Removing Cycles from Proofs*

Andrea Aler Tubella[1], Alessio Guglielmi[2], and Benjamin Ralph[3]

1 **IRIF, CNRS et Université Paris Diderot**
`Andrea.Aler@irif.fr`
2 **Department of Computer Science, University of Bath, UK**
`a.guglielmi@bath.ac.uk`
3 **Department of Computer Science, University of Bath, UK**
`b.d.ralph@bath.ac.uk`

Abstract

If we track atom occurrences in classical propositional proofs in deep inference, we see that they can form cyclic structures between cuts and identity steps. These cycles are an obstacle to a very natural form of normalisation, that simply unfolds all the contractions in a proof. This mechanism, which we call 'decomposition', has many points of contact with explicit substitutions in lambda calculi. In the presence of cycles, decomposition does not terminate, and this is an obvious drawback if we want to interpret proofs computationally. One way of eliminating cycles is eliminating cuts. However, we could ask ourselves whether it is possible to eliminate cycles independently of (general) cut elimination. This paper shows an efficient way to do so, therefore establishing the independence of decomposition from cut elimination. In other words, cut elimination in propositional logic can be separated into three separate procedures: 1) cycle elimination, 2) unfolding of contractions, 3) elimination of cuts in the linear fragment.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases proof theory, deep inference, proof complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.9

1 Introduction

It is well known, in classical and other logics, that cuts compress proofs. Conversely, eliminating cuts, i.e., normalising, expands proofs, and normalisation can be interpreted computationally. While this is a general phenomenon, there is a special case where the situation is not so clear. In [4], Buss introduces the concept of 'logical flow graph' as a useful tool to analyse the complexity of proofs. A logical flow graph is a graph obtained by tracking subformulae in a proof, and the topological information that it exposes can be directly connected to the complexity of the proof. In [5], Buss notes that it is possible to form cycles in the flow graphs of classical logic proofs, where a cycle is essentially a loop involving cuts and identity axioms. In that paper, Buss states the problem of determining whether using cuts in cycles helps to compress a proof significantly, or not, and he offers examples where the cycles can be eliminated at no cost. In [6], Carbone proves that n cycles can be eliminated from a classical propositional proof of k lines to give an acyclic proof of $\mathcal{O}(k^{n+1})$ lines. All those results apply to the sequent calculus. This paper contributes some results concerning essentially the same problem for deep-inference classical propositional proofs [1] and provides a procedure that eliminates cycles.

* This research has been supported by EPSRC Project EP/K018868/1 *Efficient and Natural Proof Systems* and ANR project FISP ANR-15-CE25-0014-01

Let us first introduce the wider context of the present work. Normalisation in deep inference allows a finer control over complexity than Gentzen's theory does, in particular, because it separates two composition mechanisms that in the normalisation of the sequent calculus are conflated: cut and contraction. The notion of logical flow graph becomes much simpler, because, in deep inference, we can reduce all the structural rules to their atomic variants. These simplified flow graphs are called 'atomic flows' [8, 10]. It turns out that we can control normalisation directly from atomic flows because every operation on the graphs can be mimicked by a corresponding operation on inference steps of minimal (atomic) granularity. In deep inference, we have atomic cuts, atomic identities, atomic contractions and atomic cocontractions; inference steps of these kinds can all be moved and normalised upon separately, which is not possible in Gentzen theory. The finer control on complexity that we obtain this way leads to the surprising result that eliminating cuts in propositional logic only has a quasipolynomial cost (as opposed to exponential in Gentzen's theory) [2, 12]. This development depends, crucially, on the use of contractions as a sharing mechanism in a structure that we call a 'sausage'. It is an open problem whether sausages can be, in turn, removed at polynomial cost or not [7]. In other words, in deep inference, part of the complexity that in Gentzen's theory is controlled by cuts has been isolated into contractions.

What about cycles in deep inference? Can they (and their cuts) be eliminated at a lower cost than general cut elimination? This paper makes some progress towards answering that question, in the first place by providing a cycle elimination procedure (independent from general cut elimination) and secondarily by confining the creation of most proof complexity in one specific step of the procedure. As one could expect, again, some of the complexity that was controlled by cuts is shifted to contractions: our procedure produces sausages. However, what is totally unexpected is the way sausages are created. This happens only via normalising through associativity steps, i.e., complexity is created in inference steps that have no logical content in terms of deduction. Why is it so? Is it just an artefact of our procedure? At present, we are unable to devise a procedure that avoids the problem, but we also are unable to design proofs with cycles where the use of the offending associativity is crucial. In other words, it seems possible to enhance our procedure in such a way that some preprocessing of the given proof would avoid the creation of sausages.

Apart from the progress in dealing with proof complexity, cycle elimination helps normalisation theory in general, essentially because it provides a simple induction measure. The experience of a decade of work with atomic flows (and almost three decades with logical flow graphs) shows that paths in proofs play a crucial role in understanding where the pieces of proofs move during normalisation. Typically, subproofs move along paths; for example, contractions move along atomic flows, in a process called 'decomposition' in the deep inference literature. The absence of cycles allows us to use the length of paths as a straightforward induction measure for decomposition. Therefore we expect applications of this research in computational interpretations. A further benefit should be in the development of a proof semantics that takes into consideration proof complexity. More generally, this result fits in a wider ongoing research on separating the various compression mechanisms of proofs. For example, and thanks to this result, cut elimination in propositional logic can be separated into three separate procedures: 1) cycle elimination, 2) decomposition, 3) elimination of cuts in the linear fragment (a process called 'splitting' in the linear logic literature). Finally, we note that the technique employed in this paper is not restricted to classical logic, and, in fact, can be generalised to a wide class of logics [13].

As is well known, the issues of compression and circularity pop up everywhere in computational logic, so, at a superficial level, our work could be deemed to have connections to

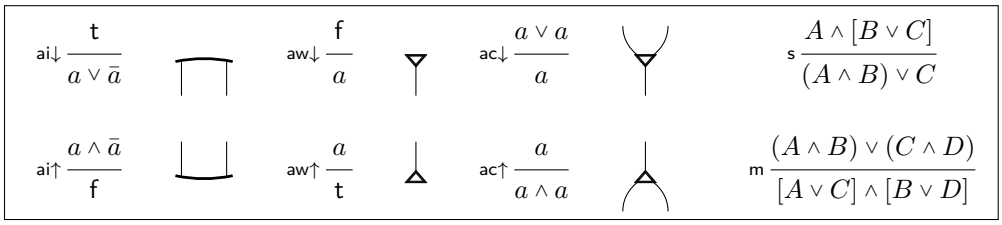


Figure 1 The six structural rules and two logical rules of SKS. The structural rules are shown with their respective atomic flow vertices [3, 11].

several other investigations. On the other hand, it seems that our cycle-elimination procedure only makes sense in proof systems where the full descriptive power of atomic flows can be exploited. This basically means that we need to work on fully localised proof systems, i.e., proof systems where the cost of checking a rule instance is bounded by a constant. This is a feature that (to the best of our knowledge) is only achievable in deep inference.

There is no way that this paper can be made self-contained because of the limitations of the conference format. Luckily, good papers exist on the fundamentals of deep inference. On the other hand, we made some effort towards making the paper self-explanatory regarding atomic flows. In other words, while the reader who does not know deep inference needs to refer to the cited literature, there is no need to study atomic flows elsewhere.

