4 \,,\, 5 \,\},$$
$$\{\, \nu_1 \mapsto \mathsf{ai}{\uparrow} \,,\, \nu_2 \mapsto \mathsf{ac}{\uparrow} \,,\, \nu_3 \mapsto \mathsf{ai}{\uparrow} \,\},$$
$$\{\, 1 \mapsto \top \,,\, 2 \mapsto \top \,,\, 3 \mapsto \nu_2 \,,\, 4 \mapsto \nu_2 \,,\, 5 \mapsto \top \,\},$$
$$\{\, 1 \mapsto \nu_1 \,,\, 2 \mapsto \nu_2 \,,\, 3 \mapsto \nu_1 \,,\, 4 \mapsto \nu_3 \,,\, 5 \mapsto \nu_3 \,\}) \quad;$$

the following are three of its possible representations:
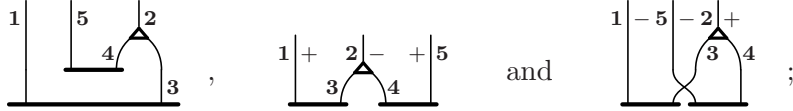
in the last two diagrams, we also indicated each of the two possible polarity assignments. This flow has one cocontraction and two cointeraction vertices; it has three upper edges, $\mathbf{1}$, $\mathbf{2}$ and $\mathbf{5}$, and no lower edges.

Example 3.4. The graph is not an atomic flow, for lack of a polarity assignment.

We now define the mapping from derivations to atomic flows. As we said, the idea is that structural rules map to the respective atomic-flow vertices, and the edges trace the atoms between inference steps. We first state a fact, whose proof is immediate.

Proposition 3.5. *Given an* SKS *derivation* Φ, *there is a unique atomic flow* A *(modulo isomorphisms) such that:*

(1) *there is a surjective map between the set of atom occurrences of* Φ *and the set of edges of* A;

(2) *for each inference step* $\rho\dfrac{\xi\{\alpha\}}{\xi\{\beta\}}$ *of* Φ, *where* $\rho \in \{\mathsf{ai}{\downarrow}, \mathsf{ai}{\uparrow}, \mathsf{aw}{\downarrow}, \mathsf{aw}{\uparrow}, \mathsf{ac}{\downarrow}, \mathsf{ac}{\uparrow}\}$ *and* $\rho\dfrac{\alpha}{\beta}$ *is a rule instance, all atom occurrences in* $\xi\{\ \}$ *in the premiss are respectively mapped to the same edges of* A *as the atom occurrences in* $\xi\{\ \}$ *in the conclusion; the atom occurrences in* $\rho\dfrac{\alpha}{\beta}$ *are mapped to edges of* A *such that the edges are related*

with vertices as indicated below, for each possible case of the inference step:

$$\mathsf{ai}{\downarrow}\,\frac{\mathsf{t}}{a^{\mathbf{1}} \vee \bar{a}^{\mathbf{2}}} \quad to \quad \text{<image>} \quad , \qquad \mathsf{ai}{\uparrow}\,\frac{a^{\mathbf{1}} \wedge \bar{a}^{\mathbf{2}}}{\mathsf{f}} \quad to \quad \text{<image>} \quad ,$$

$$\mathsf{aw}{\downarrow}\,\frac{\mathsf{f}}{a^{\mathbf{1}}} \quad to \quad \text{<image>} \quad , \qquad \mathsf{aw}{\uparrow}\,\frac{a^{\mathbf{1}}}{\mathsf{t}} \quad to \quad \text{<image>} \quad ,$$

$$\mathsf{ac}{\downarrow}\,\frac{a^{\mathbf{1}} \vee a^{\mathbf{2}}}{a^{\mathbf{3}}} \quad to \quad \text{<image>} \quad , \qquad \mathsf{ac}{\uparrow}\,\frac{a^{\mathbf{1}}}{a^{\mathbf{2}} \wedge a^{\mathbf{3}}} \quad to \quad \text{<image>} \quad ,$$

where the mapping is indicated by small numerals.

(3) *for each inference step of Φ of kind*

$$\mathsf{s}\,\frac{\xi\{\alpha \wedge [\beta \vee \gamma]\}}{\xi\{(\alpha \wedge \beta) \vee \gamma\}} \quad , \qquad \mathsf{m}\,\frac{\xi\{(\alpha \wedge \beta) \vee (\gamma \wedge \delta)\}}{\xi\{[\alpha \vee \gamma] \wedge [\beta \vee \delta]\}} \quad ,$$

$$=\frac{\xi\{\alpha \vee \beta\}}{\xi\{\beta \vee \alpha\}} \quad , \qquad =\frac{\xi\{\alpha \wedge \beta\}}{\xi\{\beta \wedge \alpha\}} \quad ,$$

$$=\frac{\xi\{[\alpha \vee \beta] \vee \gamma\}}{\xi\{\alpha \vee [\beta \vee \gamma]\}} \quad , \qquad =\frac{\xi\{\alpha \vee [\beta \vee \gamma]\}}{\xi\{[\alpha \vee \beta] \vee \gamma\}} \quad ,$$

$$=\frac{\xi\{(\alpha \wedge \beta) \wedge \gamma\}}{\xi\{\alpha \wedge (\beta \wedge \gamma)\}} \quad , \qquad =\frac{\xi\{\alpha \wedge (\beta \wedge \gamma)\}}{\xi\{(\alpha \wedge \beta) \wedge \gamma\}} \quad ,$$

$$=\frac{\xi\{\alpha \vee \mathsf{f}\}}{\xi\{\alpha\}} \quad , \qquad =\frac{\xi\{\alpha\}}{\xi\{\alpha \vee \mathsf{f}\}} \quad , \qquad =\frac{\xi\{\alpha \wedge \mathsf{t}\}}{\xi\{\alpha\}} \quad and \qquad =\frac{\xi\{\alpha\}}{\xi\{\alpha \wedge \mathsf{t}\}}$$

all the atom occurrences in $\xi\{\ \}$, α, β, γ and δ in the premiss are respectively mapped to the same edges of A as the atom occurrences in $\xi\{\ \}$, α, β, γ and δ in the conclusion.

Definition 3.6. Given a derivation Φ, we say that the unique atomic flow A defined in Proposition 3.5 is the atomic flow *associated with* the derivation Φ. Sometimes, when an atom occurrence a in Φ maps to an edge ϵ in A, we decorate ϵ with the label a.

Example 3.7. Figure 1 has some examples of atomic flows associated with derivations.

Inference rules are usually called linear when they do not 'create' nor 'destroy' atoms. Linear rules of SKS are switch, medial and (every equation defining) rule =. Note that linear inference rules do not introduce any vertices in atomic flows.

We now show that there is no such thing as an 'invalid atomic flow', or, in other words, the mapping from derivations to atomic flows is surjective.

Theorem 3.8. *Every atomic flow is associated with some derivation.*

Proof. First, we construct a derivation scheme Φ that 'glues together' any two atomic flows without introducing any vertices. For every α, β, the atomic flow of the following derivation
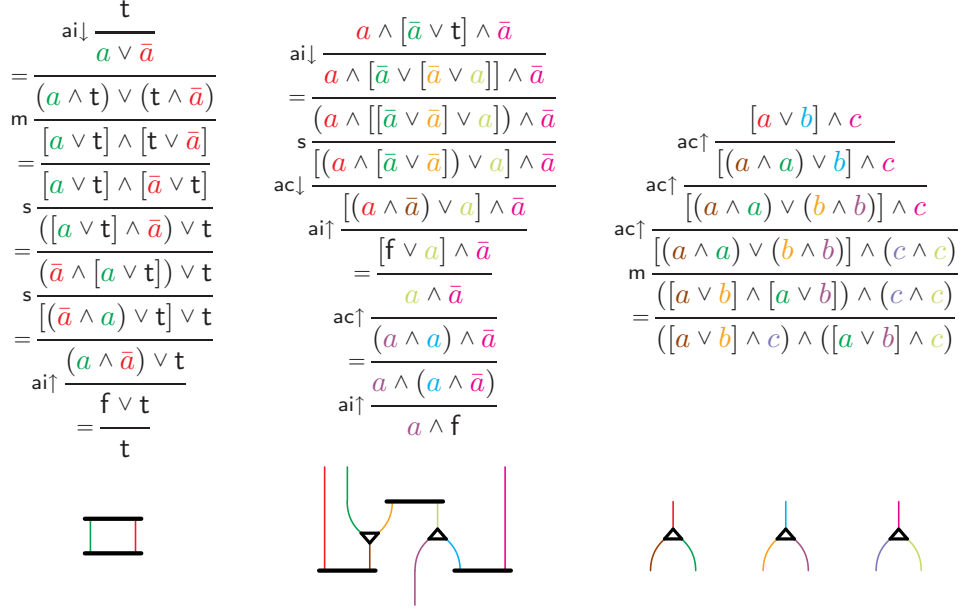
$$
\mathsf{ai}\!\downarrow \frac{\mathsf{t}}{a \vee \bar{a}}
$$
$$
= \frac{(a \wedge \mathsf{t}) \vee (\mathsf{t} \wedge \bar{a})}{\mathsf{m}}
$$
$$
= \frac{[a \vee \mathsf{t}] \wedge [\mathsf{t} \vee \bar{a}]}{\mathsf{s}}
$$
$$
= \frac{[a \vee \mathsf{t}] \wedge [\bar{a} \vee \mathsf{t}]}{([a \vee \mathsf{t}] \wedge \bar{a}) \vee \mathsf{t}}
$$
$$
= \frac{(\bar{a} \wedge [a \vee \mathsf{t}]) \vee \mathsf{t}}{\mathsf{s}}
$$
$$
= \frac{[(\bar{a} \wedge a) \vee \mathsf{t}] \vee \mathsf{t}}{\mathsf{ai}\!\uparrow}
$$
$$
\mathsf{ai}\!\uparrow \frac{(a \wedge \bar{a}) \vee \mathsf{t}}{\mathsf{f} \vee \mathsf{t}}
$$
$$
= \frac{\mathsf{f} \vee \mathsf{t}}{\mathsf{t}}
$$

$$
\mathsf{ai}\!\downarrow \frac{a \wedge [\bar{a} \vee \mathsf{t}] \wedge \bar{a}}{a \wedge [\bar{a} \vee [\bar{a} \vee a]] \wedge \bar{a}}
$$
$$
= \frac{(a \wedge [[\bar{a} \vee \bar{a}] \vee a]) \wedge \bar{a}}{\mathsf{s}}
$$
$$
\mathsf{s} \frac{[(a \wedge [\bar{a} \vee \bar{a}]) \vee a] \wedge \bar{a}}{\mathsf{ac}\!\downarrow}
$$
$$
\mathsf{ac}\!\downarrow \frac{[(a \wedge \bar{a}) \vee a] \wedge \bar{a}}{\mathsf{ai}\!\uparrow}
$$
$$
= \frac{[\mathsf{f} \vee a] \wedge \bar{a}}{a \wedge \bar{a}}
$$
$$
\mathsf{ac}\!\uparrow \frac{(a \wedge a) \wedge \bar{a}}{\mathsf{ai}\!\uparrow}
$$
$$
= \frac{a \wedge (a \wedge \bar{a})}{a \wedge \mathsf{f}}
$$

$$
\mathsf{ac}\!\uparrow \frac{[a \vee b] \wedge c}{[(a \wedge a) \vee b] \wedge c}
$$
$$
\mathsf{ac}\!\uparrow \frac{[(a \wedge a) \vee (b \wedge b)] \wedge c}{[(a \wedge a) \vee (b \wedge b)] \wedge (c \wedge c)}
$$
$$
\mathsf{m} \frac{([a \vee b] \wedge [a \vee b]) \wedge (c \wedge c)}{([a \vee b] \wedge c) \wedge ([a \vee b] \wedge c)}
$$

Figure 1: Examples of atomic flows associated with derivations.

consists only of edges (there are no vertices):

$$
\Psi = \quad
\begin{array}{c}
= \dfrac{[\alpha \vee \beta] \vee \mathsf{t}}{\mathsf{m} \dfrac{[(\alpha \wedge \mathsf{t}) \vee (\mathsf{t} \wedge \beta)] \vee \mathsf{t}}{([\alpha \vee \mathsf{t}] \wedge [\mathsf{t} \vee \beta]) \vee \mathsf{t}}} \\[6pt]
= \dfrac{([\alpha \vee \mathsf{t}] \wedge [\beta \vee \mathsf{t}]) \vee \mathsf{t}}{\mathsf{s} \dfrac{[([\alpha \vee \mathsf{t}] \wedge \beta) \vee \mathsf{t}] \vee \mathsf{t}}{(\beta \wedge [\alpha \vee \mathsf{t}]) \vee \mathsf{t}}} \\[6pt]
= \dfrac{[(\beta \wedge \alpha) \vee \mathsf{t}] \vee \mathsf{t}}{(\alpha \wedge \beta) \vee \mathsf{t}}
\end{array}
\quad .
$$

We use Ψ and ss (see Remark 2.5) to 'move' an atom a from one context $\zeta\{\ \}$ to another context $\xi\{\ \}$, again with an associated atomic flow that is free of vertices:

$$
\mathsf{ss} \frac{(\xi\{\mathsf{t}\} \wedge \zeta\{a\}) \vee \mathsf{t}}{[\xi\{a\} \vee \zeta\{\mathsf{f}\}] \vee \mathsf{t}}
$$
$$
\Psi \,\Big\|\, \{\mathsf{s},\mathsf{m}\}
$$
$$
(\xi\{a\} \wedge \zeta\{\mathsf{f}\}) \vee \mathsf{t}
$$
.

This construction can be used zero or more times to get the desired Φ, for $h \geq 0$:

$$
(\xi\{\mathsf{t}\} \cdots \{\mathsf{t}\} \wedge \zeta\{a_1\} \cdots \{a_h\}) \vee \mathsf{t}
$$
$$
\Phi \,\Big\|\, \{\mathsf{s},\mathsf{m}\}
$$
$$
(\xi\{a_1\} \cdots \{a_h\} \wedge \zeta\{\mathsf{f}\} \cdots \{\mathsf{f}\}) \vee \mathsf{t}
$$
.

