Proof Complexity and Proof Compression via Combinatorial Flows

Lutz Straßburger
Inria

***Abstract*—Combinatorial flows are a novel way of representing proofs in a "syntax-free" way that also includes cut and substitution as proof compression mechanisms. In this paper we show how combinatorial flows are translated into syntactic proofs and vice versa, establishing that combinatorial flows are p-equivalent to Frege systems with substitution.**[1]

1. Introduction

The *cut* can be seen as a method of proof compression, in the sense that a proof with cuts is in general much smaller than the proof that is obtained by elimination the cuts [1]. In the area of structural proof theory this is a well studied and well understood phenomenon. However, there are other methods of proof compression, namely *extension* and *substitution* [4], that are mainly studied in the area of proof complexity. In fact, it is one of the major open problems of proof complexity whether Frege systems without extension can p-simulate Frege-systems with extension. But from the viewpoint of structural proof theory, the notions of extension and substitution have not yet been much investigated [3], [12], [10].

It has long been known that in the presence of cut, extension and substitution have the same strength with respect to p-simulation (shown in [4] and [9]). But only recently it has been shown that also in the absence of cut, systems with extension and systems with substitution can p-simulate each other [12], [10]. For this, a deep inference proof systems has been used that can speak about cut elimination and extension elimination at the same time so that the two proof compression mechanisms can be studied together. For Frege-systems, which are the ordinary vehicle for studying extension and substitution, there is no "cut-free" version.

In any case, so far, there is no "syntax-free" proof presentation (in the sense of linear logic proof nets) that can deal with proofs using extension or substitution.

In this paper I will present *combinatorial flows* that join ideas from *atomic flows* [5], [6] and *combinatorial proofs* by Hughes [7]. They represent proofs that can be composed via binary connectives, via cut, and via substitution. More precisely, instead of substituting formulas into formulas, combinatorial flows allow to substitute proofs into proofs.

2. Preliminaries: from combinatorial proofs to combinatorial flows

Combinatorial proofs have been introduced by Hughes in [7] as a way to present proofs of classical logic independent from a syntactic proof system.

We consider formulas (denoted by capital Latin letters $A, B, C, \ldots$) in negation normal form (NNF), generated from a countable set $\mathcal{V} = \{a, b, c, \ldots\}$ of (propositional) variables by the following grammar:

$$A, B ::= a \mid \bar{a} \mid A \wedge B \mid A \vee B \tag{1}$$

where $\bar{a}$ is the negation of a. The negation can then be defined for all formulas using the De Morgan laws $\bar{\bar{A}} = A$ and $\overline{A \wedge B} = \bar{A} \vee \bar{B}$. An *atom* is a variable or its negation. We use $\mathcal{A}$ to denote the set of all atoms. We use $A \Rightarrow B$ as abbreviation for $\bar{A} \vee B$, and $A \Leftrightarrow B$ as abbreviation for $(A \Rightarrow B) \wedge (B \Rightarrow A)$.

A *sequent* Γ is a multiset of formulas, written as a list separated by comma: $\Gamma = A_1, A_2, \ldots, A_n$. We write $\bar{\Gamma}$ to denote the sequent $\bar{A}_1, \bar{A}_2, \ldots, \bar{A}_n$. We define the *size* of a sequent Γ, denoted by $|\Gamma|$, to be the number of atom occurrences in it. We write $\bigwedge\Gamma$ (resp. $\bigvee\Gamma$) for the conjunction (res. disjunction) of the formulas in Γ.

Remark 2.1. For simplicity we do not include the constants $\top$ and $\bot$ (for *truth* and *falsum*, respectively) into the language. We can always recover them by letting $\top = a_0 \vee \bar{a}_0$ and $\bot = a_0 \wedge \bar{a}_0$ for some fresh a_0.

