Combinatorial Flows as Bicolored Atomic Flows

Giti Omidvar, Lutz Straßburger

Combinatorial Flows as Bicolored Atomic Flows

Giti Omidvar and Lutz Straßburger

Inria Saclay and Ecole Polytechnique Paris, France

Abstract. We introduce combinatorial flows as a graphical representation of proofs. They can be seen as a generalization of atomic flows on one side and of combinatorial proofs on the other side. From atomic flows, introduced by Guglielemi and Gundersen, they inherit the close correspondence with open deduction and the possibility of tracing the occurrences of atoms in a derivation. From combinatorial proofs, introduced by Hughes, they inherit the correctness criterion that allows to reconstruct the derivation from the flow. In fact, combinatorial flows form a proof system in the sense of Cook and Reckhow. We show how to translate between open deduction derivations and combinatorial flows, and we show how they are related to combinatorial proofs with cuts.

Keywords: Proof identity · Proof invariants · Combinatorial flows · Open deduction.

1 Introduction

The question of when two proofs are the same is older than proof theory itself, as it has been posed by Hilbert [31] as early as 1900. Nowadays there are essentially two approaches to the problem. The first is to find suitable proof transformations and postulate that two proofs are the same if they can be transformed into each other using these transformations. This can be achieved via proof normalization [24] or rule permutations [21]. The second way is to define suitable canonical proof representations. The most prominent examples are λ-terms [10], proof nets [9], and combinatorial proofs [18].

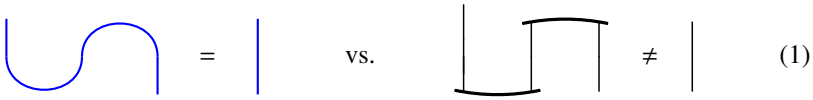
A recurring theme in this second setting is the idea to trace the occurrences of formulas or atoms inside the proofs. It has first been used by Kelly and Maclane in [20] (via *coherence graphs*) to determine the identity of morphisms in a category. This idea has then been rediscovered in Girard's *proof nets* [7,9] and is also used nowadays in *string diagrams* [27]. All these notions—proof nets, coherence graphs, string diagrams—work remarkably well in the linear setting, where no contraction or weakening is present. Indeed, proof nets form a canonical representation for multiplicative linear logic (MLL) [9].

In a classical setting, where contraction and weakening are present, the idea of tracing formulas in a derivation has first been used by Buss [3] in the form of *logical flow graphs*, which have later been studied by Carbone [4,5] to investigate the relation between cuts and contractions and cycles in these flow graphs.

To get a more precise grip on these cycles—which do not exist in the linear setting—and to eliminate them, Guglielmi and Gundersen developed *atomic flows* [11] as a refinement of logical flow graphs, by only tracing the atoms and by completely detaching

the flow from the derivation. In this way, atomic flows are visually closer to coherence graphs and string diagrams, they can be used as invariants for proofs and they can be used to devise normalization procedures for deep inference derivations [11,13,32]. However, atomic flows have two major drawbacks:

1. *We cannot read back a proof from an atomic flow.* They lose too much information about the proof. Not only do they *not* form a canonical proof representation, there is also no polynomial correctness criterion.[1] Therefore they do not form a proof system in the sense of Cook and Reckhow [6].
2. *Yanking is not possible in atomic flows.* One of the main advantages of coherence graphs or string diagrams is that they can abstract away from superfluous "bends". This is indicated on the left below, where we simply can "pull" the ends of the wire. In MLL, this process corresponds to cut elimination [17].



(1)

   In atomic flows, this situation is represented as on the right above, where we cannot straighten the edge. The reason lies in the interference of contraction with cut elimination.

The purpose of this paper is to address these two problems. In fact, the first problem has already been solved by *combinatorial proofs* [18], but for too high a price: a total separation of the linear part and the resource management part of the proof. Performing this separation is as expensive as cut elimination, and therefore leads to a size explosion.

Our idea here is to use two colors in the atomic flows, to distinguish between the linear parts that can be yanked (blue) and the resource management parts that cannot be yanked (purple). This is similar to what happens in combinatorial proofs, but is less restrictive, as the two parts can be mixed—there is no global separation between the linear part and the resource management. In fact, what we present here can be seen as a merging of atomic flows and combinatorial proofs, and we use the term *combinatorial flows*[2].

Special care has to be taken with respect to the units, as atomic flows and combinatorial proofs are unit-free.[3] However, in order to establish a proper correspondence with the deep inference proof system, units are necessary.

This paper is organized as follows. In the next two sections we recall basic notions of deep inference and show how derivations are translated into objects that we call *preflows*, as the correctness criterion is not yet established. To present the correctness criterion, we recall in Section 4 Retoré's *RB-cographs*. Then, in Sections 5 and 6 we define the correctness for the linear parts and the resource management parts, respectively. Finally, in Section 7 we define combinatorial flows as correct preflows. In Section 8, we introduce *purification* which is a form of normalization via rewriting that simplifies the

[1] Das has shown in [8] that no such criterion is possible, under the assumption that integer factoring is hard for *P/poly*.

[2] This term has already been used in [29] for compositions of combinatorial proofs. Here we are more general: the components of our combinatorial flows are not combinatorial proofs.

[3] Combinatorial proofs can be presented with units [18] but then the units are treated as atoms.

flow by eliminating irrelevant parts. Finally, in Section 9, we establish the relation between our combinatorial flows and Hughes' combinatorial proof with cuts, as they are presented in [19].

2 Preliminaries on Open Deduction

Open deduction [12] is a deep inference formalism [14], allowing to write derivations in a way such that the same operations that are used to build formulas from atoms are also used to build derivations from inference rules.

We define the ***formulas*** of classical propositional logic, denoted by $A, B, \dots$ to be generated from a countable set of propositional variables $\{a, b, \dots\}$ and their negations $\{\overline{a}, \overline{b}, \dots\}$ with the following grammar:

$$A, B := \mathsf{t} \mid \mathsf{f} \mid a \mid \overline{a} \mid A \vee B \mid A \wedge B$$

Negation can be also extended to all formulas via De Morgan laws:

$$\overline{\mathsf{t}} = \mathsf{f} \qquad \overline{\mathsf{f}} = \mathsf{t} \qquad \overline{\overline{a}} = a \qquad \overline{A \wedge B} = \overline{A} \vee \overline{B} \qquad \overline{A \vee B} = \overline{A} \wedge \overline{B}$$

We define the equivalence relation $\equiv$ on formulas:

$$\begin{array}{llll} A \wedge B \equiv B \wedge A & (A \wedge B) \wedge C \equiv A \wedge (B \wedge C) & A \wedge \mathsf{t} \equiv A & \mathsf{t} \vee \mathsf{t} \equiv \mathsf{t} \\ A \vee B \equiv B \vee A & (A \vee B) \vee C \equiv A \vee (B \vee C) & A \vee \mathsf{f} \equiv A & \mathsf{f} \wedge \mathsf{f} \equiv \mathsf{f} \end{array} \tag{2}$$

A formula A is a ***unit*** if $A = \mathsf{t}$ or $A = \mathsf{f}$. A formula A is ***unit-free*** if it does not contain any units. A formula A is ***pure*** if either $A \equiv \mathsf{t}$ or $A \equiv \mathsf{f}$ or $A \equiv A'$ for some unit-free formula A'. An ***atom*** is an element in $\mathcal{A} = \{a, b, \dots\} \cup \{\overline{a}, \overline{b}, \dots\}$, and a ***sequent*** $\Gamma = A_1, \dots, A_n$ is a finite non-empty multiset of formulas.

Figure 1 shows the inference rules of system SKS [2]. They have to be understood as rule schemas, where a can stand for an arbitrary atom and A, B, C, D for arbitrary formulas. We call an ***inference system*** any such set of inference rules. The rules can be composed to ***derivations***, which are defined inductively below, and which are denoted as $\mathcal{D} \begin{Vmatrix} A \\ B \end{Vmatrix} \mathsf{S}$ where A is the ***premise*** and B is the ***conclusion*** of the derivation $\mathcal{D}$, and S is the set of inference rules used in $\mathcal{D}$.