We can now prove the theorem by induction on the number of vertices of a given atomic flow A. The cases where A only has zero or one vertex are trivial. Let us then suppose that A has more than one vertex; then A can be considered as composed of two flows B and C, each with less vertices than A, as follows:
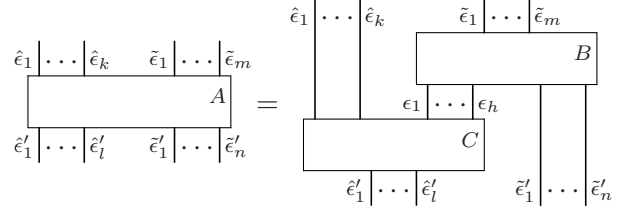
where $h, k, l, m, n \geq 0$ (this can possibly be done in many different ways). By the inductive hypothesis, there exist derivations $\Phi_B \Big\| \begin{matrix} \gamma \\ \zeta\{a_1^{\epsilon_1}\}\cdots\{a_h^{\epsilon_h}\} \end{matrix}$ and $\Phi_C \Big\| \begin{matrix} \xi\{a_1^{\epsilon_1}\}\cdots\{a_h^{\epsilon_h}\} \\ \delta \end{matrix}$ whose flows are, respectively, B and C. Using these, we can build

$$
\begin{matrix}
(\xi\{\mathsf{t}\}\cdots\{\mathsf{t}\} \wedge \gamma) \vee \mathsf{t} \\
(\xi\{\mathsf{t}\}\cdots\{\mathsf{t}\}\wedge\Phi_B)\vee\mathsf{t} \Big\| \\
(\xi\{\mathsf{t}\}\cdots\{\mathsf{t}\} \wedge \zeta\{a_1^{\epsilon_1}\}\cdots\{a_h^{\epsilon_h}\}) \vee \mathsf{t} \\
\Phi \Big\| \\
(\xi\{a_1^{\epsilon_1}\}\cdots\{a_h^{\epsilon_h}\} \wedge \zeta\{\mathsf{f}\}\cdots\{\mathsf{f}\}) \vee \mathsf{t} \\
(\Phi_C\wedge\zeta\{\mathsf{f}\}\cdots\{\mathsf{f}\})\vee\mathsf{t} \Big\| \\
(\delta \wedge \zeta\{\mathsf{f}\}\cdots\{\mathsf{f}\}) \vee \mathsf{t}
\end{matrix}
\quad,
$$

whose flow is A. $\qquad\square$

The derivations in the constructions for the theorem above involve tautologies of the kind $\alpha \vee \mathsf{t}$. There, the 'logical content' of α does not matter because it is trivialized by t, and we are free to build derivations with any premiss and conclusion. A challenging question is whether a derivation exists with a given associated flow and with given premiss and conclusion; in this case, we cannot resort to logical units to trivialize derivations. In the following, in particular in Sections 4 and 5, we reason about derivations whose premiss and conclusion are arbitrary and have to be preserved through transformations of derivations and their associated flows.

3.2. **Paths and Cycles.** We now define the notions of 'ai-path' and 'ai-cycle' in atomic flows. Paths are sequences of adjacent edges that only 'go down' or only 'go up'; ai-paths are formed by joining paths at interaction or cointeraction vertices; ai-cycles are circular ai-paths. We also define the notion of 'simple edge', *i.e.*, an edge connecting an interaction and a cointeraction, as in the first diagram in this paper.

Definition 3.9. Given an atomic flow (V, E, η, up, lo) and $\epsilon_1, \ldots, \epsilon_h \in E$ such that, for $1 \leq i < h$, we have $lo(\epsilon_i) = up(\epsilon_{i+1})$, $up(\epsilon_1) = \nu$ and $lo(\epsilon_h) = \nu'$, we say that $\epsilon_1, \ldots, \epsilon_h$ is a *path from ν to ν'* and that $\epsilon_h, \ldots, \epsilon_1$ is a *path from ν' to ν*; both paths have *length* h. An ai-*path from ν to ν'* of *length* h is either a path from ν to ν' of length h or a sequence of edges $\epsilon_1, \ldots, \epsilon_k, \epsilon_{k+1}, \ldots, \epsilon_h$ such that $\epsilon_k \neq \epsilon_{k+1}$ and, for some $\nu'' \in V$ with $\eta(\nu'') \in \{\mathsf{ai}\!\downarrow, \mathsf{ai}\!\uparrow\}$, we have that $\epsilon_1, \ldots, \epsilon_k$ is an ai-path from ν to ν'' and $\epsilon_{k+1}, \ldots, \epsilon_h$ is an ai-path from ν'' to ν'.

An ai-path of length h is *maximal* if no ai-path containing its edges has length greater than h. An ai-path from (resp., to) ν of length h is a *maximal* ai-*path from* (resp., *to*) ν if no ai-path from (resp., to) ν containing its edges has length greater than h. A path from an interaction to a cointeraction vertex or vice versa is called an ai-*connection*.

Example 3.10. The atomic flow on the left has the ai-paths on the right, and the paths are marked with an asterisk:
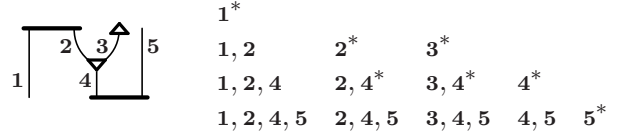
In addition, the flow has the paths and ai-paths obtained from the shown ones by inverting the order of edges, for example $5, 4, 2, 1$ is an ai-path. The ai-paths from the interaction vertex are 1 and 2 and $2, 4$ and $2, 4, 5$; the ai-paths to the contraction vertex are $1, 2$ and 2 and 3 and 4 and $5, 4$; of all the ai-paths to the cointeraction vertex, $2, 4$ is the only ai-connection; the only other ai-connection in the flow is $4, 2$. The maximal ai-paths are $1, 2, 4, 5$ and $3, 4, 5$ and their inverses. The maximal ai-paths from the cointeraction vertex are $4, 2, 1$ and $4, 3$ and 5; the maximal ai-paths to the contraction vertex are $1, 2$ and 3 and $5, 4$.

Simple edges represent immediate causality relations between axioms and cuts. They play a crucial role in the following, in particular when they belong to ai-cycles.

Definition 3.11. An ai-connection consisting of a single edge is called a *simple edge*. A *clean path* is an ai-path where every ai-connection is a simple edge. An ai-*cycle* is an ai-path from a vertex to itself, where no edge appears twice; we do not distinguish ai-cycles that only differ for cyclic permutations of their edges or for inversion, so, if $\epsilon_1, \ldots, \epsilon_h$ is an ai-cycle, then $\epsilon_2, \ldots, \epsilon_h, \epsilon_1$ and $\epsilon_h, \ldots, \epsilon_1$ are the same ai-cycle. A *fragile cycle* is an ai-cycle containing a simple edge. Atomic flows and ai-paths are both called *cycle-free* if they do not contain ai-cycles.

Example 3.12. In the following cycle-free flow all the ai-paths are clean paths; 1, 2 and 3 are ai-connections and simple edges:
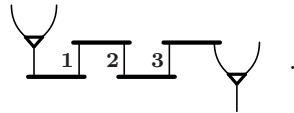
**Example 3.13.** Consider the following atomic flow:
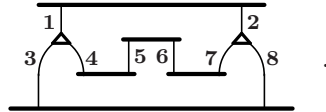
The flow contains two ai-cycles: $1, 4, 5, 6, 7, 2$ and $1, 3, 8, 2$. The first ai-cycle contains the two simple edges 5 and 6, so it is a fragile cycle; the second ai-cycle does not contain any simple edge. Note that the two ai-cycles are 'overlapping', in the sense that edges 1 and 2 belong to both.

**Remark 3.14.** If an ai-path is maximal, then it is cycle-free.

3.3. **Streamlined Derivations.** Intuitively, we can consider interaction and weakening as creators of paths, and we can consider cointeraction and coweakening as their destroyers. We call a derivation 'streamlined' if no path is both created and destroyed.

Definition 3.15. An SKS derivation is *streamlined* if, in its associated atomic flow, there are no paths from interaction or weakening vertices to cointeraction or coweakening vertices.

Remark 3.16. It immediately follows from the definition that the diagram below describes the shape of a streamlined derivation; the boxes stand for flows obtained by freely composing edges and vertices whose labels are only those indicated on the boxes:
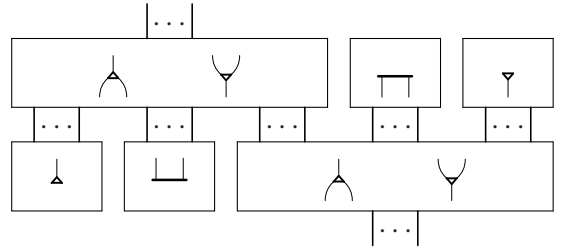
Example 3.17. The first flow is not streamlined, the other two are streamlined:
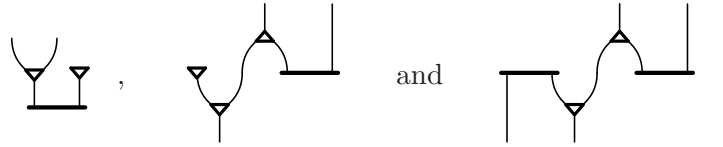
Remark 3.18. A streamlined SKS proof is cut-free. In fact, consider the diagram in Remark 3.16: there can be no ai-cycles, and any maximal ai-path from a cut goes all the way to the top. So, if there are cuts, there must be atoms in the premiss of the derivation, and it cannot be a proof. Note also that a cut-free proof is not necessarily streamlined, as it might have paths from interactions to coweakenings.

Definition 3.19. We say that an algorithm P *streamlines* a derivation Φ if the output of P on Φ is a streamlined derivation that has the same premiss and conclusion as Φ, in the same deductive system.

We know that a Craig interpolant of formulae α and β is a formula γ such that $\alpha \to \gamma$ and $\gamma \to \beta$, and all the atoms in γ appear in α and β. Consider a derivation whose premiss is α and conclusion is β and consider all the ai-paths from atoms of α that are not in β and from atoms of β that are not in α. While, in general, each of these ai-paths can be composed of an arbitrary number of paths, in a streamlined derivation they consist of at most two paths. We would then be tempted to exploit this simplification to try and read interpolants as some intermediate formulae in streamlined derivations. Unfortunately, this does not work so simply, as the following example shows.

Example 3.20. The two following streamlined derivations have the same premiss $\alpha = a \wedge \bar{a}$ and conclusion $\beta = b \vee \bar{b}$ and the same atomic flow (at the centre):

$$
\mathsf{ai}{\uparrow}\frac{a \wedge \bar{a}}{\mathsf{f}}
$$
$$
=\frac{\mathsf{f} \wedge [\mathsf{f} \vee \mathsf{t}]}{}
$$
$$
\mathsf{s}\frac{(\mathsf{f} \wedge \mathsf{f}) \vee \mathsf{t}}{}
$$
$$
=\frac{\mathsf{t}}{}
$$
$$
\mathsf{ai}{\downarrow}\frac{}{b \vee \bar{b}}
$$

$$
=\frac{a \wedge \bar{a}}{(a \wedge \bar{a}) \wedge [\mathsf{f} \vee \mathsf{t}]}
$$
$$
\mathsf{ai}{\downarrow}\frac{}{(a \wedge \bar{a}) \wedge [\mathsf{f} \vee [b \vee \bar{b}]]}
$$
$$
\mathsf{s}\frac{}{((a \wedge \bar{a}) \wedge \mathsf{f}) \vee [b \vee \bar{b}]}
$$
$$
\mathsf{ai}{\uparrow}\frac{}{(\mathsf{f} \wedge \mathsf{f}) \vee [b \vee \bar{b}]}
$$
$$
=\frac{}{b \vee \bar{b}} \quad .
$$

The formulae that appear in the derivation on the left, apart from the premiss and conclusion, are interpolants of α and β. In the derivation on the right, the same is not true.

The results we get in this paper for derivations share some characteristics with Craig's interpolation for formulae: 1) we focus on a normal form that essentially depends on the atoms in common between premiss and conclusion, 2) the cost of getting it is exponential (to the best of our knowledge), and 3) it is intimately related to cut elimination.

We are able to use the methods presented here in order to read interpolants from derivations, so overcoming the problem shown in the previous example. We start from streamlined derivations and then we perform some further constructions; this way, we obtain a normal form such that each inference step of the kind $\mathsf{ai}{\uparrow}$, $\mathsf{aw}{\uparrow}$ or $\mathsf{ac}{\uparrow}$ is above all inference steps of the kind $\mathsf{ai}{\downarrow}$, $\mathsf{aw}{\downarrow}$ or $\mathsf{ac}{\downarrow}$. This normal form is obtained by Kai Brünnler in [Brü06b] by resorting to the sequent calculus, while we can obtain it directly in the calculus of structures. These results are presented in [GG08].

4. Reductions of Atomic Flows

We control normalisation of derivations by manipulating atomic flows, in the sense of graph rewriting. There are two kinds of flow reductions: local and global ones. In local reductions, a bounded subflow in a flow is substituted by another subflow that fits in the context. In global reductions, the entire flow is rewritten: normally, two slightly altered copies of a flow are connected together. In this section, we see local transformations, which are based on reduction rules; in Section 5, we deal with global reductions. It is convenient to classify reduction rules into those for weakening and those for contraction. After seeing flow reductions and tying them with derivations, in Subsection 4.1, we explore some of their basic properties, in the two short Subsections 4.2 and 4.3.

4.1. **Reductions.** We introduce reductions for atomic flows, for which we define a concept of soundness. We will soon see that, corresponding to every sound reduction, there is a transformation on derivations that preserves premiss and conclusion. We will then be able to control complex proof transformations by simple atomic flow transformations.

We start by defining some flow reductions: they are relations $A \to B$ between flows, which we interpret as the possibility of replacing flow A with flow B.

Definition 4.1. In Figure 2, we define graphical expressions of the kind $r\colon A \to B$, where r is a name and A and B are flows.