2 Preliminaries on Deep Inference and Atomic Flows

In this paper, we will be working in the deep inference system SKS for classical propositional logic, using the formalism open deduction [9]. We will assume a working knowledge of this proof system, a full exposition of which can be found in [7]. The structural and logical rules of SKS can be found in Figure 1. We do not include the equality (associativity and commutativity of connectives) and unit rules. We will sometimes use the non-atomic versions of the six structural rules (denoted without the 'a'). Each non-atomic structural rule ρ can be thought of as abbreviating a derivation using the rules $\mathsf{a}\rho, \mathsf{s}$ and m [3].

2.1 Atomic Flows

An *(atomic) flow* is a geometric invariant of an SKS derivation that follows the occurrences of atoms. They can be seen as composite diagrams that are freely generated from a set of six elementary diagrams, or as labelled directed graphs, where the six possible labels for the vertices are given in Figure 1. We can associate an atomic flow to every derivation in SKS in a natural way: every edge follows the occurrence of an atom in the derivation, and each vertex label corresponds to the occurrence of a structural rule where atoms are created or destroyed ($\mathsf{ai}{\downarrow}, \mathsf{ai}{\uparrow}, \mathsf{aw}{\downarrow}, \mathsf{aw}{\uparrow}$), or duplicated ($\mathsf{ac}{\downarrow}, \mathsf{ac}{\uparrow}$). The units f and t are not represented in the flow. See Figure 2 for examples of SKS derivations and their respective flows. Technically, there are some polarity restrictions on the construction of the flows to guarantee that for every flow there is an associated SKS derivation. However, an intuitive understanding of the flows is sufficient to follow the graphical representation of the reduction rules and the measure presented below, and this is what we are seeking to provide. Again, the interested reader is invited to refer to [7] for further technical details on the definition of atomic flows.

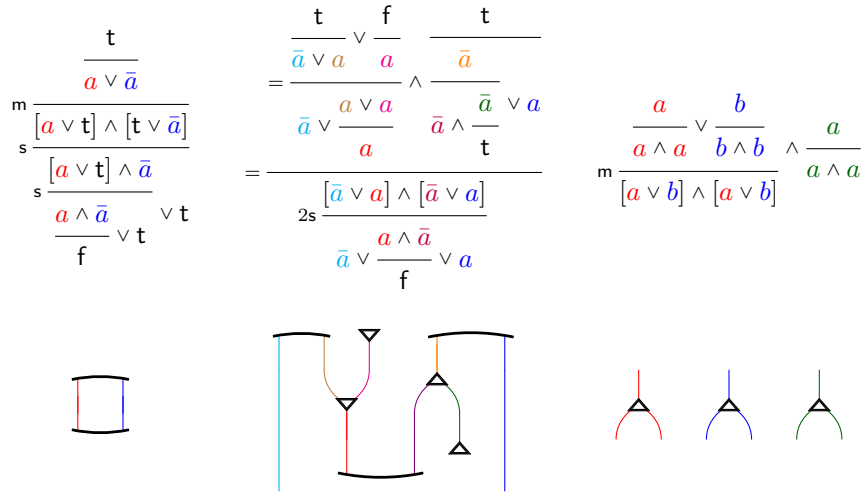


Figure 2 Three examples of SKS derivations and the atomic flows associated to them.

2.2 Decomposition

One major use of atomic flows is to better understand normalisation in deep inference [7, 8]. In particular, we can use atomic flows to describe the aspect of normalisation that deals with (co)contractions and (co)weakenings, preliminary to cut elimination. This stage of normalisation is called *decomposition* [13].

▶ **Definition 1.** An SKS derivation ϕ from A to B can be *decomposed* if we can convert it to the following form:

$$\begin{array}{ccc} A & & A \\ \phi \,\|\, \mathsf{SKS} & \longrightarrow & \|\, \mathsf{aw}{\uparrow} \\ B & & A_1 \\ & & \|\, \mathsf{ac}{\uparrow} \\ & & A_2 \\ & & \|\, \{\mathsf{s},\mathsf{m},\mathsf{ai}{\uparrow},\mathsf{ai}{\downarrow}\} \\ & & A_3 \\ & & \|\, \mathsf{ac}{\downarrow} \\ & & A_4 \\ & & \|\, \mathsf{aw}{\downarrow} \\ & & B \end{array}$$

We will first show that every acyclic proof can be decomposed, a result first shown in [8], and then extend the result to proofs containing cycles.

2.3 The rewriting system C

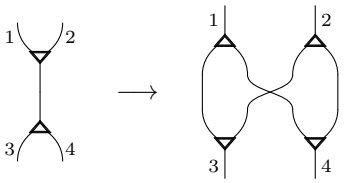
▶ **Definition 2.** A *reduction rule* r is a pair (ϕ', ψ') where ϕ' and ψ' are derivations in SKS with the same premise and conclusion. We write $r : \phi' \to \psi'$.

For every reduction rule $r : \phi' \to \psi'$ we define the reduction $\to_r$ such that $\phi \to_r \psi$ if and only if ψ' is a subderivation of ϕ and ψ is obtained from ϕ by replacing ϕ' by ψ'.

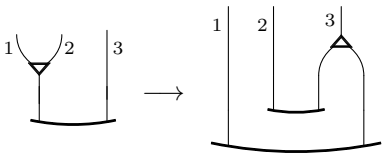
We call a finite set R of reduction rules a *rewriting system*. Given a set S of derivations, we say that rewriting system R is *terminating on* S if there is no infinite chain $\phi \to_{r_1} \phi_1 \to_{r_2} \dots$ with $r_i \in R$ for any $\phi \in S$.

▶ **Definition 3.** We define the following reduction rules for SKS:

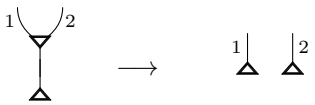
$$\mathsf{ac}{\downarrow} - \mathsf{ac}{\uparrow}: \quad \mathsf{ac}{\uparrow}\dfrac{\mathsf{ac}{\downarrow}\dfrac{a \vee a}{a}}{a \wedge a} \longrightarrow \mathsf{m}\dfrac{\mathsf{ac}{\uparrow}\dfrac{a}{a \wedge a} \vee \mathsf{ac}{\uparrow}\dfrac{a}{a \wedge a}}{\mathsf{ac}{\downarrow}\dfrac{a \vee a}{a} \wedge \mathsf{ac}{\downarrow}\dfrac{a \vee a}{a}}$$



$$\mathsf{ac}{\downarrow} - \mathsf{ai}{\uparrow}: \quad \mathsf{ai}{\uparrow}\dfrac{\mathsf{ac}{\downarrow}\dfrac{a \vee a}{a} \wedge \bar{a}}{\mathsf{f}} \longrightarrow \begin{array}{l}\mathsf{2s}\dfrac{[a \vee a] \vee \mathsf{ac}{\uparrow}\dfrac{\bar{a}}{\bar{a} \wedge \bar{a}}}{\mathsf{ai}{\uparrow}\dfrac{a \wedge \bar{a}}{\mathsf{f}} \vee \mathsf{ai}{\uparrow}\dfrac{a \wedge \bar{a}}{\mathsf{f}}} \\ = \dfrac{}{\mathsf{f}}\end{array}$$

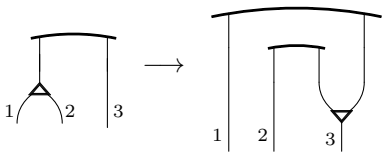


$$\mathsf{ac}{\downarrow} - \mathsf{aw}{\uparrow}: \quad \mathsf{aw}{\uparrow}\dfrac{\mathsf{ac}{\downarrow}\dfrac{a \vee a}{a}}{\mathsf{t}} \longrightarrow = \dfrac{\mathsf{aw}{\uparrow}\dfrac{a}{\mathsf{t}} \vee \mathsf{aw}{\uparrow}\dfrac{a}{\mathsf{t}}}{\mathsf{t}}$$