1. A formula A is a derivation with premise A and conclusion A.
2. Every inference rule $\dfrac{A_1 \quad A_2}{B}\,\rho$ in S, is a derivation with premise A and conclusion B.
3. If $\mathcal{D}_1 \begin{Vmatrix} B_1 \\ \end{Vmatrix} \mathsf{S}$ and $\mathcal{D}_2 \begin{Vmatrix} B_2 \\ \end{Vmatrix} \mathsf{S}$ are derivations, then the compositions $\mathcal{D}_1 \wedge \mathcal{D}_2$ and $\mathcal{D}_1 \vee \mathcal{D}_2$ are derivations and denoted as $\begin{matrix} A_1 & A_2 \\ \mathcal{D}_1 \| \mathsf{S} \wedge \mathcal{D}_2 \| \mathsf{S} \\ B_1 & B_2 \end{matrix}$ and $\begin{matrix} A_1 & A_2 \\ \mathcal{D}_1 \| \mathsf{S} \vee \mathcal{D}_2 \| \mathsf{S} \\ B_1 & B_2 \end{matrix}$, respectively.

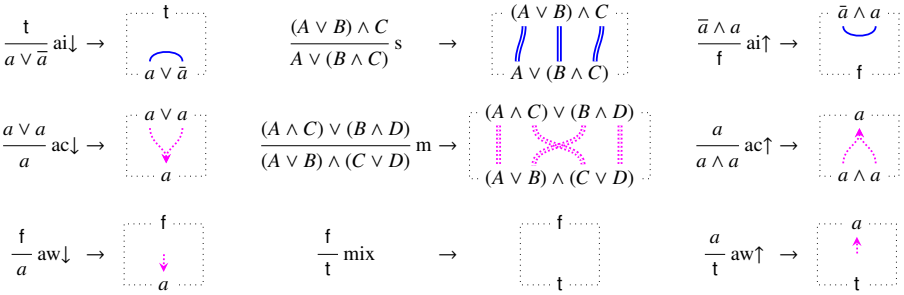


Fig. 1: Inference rules of system SKS and their translation into flowboxes

4. If $\mathcal{D}_1$ is a derivation with premise A_1 and conclusion B_1, and $\mathcal{D}_2$ is a derivation with premise A_2 and conclusion B_2, and $A_2 \equiv B_1$, we can compose $\mathcal{D}_1$ and $\mathcal{D}_2$ directly to $\mathcal{D}_1 \circ \mathcal{D}_2$ denoted as:

$$\begin{array}{c} A_1 \\ \mathcal{D}_1 \| \mathsf{s} \\ \dfrac{B_1}{A_2} \equiv \\ \mathcal{D}_2 \| \mathsf{s} \\ B_2 \end{array} \quad \text{or} \quad \begin{array}{c} A_1 \\ \mathcal{D}_1 \| \mathsf{s} \\ A_2 \\ \mathcal{D}_2 \| \mathsf{s} \\ B_2 \end{array} \quad \text{or} \quad \begin{array}{c} A_1 \\ \mathcal{D}_1 \| \mathsf{s} \\ B_1 \\ \mathcal{D}_2 \| \mathsf{s} \\ B_2 \end{array} \tag{3}$$

Figure 2 shows on the left an example of an SKS derivation.

3 From Derivations to Flows

Definition 3.1 We say that a set X is ***$\mathcal{A}$-labelled*** if it is equipped with a ***labelling function*** $\ell_X \colon X \to \mathcal{A}$, mapping each element $x \in X$ to an atom. A binary relation $\mathbb{B} \subseteq X \times X$ on an $\mathcal{A}$-labelled set X is ***well-matched*** if it is symmetric and we have that $x\mathbb{B}y$ implies $\ell_X(x) = \overline{\ell_X(y)}$. We say that $\mathbb{B}$ is ***perfectly matched*** if it is well-matched and for every $x \in X$ there is exactly one $y \in X$ with $x\mathbb{B}y$.

Definition 3.2 Let $\mathbb{R}$ and $\mathbb{S}$ be binary relations on $X \uplus Y$ and $Y \uplus Z$ where $X \cap Z = \emptyset$. The composition of the two relations $\mathbb{R}$ and $\mathbb{S}$ (denoted as $\mathbb{R} \circ \mathbb{S}$) is a binary relation on $X \uplus Z$ where $(a, c) \in \mathbb{R} \circ \mathbb{S}$ if and only if there exists a sequence $v_1, v_2, \ldots, v_n$ of elements in $X \uplus Y \uplus Z$ such that $a = v_1$, $c = v_n$, and for every $i \in \{1, \ldots, n-1\}$, we have $v_i \mathbb{R} v_{i+1}$ if i is odd and $v_i \mathbb{S} v_{i+1}$ if i is even.

For a formula A, we write $\lfloor A \rfloor$ to denote the set of leaves of the formula tree of A. This set is $\mathcal{A}$-labelled, with the labelling function $\ell_{\lfloor A \rfloor} \colon \lfloor A \rfloor \to \mathcal{A}$ mapping each leaf of the formula tree to the atom occurring in that position. Note that $A \equiv A'$ implies $\lfloor A \rfloor = \lfloor A' \rfloor$.

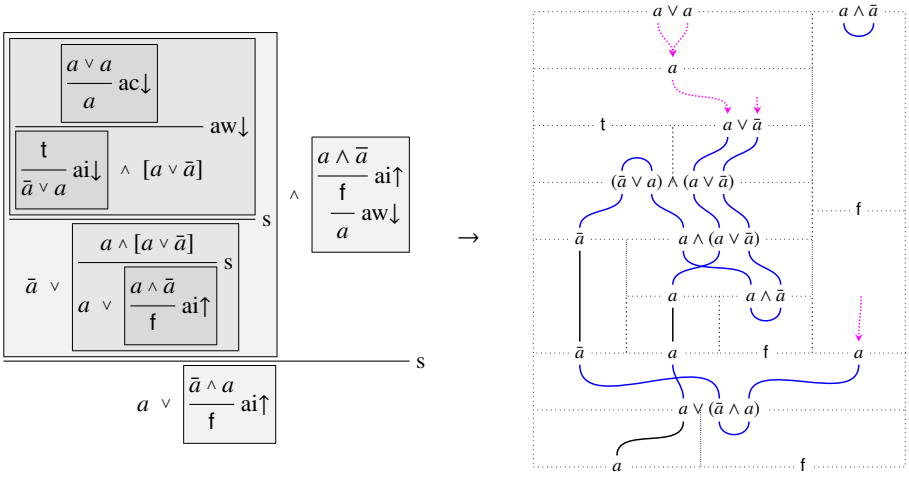
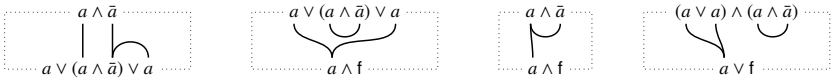


Fig. 2: A derivation from $(a \vee a) \wedge (a \wedge \overline{a})$ to $a \vee \mathsf{f}$ and its translation to a preflow.

Definition 3.3 A ***flowbox*** is a triple $\phi = \langle A, B, \mathbb{B}_\phi \rangle$, where A and B are formulas and $\mathbb{B}_\phi$ is a well-matched relation on $\lfloor \overline{A} \rfloor \uplus \lfloor B \rfloor$. For flowboxes $\phi = \langle A, B, \mathbb{B}_\phi \rangle$ and $\psi = \langle C, D, \mathbb{B}_\psi \rangle$, we define $\phi \wedge \psi = \langle A \wedge C, B \wedge D, \mathbb{B}_\phi \uplus \mathbb{B}_\psi \rangle$ and $\phi \vee \psi = \langle A \vee C, B \vee D, \mathbb{B}_\phi \uplus \mathbb{B}_\psi \rangle$, which are also flowboxes. If $\phi = \langle A, B_1, \mathbb{B}_\phi \rangle$ and $\psi = \langle B_2, C, \mathbb{B}_\psi \rangle$ are flowboxes with $B_1 \equiv B_2$, then $\phi \circ \psi = \langle A, C, \mathbb{B}_\phi \circ \mathbb{B}_\psi \rangle$ is also a flowbox.

Example 3.4 Here are some examples of flowboxes. We denote them by writing the formulas on top of each other and indicating the relation $\mathbb{B}$ by edges between the atom occurrences.