$w{\downarrow}\text{-}c{\downarrow}:$ [flow] $\rightarrow$ [flow] $1,2$ $c{\uparrow}\text{-}w{\uparrow}:$ [flow] $\rightarrow$ [flow] $1,2$

$w{\downarrow}\text{-}i{\uparrow}:$ [flow] 1 $\rightarrow$ [flow] 1 $i{\downarrow}\text{-}w{\uparrow}:$ [flow] 1 $\rightarrow$ [flow] 1

$w{\downarrow}\text{-}w{\uparrow}:$ [flow] $\rightarrow$

$w{\downarrow}\text{-}c{\uparrow}:$ 1 [flow] 2 $\rightarrow$ 1 [flow] [flow] 2 $c{\downarrow}\text{-}w{\uparrow}:$ 1 [flow] 2 $\rightarrow$ 1 [flow] [flow] 2

$c{\downarrow}\text{-}i{\uparrow}:$ 1 [flow] 2 3 $\rightarrow$ 1 2 3 [flow] $i{\downarrow}\text{-}c{\uparrow}:$ 1 [flow] 2 3 $\rightarrow$ 1 2 3 [flow]

$c{\downarrow}\text{-}c{\uparrow}:$ 1 [flow] 2 / 3 [flow] 4 $\rightarrow$ 1 2 / 3 4 [flow]
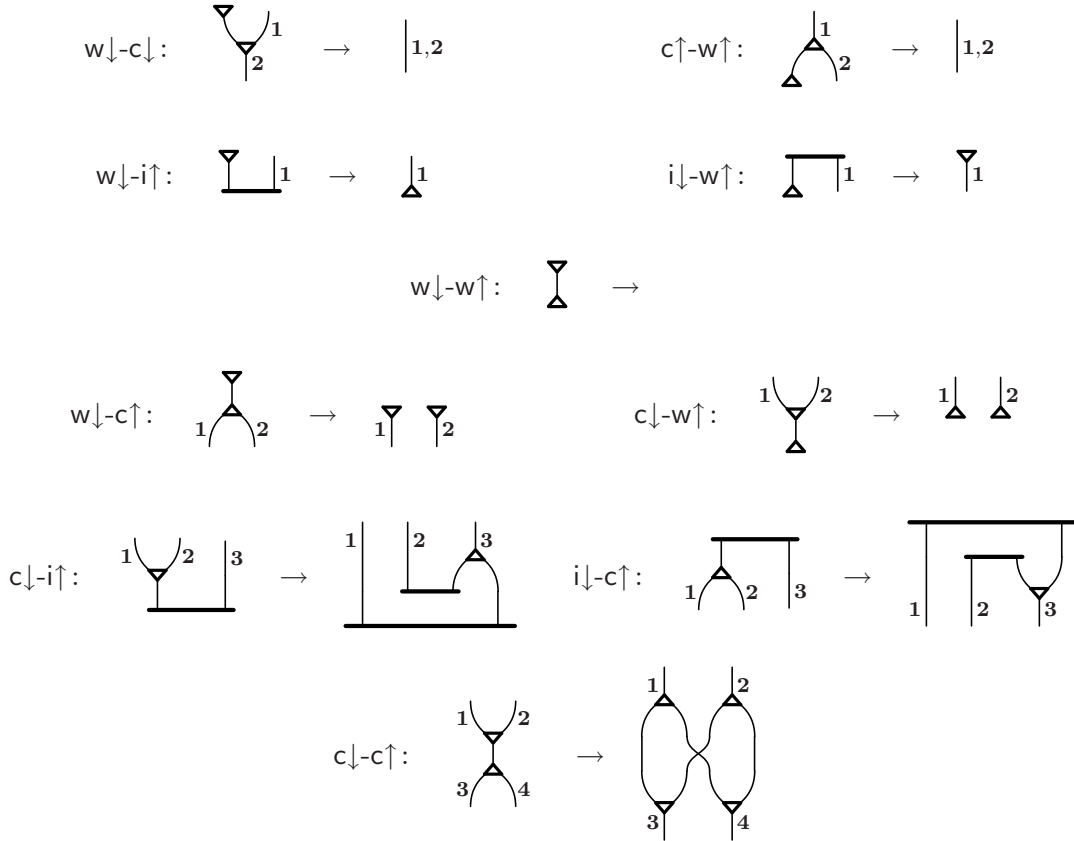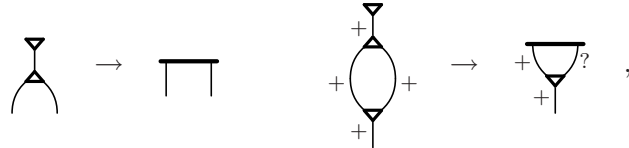
Figure 2: Atomic-flow reduction rules.

We would like to use the reductions in Figure 2 as rules for rewriting inside generic atomic flows. To do so, in general, we should have matching upper and lower edges in the flows that participate in the reduction, and the reductions in the figure clearly do so. However, we also have to pay attention to polarities, not to disrupt atomic flows. In fact, consider the following example.

Example 4.2. The 'reduction' on the left, when used inside a larger atomic flow, might create a situation as on the right:

[flow] $\rightarrow$ [flow] $+$ [flow] $+$ $\rightarrow$ $+$ [flow] $?$,

where the graph at the right is not an atomic flow, for lack of a polarity assignment.

This prompts us to define reduction rules and reductions for atomic flows as follows.

Definition 4.3. An (*atomic-flow*) *reduction rule* r *from flow* A *to flow* B is a quadruple (A, B, f, g) such that:

   (1) f is a one-to-one map from the upper edges of A to the upper edges of B,

(2) g is a one-to-one map from the lower edges of A to the lower edges of B,

(3) for every polarity assignment π for A, there is a polarity assignment π' for B such that $\pi'(f(\epsilon)) = \pi(\epsilon)$ and $\pi'(g(\epsilon')) = \pi(\epsilon')$, for any upper edge ϵ and any lower edge ϵ' of A;

we define reduction rules with graphical expressions $r\colon A \to B$, where f and g are indicated by labelling edges. A binary relation R on the set of atomic flows is called an (*atomic-flow*) *reduction* if, whenever $C\ R\ D$, there is a one-to-one map from the upper edges of C to the upper edges of D and a one-to-one map from the lower edges of C to the lower edges of D. For every reduction rule $r\colon A \to B$, the reduction $\to_r$ is defined, such that $C \to_r D$ if and only if A appears as a subgraph in C and we obtain D by replacing A with B in C, while respecting the correspondence of edges; we call this operation a *reduction by r*.

Remark 4.4. The condition on polarity assignments for a reduction rule r guarantees that the D in $C \to_r D$ is a proper atomic flow, if C is one.

Remark 4.5. Because of the condition on polarity assignments for reduction rules, two distinct connected components in a flow cannot be connected by a reduction. To see that this is impossible, consider the following 'reduction rule', which violates the condition on polarity assignments:

For this 'reduction rule' there exist both valid (left) and invalid (right) polarity assignments:

It is immediate to check:

Proposition 4.6. *The graphical expressions in Figure 2 are atomic-flow reduction rules.*

Our reduction rules bear a striking resemblance to many rewriting systems on graphs, and in particular with interaction nets [Laf97]. It is certainly possible that some interesting connections with other formalisms can be drawn at some point, but, at this time, we are not aware of any. We note that the resemblance might simply be due to there being very few things that we can do with atoms: we can carry them through, delete them or duplicate them, and it is difficult to think of anything else. We might think of making several copies of an atom at once, instead of just two, and this indeed has some uses in the fight against the bureaucracy related to associativity of (co)contraction.

What is peculiar to our work is the fact that reducing flows corresponds to transforming derivations in a very direct way. The correspondence is captured by the notion of soundness, which we now define.

Definition 4.7. A reduction R is *sound* if, for every A and B such that $A\ R\ B$ and for every derivation Φ with atomic flow A, there is a derivation Ψ with atomic flow B such that Φ and Ψ have the same premise and conclusion; in this case we write $\Phi\ R\ \Psi$. A reduction rule r is *sound* if $\to_r$ is sound.

The proof of the following theorem is essentially contained in Figures 3 and 4.

Theorem 4.8. *The reduction rules* w↓-c↓, w↓-i↑, w↓-w↑, w↓-c↑, c↓-i↑, c↓-c↑, c↑-w↑, i↓-w↑, c↓-w↑ *and* i↓-c↑ *are sound.*

$$\mathsf{w}{\downarrow}\text{-}\mathsf{c}{\downarrow}: \quad \rightarrow \quad \Big|\, 1,2 \qquad \mathsf{aw}{\downarrow}\frac{\xi\{\mathsf{f}\}}{\xi\{a^{\mathbf{3}}\}} \quad \Phi \,\Big\| \quad \mathsf{ac}{\downarrow}\frac{\zeta\{a^{\mathbf{3}} \vee a^{\mathbf{1}}\}}{\zeta\{a^{\mathbf{2}}\}} \quad \rightarrow_{\mathsf{w}{\downarrow}\text{-}\mathsf{c}{\downarrow}} \quad \xi\{\mathsf{f}\} \quad \Phi\{a^{\mathbf{3}}{\leftarrow}\mathsf{f}\}\,\Big\| \quad =\frac{\zeta\{\mathsf{f} \vee a^{\mathbf{1},\mathbf{2}}\}}{\zeta\{a^{\mathbf{1},\mathbf{2}}\}}$$

$$\mathsf{w}{\downarrow}\text{-}\mathsf{i}{\uparrow}: \quad \rightarrow \quad \mathsf{aw}{\downarrow}\frac{\xi\{\mathsf{f}\}}{\xi\{a^{\mathbf{2}}\}} \quad \Phi \,\Big\| \quad \mathsf{ai}{\uparrow}\frac{\zeta\{a^{\mathbf{2}} \wedge \bar{a}^{\mathbf{1}}\}}{\zeta\{\mathsf{f}\}} \quad \rightarrow_{\mathsf{w}{\downarrow}\text{-}\mathsf{i}{\uparrow}} \quad \xi\{\mathsf{f}\} \quad \Phi\{a^{\mathbf{2}}{\leftarrow}\mathsf{f}\}\,\Big\| \quad \mathsf{aw}{\uparrow}\frac{\zeta\{\mathsf{f} \wedge \bar{a}^{\mathbf{1}}\}}{\zeta\{\mathsf{f} \wedge \mathsf{t}\}} \quad =\frac{}{\zeta\{\mathsf{f}\}}$$

$$\mathsf{w}{\downarrow}\text{-}\mathsf{w}{\uparrow}: \quad \rightarrow \quad \mathsf{aw}{\downarrow}\frac{\xi\{\mathsf{f}\}}{\xi\{a^{\mathbf{1}}\}} \quad \Phi \,\Big\| \quad \mathsf{aw}{\uparrow}\frac{\zeta\{a^{\mathbf{1}}\}}{\zeta\{\mathsf{t}\}} \quad \rightarrow_{\mathsf{w}{\downarrow}\text{-}\mathsf{w}{\uparrow}} \quad \xi\{\mathsf{f}\} \quad \Phi\{a^{\mathbf{1}}{\leftarrow}\mathsf{f}\}\,\Big\| \quad =\frac{\zeta\{\mathsf{f}\}}{\zeta\{\mathsf{f} \wedge [\mathsf{f} \vee \mathsf{t}]\}} \quad \mathsf{s}\frac{}{\zeta\{(\mathsf{f} \wedge \mathsf{f}) \vee \mathsf{t}\}} \quad =\frac{}{\zeta\{\mathsf{t}\}}$$

$$\mathsf{w}{\downarrow}\text{-}\mathsf{c}{\uparrow}: \quad \rightarrow \quad \mathsf{aw}{\downarrow}\frac{\xi\{\mathsf{f}\}}{\xi\{a^{\mathbf{3}}\}} \quad \Phi \,\Big\| \quad \mathsf{ac}{\uparrow}\frac{\zeta\{a^{\mathbf{3}}\}}{\zeta\{a^{\mathbf{1}} \wedge a^{\mathbf{2}}\}} \quad \rightarrow_{\mathsf{w}{\downarrow}\text{-}\mathsf{c}{\uparrow}} \quad \xi\{\mathsf{f}\} \quad \Phi\{a^{\mathbf{3}}{\leftarrow}\mathsf{f}\}\,\Big\| \quad =\frac{\zeta\{\mathsf{f}\}}{\zeta\{\mathsf{f} \wedge \mathsf{f}\}} \quad \mathsf{aw}{\downarrow}\frac{}{\zeta\{a^{\mathbf{1}} \wedge \mathsf{f}\}} \quad \mathsf{aw}{\downarrow}\frac{}{\zeta\{a^{\mathbf{1}} \wedge a^{\mathbf{2}}\}}$$

Figure 3: 'Downwards' reduction rules for weakening and their soundness.

Proof. For $r \in \{\mathsf{w}{\downarrow}\text{-}\mathsf{c}{\downarrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{i}{\uparrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{c}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{i}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{c}{\uparrow}\}$ and $r\colon A \to B$ as in the left columns of Figures 3 and 4, for every C and D such that $C \to_r D$ and for every Ψ with flow C, the right columns of the tables provide reductions $\Psi \to_r \Psi'$, where Ψ' has flow D, as follows. If $\Phi' \to_r \Phi''$ is the reduction provided by the table, then

$$\Psi = \begin{array}{c} \alpha \\ \Psi_1 \,\Big\| \\ \alpha' \\ \Phi' \,\Big\| \\ \beta' \\ \Psi_2 \,\Big\| \\ \beta \end{array} \qquad \text{and} \qquad \Psi' = \begin{array}{c} \alpha \\ \Psi_1 \,\Big\| \\ \alpha' \\ \Phi'' \,\Big\| \\ \beta' \\ \Psi_2 \,\Big\| \\ \beta \end{array} \quad .$$

We can deal with the remaining rules by employing dual derivations to the ones shown. □

c↓-i↑: (atomic flow diagram) → (atomic flow diagram)

$$ac\downarrow\frac{\xi\{a^{\mathbf{1}} \vee a^{\mathbf{2}}\}}{\xi\{a^{\mathbf{4}}\}}$$
$$\Phi \,\Big\|$$
$$ai\uparrow\frac{\zeta\{a^{\mathbf{4}} \wedge \bar{a}^{\mathbf{3}}\}}{\zeta\{f\}}$$

$$\rightarrow_{c\downarrow\text{-}i\uparrow}$$

$$\xi\{a^{\mathbf{1}} \vee a^{\mathbf{2}}\}$$
$$\Phi\{a^{\mathbf{4}}\leftarrow a\vee a\} \,\Big\|$$
$$ac\uparrow\frac{\zeta\{[a^{\mathbf{1}} \vee a^{\mathbf{2}}] \wedge \bar{a}^{\mathbf{3}}\}}{\zeta\{[a^{\mathbf{1}} \vee a^{\mathbf{2}}] \wedge (\bar{a} \wedge \bar{a})\}}$$
$$=\frac{}{\zeta\{(\bar{a} \wedge [a^{\mathbf{2}} \vee a^{\mathbf{1}}]) \wedge \bar{a}\}}$$
$$s\frac{}{\zeta\{[(\bar{a} \wedge a^{\mathbf{2}}) \vee a^{\mathbf{1}}] \wedge \bar{a}\}}$$
$$ai\uparrow\frac{}{\zeta\{[f \vee a^{\mathbf{1}}] \wedge \bar{a}\}}$$
$$=\frac{}{\zeta\{a^{\mathbf{1}} \wedge \bar{a}\}}$$
$$ai\uparrow\frac{}{\zeta\{f\}}$$

c↓-c↑: (atomic flow diagram) → (atomic flow diagram)

$$ac\downarrow\frac{\xi\{a^{\mathbf{1}} \vee a^{\mathbf{2}}\}}{\xi\{a^{\mathbf{5}}\}}$$
$$\Phi \,\Big\|$$
$$ac\uparrow\frac{\zeta\{a^{\mathbf{5}}\}}{\zeta\{a^{\mathbf{3}} \wedge a^{\mathbf{4}}\}}$$

$$\rightarrow_{c\downarrow\text{-}c\uparrow}$$

$$\xi\{a^{\mathbf{1}} \vee a^{\mathbf{2}}\}$$
$$\Phi\{a^{\mathbf{5}}\leftarrow a\vee a\} \,\Big\|$$
$$ac\uparrow\frac{\zeta\{a^{\mathbf{1}} \vee a^{\mathbf{2}}\}}{\zeta\{a^{\mathbf{1}} \vee (a \wedge a)\}}$$
$$ac\uparrow\frac{}{\zeta\{(a \wedge a) \vee (a \wedge a)\}}$$
$$m\frac{}{\zeta\{[a \vee a] \wedge [a \vee a]\}}$$
$$ac\downarrow\frac{}{\zeta\{a^{\mathbf{3}} \wedge [a \vee a]\}}$$
$$ac\downarrow\frac{}{\zeta\{a^{\mathbf{3}} \wedge a^{\mathbf{4}}\}}$$

Figure 4: 'Downwards' reduction rules for contraction and their soundness.

