

From Syntactic Proofs to Combinatorial Proofs

Matteo Acclavio and Lutz Straßburger

Inria Saclay & LIX, Ecole Polytechnique, France

Abstract. In this paper we investigate Hughes' combinatorial proofs as a notion of proof identity for classical logic. We show for various syntactic formalisms including sequent calculus, analytic tableaux, and resolution, how they can be translated into combinatorial proofs, and which notion of identity they enforce. This allows the comparison of proofs that are given in different formalisms.

1 Introduction

Proof theory plays an important role in many areas of computer science. However, unlike many other mathematical fields, it is not able to identify its objects. We do not have a clear understanding of when two proofs are the same. The standard proof theoretical answer to this question is *normalization*: two proofs are the same, if they have the same normal form. This certainly makes perfect sense from the viewpoint of functional programming and the Curry-Howard-correspondence, where proofs are programs and the proof normalization is the execution of the program. However, from the viewpoint of logic programming and proof search, this only makes little sense, since all considered proofs are already in normal form.

An alternative approach to the question of proof identity is based on rule permutations. Two proofs are considered the same if they can be transformed into each other by a series of simple rule permutation steps. The fundamental problem with this approach is that both proofs have to be presented in the same proof system. In fact, one can say that proof theory, in its current form, *is not the theory of proofs but the theory of proof systems*. The question of comparing two proofs that are given in two different proof systems (for example, analytic tableaux and resolution) does not even make sense. And most of the important theorems of proof theory, like soundness, completeness, cut admissibility, proof complexity, or focusing, are not about proofs but about proof systems.

Combinatorial proofs [10, 11] have been introduced by Hughes to address this problem. They are graphical presentations of proofs, independent from the syntactic restrictions of proof formalisms. Nonetheless, combinatorial proofs form a proof system in the sense of Cook and Reckhow [3], the correctness of a combinatorial proof can be checked in polynomial time in the size of the proof. However, the precise relation between combinatorial proofs and syntactic proofs has so far been discussed only on a superficial level. In [11], Hughes shows the relation between combinatorial proofs and a nonstandard version of the sequent calculus LK, and in [18], the relation to the deep inference system SKS is shown.

In this paper we explore the relation between combinatorial proofs and syntactic proofs in various formalisms, in particular, we look at a one-sided variant of LK [8] which has an explicit contraction and weakening rule and in which the conjunction rule is multiplicative, and at G3p [19] in which the conjunction rule is additive and there are no contraction and weakening rules. Then we also look at analytic tableaux [15], and resolution. We will show how a syntactic proof in each of these formalisms is translated into a combinatorial proof, and when a combinatorial proof can be translated back. Note that this is not always possible. Even though all systems are semantically complete, i.e., can prove all theorems, they do not see all proofs.

We will also define for analytic tableaux and for resolution a syntactic equivalence on proofs, and then show that this equivalence coincides with the one imposed by combinatorial proofs. This justifies the use of combinatorial proofs for proof identity: two proofs are the same if they are mapped to the same combinatorial proof.¹ To our knowledge, this is the first proposal for a notion of proof identity that allows us to compare syntactic proofs in different formalisms.

The paper is organized as follows: We first give some preliminaries on combinatorial proofs in Section 2. Then we discuss the relation between combinatorial proofs and sequent calculus in Section 3, and finally, in Sections 4 and 5, we investigate the translation from tableaux and resolution into combinatorial proofs.