(4)

The vertical composition of the first two flowboxes results in the third flowbox.

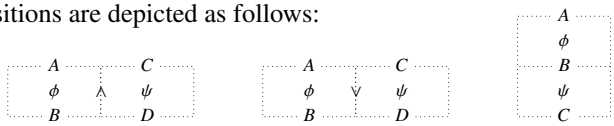
Observe for each formula there is an ***identity*** flowbox $\mathsf{id}_A = \langle A, A, \mathbb{B}_{\mathsf{id}} \rangle$, and that the operation $\circ$ on flowboxes is associative with id as unit. In fact, flowboxes form a category and are essentially the same as the B-nets of [22], and the operation $\circ$ corresponds to the cut elimination of B-nets. As already observed in [22], the composition $\circ$ forgets too much information. In general, we cannot recover a derivation from a flowbox. To gain more control, we define a formal compostion that keeps the structure.

Definition 3.5 We define the set of ***preflows*** to be inductively constructed as follows:

1. A flowbox $\phi = \langle A, B, \mathbb{B}_\phi \rangle$ is a preflow $\phi \colon A \vdash B$ with ***premise*** A and ***conclusion*** B.
2. If $\phi \colon A \vdash B$ and $\psi \colon C \vdash D$ are preflows, Then their ***horizontal compositions*** $\phi \mathbin{⦶} \psi \colon A \wedge C \vdash B \wedge D$ and $\phi \mathbin{⦷} \psi \colon A \vee C \vdash B \vee D$ are preflows.

3. Let $\phi\colon A \vdash B_1$ and $\psi\colon B_2 \vdash C$ be preflows with $B_1 \equiv B_2$, then the ***vertical composition*** $\phi \circledcirc \psi\colon A \vdash C$ is a preflow.

These compositions are depicted as follows:



As suggested by this graphical notation, we consider preflows to be equivalent modulo

$$\begin{array}{ll} (\phi \mathbin{⍓} \psi) \mathbin{⍓} \xi = \phi \mathbin{⍓} (\psi \mathbin{⍓} \xi) & (\phi \mathbin{⍓} \psi) \circledcirc (\xi \mathbin{⍓} \pi) = (\phi \circledcirc \xi) \mathbin{⍓} (\psi \circledcirc \pi) \\ (\phi \mathbin{⍒} \psi) \mathbin{⍒} \xi = \phi \mathbin{⍒} (\psi \mathbin{⍒} \xi) & (\phi \mathbin{⍒} \psi) \circledcirc (\xi \mathbin{⍒} \pi) = (\phi \circledcirc \xi) \mathbin{⍒} (\psi \circledcirc \pi) \end{array} \tag{5}$$

We can now translate derivations into preflows by translating inference rules into flowboxes as indicated in Figure 1—the colors and arows used in the figure will be explained later—and then use the operations ⍓, ⍒, and ◎ to compose them.

Definition 3.6 The ***translation*** T of a derivation $\mathcal{D}$, denoted as $\mathsf{T}(\mathcal{D})$, is the preflow inductively obtained as follows. If $\mathcal{D}$ is a formula A, then its translation is id_A. If $\mathcal{D}$ is a rule instance, then the translation is shown in Figure 1. Finally, $\mathsf{T}(\mathcal{D}_1 \wedge \mathcal{D}_2) = \mathsf{T}(\mathcal{D}_1) \mathbin{⍓} \mathsf{T}(\mathcal{D}_2)$ and $\mathsf{T}(\mathcal{D}_1 \vee \mathcal{D}_2) = \mathsf{T}(\mathcal{D}_1) \mathbin{⍒} \mathsf{T}(\mathcal{D}_2)$ and $\mathsf{T}(\mathcal{D}_1 \circ \mathcal{D}_2) = \mathsf{T}(\mathcal{D}_1) \circledcirc \mathsf{T}(\mathcal{D}_2)$.

Definition 3.7 We define the ***collapse*** $[\![\phi]\!]$ of a preflow ϕ to be the flowbox inductively obtained via $[\![\phi \mathbin{⍓} \psi]\!] = [\![\phi]\!] \wedge [\![\psi]\!]$ and $[\![\phi \mathbin{⍒} \psi]\!] = [\![\phi]\!] \vee [\![\psi]\!]$ and $[\![\phi \circledcirc \psi]\!] = [\![\phi]\!] \circ [\![\psi]\!]$.

In other words, the collapse "executes" the operations that are used to define preflows.

Example 3.8 The collapse of the preflow in Figure 2 is the rightmost flowbox in (4)

Clearly, preflows contain too much information—essentially the same as the open deduction derivation—and the collapse forgets too much of it. Atomic flows are somewhere in the middle. However they forget too much (we still cannot reconstruct the derivation) and too little (we cannot yank) at the same time.

To better control the information that can be removed, we propose here a solution that assigns colors (blue and purple) to flowboxes and only allows to collapse then if they have the same color. The idea is that the blue flowboxes encode linear (or multiplicative) behaviour, and therefore blue wires can be yanked. The purple flowboxes encode the resource management (or the additive behaviour) of the proof, and therefore purple wires cannot be yanked. [4]

Furthermore, both kinds of flowboxes obey different correctness criteria. To define these, we need to recall Retoré's RB-cographs [25,26].

[4] To indicate the color in a flowbox with empty $\mathbb{B}$ (because of weakening), we sometimes draw half arrows to the atoms, as done in Figure 1.

4 Preliminaries on RB-cographs

An ***(undirected) graph*** $\mathcal{G} = \langle V_\mathcal{G}, R_\mathcal{G}\rangle$ is a set of vertices $V_\mathcal{G}$ accompanied with a binary edge relation $R_\mathcal{G} \subseteq V_\mathcal{G} \times V_\mathcal{G}$ which is irreflexive and symmetric. We omit the index $\mathcal{G}$ in $V_\mathcal{G}$ and $R_\mathcal{G}$ when it is clear from the context. All graphs in this paper have an ***$\mathcal{A}$-labelled*** vertex set. For two graphs $\mathcal{G}$ and $\mathcal{H}$ we define their ***disjoint union*** as $\mathcal{G} \uplus \mathcal{H} = \langle V_\mathcal{G} \cup V_\mathcal{H}, R_\mathcal{G} \cup R_\mathcal{H}\rangle$ and their ***join*** as $\mathcal{G} \bowtie \mathcal{H} = \langle V_\mathcal{G} \cup V_\mathcal{H}, R_\mathcal{G} \cup R_\mathcal{H} \cup \{(u,v),(v,u) \mid u \in V_\mathcal{G}, v \in V_\mathcal{H}\}\rangle$. The ***complement*** of $\mathcal{G}$ is $\overline{\mathcal{G}} = \langle V_\mathcal{G}, \{(v,u) \mid (v,u) \notin R_\mathcal{G} \text{ and } v \neq u\}\rangle$. The labels are preserved for $\mathcal{G} \uplus \mathcal{H}$ and $\mathcal{G} \bowtie \mathcal{H}$ and negated in $\overline{\mathcal{G}}$. A graph $\mathcal{G}$ is a ***cograph*** if and only if it is constructed from single vertices using the operations $\uplus$ and complement.

Definition 4.1 The ***graph of*** a formula A, denoted as $\mathcal{G}(A)$, is defined inductively via $\mathcal{G}(\mathsf{t}) = \mathcal{G}(\mathsf{f}) = \langle \emptyset, \emptyset\rangle$ (the empty graph), $\mathcal{G}(a) = \langle \{\bullet_a\}, \emptyset\rangle$ (a single vertex graph whose vertex is labelled by a), and $\mathcal{G}(B \vee C) = \mathcal{G}(B) \uplus \mathcal{G}(C)$, and $\mathcal{G}(B \wedge C) = \mathcal{G}(B) \bowtie \mathcal{G}(C)$.