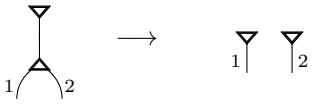


And their duals:

$$\mathsf{ai}{\downarrow} - \mathsf{ac}{\uparrow}: \quad \mathsf{ai}{\downarrow}\dfrac{\mathsf{t}}{\mathsf{ac}{\uparrow}\dfrac{a}{a \wedge a} \vee \bar{a}} \longrightarrow \begin{array}{l}= \dfrac{\mathsf{t}}{\mathsf{ai}{\downarrow}\dfrac{\mathsf{t}}{a \vee \bar{a}} \wedge \mathsf{ai}{\downarrow}\dfrac{\mathsf{t}}{a \vee \bar{a}}} \\ \mathsf{2s}\dfrac{}{(a \wedge a) \wedge \mathsf{ac}{\downarrow}\dfrac{\bar{a} \vee \bar{a}}{\bar{a}}}\end{array}$$



$$\mathsf{aw}{\downarrow} - \mathsf{ac}{\uparrow}: \quad \mathsf{ac}{\uparrow}\dfrac{\mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{a}}{a \wedge a} \longrightarrow = \dfrac{\mathsf{f}}{\mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{a} \wedge \mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{a}}$$



Last, we define the trivial family of reduction rules:

$$\mathsf{ac}{\downarrow} - \rho_H: \quad \rho\dfrac{H\left\{\mathsf{ac}{\downarrow}\dfrac{a \vee a}{a}\right\}}{H'\{a\}} \longrightarrow \rho\dfrac{H\{a \vee a\}}{H'\left\{\mathsf{ac}{\downarrow}\dfrac{a \vee a}{a}\right\}}$$

$$\rho_H - \mathsf{ac}{\uparrow}: \quad \rho\dfrac{H'\{a\}}{H\left\{\mathsf{ac}{\uparrow}\dfrac{a}{a \wedge a}\right\}} \longrightarrow \rho\dfrac{H'\left\{\mathsf{ac}{\uparrow}\dfrac{a}{a \wedge a}\right\}}{H\{a \wedge a\}}$$

These simply correspond to drawing the (co)contraction node lower (higher) in the atomic flow.

▶ **Definition 4.** We define *rewriting system* C for SKS as the rewriting system given by the reduction rules of Definition 3.

It should be clear that if the rewriting system C terminates for a derivation, we will obtain a derivation with the same premise and conclusion of the following form: all the instances of ac↑ are at the top, followed by a derivation composed only of rules in the set $\{\mathsf{s}, \mathsf{m}, \mathsf{ai}\uparrow, \mathsf{ai}\downarrow, \mathsf{aw}\uparrow, \mathsf{aw}\downarrow\}$ and a bottom phase made up only of ac↓ rules.

2.4 Termination of C

In [8, Theorem 4.22] it is shown that rewriting system C terminates for the set of SKS proofs that do not contain a certain construction, called an ai-cycle. The measure used to prove termination can be easily followed in a flow: it corresponds to the length of a certain type of path.

▶ **Definition 5.** Given an edge ϵ in an atomic flow, we define $\mathsf{up}(\epsilon)$ as the upper vertex it is connected to, and $\mathsf{lo}(\epsilon)$ as the lower vertex it is connected to.

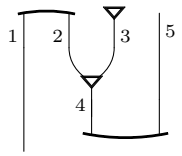
▶ **Definition 6.** Given a sequence of distinct edges $\epsilon_1, \ldots, \epsilon_n$ such that $\mathsf{lo}(\epsilon_i) = \mathsf{up}(\epsilon_{i+1})$ for $1 \leq i < n$, we say that $\epsilon_1, \ldots, \epsilon_n$ is a *path of length n from* $\mathsf{up}(\epsilon_1)$ *to* $\mathsf{lo}(\epsilon_n)$, and that $\epsilon_n, \ldots, \epsilon_1$ is a *path of length n from* $\mathsf{lo}(\epsilon_n)$ *to* $\mathsf{up}(\epsilon_1)$.

Given a sequence of edges $\epsilon_1, \ldots, \epsilon_n$, we say that $\epsilon_1, \ldots, \epsilon_n$ is an ai-*path of length n from vertex* v_1 *to vertex* v_2 if it is a path from v_1 to v_2 or if there exists a vertex v labelled by ai↑ or ai↓ such that $\epsilon_1, \ldots, \epsilon_h$ is an path from v_1 to v, $\epsilon_h \neq \epsilon_{h+1}$, and $\epsilon_{h+1}, \ldots, \epsilon_n$ is an ai-path from v to v_2.

An ai-path of length n is *maximal* if no ai-path containing its edges has length greater than n. An ai-path of length n from v is *maximal* if no ai-path from v containing its edges has length greater than n.

Intuitively, paths correspond to any non-empty sequence of edges from v_1 to v_2 that do not change direction (they either only 'go downwards' or only 'go upwards'). ai-paths are allowed to change direction, but only at ai-vertices: they are zig-zag paths that change direction at ai-nodes.

▶ **Example 7.**



- Some examples of paths in this flow are $(2, 4)$ and (5).
- Some examples of ai-paths in this flow are given by $(1, 2)$ and $(3, 4, 5)$.
- The maximal ai-paths in this flow are $(1, 2, 4, 5)$, $(3, 4, 5)$ and their respective reversals.
- The maximal ai-paths from the ac↓ vertex are $(2, 1)$, (3), and $(4, 5)$.

Informally, the ai-paths from a particular ac↓ vertex correspond to all the paths that the corresponding contraction will take when the reduction rules are applied. Thus, the maximal ai-path length corresponds to the maximal number of other rules a contraction must pass through.

For example, in a derivation whose flow is as in Example 7, when we apply the reduction rules to move the atomic contraction downwards, it will permute with one instance of the rule ai↑.

More precisely, we can assign a *rank* to every contraction and to every cocontraction of a derivation by referring to its flow. The rank of a contraction will be given by the sum of the lengths of the maximal ai-paths starting with the lower edge of its corresponding vertex in the flow. Dually, the rank of a cocontraction will be given by the sum of the lengths of the maximal ai-paths starting with the upper edge of its corresponding vertex in a flow. We will see that the reduction rules of system C reduce the sum of the ranks of the contractions and cocontractions in a derivation, effectively providing a termination measure when these ranks are finite.

▶ **Definition 8.** Given a vertex v labelled with ac↓ in a flow, we define its *rank* as the sum of the lengths of the maximal ai-paths $\epsilon_1, \ldots, \epsilon_n$ from v such that $\mathsf{up}(\epsilon_1) = v$.

Dually, given a vertex v labelled with ac↑ in a flow, we define its *rank* as the sum of the lengths of the maximal ai-paths $\epsilon_1, \ldots, \epsilon_n$ from v such that $\mathsf{lo}(\epsilon_1) = v$.

▶ **Example 9.** The rank of the ac↓ vertex of the flow of example 7 is 2: it corresponds to the length of the ai-path $(4, 5)$.

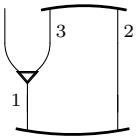
▶ **Definition 10.** Given an occurrence of the rule ac↓ in a derivation ϕ with flow ψ, we define its *rank* as the rank of its corresponding vertex in ψ.