2 Preliminaries on combinatorial proofs

For simplicity, we consider formulas (denoted by capital Latin letters A, B, C, \dots) in negation normal form², generated from a countable set $\mathcal{V} = \{a, b, c, \dots\}$ of propositional variables by the following grammar: $A, B ::= a \mid \bar{a} \mid A \wedge B \mid A \vee B$, where \bar{a} is the negation of a . The negation can then be defined for all formulas using the De Morgan laws $\overline{\bar{A}} = A$, and $\overline{A \wedge B} = \bar{A} \vee \bar{B}$ and $\overline{A \vee B} = \bar{A} \wedge \bar{B}$. An *atom* is a variable or its negation. We use \mathcal{A} to denote the set of all atoms. A *sequent* Γ is a multiset of formulas, written as a list separated by comma: $\Gamma = A_1, A_2, \dots, A_n$. We write $\bar{\Gamma}$ to denote the sequent $\bar{A}_1, \bar{A}_2, \dots, \bar{A}_n$. We define the *size* of a sequent Γ , denoted by $|\Gamma|$, to be the number of atom occurrences in it. We write $\wedge \Gamma$ (resp. $\vee \Gamma$) for the conjunction (res. disjunction) of the formulas in Γ and $F^{\vee k} := \underbrace{F \vee \dots \vee F}_k$ ($F^{\wedge k} := \underbrace{F \wedge \dots \wedge F}_k$) the disjunction (conjunction) of k copies of F .

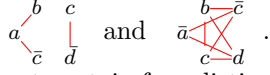
A *graph* $\mathfrak{G} = \langle V_{\mathfrak{G}}, E_{\mathfrak{G}} \rangle$ consists of a set of *vertices* $V_{\mathfrak{G}}$ and a set of *edges* $E_{\mathfrak{G}}$ which are two-element subsets of $V_{\mathfrak{G}}$. We omit the index \mathfrak{G} when it is clear from context. For $v, w \in V$ we write vw for $\{v, w\}$. 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 set L , a graph \mathfrak{G} is *L-labeled* if every vertex of \mathfrak{G} is associated with an element L , called its *label*.

¹ However, this paper does not speak about normalization of combinatorial proofs. For this topic, the reader is referred to [11, 18, 17].

² Note that this is only a cosmetic limitation. The theory of combinatorial proofs can easily be extended to the full language including implication and general negation.

If we associate to each atom a a single vertex labeled with a then every formula A uniquely determines an (\mathcal{A} -labeled) graph $\mathfrak{G}(A)$ that is constructed via the operations \wedge and \vee . We define $\mathfrak{G}(\Gamma) = \mathfrak{G}(\vee\Gamma)$. It is easy to see that 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 .

Example 2.1 Let $A = (a \wedge (b \vee \bar{c})) \vee (c \wedge \bar{d})$ then $\bar{A} = (\bar{a} \vee (\bar{b} \wedge c)) \wedge (\bar{c} \vee d)$.

The two graphs $\mathfrak{G}(A)$ and $\mathfrak{G}(\bar{A}) = \overline{\mathfrak{G}(A)}$ are: .

A graph $\langle V, E \rangle$ is called a *cograph* if V does not contain four distinct vertices u, v, w, z with $uv, vw, wz \in E$ and $vz, zu, uw \notin E$. We have the following well-known proposition, which can already be found in [6].

Proposition 2.2 *A graph is equal to $\mathfrak{G}(A)$ for some A iff it is a cograph.*

The following definitions are due to Retoré [14]. An *R&B-graph* $\mathfrak{G} = \langle V, R, B \rangle$ is a triple such that $\langle V, R \rangle$ and $\langle V, B \rangle$ are graphs and such that B is a perfect matching on V , i.e., no two edges in B are adjacent and every vertex $v \in V_{\mathfrak{G}}$ is incident to an edge in B . We write $\mathfrak{G}^{\downarrow}$ for $\langle V, R \rangle$. An *R&B-cograph* is an R&B-graph $\mathfrak{G} = \langle V, R, B \rangle$ where $\mathfrak{G}^{\downarrow} = \langle V, R \rangle$ is a cograph.

A *cordless \mathfrak{a} -cycle* in $\mathfrak{G} = \langle V, R, B \rangle$ is a set $\{v_1, \dots, v_{2n}\} \subseteq V$ of vertices such that $v_{2n}v_1, v_2v_3, \dots, v_{2n-2}v_{2n-1} \in B$ and $v_iv_j \in R$ if and only if $i = 2k + 1$ and $j = 2k + 2$ for some $0 \leq k \leq n - 1$. We say an R&B-graph is *\mathfrak{a} -acyclic* if it has no cordless \mathfrak{a} -cycle. Following [14] we will draw B -edges in blue/bold, and R -edges in red/regular. Below are four examples:



The first one is not an R&B-cograph, the other three are. The second one has a chordless \mathfrak{a} -cycle, and the last two are \mathfrak{a} -acyclic.