This means that for every formula A, we have that $V_{\mathcal{G}(A)} = \lfloor A \rfloor$, with the same labelling function. We immediately have the following properties of this translation:

Proposition 4.2 *A graph $\mathcal{G}$ is a cograph if and only if it is isomorphic to $\mathcal{G}(A)$ for some pure formula A. And for all pure formulas A, we have $\overline{\mathcal{G}(A)} = \mathcal{G}(\overline{A})$.*

Proposition 4.3 *If A is a unit-free formula, then $A \equiv B$ iff $\mathcal{G}(A) = \mathcal{G}(B)$.*

Definition 4.4 An ***RB-graph*** is a triple $\mathcal{G} = \langle V_\mathcal{G}, R_\mathcal{G}, \mathbb{B}_\mathcal{G}\rangle$ such that $\langle V_\mathcal{G}, R_\mathcal{G}\rangle$ is a labelled graph and $\mathbb{B}_\mathcal{G}$ is a perfectly matched binary relation on $V_\mathcal{G}$. An ***RB-cograph*** is an RB-graph $\mathcal{G}$ where $\langle V_\mathcal{G}, R_\mathcal{G}\rangle$ is a cograph.

Definition 4.5 An ***alternating elementary cycle***(***æ-cycle***) in a RB-graph is a sequence of vertices $x_1, \ldots, x_n$ with n even, such that $x_i \neq x_j$ for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$, and we have $x_1 R x_2 \mathbb{B} x_3 R x_4 \ldots x_n \mathbb{B} x_1$. A ***chord*** in an æ-cycle is an edge $x_i R x_j$ for $i, j \in \{1, \ldots, n\}$ that does not participate in the cycle. A ***chordless æ-cycle*** is an æ-cycle without any chords. An RB-graph $\mathcal{G}$ is ***æ-acyclic*** if it has no chordless æ-cycle.

5 Multiplicative Flows

The basic idea of the correctness criterion for combinatorial flows is the distinction between multiplicative and additive behavior of flowboxes. We begin in this section with the multiplicative part, which is based on Retoré's work on handsome proof nets [26].

Let $\phi = \langle A, B, \mathbb{B}_\phi\rangle$ be a flowbox, where $\mathbb{B}_\phi$ is perfectly matched. Then we can associate to ϕ the RB-cograph $\mathcal{G}(\phi) = \langle V_\phi, R_\phi, \mathbb{B}_\phi\rangle$, where $\langle V_\phi, R_\phi\rangle$ is $\mathcal{G}(\overline{A} \vee B)$. Observe that $V_\phi = V_{\mathcal{G}(\overline{A} \vee B)} = \lfloor \overline{A} \vee B \rfloor = \lfloor \overline{A} \rfloor \uplus \lfloor B \rfloor$, allowing us to use $\mathbb{B}_\phi$ in the graph.

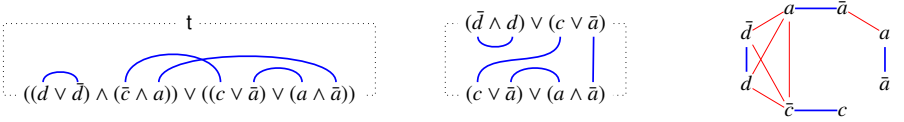
Definition 5.1 A flowbox $\phi = \langle A, B, \mathbb{B}_\phi\rangle$ is ***pure*** if A and B are pure. A ***multiplicative flow*** (or ***m-flow***) is a pure flowbox $\phi = \langle A, B, \mathbb{B}_\phi\rangle$, where $\mathbb{B}_\phi$ is perfectly matched and $\mathcal{G}(\phi)$ is æ-acyclic, and we do not have both $A \equiv \mathsf{t}$ and $B \equiv \mathsf{f}$.[5]

[5] Observe that we do allow $A \equiv \mathsf{t}$ or $B \equiv \mathsf{f}$, just not both at the same time.

Remark 5.2 Note that for each flowbox ϕ with $\mathbb{B}_\phi$ being perfectly matched, the graph $\mathcal{G}(\phi)$ is uniquely determined. However, the converse is not true. In particular, if A and B are unit-free formulas and $\phi = \langle A, B, \mathbb{B}_\phi\rangle$ and $\phi' = \langle \mathsf{t}, \overline{A} \vee B, \mathbb{B}_\phi\rangle$, then $\mathcal{G}(\phi) = \mathcal{G}(\phi')$.

Proposition 5.3 *Let A and B be unit-free formulas. Then $\phi = \langle A, B, \mathbb{B}_\phi\rangle$ is an m-flow iff $\phi' = \langle \mathsf{t}, \overline{A} \vee B, \mathbb{B}_\phi\rangle$ is an m-flow.*

Example 5.4 Below on the left are two flowboxes that are m-flows and that have the same RB-cograph which is shown on the right below.



When we compose two flowboxes ϕ and ψ with the operations $\wedge$, $\vee$, $\circ$ defined in Definition 3.3, then it is clear that the result is perfectly matched if ϕ and ψ are. From the work of Retoré [26] it follows that also the property of being æ-acyclic is preserved. However, the result does not need to be pure even if ϕ and ψ are (see Example 5.7 below). In particular, if we have m-flows $\phi = \langle \mathsf{t}, B, \mathbb{B}_\phi\rangle$ and $\psi = \langle C, D, \mathbb{B}_\psi\rangle$, where C is unit-free, then $\phi \vee \psi$ is not an m-flow because $\mathsf{t} \vee C$ is not pure. However, we have the following result:

Theorem 5.5 *Let $\mathcal{D}\,\begin{Vmatrix} A \\ B \end{Vmatrix}_{\{ai\downarrow,ai\uparrow,s,mix\}}$ be a derivation. If A and B are pure, then $[\![\mathsf{T}(\mathcal{D})]\!]$ is an m-flow.*

Proof Every flowbox occurring in $\mathsf{T}(\mathcal{D})$ is an m-flow, because $\mathcal{D}$ only contains instances of $\mathrm{ai}{\downarrow}, \mathrm{ai}{\uparrow}, \mathrm{s}, \mathrm{mix}$. By construction, $\mathbb{B}_{[\![\mathsf{T}(\mathcal{D})]\!]}$ is perfectly matched, and as already observed above, no ae-cycle is introduced in $\mathcal{G}([\![\mathsf{T}(\mathcal{D})]\!])$. Since A and B are pure, it follows that $[\![\mathsf{T}(\mathcal{D})]\!]$ is an m-flow. □

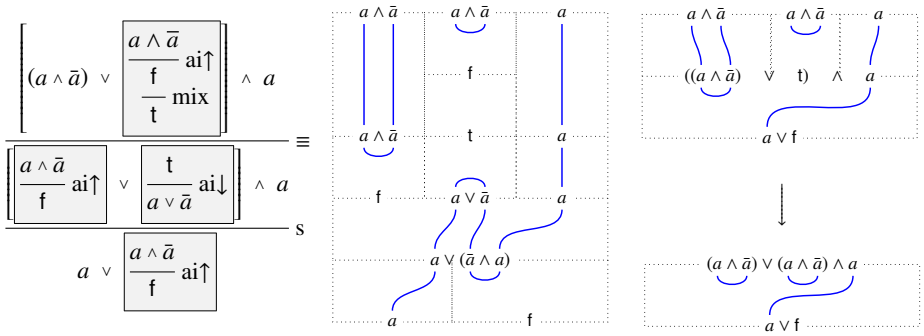
We also have the converse, which is a consequence of sequentialization of linear logic proof nets.

Theorem 5.6 *Let $\phi = \langle A, B, \mathbb{B}_\phi\rangle$ be an m-flow. Then there is a derivation $\mathcal{D}\,\begin{Vmatrix} A \\ B \end{Vmatrix}_{\{ai\downarrow,ai\uparrow,s,mix\}}$ with $[\![\mathsf{T}(\mathcal{D})]\!] = \phi$.*

Proof The simplest way to prove this is using the sequentialization result of Retoré [26] together with the correspondence between sequent calculus and deep inference [14]. A direct sequentialization from proof nets to deep inference derivations can be found in [30]. □

Example 5.7 Below we show on the left a derivation, whose translation to combinatorial flows is shown in the middle. We can naively compose the lower half, but then would obtain flowboxes that are not pure, as shown on the upper right below. Nonethe-

less, the complete composition is pure and an m-flow (shown on the lower right).



This example also shows a case of yanking.

6 Additive Flows

In the previous section we looked at flows generated by the rules ai↓, ai↑, s, mix. Now we investigate the flows generated by the rules ac↓, ac↑, aw↓, aw↑, m. The main difference is that we do no longer demand that the relation of the flowbox is perfectly matched, but that it is a function with certain properties.