Definition 2.2. A *graph* $\mathfrak{G} = \langle V_{\mathfrak{G}}, E_{\mathfrak{G}} \rangle$ consists of a set of *vertices* $V_{\mathfrak{G}}$ and a set of *edges* $E_{\mathfrak{G}}$ which are two-element subsets of $V_{\mathfrak{G}}$. For $v, w \in V$ we write vw for $\{v, w\}$. A *graph homomorphism* $f \colon \mathfrak{G} \to \mathfrak{G}'$ is a function from $V_{\mathfrak{G}}$ to $V_{\mathfrak{G}'}$ such that $vw \in E_{\mathfrak{G}}$ implies $f(v)f(w) \in E_{\mathfrak{G}'}$. A graph $\mathfrak{G}$ is called a *cograph* if it does not contain four distinct vertices u, v, w, z with $uv, vw, wz \in E$ and $vz, zu, uw \notin E$. For two graphs $\mathfrak{G} = \langle V, E \rangle$ and $\mathfrak{G}' = \langle V', E' \rangle$, we define the operations *union* $\mathfrak{G} \vee \mathfrak{G}' = \langle V \cup V', E \cup E' \rangle$ and *join* $\mathfrak{G} \wedge \mathfrak{G}' = \langle V \cup V', E \cup E' \cup \{vv' \mid v \in V, v' \in V'\} \rangle$. For a graph $\mathfrak{G} = \langle V, E \rangle$, also define its *negation* $\bar{\mathfrak{G}} = \langle V, \{vw \mid v \neq w, vw \notin E\} \rangle$.

Construction 2.3. If we associate to each atom a a single vertex labeled with a then every formula A uniquely determines a graph $\mathfrak{G}(A)$ that is constructed via the operations $\wedge$ and $\vee$. For a sequent $\Gamma = A_1, A_2, \ldots, A_n$, we define $\mathfrak{G}(\Gamma) = \mathfrak{G}(\bigvee\Gamma) = \mathfrak{G}(A_1) \vee \mathfrak{G}(A_2) \vee \cdots \vee \mathfrak{G}(A_n)$. Note that this construction entails that $\overline{\mathfrak{G}(A)} = \mathfrak{G}(\bar{A})$.

Lemma 2.4. *For two formulas A and B, we have $\mathfrak{G}(A) = \mathfrak{G}(B)$ iff A and B are equivalent modulo associativity and commutativity of $\wedge$ and $\vee$:*

$$\begin{array}{ll} A \wedge (B \wedge C) = (A \wedge B) \wedge C & A \wedge B = B \wedge A \\ A \vee (B \vee C) = (A \vee B) \vee C & A \vee B = B \vee A \end{array} \tag{2}$$

The following is well-known.

Proposition 2.5. *A graph $\mathfrak{G}$ is a cograph iff it is constructed from a formula via Construction 2.3.*

Definition 2.6. A graph homomorphism f is a *skew fibration*, denoted as $f \colon \mathfrak{G} \rightarrowtail \mathfrak{G}'$, if for every $v \in V_{\mathfrak{G}}$ and $w' \in V_{\mathfrak{G}'}$ with $f(v)w' \in E'_{\mathfrak{G}}$ there is a $w \in V_{\mathfrak{G}}$ with $vw \in E_{\mathfrak{G}}$ and $f(w)w' \notin E'_{\mathfrak{G}}$.

We are now ready to give the definition of a combinatorial proof together with the main result of [7]:

1. More technical details of this work can be found in the reserach report [13], available at https://hal.inria.fr/hal-01498468/.

Definition 2.7. A *combinatorial proof* of a sequent Γ consists of a formula C, a linear logic proof net for C, and a label-preserving skew-fibration $f\colon \mathfrak{G}(C) \rightarrowtail \mathfrak{G}(\Gamma)$.