Likewise, we define the rank of an occurrence of the rule ac↑ as the rank of its corresponding vertex.

However, it is possible that (co)contractions have an infinite rank in a derivation: these are precisely those cases when the contraction is in a *cycle*.

▶ **Definition 11.** An ai-path from v to v is called an ai-*cycle*.

▶ **Example 12.** In the following flow, the ai-path $(1, 2, 3)$ is an ai-cycle.



▶ **Definition 13.** We say that a derivation contains an ai-cycle if its atomic flow contains an ai-cycle.

We can easily see that if we repeatedly perform rewrites exclusively on a contraction inside an ai-cycle, such as in Figure 3, then the rewriting procedure will not terminate. In the absence of such cycles however, the rewriting always terminates. We will briefly outline this result and its proof as presented in [8].

▶ **Theorem 14.** *The rewriting system* C *is terminating on the set of* ai-*cycle-free derivations.*

Proof. The first observation is that it is clear by inspection of the reduction rules that the rank of (co)contractions not involved in the reduction stays the same.

Given an ai-cycle-free derivation ϕ, we consider the lexicographic order on (r, d). r is the sum of the ranks of the contractions and cocontractions in ϕ, and d is the sum of the number of rules below each contraction and the number of rules above each cocontraction when sequentialising ϕ.

We describe how each application of a reduction of C reduces (r, d):

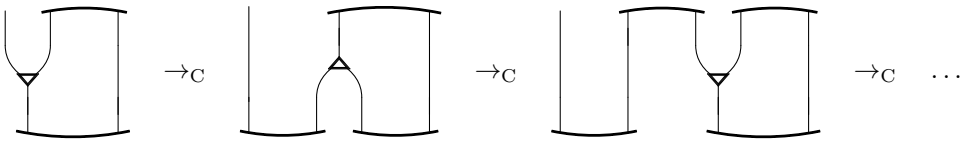


Figure 3 A flow that does not terminate under C.

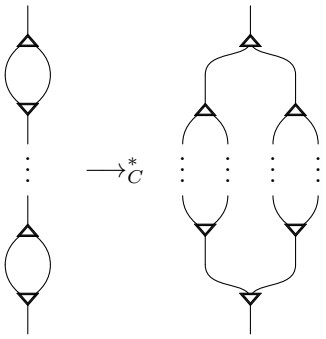


Figure 4 Exponential blow-up caused by sausages.

- Applications of the rules $\mathsf{ac}{\downarrow} - \mathsf{ac}{\uparrow}$, $\mathsf{ac}{\downarrow} - \mathsf{ai}{\downarrow}$ and $\mathsf{ai}{\downarrow} - \mathsf{ac}{\uparrow}$ reduce r in the absence of ai-cycles as is shown in the proof of Theorem 7.2.3 of [11].
- Applications of the rules $\mathsf{ac}{\downarrow} - \mathsf{aw}{\uparrow}$ and $\mathsf{aw}{\downarrow} - \mathsf{ac}{\uparrow}$ reduce r since they remove contractions and cocontractions.
- Applications of the rules $\mathsf{ac}{\downarrow} - \rho_H$ and $\rho_H - \mathsf{ac}{\uparrow}$ trivially maintain r and reduce d.

◀

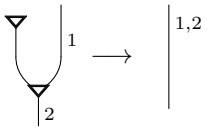
The decomposition procedure may increase the size of a proof exponentially, through the crossings of contractions and cocontractions as shown in Figure 4. We call the contraction-cocontraction pairs on the left *sausages*.

ai-cycles are evidently removed through cut-elimination, since they are caused by the connection of a cut and an introduction. In this paper we will present a local procedure to remove cycles that does not involve cut-elimination, thus proving the independence of decomposition from cut-elimination.

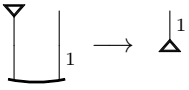
To improve this decomposition result, it can also be shown that (co)weakenings can be permuted to the bottom (top) of a derivation through the following reductions [7].

▶ **Definition 15.** We define the following reduction rules for SKS:

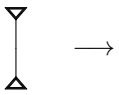
$$\mathsf{aw}{\downarrow} - \mathsf{ac}{\downarrow}: \quad \dfrac{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow} \vee a}{a}\,\mathsf{ac}{\downarrow} \longrightarrow = \dfrac{\mathsf{f} \vee a}{a}$$



$$\mathsf{aw}{\downarrow} - \mathsf{ai}{\uparrow}: \quad \dfrac{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow} \wedge \bar{a}}{\mathsf{f}}\,\mathsf{ai}{\uparrow} \longrightarrow = \dfrac{\mathsf{f} \wedge \mathsf{aw}{\uparrow}\dfrac{\bar{a}}{\mathsf{t}}}{\mathsf{f}}$$

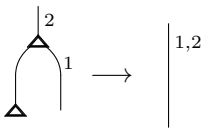


$$\mathsf{aw}{\downarrow} - \mathsf{aw}{\uparrow}: \quad \dfrac{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow}}{\mathsf{t}}\,\mathsf{aw}{\uparrow} \longrightarrow = \dfrac{= \dfrac{\mathsf{f}}{\mathsf{f} \wedge [\mathsf{f} \vee \mathsf{t}]}}{\mathsf{t}}\,\mathsf{s}$$

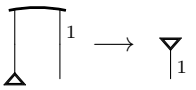


And their duals:

$$\mathsf{ac}{\uparrow} - \mathsf{aw}{\uparrow}: \quad \dfrac{a}{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow} \wedge a}\,\mathsf{ac}{\uparrow} \longrightarrow = \dfrac{a}{\mathsf{t} \wedge a}$$



$$\mathsf{ai}{\downarrow} - \mathsf{aw}{\uparrow}: \quad \dfrac{\mathsf{t}}{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow} \vee \bar{a}}\,\mathsf{ai}{\downarrow} \longrightarrow = \dfrac{\mathsf{t}}{\mathsf{t} \vee \mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{\bar{a}}}$$



And the trivial reductions:

$$\mathsf{aw}{\downarrow} - \rho_H: \quad \dfrac{H\left\{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow}\right\}}{H'\{a\}}\,\rho \longrightarrow \dfrac{H\{\mathsf{f}\}}{H'\left\{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow}\right\}}\,\rho$$

$$\rho_H - \mathsf{aw}{\uparrow}: \quad \dfrac{H'\{a\}}{H\left\{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow}\right\}}\,\rho \longrightarrow \dfrac{H'\left\{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow}\right\}}{H\{\mathsf{t}\}}\,\rho$$

▶ **Definition 16.** We define the rewriting system W as the rewriting system given by the reductions in Definition 15.

▶ **Theorem 17.** *The rewriting system* W *is terminating.*

Proof. By observing the corresponding flow reductions, it is easy to see that the non-trivial reductions of W remove edges of atomic flows. Since every application of a non-trivial reduction rule reduces the number of edges of the associated flow to a derivation, and the trivial rules reduce the number of rules below weakenings and above coweakenings, termination is clear. ◀

Note that the reductions of system W do not introduce atomic (co)contractions or medials. Thus, we get the following theorem.

▶ **Theorem 18.** *Given an* SKS *derivation ϕ from A to B not containing* ai*-cycles, we can perform decomposition.*

Proof. By applying system C followed by system W to ϕ. ◀

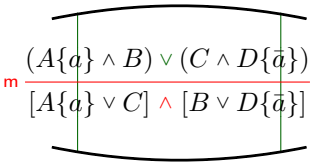
We will now present a local procedure to remove ai-cycles from derivations, effectively showing the independence of decomposition and cut-elimination.