A flowbox $\phi = \langle A, B, \mathbb{B}_\phi \rangle$ is ***function-like*** if $x \mathbb{B}_\phi y$ implies that either $x \in \lfloor A \rfloor$ and $y \in \lfloor B \rfloor$ or $x \in \lfloor B \rfloor$ and $y \in \lfloor A \rfloor$, and for every $x \in \lfloor A \rfloor$, there is a unique y with $x \mathbb{B}_\phi y$. Then $\mathbb{B}_\phi$ defines a function $f_\phi^{\downarrow} : \lfloor A \rfloor \to \lfloor B \rfloor$, and we write ϕ as $\langle A, B, f_\phi^{\downarrow} \rangle$.

Similarly, we say that ϕ is ***cofunction-like*** if if $x \mathbb{B}_\phi y$ implies that either $x \in \lfloor A \rfloor$ and $y \in \lfloor B \rfloor$ or $x \in \lfloor B \rfloor$ and $y \in \lfloor A \rfloor$, and for every $y \in \lfloor B \rfloor$ there is exactly one $x \in \lfloor A \rfloor$ with $x \mathbb{B}_\phi y$. In this case we interpret the function $\lfloor B \rfloor \to \lfloor A \rfloor$ defined by $\mathbb{B}_\phi$ as function $f_\phi^{\uparrow} : \lfloor \overline{B} \rfloor \to \lfloor \overline{A} \rfloor$, and we write ϕ as $\langle A, B, f_\phi^{\uparrow} \rangle$.

For specifying the desired properties of those functions, let us recall the notion of *skew fibration* [18].

Definition 6.1 Let $\mathcal{G}$ and $\mathcal{H}$ be graphs. A ***graph homomorphism*** $f : \mathcal{G} \to \mathcal{H}$ is a mapping $f : V_\mathcal{G} \to V_\mathcal{H}$ such that for every $v, w \in V_\mathcal{G}$, if $v R_\mathcal{G} w$ then $f(v) R_\mathcal{H} f(w)$, and for every $v \in V_\mathcal{G}$ we have $\ell_\mathcal{G}(v) = \ell_\mathcal{H}(f(v))$. A ***skew fibration*** is a graph homomorphism $f : \mathcal{G} \rightarrowtail \mathcal{H}$ where $\mathcal{H}$ is non-empty, and for every $v \in V_\mathcal{G}$ and $w \in V_\mathcal{H}$ with $f(v) R_\mathcal{H} w$, there exists a vertex z in $\mathcal{G}$ with $v R_\mathcal{G} z$ and $(f(z), w) \notin R_\mathcal{H}$.

For graph homomorphisms $f_1 : \mathcal{G}_1 \to \mathcal{H}_1$ and $f_2 : \mathcal{G}_2 \to \mathcal{H}_2$ we can define their horizontal compositions $f_1 + f_2 : \mathcal{G}_1 \uplus \mathcal{G}_2 \to \mathcal{H}_1 \uplus \mathcal{H}_2$ and $f_1 \times f_2 : \mathcal{G}_1 \bowtie \mathcal{G}_2 \to \mathcal{H}_1 \bowtie \mathcal{H}_2$, acting componentwise on the vertex set $V_{\mathcal{G}_1} \uplus V_{\mathcal{G}_2}$.

Lemma 6.2 *If $f : \mathcal{G} \rightarrowtail \mathcal{H}$ and $g : \mathcal{I} \rightarrowtail \mathcal{J}$ are skew fibrations, then so is $f + g : \mathcal{G} \uplus \mathcal{H} \rightarrowtail \mathcal{I} \uplus \mathcal{J}$. Furthermore, if either $\mathcal{G}_1 \neq \emptyset \neq \mathcal{G}_2$ or $\mathcal{G}_1 = \emptyset = \mathcal{G}_2$ then $f \times g : \mathcal{G} \bowtie \mathcal{H} \rightarrowtail \mathcal{I} \bowtie \mathcal{J}$ is also a skew fibration.*

Proof Immediate from the definition. □

Lemma 6.3 *Let $f : \mathcal{G} \rightarrowtail \mathcal{H}_1 \uplus \mathcal{H}_2$ be a skew fibration. Then $f = f_1 + f_2$ for skew fibrations $f_1 : \mathcal{G}_1 \rightarrowtail \mathcal{H}_1$ and $f_2 : \mathcal{G}_2 \rightarrowtail \mathcal{H}_2$ with $\mathcal{G} = \mathcal{G}_1 \uplus \mathcal{G}_2$.*

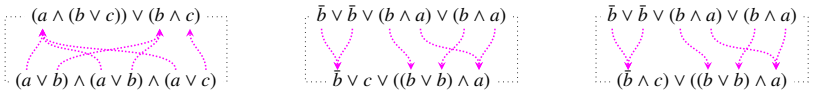
Proof Immediate from the definition. □

Lemma 6.4 *If $f : \mathcal{G} \rightarrowtail \mathcal{H}$ and $g : \mathcal{H} \rightarrowtail \mathcal{I}$ are skew fibrations and $\mathcal{G}$, $\mathcal{H}$, $\mathcal{I}$ are cographs, then the composition $f \circ g : \mathcal{G} \rightarrowtail \mathcal{I}$ is a skew fibration.*

Proof See [19] or [28]. □

Definition 6.5 An ***a$^{\downarrow}$-flow*** is a function-like flowbox $\phi = \langle A, B, f_\phi^{\downarrow} \rangle$ where A and B are pure and $A \not\equiv \mathsf{t}$ and $f_\phi^{\downarrow}$ is a skew fibration $f_\phi^{\downarrow} : \mathcal{G}(A) \rightarrow \mathcal{G}(B)$. Similarly, an ***a$^{\uparrow}$-flow*** is a cofunction-like flowbox $\phi = \langle C, D, f_\phi^{\uparrow} \rangle$ where C and D are pure and $D \not\equiv \mathsf{f}$ and $f_\phi^{\uparrow}$ is a skew fibration $f_\phi^{\uparrow} : \mathcal{G}(\overline{D}) \rightarrow \mathcal{G}(\overline{C})$. We call a$^{\downarrow}$-flows and a$^{\uparrow}$-flows also ***additive flows***.

Example 6.6 When drawing flowboxes that are a$^{\downarrow}$-flows or a$^{\uparrow}$-flows, we use purple arrows to indicate the direction of the functions. Below are three examples. The first one is an a$^{\uparrow}$-flow. The second one is an a$^{\downarrow}$-flow. The third example is not a skew fibration because in the lower graph there is an edge between the $\overline{b}$ and the c, violating the skew lifting.



When composing flowboxes that are additive flows, we are in a similar situation as for multiplicative flows in the previous section. In the general case, the horizontal composition of skew fibrations is not a skew fibration (see side condition in Lemma 6.2). However, we have an analoguous result as for m-flows:

Theorem 6.7 *Let $\mathcal{D} \Vert^{A}_{B}{}_{\{aw\downarrow,ac\downarrow,m\}}$ be a derivation. If A and B are pure, then $[\![\mathsf{T}(\mathcal{D})]\!]$ is an a$^{\downarrow}$-flow. Dually, if A and B are pure in $\mathcal{D} \Vert^{A}_{B}{}_{\{aw\uparrow,ac\uparrow,m\}}$ then $[\![\mathsf{T}(\mathcal{D})]\!]$ is an a$^{\uparrow}$-flow.*

Proof Every flowbox occurring in $\mathsf{T}(\mathcal{D})$ is an a$^{\downarrow}$-flow (resp. a$^{\uparrow}$-flow). We can conclude by a similar argument as for Theorem 5.5. □

As before, we also have the converse.