Theorem 2.8 ([7])**.** *A formula is a theorem of classical propositional logic iff it has a combinatorial proof.*

Theorem 2.9 ([7])**.** *Combinatorial proofs form a proof system in the sense of Cook and Reckhow [4].*

Definition 2.10. Given two sequents Γ and Δ, a *simple (combinatorial) flow* ϕ from Γ to Δ, denoted by $\phi\colon \Gamma \vdash \Delta$, is a combinatorial proof for the sequent $\bar{\Gamma}, \Delta$. We write $\phi\colon \circ \vdash \Delta$ (resp. $\phi\colon \Gamma \vdash \circ$) if Γ (resp. Δ) is empty. Let ϕ be given by the R&B-cograph $\mathfrak{C}$ and skew fibration $f\colon \mathfrak{C}^{\downarrow} \rightarrowtail \mathfrak{G}(\bar{\Gamma}, \Delta)$. Then the *size* of ϕ, denoted by $|\phi|$, is defined to be $|\mathfrak{C}^{\downarrow}| + |\Gamma| + |\Delta|$.

Lemma 2.11. *Let $\mathfrak{C}$, $\mathfrak{G}_1$, and $\mathfrak{G}_2$ be cographs and let $f\colon \mathfrak{C} \rightarrowtail \mathfrak{G}_1 \vee \mathfrak{G}_2$ be a skew fibration. Then there are cographs $\mathfrak{C}_1$ and $\mathfrak{C}_2$ and graph homomorphisms $f_1\colon \mathfrak{C}_1 \rightarrowtail \mathfrak{G}_1$ and $f_2\colon \mathfrak{C}_2 \rightarrowtail \mathfrak{G}_2$ such that $\mathfrak{C} = \mathfrak{C}_1 \vee \mathfrak{C}_2$ and $f = f_1 \vee f_2$.*

Notation 2.12. This lemma allows us to depict basic combinatorial flows in the following way. Let $\phi\colon \Gamma \vdash \Delta$ be given, let $f\colon \mathfrak{C}^{\downarrow} \rightarrowtail \overline{\mathfrak{G}(\Gamma)} \vee \mathfrak{G}(\Delta)$ be the defining skew fibration, and let $\mathfrak{C}_\Gamma$ and $\mathfrak{C}_\Delta$ be the cographs determined by Lemma 2.11 (i.e., $\mathfrak{C}^{\downarrow} = \mathfrak{C}_\Gamma \vee \mathfrak{C}_\Delta$). If we write $F(\overline{\mathfrak{C}_\Gamma})$ and $F(\mathfrak{C}_\Delta)$ for the formula trees corresponding to the cographs $\overline{\mathfrak{C}_\Gamma}$ and $\mathfrak{C}_\Delta$, respectively, then we can write ϕ by writing Γ, $F(\overline{\mathfrak{C}_\Gamma})$, $F(\mathfrak{C}_\Delta)$, and Δ above each other, draw the axiom-links as blue/bold-edges [11] and indicate the mapping f by thin/thistle arrows. Figure 1 shows some examples. For better readability, we do not write atom names in $F(\overline{\mathfrak{C}_\Gamma})$ and in $F(\mathfrak{C}_\Delta)$, and allow in $F(\overline{\mathfrak{C}_\Gamma})$ outermost $\wedge$ to be replaced by comma, and in $F(\mathfrak{C}_\Delta)$ outermost $\vee$ to be replaced by comma.

Definition 2.13. A *substitution* is a mapping σ from propositional variables to formulas such that $\sigma(a) \neq a$ for only finitely many a.

We write $A\sigma$ for the formula obtained from applying the substitution σ to the formula A. If $\sigma = \{a_1 \mapsto B_1, \ldots, a_n \mapsto B_n\}$ we also write $A[a_1/B_1, \ldots, a_n/B_n]$ for $A\sigma$. This normally means that not only is each occurrence of a_i in A is replaced by B_i in A, but also each occurrence of $\bar{a}_i$ is replaced by $\bar{B}_i$. We also need a notation for substitutions in which a variable a and its dual $\bar{a}$ are not replaced by dual formulas. In this case we write $A[a_1/B_1, \bar{a}_1/C_1, \ldots, a_n/B_n, \bar{a}_n/C_n]$ for the formula that is obtained from A by simultaneously replacing every a_i by B_i and every $\bar{a}_i$ by C_i for each $i \in \{1, \ldots, n\}$.