3 The Cycle Elimination Procedure

3.1 Overview

For a cycle to occur in a proof, two edges of an atomic flow that were related by $\vee$ at the top of a connected component have to be connected by $\wedge$ at the bottom of the flow. Therefore, an instance of a rule that changes the main relation between formulae from $\vee$ to $\wedge$ needs to occur, containing the atoms involved in the cycle. In SKS, the only candidate is the medial rule.

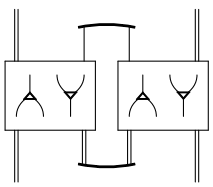
▶ **Definition 19.** A *critical medial* for a cycle is a medial that converts the link between the positive and negative atom from an $\vee$ to a $\wedge$. The picture below shows how this can happen: it depicts the simplest case, the flow containing the cycle can of course be more complicated.



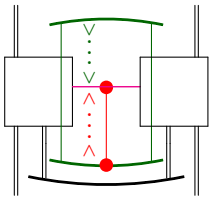
Following this observation, a strategy one can take to remove cycles becomes clear: we can permute the critical instances of the medial rule downwards (or upwards) in a proof. When the corresponding cut is reached, it is 'broken' by the critical medial, and the cycle can then be removed by performing standard deep inference rewriting techniques. It is by no means obvious that this suffices to remove cycles, but we will show that it in fact does.

▶ **Definition 20.** Following the steps below leads to the elimination of ai-cycles in a SKS derivation. An outline of the procedure is given and then the techniques in bold are explained in more detail below. Manipulations of derivations by means of the atomic flow that are not explicitly described follow [11].

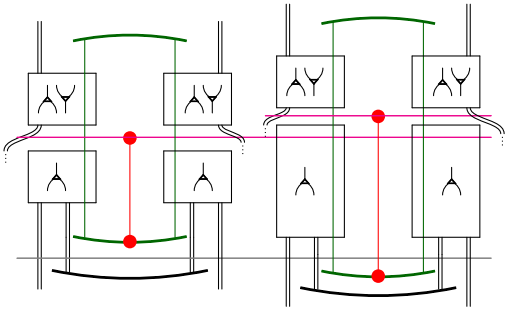
0. **Remove all trivialising units** (this is optional but simplifies step 5).
1. **Normalise every connected component** containing cycles to the following form, where double edges in an atomic flow represent an arbitrary number of edges and boxes represent free compositions of the vertices that they are labelled with:



2. Pick a normalised connected component that contains a cycle. Colour the edges of the cycle green.
3. Starting from the bottom, colour red each conjunction ($\wedge$) connecting two green edges until the critical medial for the cycle. In the same way, colour green each disjunction ($\vee$) connecting two green edges, working from the top of the cycle until the critical medial. We call the trace of $\wedge$-s the $\wedge$-*flow*.



We call the cut below the critical medial the *corresponding cut*.
4. Sequentialise the proof in such a way that the cut corresponding to the bottom critical medial is the top cut, that the cut corresponding the second critical medial from the bottom is the second cut from the top, and so on. **Remove the contractions below each critical medial**.



5. Apply the **transformation along the $\wedge$-flow** to each $\wedge$-flow. Each time apply the transformation to the most internal $\wedge$-flow. This is so no cuts are created and eventually all cycles are removed. The proof so created is broken, but will be fixed in the next phase.
6. Propagate (co)weakenings using the rewriting system W.
7. **Remove all trivialising units.**

3.2 Removing trivialising units

Apply the following transformations, then propagate the (co)-weakenings with rewriting system W:

$$\mathsf{f} \wedge A \quad \longrightarrow \quad = \dfrac{\mathsf{f} \wedge \mathsf{w}\uparrow \dfrac{A}{\mathsf{t}}}{\mathsf{aw}\downarrow \dfrac{\mathsf{f}}{A}} \qquad \text{and} \qquad \mathsf{t} \vee A \quad \longrightarrow \quad = \dfrac{\mathsf{w}\uparrow \dfrac{A}{\mathsf{t}}}{\mathsf{t} \vee \mathsf{aw}\downarrow \dfrac{\mathsf{f}}{A}}$$

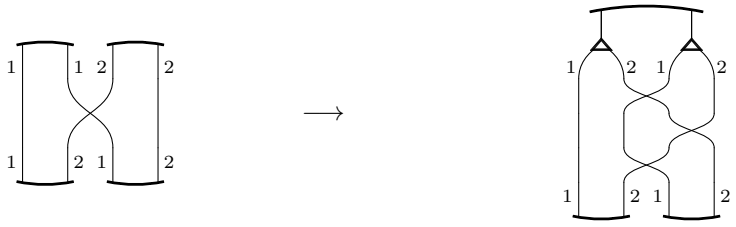
3.3 Normalise every connected component

To normalise the connected component containing a cycle, we need to collect the identities together. We will show how to do so in the case of two identities; the general case is similar.

Note that the atoms are indexed to ease understanding.

$$\dfrac{\dfrac{}{H\{\mathsf{t}\}\{a_1 \vee \bar{a}_1\}}\,\mathsf{ai}\!\downarrow \quad \dfrac{}{G\{a_1 \wedge \bar{a}_2\}\{\bar{a}_1 \wedge a_2\}}\,\mathsf{ai}\!\uparrow}{A \,\|\, B} \longrightarrow \left(\dfrac{\dfrac{\dfrac{}{\mathsf{t}}\,\mathsf{ai}\!\downarrow}{\left[\dfrac{a}{a_1 \wedge a_2}\,\mathsf{ac}\!\uparrow\right] \vee \left[\dfrac{\bar{a}}{\bar{a}_1 \wedge \bar{a}_2}\,\mathsf{ac}\!\uparrow\right]}}{[a_1 \vee \bar{a}_1] \wedge [a_2 \vee \bar{a}_2]}\,\mathsf{m} \wedge H\{\mathsf{t}\}\{\mathsf{t}\}\right)$$

$$\dfrac{\dfrac{\dfrac{}{H\{a_1 \vee \bar{a}_1\}\{a_2 \vee \bar{a}_2\}}\,\star}{G\{a_1 \wedge \bar{a}_2\}\{\bar{a}_1 \wedge a_2\}}\,\mathsf{s}}{A \,\|\, B}$$

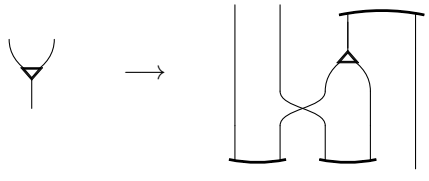


The part of the right-hand derivation labelled with a star is a standard SKS derivation, it can be found in [11, Lemma 2.3.8]. After applying this transformation, the W rewriting system can then be used to get the component into the required form.