Theorem 6.8 *Let $\phi = \langle A, B, f_\phi^{\downarrow} \rangle$ be an a$^{\downarrow}$-flow. Then there is a derivation $\mathcal{D} \Vert^{A}_{B}{}_{\{aw\downarrow,ac\downarrow,m\}}$ with $[\![\mathsf{T}(\mathcal{D})]\!] = \phi$. Dually, for every a$^{\uparrow}$-flow ψ we have $\mathcal{D} \Vert^{A}_{B}{}_{\{aw\uparrow,ac\uparrow,m\}}$ with $[\![\mathsf{T}(\mathcal{D})]\!] = \psi$.*

Proof This follows from [19] together with [2], or more directly from [28]. □

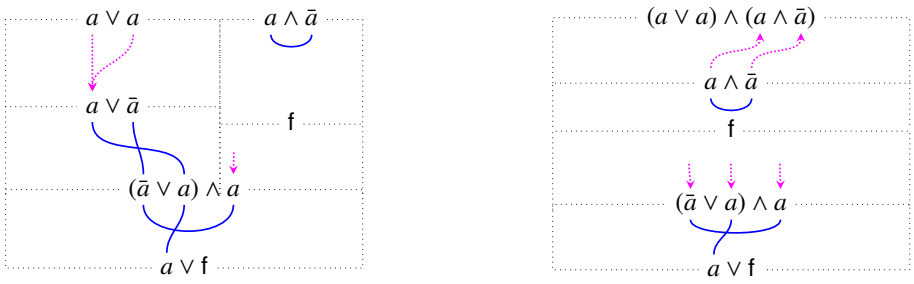


Fig. 3: A simplification of the combinatorial flow in Figure 2 and its purification

7 Combinatorial Flows

Definition 7.1 A ***combinatorial flow*** is a preflow $\phi\colon A \vdash B$ where every flowbox that occurs in ϕ is either a multiplicative flow or an additive flow.

We immediately have the following:

Theorem 7.2 *Let $\mathcal{D}$ be a derivation in* SKS. *Then* $\mathsf{T}(\mathcal{D})$ *is a combinatorial flow.*

We can simplify combinatorial flows by "executing" the operations in Definition 3.5. More precisely, a combinatorial flow $\phi\colon A \vdash B$ is a ***simplification*** of a $\psi\colon A \vdash B$ if ϕ is obtained from ψ by collapsing subflows in which all flowboxes have the same type (m-flow, $\mathrm{a}^{\downarrow}$-flow, or $\mathrm{a}^{\uparrow}$-flow).

Note that the difference between the collapse (which is uniquely determined) and a simplification (which is not uniquely determined) is that in the former everything is simplified, wheras in the latter the property of being a combinatorial flow is preserved. With this, we can state the converse of Theorem 7.2

Theorem 7.3 *Let $\phi\colon A \vdash B$ be a combinatorial flow. Then there is a derivation* $\begin{smallmatrix} A \\ \mathcal{D} \,\|\, \mathsf{SKS} \\ B \end{smallmatrix}$ *such that ϕ is a simplification of* $\mathsf{T}(\mathcal{D})$.

Proof This follows immediately from Theorems 5.6 and 6.8. □

The combinatorial flow on the left in Figure 3 is a simplification of the one in Figure 2.

Corollary 7.4 Combinatorial flows are sound and complete for classical logic.

Corollary 7.5 Combinatorial flows form a proof system in the sense of Cook and Reckhow [6].

Proof It can be checked in polynomial time if a preflow is a combinatorial flow. □

Example 7.6 Figure 4 shows a series of combinatorial flows, where we apply "Currying": flipping some premises to the conclusion or vice versa. Because blue wires can be yanked, this is easily possible, and by Theorem 7.3 there is always a corresponding

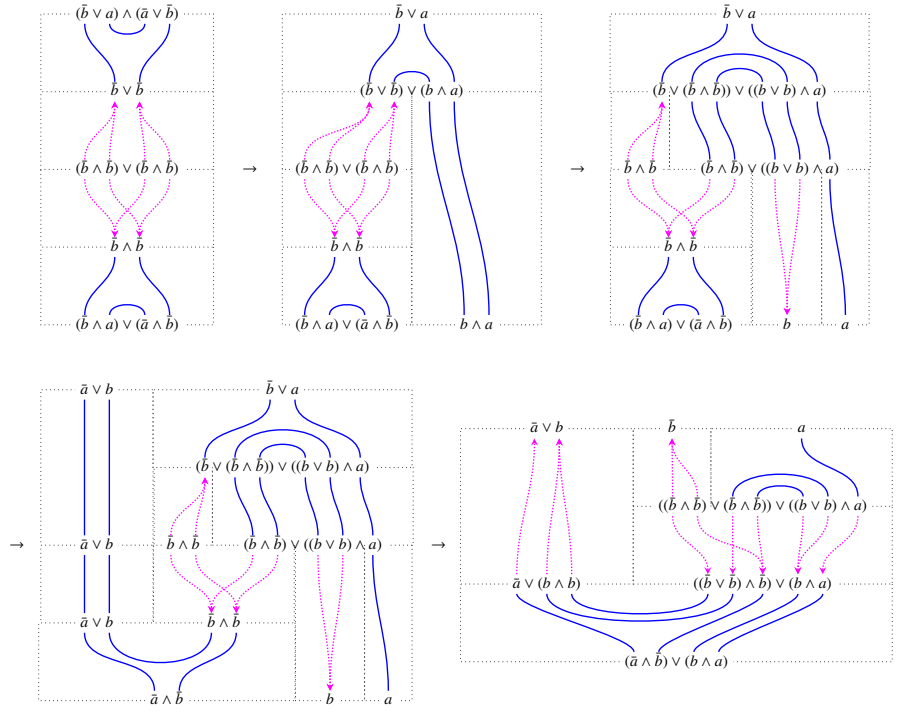
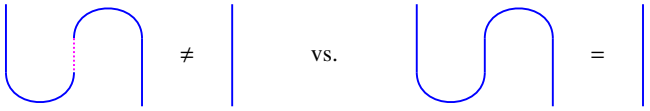


Fig. 4: Example of applying "Currying" to a combinatorial flow.

derivation. However, in the last combinatorial flow in that figure, there are two blue bends that are connected by a purple edge, and that therefore cannot be yanked:



8 Purification

We introduced the notion of ***pure*** because the correctness criteria for additive and multiplicative flows do not work if the formulas are not pure. In the multiplicative case, we do no longer have a canonical representation of an m-flow (see e.g. [1,17,23]) and checking equivalence is PSPACE-complete [15]. In the additive case, the horizontal composition can break the skew fibration property, when units are involved. This is the reason for the side condition in Lemma 6.2. With the equivalence $\equiv$, some units occurring in formulas can be removed, but not all. And the presence of these units can block further simplification in combinatorial flows.

We have $A \wedge \mathsf{f} \not\equiv \mathsf{f}$ and $A \vee \mathsf{t} \not\equiv \mathsf{t}$ because otherwise $\equiv$ would change the number of atoms in a formula, and therefore break Propositions 4.2 and 4.3. However, we have the logical equivalences $A \wedge \mathsf{f} \Longleftrightarrow \mathsf{f}$ and $A \vee \mathsf{t} \Longleftrightarrow \mathsf{t}$. Using these equivalences, we can remove all units from a combinatiorial flow, and we call this process *purification*.

Definition 8.1 The ***purification*** of a formula A, denoted as $\mathbf{p}(A)$ is defined to be the normal form of the rewriting relation $\rightsquigarrow$ below:

$$\begin{array}{llll} A \wedge \mathsf{t} \rightsquigarrow A & \mathsf{t} \wedge A \rightsquigarrow A & A \vee \mathsf{t} \rightsquigarrow \mathsf{t} & \mathsf{t} \vee A \rightsquigarrow \mathsf{t} \\ A \vee \mathsf{f} \rightsquigarrow A & \mathsf{f} \vee A \rightsquigarrow A & A \wedge \mathsf{f} \rightsquigarrow \mathsf{f} & \mathsf{f} \wedge A \rightsquigarrow \mathsf{f} \end{array} \tag{6}$$

It is easy to see that this rewriting relation is terminating and confluent, and therefore the purification of a formula is well-defined. The interesting observation is that this rewriting relation can be extended from formulas to combinatorial flows.

Definition 8.2 A ***slice*** of a combinatorial flow $\phi \colon A \vdash B$ is a formula C such that $\phi = \phi_1 \circledcirc \phi_2$ for some combinatorial flows $\phi_1 : A \vdash C_1$ and $\phi_2 : C_2 \vdash B$ with $C_1 \equiv C \equiv C_2$. A combinatorial flow ϕ is ***pure*** if every slice of ϕ is pure.