Definition 2.14. The set of *combinatorial flows* is defined inductively as follows:

- A simple combinatorial flow $\phi\colon A \vdash B$ is a combinatorial flow.
- If $\phi\colon A \vdash B$ and $\psi\colon C \vdash D$ are combinatorial flows then so are $\phi \wedge \psi\colon A \wedge B \vdash C \wedge D$ and $\phi \vee \psi\colon A \vee C \vdash B \vee D$. This operation is called *horizontal composition*.
- If $\phi\colon \Gamma \vdash A$ and $\psi\colon A \vdash \Delta$ are combinatorial flows then $\phi \bullet \psi\colon \Gamma \vdash \Delta$ is a combinatorial flow. This operation is called *vertical composition*, *concatenation*, or *cut*.
- If $\phi\colon \Gamma \vdash \Delta$ and $\psi\colon C \vdash D$ are combinatorial flows then $\phi[a/\psi]\colon \Gamma[a/C, \bar{a}/\bar{D}] \vdash \Delta[a/D, \bar{a}/\bar{C}]$ is a combinatorial flow. This operation is called *substitution*.

The *size* of a combinatorial flow ϕ, denoted by $|\phi|$, is defined to be the sum of the sizes of all simple combinatorial flows occurring in ϕ.

Theorem 2.15. *Combinatorial flows form a proof system.*

Theorem 2.16. *For each combinatorial flow $\phi\colon \Gamma \vdash \Delta$ there is a simple combinatorial flow $\phi'\colon \Gamma \vdash \Delta$ with the same premise and conclusion.*

For a proof see [13]. An important observation is that the reduction step eliminating horizontal composition does not increase the size of the flow. However, cut elimination as well as substitution elimination can lead to an exponential blow-up. Thus, both cut and substitution are mechanisms to compress the flow.

3. Relation to deep inference proofs

In the remainder of this paper we show how combinatorial flows are related to syntactic proofs given in some deductive formalism. We start with proofs in the deep inference system SKS [2], which is shown in Figure 2. The rules shown there should be read as rewrite rule schemes that can be applied inside an arbitrary (positive) formula context. In the rules $\mathsf{ai}{\downarrow}$, $\mathsf{ai}{\uparrow}$, $\mathsf{ac}{\downarrow}$, and $\mathsf{ac}{\uparrow}$, the a can stand for any atom. In all rules, A, B, C, and D, can stand for any formula, and in $\mathsf{ai}{\downarrow}$ we additionally allow A to be empty, so to have proper proofs without premise. For SKS derivations, we consider formulas equivalent modulo associativity and commutativity of $\wedge$ and $\vee$, as given in (2). The *size* of a derivation Φ, denoted by $|\Phi|$ is the sum of the sizes of the formula occurrences in Φ.

Each rule in system SKS can straightforwardly be translated into a simple combinatorial flow, as indicated in Figure 3, where the double lines indicate the identity. Note that for the m-rule there are two possible translations. Since whenever $A = B$ modulo associativity and commutativity (2) we have that $\mathfrak{G}(A) = \mathfrak{G}(B)$, an equivalence step in an SKS-proof can translated into the identity flow. This is enough to give a direct translation which proves the following:

Theorem 3.1. *Substitution-free combinatorial flows p-simulate system* SKS.

Before we look at the other direction, we first look at the expressive power of simple combinatorial flows. We have the following result:

Theorem 3.2. *Let A and B be formulas. There is a simple combinatorial flow $\phi\colon A \to B$ iff there are formulas A' and B' such that there are derivations*

$$\begin{array}{c} A \\ \{\mathsf{w}{\uparrow},\mathsf{ac}{\uparrow},\mathsf{m}\}\big\|\Phi_1 \\ A' \end{array} \quad \textit{and} \quad \begin{array}{c} A' \\ \{\mathsf{ai}{\downarrow},\mathsf{ai}{\uparrow},\mathsf{s}\}\big\|\Phi_2 \\ B' \end{array} \quad \textit{and} \quad \begin{array}{c} B' \\ \{\mathsf{w}{\downarrow},\mathsf{ac}{\downarrow},\mathsf{m}\}\big\|\Phi_3 \\ B \end{array}$$