3.4 Removing contractions below a critical medial

One could push the contractions through the proof until they are no longer in the wrong place, but, as with any procedure that pushes contractions around, this can have an exponential complexity cost. A simple way to avoid this risk is to convert deviant contractions to cocontractions in the following way, after which the connected component will need to be renormalised, as above.

$$\dfrac{a \vee a}{a}\,\mathsf{ac}\!\downarrow \longrightarrow \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{t}}}{[a \vee a] \wedge \dfrac{a}{a \wedge a} \vee a}}{[a \vee a] \wedge (\bar{a} \wedge a)}\,\mathsf{s}}{\dfrac{a \wedge \bar{a}}{\mathsf{f}} \vee \dfrac{a \wedge \bar{a}}{\mathsf{f}} \vee a}\,2\mathsf{s}}$$



3.5 Transformation along the ∧-flow

To perform the transformation, we first translate the derivation into sequential form (a dotted line in an inference rule denotes synchronal composition of derivations). We can then

transform along the $\wedge$*-flow* in the following fashion:

$$
\begin{array}{c}
= \dfrac{\chi}{\mathsf{m}\dfrac{(A\{a\} \wedge B) \vee (C \wedge D\{\bar{a}\})}{[A\{a\} \vee C] \wedge [B \vee D\{\bar{a}\}]}} \\
= \dfrac{}{\rho_1 \dfrac{K\{E_1\{a\} \wedge F_1\{\bar{a}\}\}}{\vdots}} \\
\rho_{k-1} \dfrac{}{= \dfrac{K\{E_k\{a\} \wedge F_k\{\bar{a}\}\}}{H\left\{\mathsf{ai}{\uparrow}\dfrac{a \wedge \bar{a}}{\mathsf{f}}\right\}}} \\
= \dfrac{}{\psi}
\end{array}
\quad \longrightarrow \quad
\begin{array}{c}
= \dfrac{\chi}{K\left\{= \dfrac{(A\{a\} \wedge B) \vee (C \wedge D\{\bar{a}\})}{\left(\left[\dfrac{\mathsf{f}}{A\{a\}} \vee C\right] \wedge \left[\dfrac{\mathsf{f}}{B} \vee D\{\bar{a}\}\right]\right) \vee \left(\left[A\{a\} \vee \dfrac{\mathsf{f}}{C}\right] \wedge \left[B \vee \dfrac{\mathsf{f}}{D\{\bar{a}\}}\right]\right)}\right\}} \\
= \dfrac{}{K\{(E_1\{a\} \wedge F_1\{\bar{a}\}) \vee (E_1\{a\} \wedge F_1\{\bar{a}\})\}} \\
\phi_1 \,\| \\
\vdots \\
\phi_{k-1} \,\| \\
= \dfrac{K\{(E_k\{a\} \wedge F_k\{\bar{a}\}) \vee (E_k\{a\} \wedge F_k\{\bar{a}\})\}}{H\left\{\neq \dfrac{\left(a \wedge \mathsf{aw}{\uparrow}\dfrac{\bar{a}}{\mathsf{t}}\right) \vee \left(\mathsf{aw}{\uparrow}\dfrac{a}{\mathsf{t}} \wedge \bar{a}\right)}{\mathsf{f}}\right\}} \\
= \dfrac{}{\psi}
\end{array}
$$

Each inference rule ρ_i on the left yields a derivation ϕ_i as follows. If ρ_i only affects the context $K_i\{\,\}$ or $E_i\{a\} \wedge F_i\{a\}$ then ϕ_i is trivial. Therefore we are left with four non-trivial cases, where P and $\bar{P}$ represent $E_i\{a\}$ and $E_i\{\bar{a}\}$, and Q and $\bar{Q}$ represent $F_i\{a\}$ and $F_i\{\bar{a}\}$, respectively:

$$
= \frac{(P \wedge Q) \wedge R}{P \wedge (Q \wedge R)} \quad \longrightarrow \quad \mathsf{2s}\frac{[(P \wedge Q) \vee (P \wedge Q)] \wedge \mathsf{c}{\uparrow}\dfrac{R}{R \wedge R}}{= \dfrac{(P \wedge Q) \wedge R}{P \wedge (Q \wedge R)} \vee = \dfrac{(P \wedge Q) \wedge R}{P \wedge (Q \wedge R)}}
$$

$$
= \frac{P \wedge (Q \wedge R)}{(P \wedge Q) \wedge R} \quad \longrightarrow \quad \mathsf{m}\frac{= \dfrac{P \wedge (Q \wedge R)}{(P \wedge Q) \wedge R} \vee = \dfrac{P \wedge (Q \wedge R)}{(P \wedge Q) \wedge R}}{[(P \wedge Q) \vee (P \wedge Q)] \wedge \mathsf{c}{\downarrow}\dfrac{R \vee R}{R}}
$$

$$
\mathsf{m}\frac{(P \wedge Q) \vee (R \wedge S)}{[P \vee R] \wedge [Q \vee S]} \quad \longrightarrow \quad = \frac{[(P \wedge Q) \vee (P \wedge Q)] \vee (R \wedge S)}{\left(\left[P \vee \dfrac{\mathsf{f}}{R}\right] \wedge \left[Q \vee \dfrac{\mathsf{f}}{S}\right]\right) \vee \mathsf{m}\dfrac{(P \wedge Q) \vee (R \wedge S)}{[P \vee R] \wedge [Q \vee S]}}
$$

$$
\mathsf{s}\frac{P \wedge [Q \vee R]}{(P \wedge Q) \vee R} \quad \longrightarrow \quad = \frac{\mathsf{s}\dfrac{P \wedge [Q \vee R]}{(P \wedge Q) \vee R} \vee \mathsf{s}\dfrac{P \wedge [Q \vee R]}{(P \wedge Q) \vee R}}{[(P \wedge Q) \vee (P \wedge Q)] \vee \mathsf{c}{\downarrow}\dfrac{R \vee R}{R}}
$$

The transformed derivation is valid except for the bottom inference rule, labelled $\neq$, which has premise t and conclusion f. Since a and $\bar{a}$ are not involved in contraction steps (we have removed contractions below the critical medial), the weakening steps that generate a and $\bar{a}$ can be propagated down to the invalid inference rule, converting it to the equality $= \dfrac{\mathsf{f} \wedge \mathsf{f}}{\mathsf{f}}$.

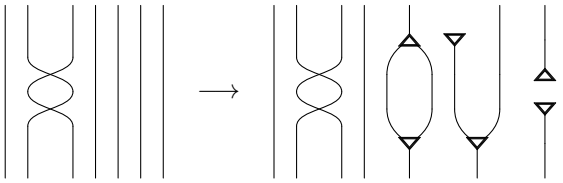


Figure 5 Possible changes to edges after eliminating cycles.

3.6 Termination of the Procedure

▶ **Theorem 21.** *The procedure described in Definition 20 terminates, removing all cycles.*

Proof. To see that this procedure to eliminate ai-cycles terminates is straightforward: after each transformation removing a critical medial there is one less cycle, and no new ai-cycles are created in the process. This fact is easy to check: no new connections between existing edges and no new cuts are created through this procedure and so the edges connected by a cut-rule after the procedure were already connected by a cut-rule before the procedure. ◀

Thus, from Theorem 21 and Theorem 18, we have that any SKS derivation ϕ from A to B can be decomposed without performing cut elimination.

4 Complexity, Confluence and Open Problems

During the transformation along the $\wedge$ flow, the only possible changes to edges outside the cycle are as follows:

- Edges may be bifurcated and then reconnected, creating a sausage.
- Edges might be joined by another edge which originates from a (co)weakening.
- Edges might be disconnected by weakenings.

Each of these three possibilities is respectively shown in Figure 5 by the three edges to the right of the flow. In the worst case, bifurcation, this incurs a linear cost in the number of inference rules. Furthermore, in this case sausages are introduced in the flow of the proof, and thus the complexity of the decomposition procedure may be exponentially increased as seen in Figure 4.