Remark 4.9. The previous soundness theorem only depends on the switch and medial rules for the reductions in Figure 4. Any system obtained from SKS by replacing s and m with linear rules that can derive them would support a soundness theorem like the one above, for the same reduction rules. For example, we could think of replacing s with the rule s$'\dfrac{[a \vee b] \wedge [c \vee d]}{(a \wedge c) \vee [b \vee d]}$, from which s is derivable.

Definition 4.10. A finite set of reduction rules is a *flow rewriting system*. For every flow rewriting system $F = \{r_1, \ldots, r_h\}$ we define $\rightarrow_F \,=\, \rightarrow_{r_1} \cup \cdots \cup \rightarrow_{r_h}$. The reflexive transitive closure of $\rightarrow_F$ is denoted by $\rightarrow_F^\star$. Given a set of atomic flows S, we say that a flow rewriting system F is *terminating on S* if there is no infinite chain $A_1 \rightarrow_F A_2 \rightarrow_F \cdots$, for every $A_1 \in S$; if F is terminating on the set of atomic flows, we say that it is *terminating*. We say that atomic flow A is *normal* for flow rewriting system F if there is no atomic flow B such that $A \rightarrow_F B$.

4.2. **Weakening and Coweakening.** The reduction rules for weakening and coweakening make for a very simple flow rewriting system. They are very 'friendly' rules, because they greatly simplify atomic flows and associated derivations.

Definition 4.11. The following flow rewriting system is called w:

$$\{\ w\downarrow\text{-}c\downarrow\ ,\ c\uparrow\text{-}w\uparrow\ ,\ w\downarrow\text{-}i\uparrow\ ,\ i\downarrow\text{-}w\uparrow\ ,\ w\downarrow\text{-}w\uparrow\ ,\ w\downarrow\text{-}c\uparrow\ ,\ c\downarrow\text{-}w\uparrow\ \}\quad .$$

Theorem 4.12. *Flow rewriting system* w *is terminating.*

Proof. At every reduction, either the number of vertices decreases, or it stays the same but the number of contraction and cocontraction vertices decreases. ☐

Remark 4.13. If flow A is normal for w, then there is no ai-path from a weakening or coweakening vertex to another vertex in A.

Since reducing by w does not introduce new edges, we have:

Proposition 4.14. *If A is cycle-free and $A \to_{\mathsf{w}}^{\star} B$ then B is cycle-free.*

4.3. **Contraction and Cocontraction.** The reduction rules for contraction and cocontraction are much less 'friendly' than weakening/coweakening ones, mainly because they create infinite reduction chains. A judicious use of these rules is the key to success for our normalisation methods.

Definition 4.15. The following flow rewriting system is called c:

$$\{\; \mathsf{c}{\downarrow}\text{-}\mathsf{i}{\uparrow} \;,\; \mathsf{i}{\downarrow}\text{-}\mathsf{c}{\uparrow} \;,\; \mathsf{c}{\downarrow}\text{-}\mathsf{c}{\uparrow} \;\} \quad .$$

Remark 4.16. Flow rewriting system c is not terminating:
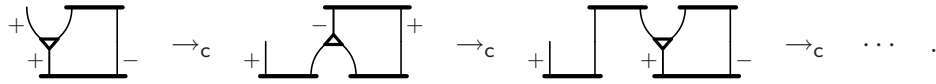
We see that if a contraction vertex belongs to an ai-cycle, reductions by c make it 'bounce' in the ai-cycle and create a trail; while bouncing, the vertex alternates between contraction and cocontraction; if we assign a polarity to the flow, the vertex alternates between being positive and negative.

Through a simple argument by contradiction, we have:

Proposition 4.17. *If A is cycle-free and $A \to_{\mathsf{c}}^{\star} B$ then B is cycle-free.*

Again, reasoning by contradiction, we have:

Proposition 4.18. *If an atomic flow is normal for* c *then all its* ai*-paths are clean paths.*

The previous proposition could be rephrased by saying that if an atomic flow is normal for c then all its ai-connections are simple edges.

Since reducing by w does not introduce new vertices, we have:

Proposition 4.19. *If A is normal for* c *and $A \to_{\mathsf{w}}^{\star} B$ then B is normal for* c*.*

By contradiction and a simple case analysis, we have:

Proposition 4.20. *If A is normal for* w *and $A \to_{\mathsf{c}}^{\star} B$ then B is normal for* w*.*

Maximal ai-paths provide for a measure when dealing with the termination of c.

Remark 4.21. A simple inspection to the reduction rules of c convinces us that reducing by c does not change the number and length of the maximal ai-paths of a flow. The same holds for the maximal ai-paths to or from vertices that are not involved in a given reduction.

Theorem 4.22. *Flow rewriting system* c *is terminating on the set of cycle-free atomic flows.*

Proof. Let A be a cycle-free flow. We associate to each contraction (resp., cocontraction) vertex ν its *rank* $\mathsf{r}_\nu = \sum_{p_i \in I_\nu} h_i$, where I_ν is the set of all maximal ai-paths $p_i = \epsilon_1^i, \ldots, \epsilon_{h_i}^i$ from ν, such that ϵ_1^i is the lower (resp., upper) edge of ν (so, the rank of a vertex is the sum of the lengths of certain maximal ai-paths from it). Note that every (co)contraction vertex has non-zero rank. We prove that a reduction of A by c decreases the sum of the ranks of the (co)contraction vertices of A. First note that the rank of the vertices not involved in the reduction step stays the same (see Remark 4.21). We then need to show that the sum of the ranks decreases for the vertices involved. There are three cases, depending on the reduction rule:

$\mathsf{c}\!\downarrow\text{-}\mathsf{i}\!\uparrow$: a contraction vertex ν is replaced by a cocontraction vertex ν', and $\mathsf{r}_{\nu'} = \mathsf{r}_\nu - n$, where $n > 0$ is the number of maximal ai-paths from ν whose first edge is the lower edge of ν;

$\mathsf{i}\!\downarrow\text{-}\mathsf{c}\!\uparrow$: this is dual to the previous case;

$\mathsf{c}\!\downarrow\text{-}\mathsf{c}\!\uparrow$: a contraction vertex ν and a cocontraction vertex ν' are replaced by two contraction vertices ν_1 and ν_2 and two cocontraction vertices ν_1' and ν_2'; we have $\mathsf{r}_{\nu_1} + \mathsf{r}_{\nu_2} = \mathsf{r}_\nu - n$, where $n > 0$ is the number of maximal ai-paths from ν whose first edge is the lower edge of ν; analogously, we have $\mathsf{r}_{\nu_1'} + \mathsf{r}_{\nu_2'} = \mathsf{r}_{\nu'} - n'$, where $n' > 0$ is the number of maximal ai-paths from ν' whose first edge is the upper edge of ν'. $\qquad\square$

Remark 4.23. Normalising by c can blow the size of atomic flows exponentially, in particular in a situation like the following (noted by Lutz Straßburger):
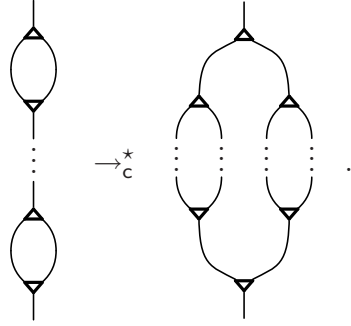
In fact, if there are n couples cocontraction/contraction like the two shown above on the left, then there are 2^n maximal ai-paths, and their number (and length) is conserved by $\to_\mathsf{c}^\star$ (see Remark 4.21). Exactly one ai-path passes through each edge in the middle portion of the flow on the right.

Normalising flows via the rewriting system c is important, and we know from Remark 4.16 and Theorem 4.22 that we can only have termination for cycle-free atomic flows. It turns out that we can 'break' ai-cycles if we manage to move contractions and cocontractions away from at least one ai-connection per ai-cycle, so to create a simple edge. As we saw in Definition 3.11, we call the ai-cycles exhibiting this property 'fragile'. We now see how to transform every ai-cycle into a fragile cycle, and Subsection 5.2 shows how to break fragile cycles.

Theorem 4.24. *For every atomic flow A there is an atomic flow B such that $A \to_\mathsf{c}^\star B$ and all the ai-cycles in B are fragile cycles.*

Proof. Given any polarity assignment for A, consider all contractions and cocontractions with some edge belonging to an ai-cycle and mapping to $-$. Apply the rules of c to these

vertices until they change polarity. Now, every ai-cycle contains at least one ai-connection mapping to −, and this cannot be anything else than a simple edge. □

Example 4.25. The negative ai-connection $1, 2, 3$ can be made into a negative simple edge:
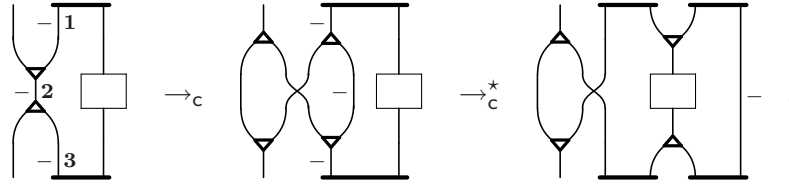
Example 4.26. The following atomic flow reduction (also present in Figure 6 on page 30) shows another application of the previous theorem:
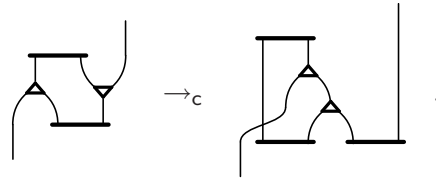

5. Streamlining Algorithms

In this section we prove our main results. The basis of our normalisation technique is the elimination of a simple edge, which means eliminating also the axiom and cut it connects. This entails a duplication of the entire flow/derivation, and the repeated process generates an exponential growth in size. Of course, this is what we expect from an algorithm that generalises cut elimination for propositional logic.

We start by studying a single elimination of a simple edge. We will then study two algorithms, based on simple-edge elimination, that, in combination, achieve the desired normalisation result. The final part of this section presents two normalisation algorithms, of varying strength, and discusses possible variations.

5.1. **Elimination of a Simple Edge.** Definition 5.1 contains a graphical representation of the main idea of this section. It is a reduction of atomic flows and associated derivations, whose purpose is eliminating one simple edge, and at the same time removing one interaction and one cointeraction vertex. Contrary to previous reduction rules, we are dealing here with a global transformation of atomic flows and associated derivations: a reduction can involve an entire flow/derivation and not just a local subflow/subderivation.

The reduction of simple edges that we are about to define is not strictly necessary for getting our results, because we could jump directly to the more general Definition 5.6. However, it is important to understand this reduction in isolation, both because it is simpler, and also because it provides the basis for different reduction strategies from those that we discuss in this paper.

Definition 5.1. We define the reduction $\rightarrow_{\mathsf{se}}$ (where se stands for *simple edge*) as follows, for every atomic flow A:
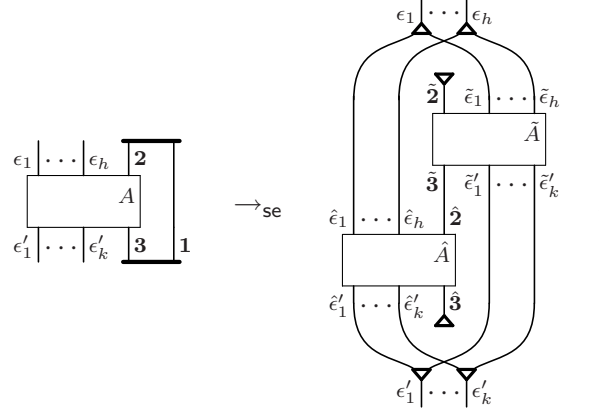
where $h, k \geq 0$, edges have been renamed with $\hat{\ }$ and $\tilde{\ }$ accents, flows $\tilde{A}$ and $\hat{A}$ are both isomorphic to A, and edges $\hat{\mathbf{2}}$ and $\tilde{\mathbf{3}}$ are identified.

A simple inspection of the definition of $\rightarrow_{\mathsf{se}}$ suffices to prove the following statement, about $\rightarrow_{\mathsf{se}}$ not introducing any ai-cycles.