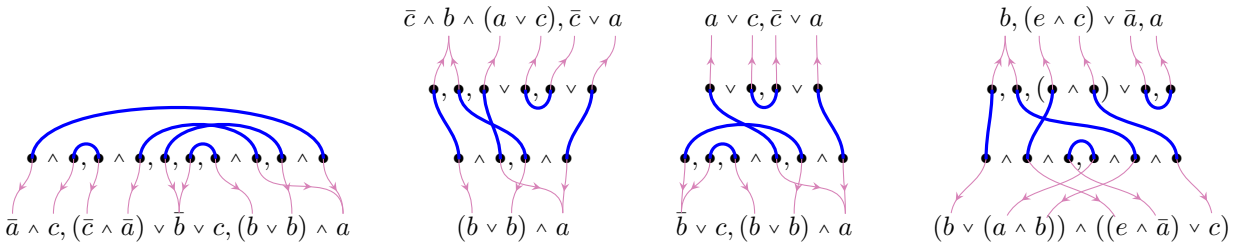


Figure 1. Examples of simple combinatorial flows

$$\mathsf{ai}{\downarrow}\ \frac{A}{A \wedge (a \vee \bar{a})} \qquad \mathsf{s}\ \frac{(A \vee B) \wedge C}{A \vee (B \wedge C)} \qquad \mathsf{ai}{\uparrow}\ \frac{(\bar{a} \wedge a) \vee A}{A}$$

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

$$\mathsf{w}{\downarrow}\ \frac{A}{A \vee B} \qquad \mathsf{w}{\uparrow}\ \frac{B \wedge A}{A}$$

Figure 2. Deep inference system SKS

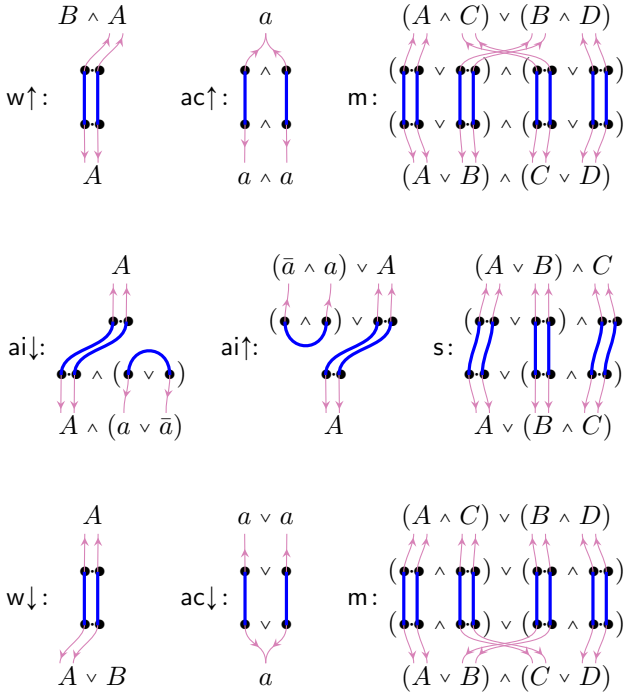


Figure 3. Simple combinatorial flows for the rules in Figure 2

and such that $|\phi| = O(|\Phi_1| + |\Phi_2| + |\Phi_3|)$. *Similarly, there is a simple flow* $\psi \colon \circ \vdash B$ *iff there are derivations*

$$\overset{B'}{\underset{\{\mathsf{ai}{\downarrow},\mathsf{s}\}}{\Big\|}\Phi_2}\ B' \qquad \textit{and} \qquad \overset{B'}{\underset{\{\mathsf{w}{\downarrow},\mathsf{ac}{\downarrow},\mathsf{m}\}}{\Big\|}\Phi_3}\ B$$

such that $|\psi| = O(|\Psi_2| + |\Psi_3|)$

Figure 4 depicts on the left the two statements of this theorem, and shows on the right two examples.

The cut-free variant of SKS is called KS, and consists of the rules $\{\mathsf{ai}{\downarrow}, \mathsf{s}, \mathsf{ac}{\downarrow}, \mathsf{aw}{\downarrow}, \mathsf{m}\}$.