Remarkably, what creates the sausage through bifurcation is two opposite instances of the associativity rule: in Figure 6, this happens in a completely redundant section of proof. This leads to the counter-intuitive conclusion that two proofs with cut, equivalent modulo equality rules, can have an exponential difference in the size of their cut-free proofs when cut elimination involves the above procedure. Clearly, we could avoid the sausage by simply eliminating the associativity rules in the left-hand proof. It is not known whether there are proofs where this cannot be done in a way that does not change the proof in a semantically significant way. An extended version of this paper is being written for journal publication, in which complexity issues will be addressed comprehensively.

Another less startling observation is that the transformation along the $\wedge$-flow is non confluent. This can be seen in the third non-trivial case of converting p_i to ϕ_i, involving medial. On the left, weakenings are introduced, but not on the right. Obviously, there is no reason why this should not be the other way around, and so non-confluence is introduced.

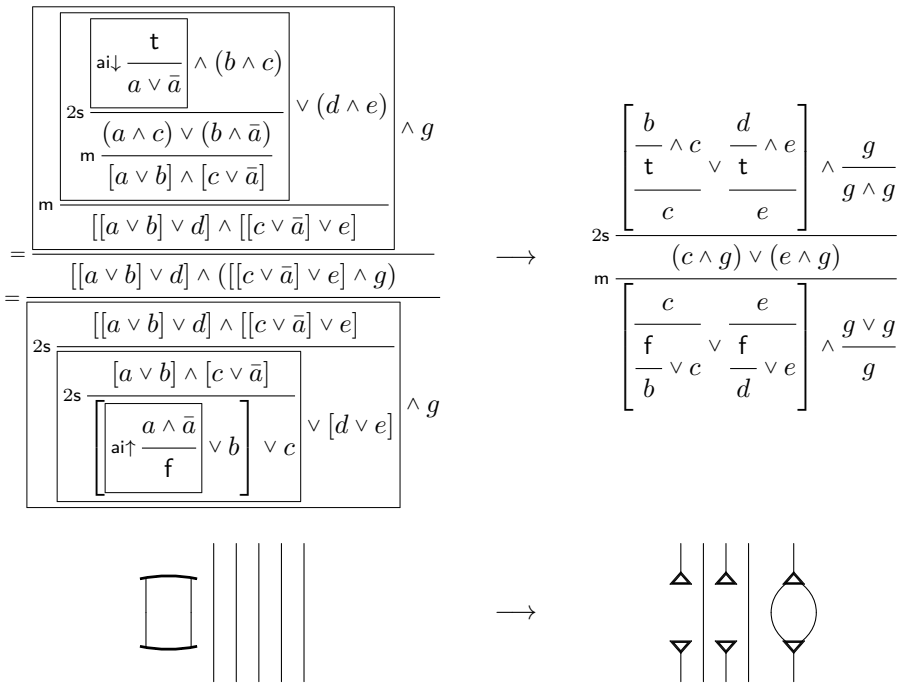


Figure 6 Potential creation of complexity by consecutive associativity rules

References

1 Paola Bruscoli and Alessio Guglielmi. On the proof complexity of deep inference. *ACM Transactions on Computational Logic (TOCL)*, 10(2):14, 2009. doi:10.1145/1462179.1462186.

2 Paola Bruscoli, Alessio Guglielmi, Tom Gundersen, and Michel Parigot. Quasipolynomial normalisation in deep inference via atomic flows and threshold formulae. *Logical Methods in Computer Science*, 12(1):5:1–30, 2016. URL: http://cs.bath.ac.uk/ag/p/QuasiPolNormDI.pdf, doi:10.2168/LMCS-12(2:5)2016.

3 Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2250, pages 347–361. Springer, 2001. doi:10.1007/3-540-45653-8_24.

4 Samuel R Buss. The undecidability of k-provability. *Annals of Pure and Applied Logic*, 53(1):75–102, 1991. doi:10.1016/0168-0072(91)90059-U.

5 Samuel R Buss. Some remarks on lengths of propositional proofs. *Archive for Mathematical Logic*, 34(6):377–394, 1995. doi:10.1007/BF02391554.

6 Alessandra Carbone. The cost of a cycle is a square. *The Journal of Symbolic Logic*, 67(01):35–60, 2002. doi:10.2178/jsl/1190150028.

7 Anupam Das. On the relative proof complexity of deep inference via atomic flows. *Logical Methods in Computer Science*, 11(1):4:1–27, 2015. URL: http://arxiv.org/pdf/1502.05860.pdf, doi:10.2168/LMCS-11(1:4)2015.

8 Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1):9:1–36, 2008. URL: http://arxiv.org/pdf/0709.1205.pdf, doi:10.2168/LMCS-4(1:9)2008.

9 Alessio Guglielmi, Tom Gundersen, and Michel Parigot. A proof calculus which reduces syntactic bureaucracy. In *21st International Conference on Rewriting Techniques and Applications (RTA)*, volume 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 135–150. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2010. URL: `http://drops.dagstuhl.de/opus/volltexte/2010/2649`, `doi:10.4230/LIPIcs.RTA.2010.135`.

10 Alessio Guglielmi, Tom Gundersen, and Lutz Straßburger. Breaking paths in atomic flows for classical logic. In *Logic in Computer Science (LICS), 2010 25th Annual IEEE Symposium on*, pages 284–293. IEEE, 2010. URL: `http://www.lix.polytechnique.fr/~lutz/papers/AFII.pdf`, `doi:10.1109/LICS.2010.12`.

11 Tom Erik Gundersen. *A general view of normalisation through atomic flows*. Thesis, University of Bath, 2009. URL: `https://tel.archives-ouvertes.fr/file/index/docid/509241/filename/thesis.pdf`.

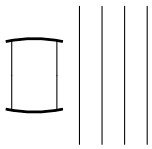
12 Emil Jeřábek. Proof complexity of the cut-free calculus of structures. *Journal of Logic and Computation*, 19(2):323–339, 2008. URL: `http://users.math.cas.cz/~jerabek/papers/cos.pdf`, `doi:10.1093/logcom/exn054`.

13 Andrea Aler Tubella. *A study of normalisation through subatomic logic*. Thesis, University of Bath, 2017. URL: `https://arxiv.org/pdf/1703.10258.pdf`.

A An example of cycle elimination

The following proof has a cycle in it, specifically in the atom a. The critical medial is coloured red.

$$= \dfrac{\mathsf{2s}\dfrac{\mathsf{m}\dfrac{\mathsf{ai}{\downarrow}\dfrac{\mathsf{t}}{a \vee \bar{a}} \wedge (b \wedge c)}{(a \wedge c) \vee (b \wedge \bar{a})} \wedge (d \wedge e)}{[a \vee b] \wedge [c \vee \bar{a}]}}{\mathsf{2s}\dfrac{\mathsf{s}\dfrac{[a \vee b] \wedge d}{a \vee (b \wedge d)} \wedge \mathsf{s}\dfrac{[\bar{a} \vee c] \wedge e}{\bar{a} \vee (c \wedge e)}}{\mathsf{ai}{\uparrow}\dfrac{a \wedge \bar{a}}{\mathsf{f}} \vee [(b \wedge d) \vee (c \wedge e)]}}$$