Proposition 5.2. *If atomic flow B is cycle-free and $B \rightarrow_{\mathsf{se}} C$, then C is cycle-free.*

Theorem 5.3. *Reduction $\rightarrow_{\mathsf{se}}$ is sound.*

Proof. Let Φ be a derivation with flow B, such that $B \rightarrow_{\mathsf{se}} C$. We show that there exists a derivation Ψ with flow C and with the same premiss and conclusion as Φ. In the following, we refer to the figure in Definition 5.1. We assume that Φ has premiss $\xi\{\mathsf{t}^\bullet\}$ and conclusion $\zeta\{\mathsf{f}^\bullet\}$, where the evidenced and labelled $\mathsf{t}^\bullet$ and $\mathsf{f}^\bullet$ can be traced to the interaction and cointeraction vertices eliminated by $\rightarrow_{\mathsf{se}}$, respectively (this can always be done by using switches and unit equations). Intuitively, we can think of $\mathsf{t}^\bullet$ and $\mathsf{f}^\bullet$ as mapping to special 'unit edges', which can be substituted just like normal edges. So, we assume that Φ is

$$
\begin{array}{c}
\xi\{\mathsf{t}^\bullet\} \\
\Phi_1 \, \big\| \\
\mathsf{ai}{\downarrow} \dfrac{\xi'\{\mathsf{t}^\bullet\}}{\xi'\{\bar{a}^{\mathbf{2}} \vee a^{\mathbf{1}}\}} \\
\Phi_2 \, \big\| \\
\mathsf{ai}{\uparrow} \dfrac{\zeta'\{\bar{a}^{\mathbf{3}} \wedge a^{\mathbf{1}}\}}{\zeta'\{\mathsf{f}^\bullet\}} \\
\Phi_3 \, \big\| \\
\zeta\{\mathsf{f}^\bullet\}
\end{array}
\quad .
$$

We obtain the two derivations Ψ' and Ψ'' from Φ as follows:

$$
\Psi' = \quad
\begin{array}{c}
\xi\{t\} \\
\Phi_1 \, \big\| \\
= \dfrac{\xi'\{t\}}{\xi'\{f \vee t\}} \\
\mathsf{aw}{\downarrow} \, \dfrac{\ }{\xi'\{\bar{a}^{\tilde{\mathbf{2}}} \vee t\}} \\
\Phi_2\{a^{\mathbf{1}} \leftarrow t\} \, \big\| \\
= \dfrac{\zeta'\{\bar{a}^{\tilde{\mathbf{3}}} \wedge t\}}{\zeta'\{\bar{a}^{\tilde{\mathbf{3}}}\}} \\
\Phi_3\{f^{\bullet} \leftarrow \bar{a}\} \, \big\| \\
\zeta\{\bar{a}^{\tilde{\mathbf{3}}}\}
\end{array}
\qquad \text{and} \qquad
\Psi'' = \quad
\begin{array}{c}
\xi\{\bar{a}^{\hat{\mathbf{2}}}\} \\
\Phi_1\{t^{\bullet} \leftarrow \bar{a}\} \, \big\| \\
= \dfrac{\xi'\{\bar{a}^{\hat{\mathbf{2}}}\}}{\xi'\{\bar{a}^{\hat{\mathbf{2}}} \vee f\}} \\
\Phi_2\{a^{\mathbf{1}} \leftarrow f\} \, \big\| \\
\mathsf{aw}{\uparrow} \, \dfrac{\zeta'\{\bar{a}^{\hat{\mathbf{3}}} \wedge f\}}{\zeta'\{t \wedge f\}} \\
= \dfrac{\ }{\zeta'\{f\}} \\
\Phi_3 \, \big\| \\
\zeta\{f\}
\end{array}
\quad .
$$

Derivation Ψ' has flow B' and Ψ'' has B'':

$$
B' = \quad
\begin{array}{c}
\tilde{\mathbf{2}}\,\blacktriangledown \; \tilde{\epsilon}_1 \big| \cdots \big| \tilde{\epsilon}_h \\
\boxed{\tilde{A}} \\
\tilde{\mathbf{3}} \; \tilde{\epsilon}'_1 \big| \cdots \big| \tilde{\epsilon}'_k
\end{array}
\qquad \text{and} \qquad
B'' = \quad
\begin{array}{c}
\hat{\epsilon}_1 \big| \cdots \big| \hat{\epsilon}_h \; \big|\hat{\mathbf{2}} \\
\boxed{\hat{A}} \\
\hat{\epsilon}'_1 \big| \cdots \big| \hat{\epsilon}'_k \; \big|\hat{\mathbf{3}}\,\blacktriangle
\end{array}
\quad .
$$

We combine Ψ' and Ψ'' to get the desired derivation Ψ with flow C and the same premiss and conclusion as Φ:

$$
\Psi = \quad
\begin{array}{c}
\mathsf{c}{\uparrow} \, \dfrac{\xi\{t\}}{\xi\{t\} \wedge \xi\{t\}} \\
\xi\{t\} \wedge \Psi' \, \big\| \\
\mathsf{ss} \, \dfrac{\xi\{t\} \wedge \zeta\{\bar{a}\}}{\xi\{\bar{a}\} \vee \zeta\{f\}} \\
\Psi'' \vee \zeta\{f\} \, \big\| \\
\mathsf{c}{\downarrow} \, \dfrac{\zeta\{f\} \vee \zeta\{f\}}{\zeta\{f\}}
\end{array}
\quad ,
$$

where ss, $\mathsf{c}{\downarrow}$ and $\mathsf{c}{\uparrow}$ are 'macro' rules introduced in Remarks 2.5 and 2.6. $\qquad \square$

The reduction $\to_{\mathsf{se}}$ on atomic flows is symmetric and deterministic, and this might be a surprise, because we tend to expect a non-deterministic choice in the elimination of a cut in classical propositional logic. The corresponding construction on derivations might appear symmetric and deterministic, too, but, subtly, it is not. In fact, we have two ways of realising the macro rule $\mathsf{ss} \, \dfrac{\xi\{t\} \wedge \zeta\{\alpha\}}{\xi\{\alpha\} \vee \zeta\{f\}}$ by using s; informally, either we 'put $\xi\{t\}$ inside $\zeta\{\ \}$' or we 'put $\zeta\{\alpha\}$ inside $\xi\{\ \}$' (see Remark 2.5). Note, however, that $\mathsf{s} \, \dfrac{\beta \wedge [\alpha \vee \gamma]}{(\beta \wedge \alpha) \vee \gamma}$ is a special case of ss (modulo $=$); so, we might replace s with ss inside SKS, and we would eliminate this non-determinism. Contrary to s, the rule ss is not local (in the sense that it cannot be checked in time bounded by a constant); we do not know whether a local rule exists that could make for a symmetric and deterministic derivation corresponding to flows obtained by $\to_{\mathsf{se}}$.

Repeated $\to_{\mathsf{se}}$ reductions can lead to infinite chains, because at each step a flow is duplicated. In fact, if the flow contains more than one simple edge the number of simple edges after each $\to_{\mathsf{se}}$ reduction increases.

5.2. **Breaking Fragile Cycles and Eliminating Clean Paths.** Since indiscriminately composing $\to_{\mathsf{se}}$ reductions might lead to infinite chains, in order to normalise proofs, we have to impose some discipline on $\to_{\mathsf{se}}$ reductions. Moreover, it is not guaranteed that $\to_{\mathsf{se}}$ can achieve normalisation without preliminarily 'preparing' the atomic flow. All that said, we are interested in as free as possible reduction mechanisms.

It turns out that we can achieve our objectives by defining two similar, recursive reductions based on $\to_{\mathsf{se}}$. The main idea is to constrain $\to_{\mathsf{se}}$ into a very simple binary recursion scheme, controlled by two different conditions on simple edges. These conditions provide induction measures that basically amount to counting the number of simple edges in the flow to reduce. As we are about to see, we are able to keep the symmetry already exhibited by $\to_{\mathsf{se}}$.

The following two reductions, $\to_{\mathsf{bc}}$ and $\to_{\mathsf{ex}}$, should be understood as 'one-shot' reductions, meaning that they will only be needed once each in the normalisation process. They consist, basically, of chains of $\to_{\mathsf{se}}$ reductions. In order to define $\to_{\mathsf{ex}}$, we need the notion of 'extremal simple edge'.

Definition 5.4. If $\epsilon_1, \ldots, \epsilon_k, \ldots, \epsilon_h$ is a maximal clean path (*i.e.*, a clean path that is also a maximal ai-path), ϵ_k is a simple edge and the edges ϵ_{k+1}, $\ldots$, ϵ_h are not simple edges, then ϵ_k is an *extremal simple edge*.

Remark 5.5. In every maximal clean path there are at most two extremal simple edges, one for each 'direction of the path'. In Example 3.12 (page 12) there are two maximal clean paths, and edges 1 and 3 are the extremal simple edges of both.

Definition 5.6. We inductively define the reductions $\to_{\mathsf{bc}}$ and $\to_{\mathsf{ex}}$ (where bc stands for *break* ai-*cycles* and ex for *eliminate extremal simple edges*) as follows. Given a flow B, the base cases are:

- if there are no fragile cycles in B then $B \to_{\mathsf{bc}} B$;
- if there are no extremal simple edges in B then $B \to_{\mathsf{ex}} B$.

For the inductive cases, let us suppose that
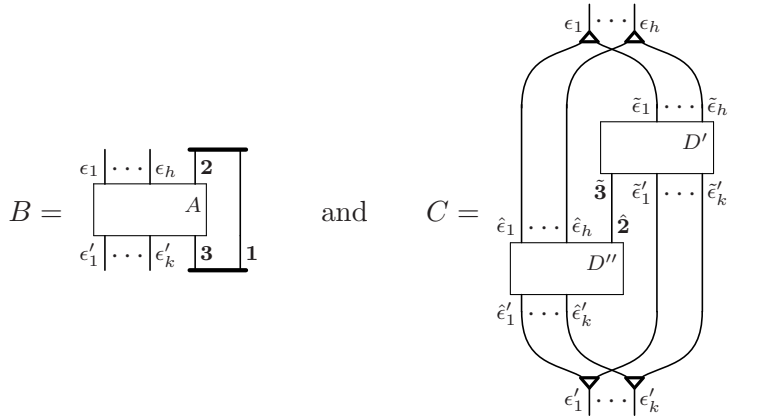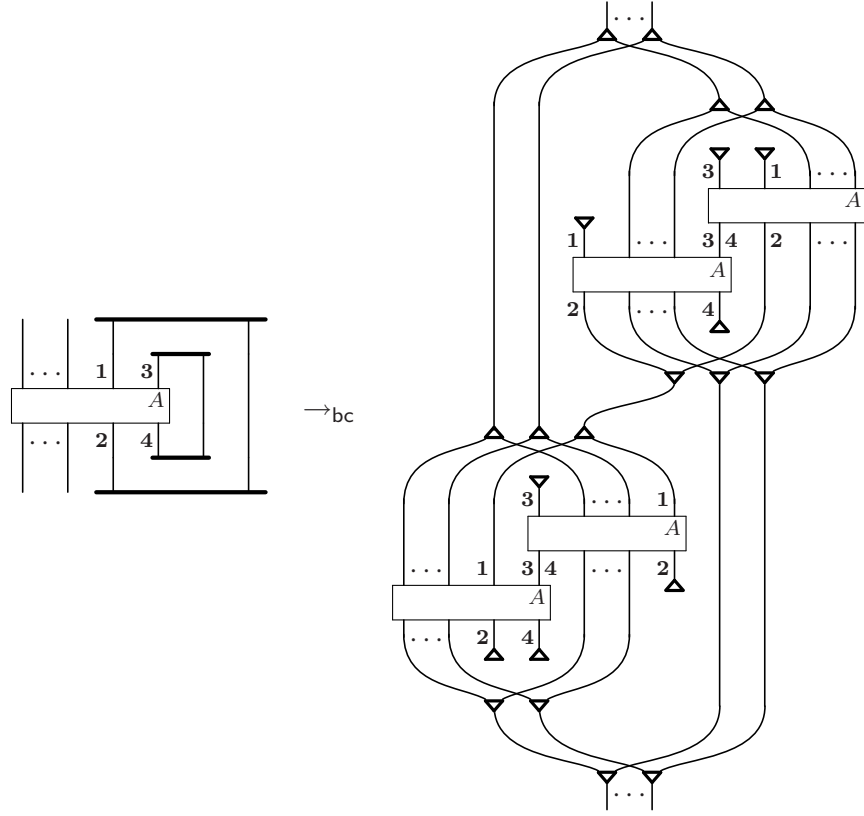
Figure 5: Example of a two-step $\rightarrow_{bc}$ (or $\rightarrow_{ex}$) reduction.

where $h, k \geq 0$, and let

$$B' = \boxed{\tilde{A}} \qquad \text{and} \qquad B'' = \boxed{\hat{A}} \quad ,$$

where the correspondence of edges has been indicated by adding accents to their labels. We have that:

- if 1 is an edge belonging to an ai-cycle, $B' \rightarrow_{bc} D'$ and $B'' \rightarrow_{bc} D''$ then $B \rightarrow_{bc} C$;
- if 1 is an extremal simple edge, $B' \rightarrow_{ex} D'$ and $B'' \rightarrow_{ex} D''$ then $B \rightarrow_{ex} C$.

Example 5.7. Consider the atomic flow to the left in Figure 5. Assuming that the two evidenced simple edges both belong to ai-cycles and that the box A stands for a cycle-free flow, then the atomic flow on the right is the result of a $\rightarrow_{bc}$ reduction. Similarly, if the two evidenced simple edges are extremal simple edges, and the box stands for a flow that contains no simple edges, then the atomic flow on the right is the result of a $\rightarrow_{ex}$ reduction.

Notice that the flow in Figure 5 represents the 'external' shape of any flow after eliminating any two simple edges. Eliminating more simple edges would follow the same pattern.

Remark 5.8. It is possible to generalise the construction in Figure 5 to any number n of simple edges: for any n, there is an atomic flow of the same nature as the one at the right

of the figure, with 2^n boxes. So, a $\rightarrow_{bc}$ or $\rightarrow_{ex}$ reduction can be 'executed' in one step if the simple edges involved, and their order, is known in advance.

The following two theorems guarantee properties of flows after reducing by $\rightarrow_{bc}$ and $\rightarrow_{ex}$, provided that the flow to which we apply them meets some conditions. These conditions can be achieved by a careful use of the flow rewriting system c for contractions, as we see later on.

Theorem 5.9. *If all the* ai*-cycles in atomic flow B are fragile cycles then there exists a cycle-free flow C such that $B \rightarrow_{bc} C$.*

Proof. By induction on the number of ai-cycles: follow Definition 5.6 and note that when composing D' and D'' in C, no ai-cycles are created. □

Example 5.10. Notice that removing one simple edge might break more than one ai-cycle. The following flows have, respectively, two, three and two fragile cycles. Removing edges 1, 2, 3 or 4 breaks two ai-cycles each:
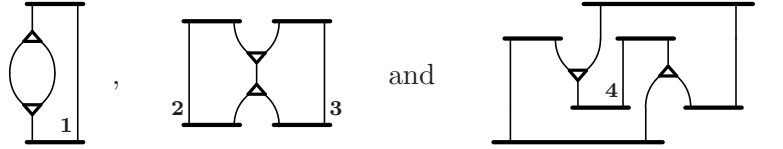
Remark 5.11. Reductions $\rightarrow_{se}$ and $\rightarrow_{bc}$ can introduce new simple edges along the edge labelled $\hat{2}$ and $\tilde{3}$ in Definitions 5.1 and 5.6. This phenomenon is the reason for having introduced the notion of extremal simple edge. As an example, consider the following application of $\rightarrow_{se}$ to a non-extremal simple edge. The problem occurs when the two copies of the given atomic flow (neither of which contains a simple edge) are combined, and the edge that connects them becomes a simple edge:
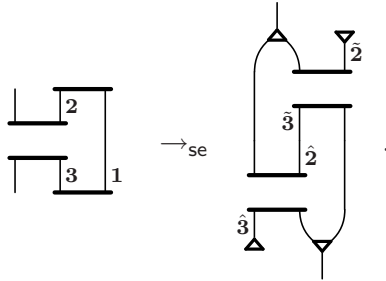
Reducing over extremal simple edges avoids the problem.