Theorem 8.3 *For every combinatorial flow $\phi \colon A \vdash B$, there is a pure combinatorial flow $\boldsymbol{p}(\phi) \colon \boldsymbol{p}(A) \vdash \boldsymbol{p}(B)$.*

Proof First, observe that every flowbox occurring in ϕ is pure, i.e., premise and conclusion can be written as a unit-free formula or a unit. If a slice (recall the eqivalences in (5)) is not pure, then it must have a subformula of the shape $A \wedge \mathsf{f}$ or $A \vee \mathsf{t}$, which is the consequence of the horizontal compositions of flowboxes. We can list all possible cases and design a rewrite system that extends (6) to combinatorial flows. A list of these cases can be found in Figure 5. Below are two representative cases:

(7)

Each step reduces the number of flowboxes in ϕ that have a unit as premise or conclusion. Therefore the rewriting is terminating. Furthermore, the result is pure. □

The combinatorial flow on the right in Figure 3 is a purification of the one on the left.

9 Combinatorial Flows and Combinatorial Proofs

Definition 9.1 A ***combinatorial proof*** [18] of a pure formula A is a skew fibration $f \colon \mathcal{H} \to \mathcal{G}(A)$ from an æ-acyclic RB-cograph $\mathcal{H}$ to the graph of A.[6]

[6] In [18], the units t and f are treated like atoms in the translation to graphs, so that the restriction to pure formulas was not needed. However, this would make composition difficult to define, and for this reason in [19] combinatorial proofs have been restricted to the unit-free setting.

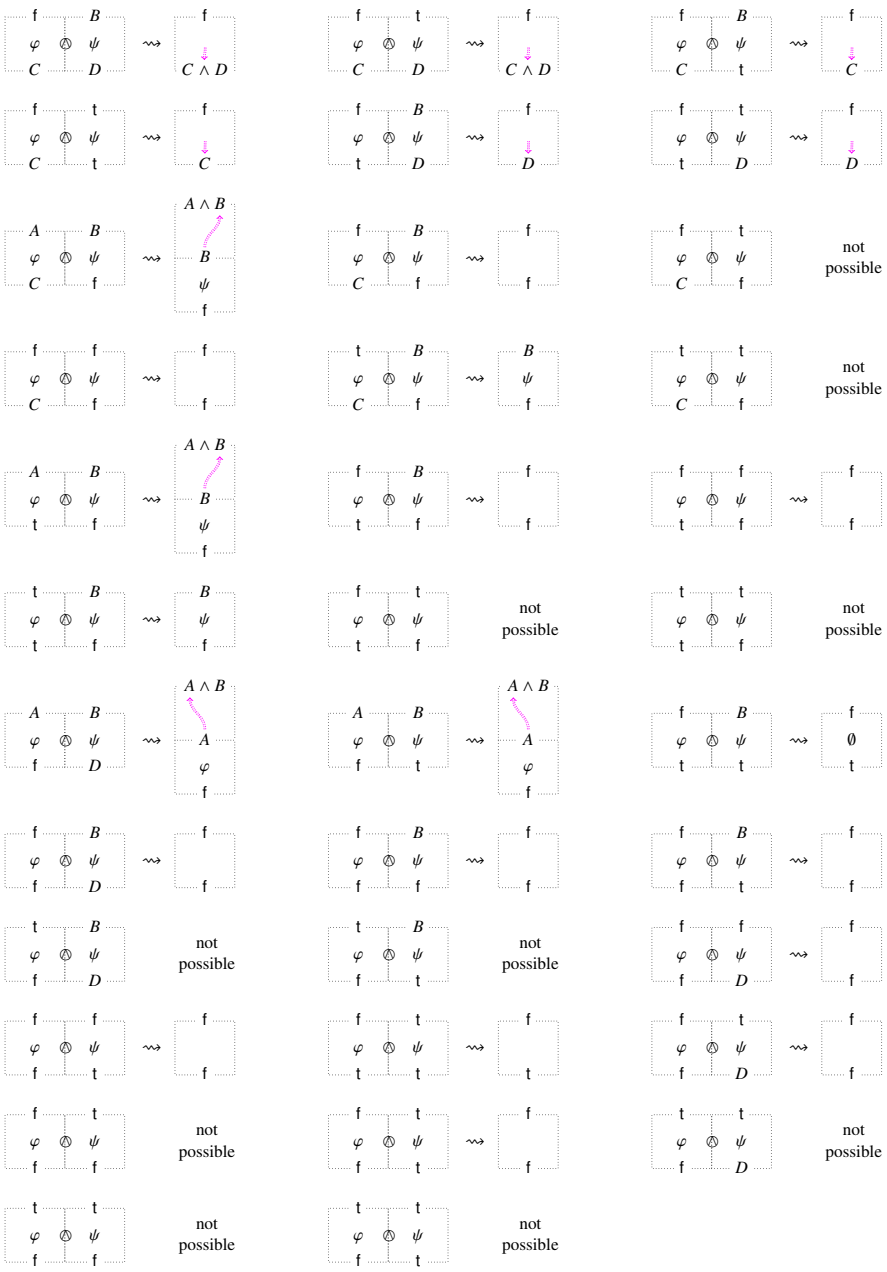


Fig. 5: Purification cases for conjunction. The cases for disjunction are dual.

Tranlated to the setting of this paper, a combinatorial proof is the composition $\phi \circledcirc \psi$ of an m-flow $\phi = \langle \mathsf{t}, H, \mathbb{B}_\phi \rangle$ and an $\mathsf{a}^\downarrow$-flow $\psi = \langle H, A, f^\downarrow_\psi \rangle$. In other words, combinatorial

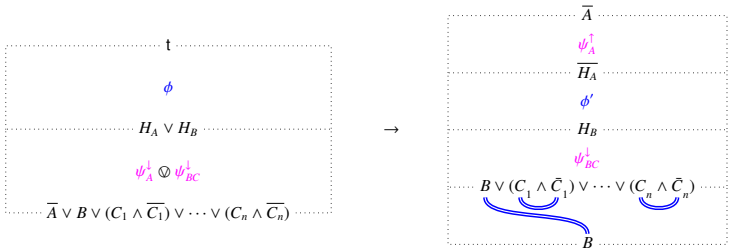


Fig. 6: Translating combinatorial proofs with cuts into combinatorial flows.

proofs are a special case of combinatorial flows that make a global separation between the multiplicative and the additive parts of a cut-free proof. In order to deal with cuts, the notion of combinatorial proof has been extended to sequents in [19].

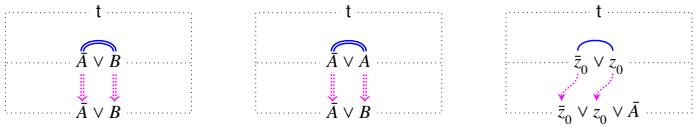
Definition 9.2 A ***combinatorial proof with cuts*** of a sequent $\Gamma = A_1, \ldots, A_n$ of unit-free formulas is a skew fibration $f \colon \mathcal{H} \to \mathcal{G}(A_1 \vee \cdots \vee A_n \vee (\overline{C}_1 \wedge C_1) \vee \cdots \vee (\overline{C}_k \wedge C_k))$ from an ae-acyclic RB-cograph $\mathcal{H}$ to the graph of $\Gamma, C_1 \wedge C_1, \ldots, \overline{C}_k \wedge C_k$, where $C_1, \ldots, C_k$ are arbitrary unit-free formulas and are called the ***cut formulas*** of the proof.

These cuts can be simulated by an m-flow in combinatorial flows. More precisely, assume we have a combinatorial proof with cuts for a sequent $\Gamma = \overline{A}, B$, as shown on the left in Figure 6. We have that $H = H_A \vee H_B$ and $\psi = \psi_A \oslash \psi_{BC}$ because of Lemma 6.3. We can translate this into a combinatorial flow $\phi \colon A \vdash B$, as shown on the right in Figure 6. There, the m-flow ϕ' exists by Proposition 5.3.