A *homomorphism* $f: \mathfrak{G} \rightarrow \mathfrak{G}'$ 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 *skew fibration*, denoted as $f: \mathfrak{G} \rightsquigarrow \mathfrak{G}'$, is a graph homomorphism such that 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 $f(w)w' \notin E'_{\mathfrak{G}}$.

Let $\mathfrak{C} = \langle V, R, B \rangle$ be an R&B-graph and $f: \mathfrak{C}^{\downarrow} \rightarrow \mathfrak{G}$ be a homomorphism and let \mathfrak{G} be \mathcal{A} -labeled (where \mathcal{A} is the set of atoms). We say f is *axiom preserving* iff $xy \in B$ implies that the labels of $f(w)$ and $f(v)$ are dual to each other. We are now ready to give the definition of a combinatorial proof.

Definition 2.3 *A combinatorial proof of a sequent Γ consists of a non-empty \mathfrak{a} -acyclic R&B-cograph \mathfrak{C} and an axiom preserving skew fibration $f: \mathfrak{C}^{\downarrow} \rightsquigarrow \mathfrak{G}(\Gamma)$.*

In [10], Hughes has shown that combinatorial proofs form a proof system in the sense of Cook and Reckhow [3], i.e., correctness can be checked in polynomial time, that is, given a formula A and a R&B-graph $\langle V, R, B \rangle$ and a map $f: \langle V, R \rangle \rightarrow \mathfrak{G}(A)$, it can be checked in polynomial time in the size of the input whether (1) $\langle V, R \rangle$ is a cograph, (2) $\langle V, R, B \rangle$ is \mathfrak{a} -acyclic, and (3) f is axiom preserving and a skew fibration.

We follow the notational convention of [18, 17] and write $\phi: \Gamma \vdash \Delta$, to denote a combinatorial proof for the sequent $\bar{\Gamma}, \Delta$, and say that Γ is its *premise* and Δ its *conclusion*. We write $\phi: \circ \vdash \Delta$ (resp. $\phi: \Gamma \vdash \circ$) if Γ (resp. Δ) is empty.³ Note that if $\phi: \Gamma \vdash \Sigma, \Delta$ is a combinatorial proof then so is $\phi = \phi': \bar{\Sigma}, \Gamma \vdash \Delta$.

Following [18], we draw combinatorial proofs as follow: let $\phi: \Gamma \vdash \Delta$ be a combinatorial proof with skew fibration $f: \bar{\mathfrak{C}}_\Gamma \vee \mathfrak{C}_\Delta \mapsto \bar{\mathfrak{G}}(\Gamma) \vee \mathfrak{G}(\Delta)$, and let $F(\bar{\mathfrak{C}}_\Gamma)$ and $F(\mathfrak{C}_\Delta)$ be the formula trees corresponding to the cographs $\bar{\mathfrak{C}}_\Gamma$ and \mathfrak{C}_Δ respectively. We write $\Gamma, F(\bar{\mathfrak{C}}_\Gamma), F(\mathfrak{C}_\Delta)$, and Δ above each other, and we draw the B -edges in bold/blue and the map f by thin/purple arrows (see Fig. 1).

The relation between combinatorial proofs and deep inference proofs in system SKS [2] has been detailed out in [18, 17]. We will not go into details here, but we will make heavy use of the following theorem, originally shown in [11, 16]:

Theorem 2.4 *Let A and B be formulas. Then the following are equivalent:*

- *There is a skew fibration $f: \mathfrak{G}(A) \mapsto \mathfrak{G}(B)$;*
- *There is a derivation Φ from A to B using only deep contraction $\mathfrak{c}\downarrow: A \vee A \rightarrow A$ and deep weakening $\mathfrak{w}\downarrow: A \rightarrow A \vee B$, modulo associativity and commutativity of \wedge and \vee , denoted as $A \models_{\mathfrak{w}\downarrow, \mathfrak{c}\downarrow} B$;*
- *There is a derivation $\bar{\Phi}$ from \bar{B} to \bar{A} using only deep cocontraction $\mathfrak{c}\uparrow: A \rightarrow A \wedge A$ and deep coweakening $\mathfrak{w}\uparrow: A \wedge B \rightarrow A$, modulo associativity and commutativity of \wedge and \vee , denoted as $\bar{B} \models_{\mathfrak{w}\uparrow, \mathfrak{c}\uparrow} \bar{A}$.*

This suggests the following definition:

Definition 2.5 A formula F' is a *skew* of a formula F iff $F \models_{\mathfrak{c}\uparrow, \mathfrak{w}\uparrow} F'$.