The algorithms we show in this paper would terminate even if we did not insist on reducing over extremal simple edges, but doing so simplifies the induction measure.

Theorem 5.12. *If all the* ai*-paths in cycle-free atomic flow B are clean paths then there exists a flow C such that no* ai*-connections appear in it and $B \rightarrow_{ex} C$.*

Proof. By induction on the number of simple edges. We follow Definition 5.6 and its notation. We have to verify that, when composing D' and D'' in C, no ai-connections are created. This could only happen if, in A, edge 2 were upper edge of a cointeraction and edge 3 were lower edge of an interaction; this is impossible, because 1 is an extremal simple edge. □

Theorem 5.13. *Reductions* $\rightarrow_{\mathsf{bc}}$ *and* $\rightarrow_{\mathsf{ex}}$ *are sound.*

Proof. The proof is almost identical for $\rightarrow_{\mathsf{bc}}$ and $\rightarrow_{\mathsf{ex}}$, so, to fix ideas, we prove the theorem for $\rightarrow_{\mathsf{bc}}$, by induction on its definition. The base case is trivial. For the inductive case, the construction is similar to the one in the proof of Theorem 5.3, and we see it briefly. Consider

$$
\Phi =
\begin{array}{c}
\xi\{t^\bullet\} \\
\Phi_1 \, \| \\
\mathsf{ai}{\downarrow} \dfrac{\xi'\{t^\bullet\}}{\xi'\{\bar{a}^2 \vee a^1\}} \\
\Phi_2 \, \| \\
\mathsf{ai}{\uparrow} \dfrac{\zeta'\{\bar{a}^3 \wedge a^1\}}{\zeta'\{f^\bullet\}} \\
\Phi_3 \, \| \\
\zeta\{f^\bullet\}
\end{array}
\quad,\qquad
\Psi' =
\begin{array}{c}
\xi\{t\} \\
\Phi_1 \, \| \\
= \dfrac{\xi'\{t\}}{\xi'\{f \vee t\}} \\
\mathsf{aw}{\downarrow} \dfrac{\xi'\{f \vee t\}}{\xi'\{\bar{a}^{\tilde{2}} \vee t\}} \\
\Phi_2\{a^1 \leftarrow t\} \, \| \\
= \dfrac{\zeta'\{\bar{a}^{\tilde{3}} \wedge t\}}{\zeta'\{\bar{a}^{\tilde{3}}\}} \\
\Phi_3\{f^\bullet \leftarrow \bar{a}\} \, \| \\
\zeta\{\bar{a}^{\tilde{3}}\}
\end{array}
\qquad\text{and}\qquad
\Psi'' =
\begin{array}{c}
\xi\{\bar{a}^{\hat{2}}\} \\
\Phi_1\{t^\bullet \leftarrow \bar{a}\} \, \| \\
= \dfrac{\xi'\{\bar{a}^{\hat{2}}\}}{\xi'\{\bar{a}^{\hat{2}} \vee f\}} \\
\Phi_2\{a^1 \leftarrow f\} \, \| \\
\mathsf{aw}{\uparrow} \dfrac{\zeta'\{\bar{a}^{\hat{3}} \wedge f\}}{\zeta'\{t \wedge f\}} \\
= \dfrac{\zeta'\{t \wedge f\}}{\zeta'\{f\}} \\
\Phi_3 \, \| \\
\zeta\{f\}
\end{array}
\quad,
$$

where Φ has flow B, such that $B \rightarrow_{\mathsf{bc}} C$, and Ψ' and Ψ'' are obtained from Φ and have flows B' and B'', respectively, as per Definition 5.6. By induction hypothesis, there are derivations $\hat{\Psi}'$ and $\hat{\Psi}''$ with flows D' and D'' such that $B' \rightarrow_{\mathsf{bc}} D'$ and $B'' \rightarrow_{\mathsf{bc}} D''$ and with the same premises and conclusions as Ψ and Ψ'. We can compose these derivations into the following, whose flow is C:

$$
\begin{array}{c}
\mathsf{c}{\uparrow} \dfrac{\xi\{t\}}{\xi\{t\} \wedge \xi\{t\}} \\
\xi\{t\} \wedge \hat{\Psi}' \, \| \\
\mathsf{ss} \dfrac{\xi\{t\} \wedge \zeta\{\bar{a}\}}{\xi\{\bar{a}\} \vee \zeta\{f\}} \\
\hat{\Psi}'' \vee \zeta\{f\} \, \| \\
\mathsf{c}{\downarrow} \dfrac{\zeta\{f\} \vee \zeta\{f\}}{\zeta\{f\}}
\end{array}
\quad.
$$

$\square$

Remark 5.14. Similarly to what we do in Remark 4.9, we observe here that the previous soundness theorem holds for any proof system containing the same structural rules as SKS and whose logical rules are such that the $\mathsf{c}{\uparrow}$, ss and $\mathsf{c}{\downarrow}$ rules are derivable. So, the soundness theorem depends only very loosely on the choice of logical rules.

Remark 5.15. After a $\rightarrow_{\mathsf{bc}}$ or $\rightarrow_{\mathsf{ex}}$ reduction, the size of an atomic flow grows by a factor of $O(2^n)$, where n is the number of simple edges involved in the reduction (see Remark 5.8). This accounts for an equally exponential growth in the corresponding derivation.

Out of the two reductions $\rightarrow_{\mathsf{bc}}$ and $\rightarrow_{\mathsf{ex}}$, we define two algorithms, BC and EX, which use flow rewriting system c to achieve the necessary preconditions for the two reductions.

Definition 5.16. For every SKS derivation Φ_0 with atomic flow A_0, *algorithm* BC (for *break* ai-*cycles*) performs the following steps:

(1) *Make* ai-*cycles fragile.* Transform Φ_0 into Φ_1, whose flow A_1 is obtained by assigning a polarity to A_0 and applying c over negative contraction and cocontraction vertices belonging to ai-cycles, until they are all positive.

(2) *Break* ai-*cycles.* Transform Φ_1 into the algorithm's output Φ_2, whose flow A_2 is such that $A_1 \rightarrow_{\mathsf{bc}} A_2$.

Theorem 5.17. *Given any* SKS *derivation, algorithm* BC *produces a derivation with the same premiss and conclusion and such that its atomic flow is cycle-free.*

Proof. We refer to Definition 5.16. Step 1 transforms the given derivation Φ_0 into Φ_1, which, by Theorems 4.24 and 4.8, exists and all its ai-cycles are fragile cycles and its premiss and conclusion are the same as those of Φ_0. Then, by Theorems 5.9 and 5.13 we conclude that Φ_2, obtained from Φ_1 in Step 2, exists, is cycle-free and its premiss and conclusion are the same as those of Φ_1. □

The first and second reduction steps in Figure 6 provide an example of application of algorithm BC.

Definition 5.18. For every SKS derivation Φ_2, with cycle-free atomic flow A_2, *algorithm* EX (for *eliminate extremal simple edges*) performs the following steps:

(1) *Make* ai-*paths clean.* Transform Φ_2 into Φ_3, whose flow A_3 is normal for c.

(2) *Remove simple edges.* Transform Φ_3 into the algorithm's output Φ_4, whose flow A_4 is such that $A_3 \rightarrow_{\mathsf{ex}} A_4$.

Theorem 5.19. *Given any* SKS *derivation whose atomic flow is cycle-free, algorithm* EX *produces a derivation with the same premiss and conclusion and such that, in its atomic flow, there are no* ai-*connections.*

Proof. We refer to Definition 5.18. Step 1 transforms the given derivation Φ_2 into Φ_3, which, by Theorems 4.22 and 4.8 and by Propositions 4.17 and 4.18, exists, is cycle-free and all its ai-paths are clean paths and its premiss and conclusion are the same as those of Φ_2. Then, by Theorems 5.12 and 5.13 we conclude that Φ_4, obtained from Φ_3 in Step 2, exists, there are no ai-connections in its atomic flow, and its premiss and conclusion are the same as for Φ_3. □

The output of BC and EX is not normal for c, in general, as a quick look at Figure 5 shows.

5.3. **Streamlining Algorithms.** We obtain here our main result: normalisation algorithms for propositional logic that normalise generic derivations, and that entail cut elimination on proofs. We show two (similar) such algorithms, but it is clear that there are many possible variations. These algorithms are obtained as combinations of the reductions $\rightarrow_{\mathsf{w}}$, $\rightarrow_{\mathsf{c}}$, $\rightarrow_{\mathsf{bc}}$ and $\rightarrow_{\mathsf{ex}}$. We strengthen the notion of streamlined derivation introduced in the beginning, in order to appreciate the variations in the design of the normalising algorithms.

Definition 5.20. For every SKS derivation Φ_0, *algorithm* Str (for *streamlining*) performs the following steps:

(1) *Break* ai-*cycles.* Apply BC to Φ_0 and obtain Φ_2.

(2) *Eliminate* ai-*connections.* Apply EX to Φ_2 and obtain Φ_4.

(3) *Move away weakenings and coweakenings.* Transform Φ_4 into algorithm's output Φ_5, whose flow A_5 is normal for w.

Example 5.21. In Figure 6, algorithm Str is applied. For each flow the corresponding derivation is shown. Inference rules are used modulo the relation = for formulae. Trivial rule instances are omitted. The atomic flow we start out with has an ai-cycle, but no simple edge. The first step is to move a contraction to create a simple edge. The second step removes the simple edge. The remaining steps are weakening reductions. The reader can apply the construction in the proof of Theorem 5.3 (or 5.13) in order to generate the third derivation. We recall that adding some switch instances can be necessary in order to 'pull' the units involved in interactions and cointeractions to the premiss and conclusion of the derivation to be reduced.

We are about to prove that algorithm Str does more than just streamlining derivations: it turns out that Step 3 performs unnecessary reductions on weakening and coweakening vertices connected with contraction and cocontraction ones. However, the shape of the derivations produced by Str is notable, so we prefer to present Str this way, and it is understood that its definition can be weakened if strict streamlining is all that is required: we only need to apply the weakening reductions that are necessary to get streamlined derivations, and no more.

We now make precise the notion of streamlining obtained by Str (called 'super-stream-lining'), and a further strengthening that we explore in the rest of this section.

Definition 5.22. An SKS derivation is *super-streamlined* if it is streamlined and its associated atomic flow is normal for w. An SKS derivation is *hyper-streamlined* if it is super-streamlined and its associated atomic flow is normal for c. We say that an algorithm *P super-streamlines* (resp., *hyper-streamlines*) a derivation Φ if the output of P on Φ is a super-streamlined (resp., hyper-streamlined) derivation that has the same premiss and conclusion as Φ.

Example 5.23. In Example 3.17 (page 13), the second flow is streamlined, but not super-streamlined, and the third is hyper-streamlined (so, super-streamlined as well).

Remark 5.24. Consider the following figure
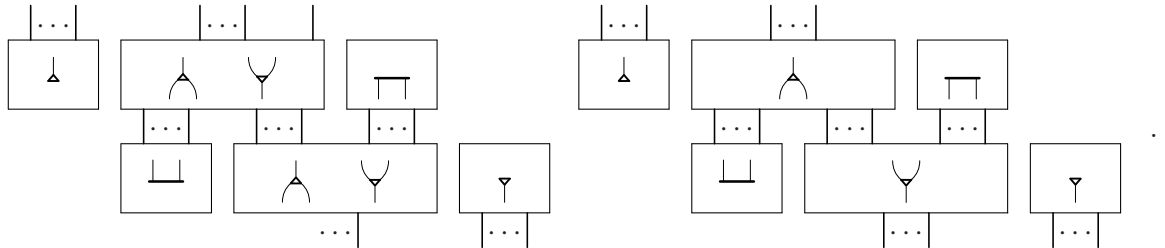
We see on the left the shape of flows of super-streamlined derivations and on the right that of flows of hyper-streamlined derivations, where the boxes represent flows obtained by freely composing edges and vertices whose labels are only those indicated on the boxes. Compare to Remark 3.16. This is the shape of the atomic flow of a hyper-streamlined proof:

$$
ai\downarrow\frac{\bar{a}}{[a \vee \bar{a}] \wedge \bar{a}}
$$
$$
s\frac{}{(a \wedge \bar{a}) \vee \bar{a}}
$$
$$
m\frac{}{[a \vee t] \wedge [\bar{a} \vee \bar{a}]}
$$
$$
ac\downarrow\frac{}{[a \vee t] \wedge \bar{a}}
$$
$$
ac\uparrow\frac{}{[(a \wedge a) \vee t] \wedge \bar{a}}
$$
$$
s\frac{}{(a \wedge a \wedge \bar{a}) \vee t}
$$
$$
ai\uparrow\frac{}{(a \wedge f) \vee t}
$$

$$c\downarrow$$

$$
ai\downarrow\frac{\bar{a}}{[\bar{a} \vee a] \wedge \bar{a}}
$$
$$
s\frac{}{\bar{a} \vee (a \wedge \bar{a})}
$$
$$
m\frac{}{[a \vee t] \wedge [\bar{a} \vee \bar{a}]}
$$
$$
ac\uparrow\frac{}{[(a \wedge a) \vee t] \wedge [\bar{a} \vee \bar{a}]}
$$
$$
s\frac{}{(a \wedge a \wedge [\bar{a} \vee \bar{a}]) \vee t}
$$
$$
ac\uparrow\frac{}{(a \wedge a \wedge a \wedge [\bar{a} \vee \bar{a}]) \vee t}
$$
$$
s\frac{}{(a \wedge a \wedge [(\bar{a} \wedge a) \vee \bar{a}]) \vee t}
$$
$$
ai\uparrow\frac{}{(a \wedge a \wedge \bar{a}) \vee t}
$$
$$
ai\uparrow\frac{}{(a \wedge f) \vee t}
$$