We are now going to show the converse, i.e., we will give a polynomial translation from combinatorial flows to combinatorial proofs with cuts. For this, we have to be a bit more careful with the units, as combinatorial proofs with cuts are only defined for unit-free formulas. We let z_0 be a fresh propositional variable, and we define the function $(\cdot)^{\mathbf{o}}$ on pure formulas as follows: If $A \equiv B$ for some unit-free formula B, then $A^{\mathbf{o}} = B$. If $A \equiv \mathsf{t}$, then $A^{\mathbf{o}} = \overline{z}_0 \vee z_0$. If $A \equiv \mathsf{f}$, then $A^{\mathbf{o}} = \overline{z}_0 \wedge z_0$.

Now assume we have a combinatorial flow $\phi \colon A \vdash B$, We can translate this inductively into a combinatorial proof with cuts of the sequent $\Gamma = \overline{A^{\mathbf{o}}}, B^{\mathbf{o}}$.

1. First, every m-flow, $\mathrm{a}^{\downarrow}$-flow, and $\mathrm{a}^{\uparrow}$-flow can be immediately translated into a combinatorial proof. For the cases $\langle A, B, \mathbb{B}_{\phi} \rangle$ and $\langle A, B, f_{\psi}^{\downarrow} \rangle$ and $\langle A, \mathsf{t}, f_{\psi}^{\uparrow} \rangle$ these are shown below. The others are similar.



2. If $\phi = \phi_1 \oslash \phi_2$ with $\phi_1 \colon A_1 \vdash B_1$ and $\phi_2 \colon A_2 \vdash B_2$ then we have by induction hypothesis combinatorial proofs with cuts of $\Gamma_1 = \overline{A_1^{\mathbf{o}}}, B_1^{\mathbf{o}}$ and $\Gamma_2 = \overline{A_2^{\mathbf{o}}}, B_2^{\mathbf{o}}$. By the construction in [19], we get one with conclusion $\Gamma = \overline{A_1^{\mathbf{o}}}, \overline{A_2^{\mathbf{o}}}, B_1^{\mathbf{o}} \wedge B_2^{\mathbf{o}}$ which is equivalent to $(\overline{A}_1 \vee \overline{A}_2)^{\mathbf{o}}, (B_1 \wedge B_2)^{\mathbf{o}}$. The case for $\phi = \phi_1 \otimes \phi_2$ is similar.

3. If $\phi = \phi_1 \circledcirc \phi_2$ with $\phi_1 : A \vdash D_1$ and $\phi_2 : D_2 \vdash B$ with $D_1 \equiv D_2$, we proceed similarly, with the difference that we add a new cut formula $D_1^{\mathbf{o}}$. Note that $\mathcal{G}(D_1^{\mathbf{o}}) = \mathcal{G}(D_2^{\mathbf{o}})$ and $\overline{D_1^{\mathbf{o}}} \wedge D_1^{\mathbf{o}}$ is added to the conclusion sequent.

10 Conclusion and Future Work

We have defined combinatorial flows as a graphical representation of classical proofs, merging features from atomic flows and combinatorial proofs. Unlike atomic flows, they allow to reconstruct a proof, and unlike combinatorial proofs they allow a more flexible mixture of additive and multiplicative parts of a proof.

In future work, we would like to use combinatorial flows to define more flexible normalization procedures. Observe that atomic flows allow local reductions [11] that have to be proceeded by a global cycle removing step [13,32], and that combinatorial proofs perform global cut reductions [16,19] similar to the sequent calculus. We hope that with combinatorial flows we can merge the advantages of both, via a "semi-local" normalization procedure that works on the level of flowboxes, similar to the purification procedure that we presented in Section 8. In fact, note that this purification can be seen as a normalization for weakening, and what remains to be done is a similar procedure for contraction.

References

1. Richard Blute, Robin Cockett, Robert Seely, and Todd Trimble. Natural deduction and coherence for weakly distributive categories. *J. of Pure and Applied Algebra*, 113:229–296, 1996.
2. Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2250 of *LNAI*, pages 347–361. Springer, 2001.
3. Samuel R. Buss. The undecidability of k-provability. *Annals of Pure and Applied Logic*, 53(1):72–102, 1991.
4. Alessandra Carbone. Interpolants, cut elimination and flow graphs for the propositional calculus. *Annals of Pure and Applied Logic*, 83:249–299, 1997.
5. Alessandra Carbone. Turning cycles into spirals. *Ann. Pure Appl. Logic*, 96(1-3):57–73, 1999.
6. Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.
7. Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Arch. Math. Log.*, 28(3):181–203, 1989.
8. Anupam Das. Rewriting with linear inferences in propositional logic. In Femke van Raamsdonk, editor, *24th International Conference on Rewriting Techniques and Applications (RTA)*, volume 21 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 158–173. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2013.
9. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
10. Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.
11. Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1:9):1–36, 2008.

12. Alessio Guglielmi, Tom Gundersen, and Michel Parigot. A Proof Calculus Which Reduces Syntactic Bureaucracy. In Christopher Lynch, editor, *21st International Conference on Rewriting Techniques and Applications, RTA 2010*, volume 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 135–150, United Kingdom, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
13. Alessio Guglielmi, Tom Gundersen, and Lutz Straßburger. Breaking paths in atomic flows for classical logic. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 284–293. IEEE Computer Society, 2010.
14. Alessio Guglielmi and Lutz Straßburger. Non-commutativity and MELL in the calculus of structures. In Laurent Fribourg, editor, *Computer Science Logic, CSL 2001*, volume 2142 of *LNCS*, pages 54–68. Springer-Verlag, 2001.
15. Willem Heijltjes and Robin Houston. Proof equivalence in MLL is pspace-complete. *Log. Methods Comput. Sci.*, 12(1), 2016.
16. Willem Heijltjes, Dominic Hughes, and Lutz Straßburger. Proof nets for first-order additive linear logic. Research Report RR-9201, Inria, 2018.
17. Dominic Hughes. Simple multiplicative proof nets with units. Preprint, 2005.
18. Dominic Hughes. Proofs Without Syntax. *Annals of Mathematics*, 164(3):1065–1076, 2006.
19. Dominic Hughes. Towards hilbert's 24^{th} problem: Combinatorial proof invariants: (preliminary version). *Electr. Notes Theor. Comput. Sci.*, 165:37–63, 2006.
20. Gregory Maxwell Kelly and Saunders Mac Lane. Coherence in closed categories. *J. of Pure and Applied Algebra*, 1:97–140, 1971.
21. S. C. Kleene. Permutability of inferences in gentzen's calculi lk and lj, memoirs of the american mathematical society. *Journal of Symbolic Logic*, 19(1):62–63, 1954.
22. François Lamarche and Lutz Straßburger. Naming proofs in classical propositional logic. In Paweł Urzyczyn, editor, *TLCA'05*, volume 3461 of *LNCS*, pages 246–261. Springer, 2005.
23. François Lamarche and Lutz Straßburger. From proof nets to the free *-autonomous category. *Logical Methods in Computer Science*, 2(4:3):1–44, 2006.
24. Dag Prawitz. *Natural Deduction, A Proof-Theoretical Study*. Almquist and Wiksell, 1965.
25. Christian Retoré. Pomset logic as a calculus of directed cographs. In V. M. Abrusci and C. Casadio, editors, *Dynamic Perspectives in Logic and Linguistics*, pages 221–247. Bulzoni, Roma, 1999. Also available as INRIA Rapport de Recherche RR-3714.
26. Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 294(3):473–488, 2003.
27. P. Selinger. A survey of graphical languages for monoidal categories. In Bob Coecke, editor, *New Structures for Physics*, pages 289–355, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
28. Lutz Straßburger. A characterization of medial as rewriting rule. In Franz Baader, editor, *Term Rewriting and Applications, 18th International Conference, RTA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4533 of *LNCS*, pages 344–358. Springer, 2007.
29. Lutz Straßburger. Combinatorial Flows and Their Normalisation. In Dale Miller, editor, *FSCD 2017*, volume 84 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:17, 2017.
30. Lutz Straßburger. Deep inference and expansion trees for second-order multiplicative linear logic. *Mathematical Structures in Computer Science*, 29(8):1030–1060, 2019.
31. Rüdiger Thiele. Hilbert's twenty-fourth problem. *American Mathematical Monthly*, 110:1–24, 2003.
32. Andrea Aler Tubella, Alessio Guglielmi, and Benjamin Ralph. Removing cycles from proofs. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic, CSL 2017, August 20-24, 2017, Stockholm, Sweden*, volume 82 of *LIPIcs*, pages 9:1–9:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.