Corollary 3.3. *System* KS *p-simulates simple combinatorial flows.*

Remark 3.4. Note that the other direction does not follow. In a KS proof, the rules s and m can be interleaved, and this behavior cannot be captured by simple combinatorial flows.

Corollary 3.5. *System* SKS *p-simulates substitution-free combinatorial flows.*

After having established the substitution-free combinatorial flows are p-equivalent to SKS, let us now investigate what happens when substitution is present. The substitution rule in a deductive system is given as follows:

$$\mathsf{sub}\ \frac{A}{A\sigma} \tag{3}$$

It replaces a formula A by the formula that is obtained by applying the substitution σ to A.

We define sSKS to be the system SKS + sub. It is important to note that unlike the other rules (shown in Figure 2) the rule sub in (3) cannot be applied inside a context. It is always applied to the whole formula. The reason is that the rule is not "strongly sound", in the sense that the premise does not imply the conclusion, as it is the case with the other inference rules. This means, in particular, that it does not make sense to speak of derivations in sSKS, but only of proofs with no premise.

Theorem 3.6. *Combinatorial flows p-simulate* sSKS.

The basic idea of the proof is to simulate the application of a substitution $\sigma = \{a_1 \mapsto B_1, \ldots, a_n \mapsto B_n\}$ in the sub-rule in sSKS by the substitution of the identity flow id_{B_i} for the variable a_i for each $i = 1..n$. But since in combinatorial flows the replacement is not performed simultaneously, we have to do a renaming first, in order to avoid unwanted variable capturing.

For the other direction, some more work is necessary. The reason is that in sSKS, substitution is a global rule, whereas in combinatorial flows it is a local activity, which is more flexible. To solve this problem, we use the notion of *extension*, which allows for abbreviations in a syntactic proof, by allowing additional proper axioms (called *extension axioms*) of the shape $a_i \Leftrightarrow B_i$ for $1 \leqslant i \leqslant k$, where k is a fixed natural number, the a_i are fresh propositional variables (called *extension variables*) which abbreviate formulas B_i (called *extension formulas*), such that a variable a_i does neither occur in the premise of the proof, nor in the conclusion of the proof, nor in any B_j with $j \leqslant i$. There are two ways to add extension to SKS. The first, as done in [3], is to use the conjunction

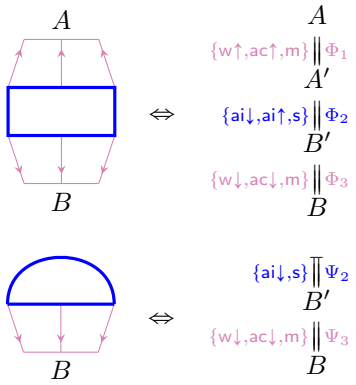
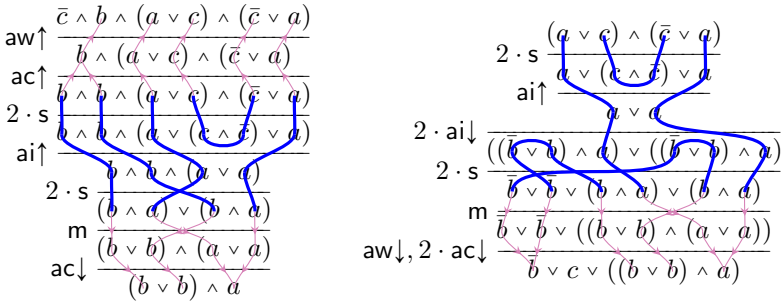


Figure 4. Left: Statements of Theorem 3.2



Right: Examples corresponding to the second and third example in Figure 1

$$\mathsf{id}\ \frac{}{a, \bar{a}} \qquad \vee\ \frac{\Gamma, A, B}{\Gamma, A \vee B} \qquad \wedge\ \frac{\Gamma, A \qquad \Delta, B}{\Gamma, \Delta, A \wedge B}$$

$$\mathsf{weak}\ \frac{\Gamma}{\Gamma, A} \qquad \mathsf{cont}\ \frac{\Gamma, A, A}{\Gamma, A} \qquad \mathsf{cut}\ \frac{\Gamma, A \qquad \Delta, \bar{A}}{\Gamma, \Delta}$$