$$\rightarrow_{bc}$$

$$
c\uparrow\frac{\bar{a}}{\bar{a} \wedge \bar{a}}
$$
$$
aw\downarrow\frac{}{[t \vee a] \wedge \bar{a} \wedge \bar{a}}
$$
$$
s\frac{}{[t \vee (a \wedge \bar{a})] \wedge \bar{a}}
$$
$$
m\frac{}{[a \vee t] \wedge [t \vee \bar{a}] \wedge \bar{a}}
$$
$$
ac\uparrow\frac{}{[(a \wedge a) \vee t] \wedge [t \vee \bar{a}] \wedge \bar{a}}
$$
$$
s\frac{}{[(a \wedge a \wedge [t \vee \bar{a}]) \vee t] \wedge \bar{a}}
$$
$$
ac\uparrow\frac{}{[(a \wedge a \wedge a \wedge [t \vee \bar{a}]) \vee t] \wedge \bar{a}}
$$
$$
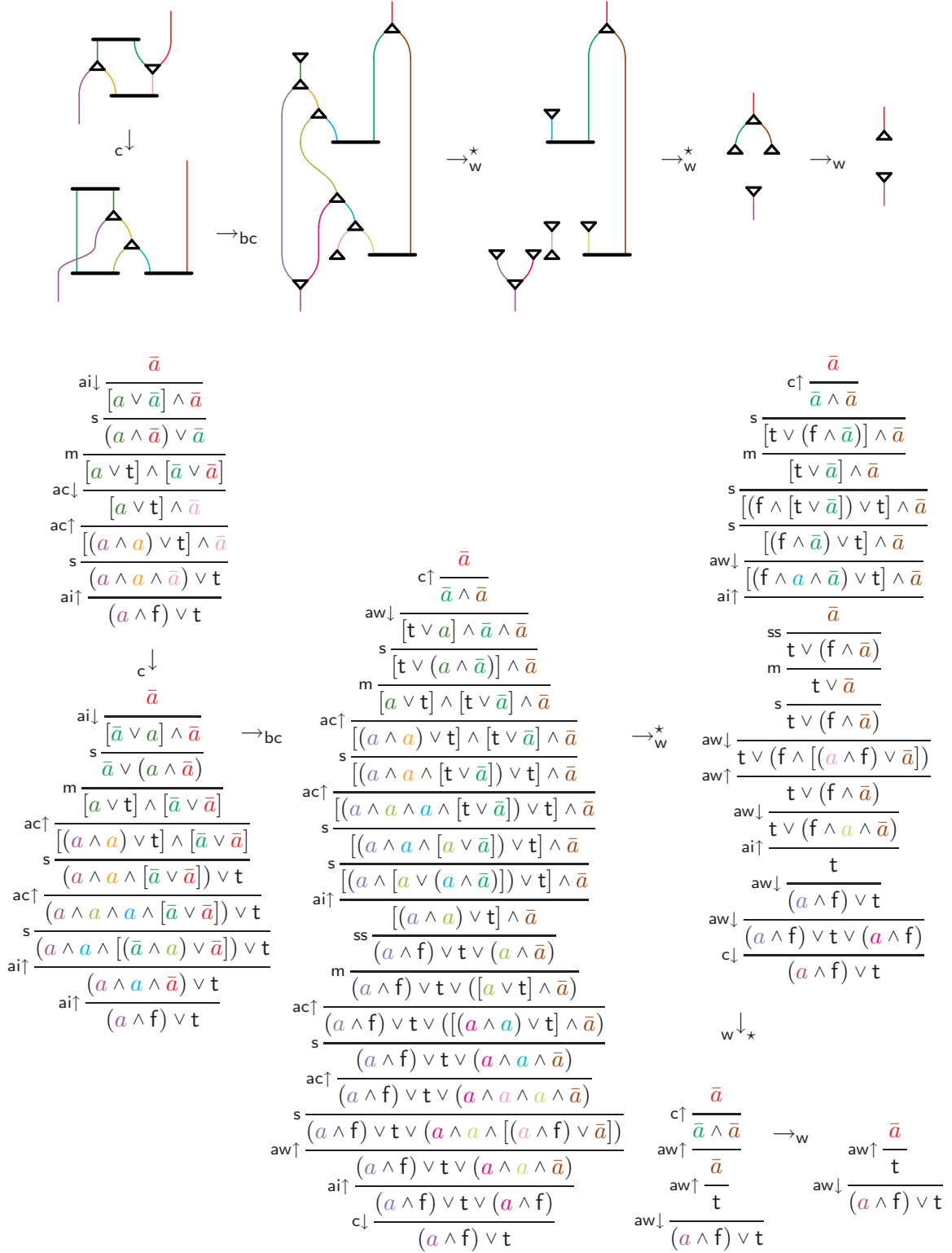s\frac{}{[(a \wedge a \wedge [a \vee \bar{a}]) \vee t] \wedge \bar{a}}
$$
$$
s\frac{}{[(a \wedge [a \vee (a \wedge \bar{a})]) \vee t] \wedge \bar{a}}
$$
$$
ai\uparrow\frac{}{[(a \wedge a) \vee t] \wedge \bar{a}}
$$
$$
ss\frac{}{(a \wedge f) \vee t \vee (a \wedge \bar{a})}
$$
$$
m\frac{}{(a \wedge f) \vee t \vee ([a \vee t] \wedge \bar{a})}
$$
$$
ac\uparrow\frac{}{(a \wedge f) \vee t \vee ([(a \wedge a) \vee t] \wedge \bar{a})}
$$
$$
s\frac{}{(a \wedge f) \vee t \vee (a \wedge a \wedge \bar{a})}
$$
$$
ac\uparrow\frac{}{(a \wedge f) \vee t \vee (a \wedge a \wedge a \wedge \bar{a})}
$$
$$
s\frac{}{(a \wedge f) \vee t \vee (a \wedge a \wedge [(a \wedge f) \vee \bar{a}])}
$$
$$
aw\uparrow\frac{}{(a \wedge f) \vee t \vee (a \wedge a \wedge \bar{a})}
$$
$$
ai\uparrow\frac{}{(a \wedge f) \vee t \vee (a \wedge f)}
$$
$$
c\downarrow\frac{}{(a \wedge f) \vee t}
$$

$$\rightarrow_{w}^{\star}$$

$$
c\uparrow\frac{\bar{a}}{\bar{a} \wedge \bar{a}}
$$
$$
s\frac{}{[t \vee (f \wedge \bar{a})] \wedge \bar{a}}
$$
$$
m\frac{}{[t \vee \bar{a}] \wedge \bar{a}}
$$
$$
s\frac{}{[(f \wedge [t \vee \bar{a}]) \vee t] \wedge \bar{a}}
$$
$$
s\frac{}{[(f \wedge \bar{a}) \vee t] \wedge \bar{a}}
$$
$$
aw\downarrow\frac{}{[(f \wedge a \wedge \bar{a}) \vee t] \wedge \bar{a}}
$$
$$
ai\uparrow\frac{}{}
$$

$$
ss\frac{\bar{a}}{t \vee (f \wedge \bar{a})}
$$
$$
m\frac{}{t \vee \bar{a}}
$$
$$
s\frac{}{t \vee (f \wedge \bar{a})}
$$
$$
aw\downarrow\frac{}{t \vee (f \wedge [(a \wedge f) \vee \bar{a}])}
$$
$$
aw\uparrow\frac{}{t \vee (f \wedge \bar{a})}
$$
$$
aw\downarrow\frac{}{t \vee (f \wedge a \wedge \bar{a})}
$$
$$
ai\uparrow\frac{}{t}
$$
$$
aw\downarrow\frac{}{(a \wedge f) \vee t}
$$
$$
aw\downarrow\frac{}{(a \wedge f) \vee t \vee (a \wedge f)}
$$
$$
c\downarrow\frac{}{(a \wedge f) \vee t}
$$

$$w\downarrow_{\star}$$

$$
c\uparrow\frac{\bar{a}}{\bar{a} \wedge \bar{a}}
$$
$$
aw\uparrow\frac{}{\bar{a}}
$$
$$
aw\uparrow\frac{}{t}
$$
$$
aw\downarrow\frac{}{(a \wedge f) \vee t}
$$

$$\rightarrow_{w}$$

$$
aw\uparrow\frac{\bar{a}}{t}
$$
$$
aw\downarrow\frac{}{(a \wedge f) \vee t}
$$

Figure 6: Example of application of algorithm Str.

Note that a hyper-streamlined SKS proof is a KS proof.

Theorem 5.25. *Algorithm* Str *super-streamlines (and so, streamlines) every* SKS *derivation.*

Proof. We refer to Definition 5.20. Theorems 5.17 and 5.19 guarantee that there are no ai-connections in Φ_4's flow A_4. By Theorem 4.12, Step 3 terminates. Since no new ai-paths are created by Step 3, and because of Remark 4.13, we can conclude that Φ_5 is super-streamlined. Since the reductions of w are sound (Theorem 4.8), the premiss and conclusion of Φ_5 are the same as those of Φ_0. $\square$

Cut elimination immediately follows (see also Remark 3.18).

Corollary 5.26 (Cut elimination). *Algorithm* Str, *when applied to an* SKS *proof, produces a cut-free proof.*

Remark 5.27. Cut-free proofs obtained from Str do not exhibit any coweakening rule, but there might be cocontraction rules, since Step 2 of algorithm EX might introduce some.

We can easily define a stronger algorithm than Str.

Definition 5.28. For every SKS derivation Φ_0, *algorithm* HStr (for *hyper-streamlining*) performs the following steps:
  (1) *Super-streamline.* Apply Str to Φ_0 and obtain Φ_5.
  (2) *Move away contractions and cocontractions.* Transform Φ_5 into algorithm's output Φ_6, whose flow A_6 is normal for c.

Theorem 5.29. *Algorithm* HStr *hyper-streamlines every* SKS *derivation; moreover, when applied to an* SKS *proof,* HStr *produces a* KS *proof.*

Proof. We appeal to Theorem 5.25 and we note that, by Proposition 4.20, the flow of the output derivation is normal for w and c. $\square$

Remark 5.30. Thanks to algorithm HStr, we can easily obtain a 'decomposition' result (as this kind of theorems are called in the deep-inference literature). Any SKS derivation can be reduced to a derivation of the form

$$
\begin{array}{c}
\alpha \\
\big\| \; \{\mathsf{ai}{\downarrow},\mathsf{aw}{\uparrow},\mathsf{ac}{\uparrow}\} \\
\beta \\
\big\| \; \{\mathsf{s},\mathsf{m}\} \\
\gamma \\
\big\| \; \{\mathsf{ai}{\uparrow},\mathsf{aw}{\downarrow},\mathsf{ac}{\downarrow}\} \\
\delta
\end{array}
\quad .
$$

In fact, a given derivation is transformed into a hyper-streamlined one, and then it is reduced to the form above by (obvious) permutations of inference steps. Actually, we can obtain

more detailed normal forms than the above, for example, any of the following:

$$
\begin{array}{ccc}
\alpha_1 & \alpha_1 & \alpha_1 \\[2pt]
\Big\Vert\, \{ai\!\downarrow\} & \Big\Vert\, \{aw\!\uparrow\} & \Big\Vert\, \{aw\!\uparrow\} \\[2pt]
\alpha_2 & \alpha_2 & \alpha_2 \\[2pt]
\Big\Vert\, \{aw\!\uparrow\} & \Big\Vert\, \{ai\!\downarrow\} & \Big\Vert\, \{ac\!\uparrow\} \\[2pt]
\alpha_3 & \alpha_3 & \alpha_3 \\[2pt]
\Big\Vert\, \{ac\!\uparrow\} & \Big\Vert\, \{ac\!\uparrow\} & \Big\Vert\, \{ai\!\downarrow\} \\[2pt]
\alpha_4 & \alpha_4 & \alpha_4 \\[2pt]
\Big\Vert\, \{s,m\} \quad , & \Big\Vert\, \{s,m\} \quad , & \Big\Vert\, \{s,m\} \quad , \text{ etc.} \\[2pt]
\alpha_4' & \alpha_4' & \alpha_4' \\[2pt]
\Big\Vert\, \{ac\!\downarrow\} & \Big\Vert\, \{ac\!\downarrow\} & \Big\Vert\, \{ai\!\uparrow\} \\[2pt]
\alpha_3' & \alpha_3' & \alpha_3' \\[2pt]
\Big\Vert\, \{aw\!\downarrow\} & \Big\Vert\, \{ai\!\uparrow\} & \Big\Vert\, \{ac\!\downarrow\} \\[2pt]
\alpha_2' & \alpha_2' & \alpha_2' \\[2pt]
\Big\Vert\, \{ai\!\uparrow\} & \Big\Vert\, \{aw\!\downarrow\} & \Big\Vert\, \{aw\!\downarrow\} \\[2pt]
\alpha_1' & \alpha_1' & \alpha_1'
\end{array}
$$

In [BT01, Brü04], this result is obtained as a consequence of cut elimination for SKS. We note that direct attempts at proving decomposition can face hard termination problems when dealing with the permutation of $ac\!\downarrow$ over $ai\!\uparrow$ inference steps (and the dual case). The problem boils down to taming infinite loops of the same nature as those mentioned in Remark 4.16. The streamlined forms of derivations, of course, take care of this. For an instructive example of this phenomenon, in a more complicated logic to which we hope to extend atomic flows, we refer the reader to [GS07].

The complexity of Str and HStr is exponential on the size of the input derivation, mainly due to the recursive duplication of derivations (see Remark 5.15). This is in line with the expectations for propositional logic normalisation. As we said in Remark 4.23, normalising via $\rightarrow_c$ also bears a potentially exponential cost, due to the contraction/cocontraction reduction. Normalising via $\rightarrow_w$, instead, reduces the complexity of derivations, as Figure 6 shows. These algorithms can be optimised, in the sense of complexity of flows and proofs, in many ways. For example, a simple method for keeping the complexity low is to always reduce weakenings as soon as possible.

There is a source of exponential speed-ups that is worth exploring in the future. The definition of $\rightarrow_{bc}$ and $\rightarrow_{ex}$ is based on recursively copying and stitching together copies of an entire flow. This is so because we need to make sure that, corresponding to the flows that we stitch together, there are proper corresponding derivations. However, it is not necessary to operate on the entire flow, it would be enough to copy and stitch only the minimal flow containing a simple edge and for which a subderivation can be found. In other words, we are duplicating entire flows for the only reason that we know that they correspond to derivations, but a more sophisticated approach could be able to do better.

We know that there is no strongly normalising algorithm based on $\rightarrow_{se}$, if by 'strongly normalising' we mean a completely liberal use of $\rightarrow_{se}$. However, natural constraints on $\rightarrow_{se}$ might lead to strong normalisation. For example, it might be possible to succeed in this if we adopt the notion of 'minimal derivation containing a simple edge' mentioned above.

6. Conclusions

We have shown a novel method of control for normalisation algorithms. It is based on a simple, graphical formalism, called 'atomic flows', similar both to proof nets and flow graphs. Atomic flows appear to capture an important aspect of the normalisation process. With a novel technique supported by atomic flows, we proved a symmetric generalisation of cut elimination for derivations, which seems to be desperately complicated when subjected to some traditional syntactic analysis.