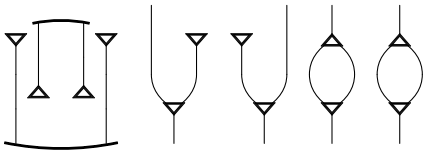


We perform the cycle removal procedure on this proof. First, we perform the transformation along the $\wedge$-flow:

$$\mathsf{2s}\,\frac{\mathsf{2s}\,\dfrac{\mathsf{ai}{\downarrow}\,\dfrac{\mathsf{t}}{a\vee\bar{a}}\wedge(b\wedge c)}{(a\wedge c)\vee(b\wedge\bar{a})}}{\left(\left[\mathsf{aw}{\downarrow}\,\dfrac{\mathsf{f}}{a}\vee b\right]\wedge\left[\mathsf{aw}{\downarrow}\,\dfrac{\mathsf{f}}{c}\vee\bar{a}\right]\right)\vee\left(\left[a\vee\mathsf{aw}{\downarrow}\,\dfrac{\mathsf{f}}{b}\right]\wedge\left[c\vee\mathsf{aw}{\downarrow}\,\dfrac{\mathsf{f}}{\bar{a}}\right]\right)}\wedge\mathsf{c}{\downarrow}\,\frac{d\wedge e}{(d\wedge e)\wedge(d\wedge e)}$$

$$=\frac{\dfrac{([a\vee b]\wedge[c\vee\bar{a}])\wedge(d\wedge e)}{\mathsf{s}\,\dfrac{[a\vee b]\wedge d}{a\vee(b\wedge d)}\wedge\mathsf{s}\,\dfrac{[c\vee\bar{a}]\wedge e}{\bar{a}\vee(c\wedge e)}}}{\mathsf{2s}\,\dfrac{}{(a\wedge\bar{a})\vee[(b\wedge d)\vee(c\wedge e)]}}\vee\frac{\dfrac{([a\vee b]\wedge[c\vee\bar{a}])\wedge(d\wedge e)}{\mathsf{s}\,\dfrac{[a\vee b]\wedge d}{a\vee(b\wedge d)}\wedge\mathsf{s}\,\dfrac{[c\vee\bar{a}]\wedge e}{\bar{a}\vee(c\wedge e)}}}{\mathsf{2s}\,\dfrac{}{(a\wedge\bar{a})\vee[(b\wedge d)\vee(c\wedge e)]}}$$

$$=\frac{}{\neq\dfrac{\left(a\wedge\mathsf{aw}{\uparrow}\,\dfrac{\bar{a}}{\mathsf{t}}\right)\vee\left(\mathsf{aw}{\uparrow}\,\dfrac{a}{\mathsf{t}}\wedge\bar{a}\right)}{\mathsf{f}}\vee\mathsf{c}{\downarrow}\,\dfrac{[(b\wedge d)\vee(c\wedge e)]\vee[(b\wedge d)\vee(c\wedge e)]}{(b\wedge d)\vee(c\wedge e)}}$$

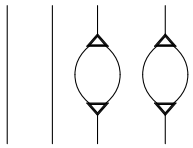


Note that sausages have been created in the atomic flow, a possible source of complexity.

Next we push the weakenings through, using the rewriting system W.

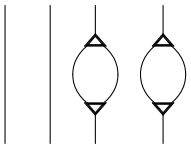
$$\mathsf{2s}\,\frac{=\dfrac{\mathsf{2s}\,\dfrac{=\dfrac{\mathsf{t}}{\mathsf{t}\vee\mathsf{t}}\wedge(b\wedge c)}{(\mathsf{t}\wedge c)\vee(b\wedge\mathsf{t})}}{([\mathsf{f}\vee b]\wedge[\mathsf{f}\vee\mathsf{t}])\vee([\mathsf{t}\vee\mathsf{f}]\wedge[c\vee\mathsf{f}])}\wedge\mathsf{c}{\downarrow}\,\dfrac{d\wedge e}{(d\wedge e)\wedge(d\wedge e)}}{=\dfrac{([\mathsf{f}\vee b]\wedge[\mathsf{f}\vee\mathsf{t}])\wedge(d\wedge e)}{\mathsf{s}\,\dfrac{[\mathsf{f}\vee b]\wedge d}{\mathsf{f}\vee(b\wedge d)}\wedge\mathsf{s}\,\dfrac{[\mathsf{f}\vee\mathsf{t}]\wedge e}{\mathsf{t}\vee(\mathsf{f}\wedge e)}}\vee\mathsf{2s}\,\dfrac{([\mathsf{t}\vee\mathsf{f}]\wedge[c\vee\mathsf{f}])\wedge(d\wedge e)}{\mathsf{s}\,\dfrac{[\mathsf{t}\vee\mathsf{f}]\wedge d}{\mathsf{t}\vee(\mathsf{f}\wedge d)}\wedge\mathsf{s}\,\dfrac{[c\vee\mathsf{f}]\wedge e}{\mathsf{f}\vee(c\wedge e)}}}{\mathsf{2s}\,\dfrac{}{(\mathsf{f}\wedge\mathsf{t})\vee[(b\wedge d)\vee(\mathsf{f}\wedge e)]}\qquad\mathsf{2s}\,\dfrac{}{(\mathsf{t}\wedge\mathsf{f})\vee[(\mathsf{f}\wedge d)\vee(c\wedge e)]}}$$

$$=\frac{}{=\dfrac{(\mathsf{f}\wedge\mathsf{t})\vee(\mathsf{t}\wedge\mathsf{f})}{\mathsf{f}}\vee\mathsf{m}\,\dfrac{=\dfrac{[(b\wedge d)\vee(\mathsf{f}\wedge e)]\vee[(\mathsf{f}\wedge d)\vee(c\wedge e)]}{(b\wedge d)\vee(\mathsf{f}\wedge d)}}{=\dfrac{b\vee\mathsf{f}}{b}\wedge\mathsf{ac}{\downarrow}\,\dfrac{d\vee d}{d}}\vee\mathsf{m}\,\dfrac{(\mathsf{f}\wedge e)\vee(c\wedge e)}{=\dfrac{\mathsf{f}\vee c}{c}\wedge\mathsf{ac}{\downarrow}\,\dfrac{e\vee e}{e}}}$$



We can simplify this proof using unit equations:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{t}}{\mathsf{t}\vee\mathsf{t}}=\wedge(b\wedge c)}{b\vee c}\,2\mathsf{s}\quad \wedge\,\dfrac{d\wedge e}{(d\wedge e)\wedge(d\wedge e)}\,\mathsf{c}\!\downarrow}{(b\wedge(d\wedge e))\wedge\dfrac{[\mathsf{t}\vee\mathsf{f}]\wedge e}{\mathsf{t}\vee(\mathsf{f}\wedge e)}\,\mathsf{s}\vee(c\wedge(d\wedge e))}\,2\mathsf{s}}{(b\wedge d)\vee\dfrac{[\mathsf{t}\vee\mathsf{f}]\wedge d}{\mathsf{t}\vee(\mathsf{f}\wedge d)}\,\mathsf{s}\vee(c\wedge e)}\,\mathsf{s}}{\dfrac{(b\wedge d)\vee(\mathsf{f}\wedge d)}{b\vee\mathsf{f}}=\wedge\,\dfrac{d\vee d}{d}\,\mathsf{ac}\!\downarrow\vee\dfrac{(\mathsf{f}\wedge e)\vee(c\wedge e)}{\mathsf{f}\vee c}=\wedge\,\dfrac{e\vee e}{e}\,\mathsf{ac}\!\downarrow}\,\mathsf{m}}$$



We can simplify even further by removing the trivialising units. Note that this eliminates the 'sausages' that appeared after the transformation along the $\wedge$-flow.

$$\dfrac{\dfrac{\dfrac{\mathsf{t}}{\mathsf{t}\vee\mathsf{t}}=\wedge(b\wedge c)}{b\vee c}\,2\mathsf{s}\quad\wedge(d\wedge e)}{(b\wedge d)\vee(c\wedge e)}\,2\mathsf{s}$$