Figure 5. A sequent calculus for classical logic

of the extension axioms as premise in an SKS-derivation. The second one, as done in [12], [10], is to transform the extension axioms into deep inference rules:

$$\mathsf{ext}{\downarrow}\ \frac{a_i}{B_i} \qquad \mathsf{ext}{\downarrow}\ \frac{\bar{a}_i}{\bar{B}_i} \qquad \mathsf{ext}{\uparrow}\ \frac{\bar{B}_i}{\bar{a}_i} \qquad \mathsf{ext}{\uparrow}\ \frac{B_i}{a_i} \tag{4}$$

Unlike the substitution rule (3), the rules in (4) can be applied deeply in any context (provided the global condition for extension is satisfied), and thus are local, and can be used to simulate the substitution in combinatorial flows. We let $\mathsf{eSKS} = \mathsf{SKS} + \mathsf{ext}{\downarrow} + \mathsf{ext}{\uparrow}$.

Theorem 3.7. *System* eSKS *p-simulates combinatorial flows.*

Corollary 3.8. *Combinatorial flows and* sSKS *and* eSKS *are all p-equivalent to each other.*

4. Sequent Calculus and Frege Systems

In [8], Hughes has already shown how to translate (cut-free) sequent calculus proofs into combinatorial proofs, using the notion of *lax combinatorial proofs*. We show here a translation of sequent proofs into simple combinatorial flows that does not need this detour. Then we show how sequent proofs with cut are translated into (substitution-free) combinatorial flows with cut.

Figure 5 shows the one-sided sequent calculus for classical propositional logic that we will use here, and that we call LK. But it should be clear that any other sound and complete sequent system could be used as well. A proof Π in LK is called *cut-free* if it does not use the cut-rule. The *size* of a proof Π in LK, denoted by $|\Pi|$ is the sum of the sizes of the sequents occurring in Π.

Theorem 4.1. *Simple combinatorial flows and cut-free* LK *are p-equivalent.*

Theorem 4.2. *Substitution-free combinatorial flows are p-equivalent to* LK.

For Frege systems we have a similar result. Recall that a *Frege system* consists of some finite (but complete) set of axioms and the rule *modus ponens* that allows to deduce B from A and $\bar{A} \vee B$. We speak of an *extended Frege system* if we add extension axioms (as discussed above). Finally, we speak of a *Frege system with substitution* if we add the sub-rule (3) to a Frege system.

Theorem 4.3. *Substitution-free combinatorial flows are p-equivalent to Frege systems. Combinatorial flows (with substitution) are p-equivalent to Frege systems with substitution and to extended Frege systems.*

References

[1] George Boolos. Don't eliminate cut. *Journal of Philosophical Logic*, 13:373–378, 1984.

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

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

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

[6] Alessio Guglielmi, Tom Gundersen, and Lutz Straßburger. Breaking paths in atomic flows for classical logic. In *LICS 2010*, 2010.

[7] Dominic Hughes. Proofs Without Syntax. *Annals of Mathematics*, 164(3):1065–1076, 2006.

[8] Dominic Hughes. Towards Hilbert's 24$^{\text{th}}$ problem: Combinatorial proof invariants: (preliminary version). *Electr. Notes Theor. Comput. Sci.*, 165:37–63, 2006.

[9] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, 1989.

[10] Novak Novakovic and Lutz Straßburger. On the power of substitution in the calculus of structures. *ACM Trans. Comput. Log.*, 16(3):19, 2015.

[11] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 294(3):473–488, 2003.

[12] Lutz Straßburger. Extension without cut. *Annals of Pure and Applied Logic*, 163(12):1995–2007, 2012.

[13] Lutz Straßburger. Combinatorial Flows and Proof Compression. Research Report RR-9048, Inria Saclay, 2017.