We and other researchers, in particular François Lamarche and Lutz Straßburger, found several times that dealing with loops similar in nature to those of Remark 4.16 is very challenging (see, for example, [GS07, LS05a, LS05b, Str07b]). The similarities suggest that the techniques of the present paper are probably applicable to other normalisation algorithms and to categorical axiomatisations of classical logic.

We have exploited the possibility, peculiar to deep inference, of designing proof systems whose structural rules are all atomic and whose logical rules are all linear. This allows a great simplification in the geometric study of dependencies between inference rules. In comparison, the sequent calculus, for example, cannot exhibit an atomic contraction rule; this means that contractions on single atom occurrences are not independent, and so, substituting atom occurrences with more complex formulae becomes practically impossible.

Our success stems from the possibility of easily manipulating complex graphs by accessing single, independent occurrences of atoms. We expect to quickly broaden the range of applications of our methods, because all the major logics enjoy presentations in deep inference with atomic structural rules and linear logical ones, contrary to what is possible in any other known formalism. However, our methods so far rely on contraction (and co-contraction for symmetric normalisations), so, ironically, it is not obvious how to use them in the case of pure linear logics.

This work shows that cut elimination is far less a delicate property than it is usually assumed. As a matter of fact, any choice of logical inference rules, provided it makes for an implicationally complete system (so that we can recover the switch and medial rules), would leave the results presented here intact. We argued that the normalisation algorithm itself is not delicate, and enjoys a vast range of possibilities for optimisation. We find it interesting that, by adopting a more liberal syntactic discipline than that of non-deep inference, we correspondingly obtain more freedom for normalisation. Some would expect the opposite to happen. We interpret this as further evidence that traditional proof theory is too syntactic, to the point that syntactic artefacts obscure a deeper and simpler reality.

Much of this paper has been about the flow rewriting systems w and c, which allow reductions of flows based on local reduction rules. We then developed global reduction mechanisms, based on the global reduction $\to_{\sf se}$, only enough to show our normalisation theorem, with the minimum conceptual effort. However, there is a multitude of interesting global reduction mechanisms, arising from the atomic flow perspective, that are worth exploring. The paper [GG08] is devoted to these global mechanisms, especially in connection with possible computational interpretations.

We are currently investigating, together with Michel Parigot, the use of atomic flows in a computational interpretation of normalisation in SKS. We are also trying to extend our technique to the case of intuitionistic [Tiu06a] and modal logics [GT07, Sto07]. Preliminary investigations with Alwen Tiu lead us to think that the algorithm can be adapted to his presentation of intuitionistic logic in the calculus of structures; in particular, flow polarities

match Tiu's polarities, which is an interesting phenomenon that we do not fully understand at this point. We anticipate that modalities will play a crucial role in understanding constructivity beyond intuitionist logic.

With Michel Parigot, we are using atomic flows to design and test the properties of a new bureaucracy-free formalism that we are temporarily calling *formalism* B. Just to give an example, in formalism B, the derivation on the right in Figure 1 (page 10) becomes

$$
\left(
\mathsf{m}\cfrac{\left[\mathsf{ac}{\uparrow}\cfrac{a}{(a\ a)}\ \mathsf{ac}{\uparrow}\cfrac{b}{(b\ b)}\right]}{([a\ b]\ [a\ b])}\ \mathsf{ac}{\uparrow}\cfrac{c}{(c\ c)}
\right)\quad,
$$

where disjunctions and conjunctions are considered modulo commutativity and associativity. Derivations in formalism B are, basically, atomic flows enriched with logical information. Consequently, in this formalism, the behaviour of normalisation algorithms controlled by atomic flows is much more natural than the same for the calculus of structures.

In the future, we want to explore the relations between atomic flows and proof nets, as in [LS05a, LS05b, LS06, Str07b, SL04]. An interesting problem is to find simple combinatorial conditions that decide whether an atomic flow is associated to some derivation with a given premiss and conclusion. Our results might help in restricting this problem to cases where no structural rules are involved, *i.e.*, to the purely linear fragment consisting of the rules switch and medial. The problem for switch in isolation is solved by the correctness criteria for multiplicative linear logic, see Christian Retoré's [Ret03]. Lutz Straßburger, in [Str07a], found a criterion for the system containing only medial. For the combination of switch and medial, no criterion is known.

We are also interested in exploring the connections between our work and Craig's interpolation theorem, especially in relation with the size of interpolants (see Alessandra Carbone's work [Car97] for a possibly related approach to ours). There appear to be connections between atomic flows and Dominic Hughes' combinatorial proofs [Hug06]; we are especially interested in generating combinatorial proofs by manipulating atomic flows. Atomic flows could be given an algebraic characterisation with Albert Burroni's polygraphs [Bur93]; Yves Guiraud investigated similar constructions in his paper [Gui06], where he analyses this way the structural bureaucracy of SKS derivations. Finally, there might be connections with the theory of matings, as in the works [And81] by Peter Andrews and [Bib81] by Wolfgang Bibel; that theory also can be considered an abstract characterisation of proofs.

References

[And81]   Peter B. Andrews. Theorem proving via general matings. *Journal of the ACM*, 28(2):193–214, 1981.

[BG08]   Paola Bruscoli and Alessio Guglielmi. On the proof complexity of deep inference. *ACM Transactions on Computational Logic*, 2008. In press. `http://cs.bath.ac.uk/ag/p/PrComplDI.pdf`.

[Bib81]   Wolfgang Bibel. On matrices with connections. *Journal of the ACM*, 28(4):633–645, 1981.

[Bru02]   Paola Bruscoli. A purely logical account of sequentiality in proof search. In Peter J. Stuckey, editor, *Logic Programming, 18th International Conference*, volume 2401 of *Lecture Notes in Computer Science*, pages 302–316. Springer-Verlag, 2002. `http://cs.bath.ac.uk/pb/bvl/bvl.pdf`.

[Brü03]   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. `http://www.iam.unibe.ch/~kai/Papers/ace.pdf`.

[Brü04]   Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. Logos Verlag, Berlin, 2004. `http://www.iam.unibe.ch/~kai/Papers/phd.pdf`.

[Brü06a] Kai Brünnler. Cut elimination inside a deep inference system for classical predicate logic. *Studia Logica*, 82(1):51–71, 2006. `http://www.iam.unibe.ch/~kai/Papers/q.pdf`.

[Brü06b] Kai Brünnler. Deep inference and its normal form of derivations. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *Computability in Europe 2006*, volume 3988 of *Lecture Notes in Computer Science*, pages 65–74. Springer-Verlag, July 2006. `http://www.iam.unibe.ch/~kai/Papers/n.pdf`.

[Brü06c] 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. `http://www.aiml.net/volumes/volume6/Bruennler.ps`.

[Brü06d] Kai Brünnler. Locality for classical logic. *Notre Dame Journal of Formal Logic*, 47(4):557–580, 2006. `http://www.iam.unibe.ch/~kai/Papers/LocalityClassical.pdf`.

[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 *Lecture Notes in Artificial Intelligence*, pages 347–361. Springer-Verlag, 2001. `http://www.iam.unibe.ch/~kai/Papers/lcl-lpar.pdf`.

[Bur93]   Albert Burroni. Higher dimensional word problems with applications to equational logic. *Theoretical Computer Science*, 115(1):43–62, (1993).

[Bus91]   Samuel R. Buss. The undecidability of k-provability. *Annals of Pure and Applied Logic*, 53(1):75–102, 1991.

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

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

[DG04]   Pietro Di Gianantonio. Structures for multiplicative cyclic linear logic: Deepness vs cyclicity. In J. Marcinkowski and A. Tarlecki, editors, *CSL 2004*, volume 3210 of *Lecture Notes in Computer Science*, pages 130–144. Springer-Verlag, 2004. `http://www.dimi.uniud.it/~pietro/papers/Soft-copy-ps/scll.ps.gz`.

[Gen69]   Gerhard Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland, Amsterdam, 1969.

[GG08]   Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows II. `http://cs.bath.ac.uk/ag/p/NormContrDIAtFl2.pdf`, 2008.

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

[GS01]   Alessio Guglielmi and Lutz Straßburger. Non-commutativity and MELL in the calculus of structures. In L. Fribourg, editor, *CSL 2001*, volume 2142 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, September 2001. `http://cs.bath.ac.uk/ag/p/NoncMELLCoS.pdf`.

[GS02]   Alessio Guglielmi and Lutz Straßburger. A non-commutative extension of MELL. In M. Baaz and A. Voronkov, editors, *LPAR 2002*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 231–246. Springer-Verlag, October 2002. `http://www.lix.polytechnique.fr/~lutz/papers/NEL.pdf`.

[GS07]   Alessio Guglielmi and Lutz Straßburger. A system of interaction and structure IV: The exponentials. In the second round of revision for Mathematical Structures in Computer Science. `http://www.lix.polytechnique.fr/~lutz/papers/NELbig.pdf`, 2007.

[GT07]   Rajeev Goré and Alwen Tiu. Classical modal display logic in the calculus of structures and minimal
          cut-free deep inference calculi for S5. *Journal of Logic and Computation*, 17(4):767–794, 2007.
          `http://users.rsise.anu.edu.au/~tiu/papers/cmdl.pdf`.
[Gug07]   Alessio Guglielmi. A system of interaction and structure. *ACM Transactions on Computational
          Logic*, 8(1):1–64, 2007. `http://cs.bath.ac.uk/ag/p/SystIntStr.pdf`.
[Gui06]   Yves Guiraud. The three dimensions of proofs. *Annals of Pure and Applied Logic*, 141(1-2):266–
          295, 2006. `http://www.loria.fr/~guiraudy/recherche/cos1.pdf`.
[Hug06]   Dominic J.D. Hughes. Proofs without syntax. *Annals of Mathematics*, 164(3):1065–1076, 2006.
[Kah06]   Ozan   Kahramanoğulları.   Reducing   nondeterminism   in   the   calculus   of   struc-
          tures.   In   M.   Hermann   and   A.   Voronkov,   editors,   *LPAR   2006*,   volume   4246
          of   *Lecture   Notes   in   Artificial   Intelligence*,   pages   272–286.   Springer-Verlag,   2006.
          `http://www.doc.ic.ac.uk/~ozank/Papers/reducingNondet.pdf`.
[Kah07a] Ozan Kahramanoğulları. Maude as a platform for designing and implementing deep in-
          ference systems. In *RULE 2007—The Eighth International Workshop on Rule-Based Pro-
          gramming*, Electronic Notes in Theoretical Computer Science. Elsevier, 2007. In press.
          `http://www.doc.ic.ac.uk/~ozank/Papers/rule07.pdf`.
[Kah07b] Ozan Kahramanoğulları. System BV is NP-complete. *Annals of Pure and Applied Logic*, 2007. In
          press. `http://www.doc.ic.ac.uk/~ozank/Papers/bv_npc_apal.pdf`.
[Laf97]   Yves Lafont. Interaction combinators. *Information and Computation*, 137:69–101, 1997.
[LS05a]   François Lamarche and Lutz Straßburger. Constructing free boolean categories. In Prakash Panan-
          gaden, editor, *20th Annual IEEE Symposium on Logic in Computer Science*, pages 209–218. IEEE,
          2005. `http://www.lix.polytechnique.fr/~lutz/papers/FreeBool-long.pdf`.
[LS05b]   François   Lamarche   and   Lutz   Straßburger.   Naming   proofs   in   classical   propositional
          logic.   In   Paweł   Urzyczyn,   editor,   *Typed   Lambda   Calculi   and   Applications*,   volume
          3461   of   *Lecture   Notes   in   Computer   Science*,   pages   246–261.   Springer-Verlag,   2005.
          `http://www.lix.polytechnique.fr/~lutz/papers/namingproofsCL.pdf`.
[LS06]   François Lamarche and Lutz Straßburger. From proof nets to the free *-autonomous category.
          Logical Methods in Computer Science, 2(4):3:1–44, 2006. `http://arxiv.org/pdf/cs.LO/0605054`.
[Ret03]   Christian Retoré. Handsome proof-nets: Perfect matchings and cographs. *Theoretical Computer
          Science*, 294(3):473–488, 2003.
[SL04]   Lutz   Straßburger   and   François   Lamarche.   On   proof   nets   for   multiplicative   linear
          logic   with   units.   In   J.   Marcinkowski   and   A.   Tarlecki,   editors,   *CSL   2004*,   volume
          3210   of   *Lecture   Notes   in   Computer   Science*,   pages   145–159.   Springer-Verlag,   2004.
          `http://www.lix.polytechnique.fr/~lutz/papers/multPN.pdf`.
[Sto07]   Phiniki Stouppa. A deep inference system for the modal logic S5. *Studia Logica*, 85(2):199–214,
          2007. `http://www.iam.unibe.ch/til/publications/pubitems/pdfs/sto07.pdf`.
[Str02]   Lutz Straßburger. A local system for linear logic. In M. Baaz and A. Voronkov, editors, *LPAR
          2002*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 388–402. Springer-Verlag, 2002.
          `http://www.lix.polytechnique.fr/~lutz/papers/lls-lpar.pdf`.
[Str03a] Lutz   Straßburger.   *Linear   Logic   and   Noncommutativity   in   the   Calcu-
          lus   of   Structures*.   PhD   thesis,   Technische   Universität   Dresden,   2003.
          `http://www.lix.polytechnique.fr/~lutz/papers/dissvonlutz.pdf`.
[Str03b] Lutz Straßburger. MELL in the calculus of structures. *Theoretical Computer Science*, 309:213–285,
          2003. `http://www.lix.polytechnique.fr/~lutz/papers/els.pdf`.
[Str07a] Lutz Straßburger. A characterisation of medial as rewriting rule. In Franz Baader, editor, *RTA
          2007*, volume 4533 of *Lecture Notes in Computer Science*, pages 344–358. Springer-Verlag, 2007.
          `http://www.lix.polytechnique.fr/~lutz/papers/CharMedial.pdf`.
[Str07b] Lutz   Straßburger.   On   the   axiomatisation   of   boolean   categories   with   and   with-
          out   medial.   *Theory   and   Applications   of   Categories*,   18(18):536–601,   2007.
          `http://www.lix.polytechnique.fr/~lutz/papers/medial.pdf`.
[Tiu06a] 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.
          `http://users.rsise.anu.edu.au/~tiu/localint.pdf`.
[Tiu06b] Alwen Tiu. A system of interaction and structure II: The need for deep inference. *Logical Methods
          in Computer Science*, 2(2):4:1–24, 2006. `http://arxiv.org/pdf/cs.LO/0512036`.