3 Sequent calculus

We recall in Fig. 2 the formulation of LK cut-free sequent calculus that we use in this paper. Moreover, we refer to MLL as the fragment of LK calculus consisting of the rules \wedge, \vee, AX only. We speak of MLL + mix if we additionally allow mix.

Theorem 3.1 ([14]) *Let A and B be formulas. There is an \mathfrak{x} -acyclic $R\mathfrak{E}B$ -cograph \mathfrak{C} with $\mathfrak{C}^\downarrow = \mathfrak{G}(\Gamma)$ iff there is a proof of Γ in MLL + mix.*

In [11], Hughes has shown how to translate an LK-proof into a combinatorial proof. In this paper we also consider the (cut-free) sequent calculus G3p [19], shown in Figure 3, which has no explicit contraction and weakening rules.

Theorem 3.2 *If $d(F)$ is a derivation of the formula F in G3p, then there is a combinatorial proof $\phi_{d(F)}: \circ \vdash F$, such that every B -edges in $\phi_{d(F)}$ correspond to an instance of the AX-rule in $d(F)$.*

³ It cannot happen that both Γ and Δ are empty.

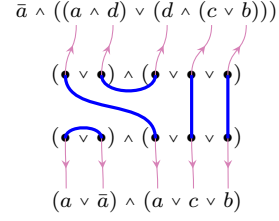


Fig. 1. A combinatorial proof

$$\frac{}{\vdash A, \bar{A}} \text{AX} \quad \frac{}{\vdash \Gamma, A, B} \vee \quad \frac{}{\vdash \Gamma, A \wedge B, \Delta} \wedge \quad \frac{}{\vdash \Gamma} \text{W} \quad \frac{}{\vdash \Gamma, A, A} \text{C} \quad \left| \quad \frac{}{\Gamma, \Delta} \text{mix} \right.$$

Fig. 2. Left: Sequent system LK (cut free) for classical logic, **Right:** The mix rule

$$\frac{}{\vdash A, \bar{A}, \Gamma} \text{AX}_{\text{G3p}} \quad \frac{}{\vdash A, B, \Gamma} \vee_{\text{G3p}} \quad \frac{}{\vdash A, \Gamma \vdash B, \Gamma} \wedge_{\text{G3p}}$$

Fig. 3. Rules of cut-free G3p sequent calculus

$$\frac{}{\vdash a, \bar{a}, c, \bar{b}} \text{AX} \quad \frac{}{\vdash a, \bar{a}, b, \bar{b}} \text{AX} \quad \xrightarrow{\wedge_{\text{G3p}}} \quad \frac{}{\vdash a, \bar{a}} \text{AX} \quad \frac{}{\vdash a, \bar{a}, c} \text{C} \quad \frac{}{\vdash b, \bar{b}} \text{AX} \quad \xrightarrow{\wedge_{\text{LK}}} \quad \begin{array}{c} (\vee) \wedge (\vee) \\ \vdots \quad \vdots \\ a, \bar{a}, (c \wedge b), \bar{b} \end{array} \quad \xrightarrow{\text{correct}} \quad \begin{array}{c} (\vee) \\ \vdots \\ a, \bar{a}, (c \wedge b), \bar{b} \end{array}$$

Fig. 4. From left to right: A G3p proof, the corresponding LK proof, the naive (incorrect) translation, and finally, the correct combinatorial proof

$$\frac{}{\vdash A, B, C, D, \Gamma} \vee \quad \frac{}{\vdash A, B, C, D, \Gamma} \vee \quad \frac{}{\vdash A, C, D, \Gamma \vdash B, C, D, \Gamma} \wedge \quad \frac{}{\vdash A, C, D, \Gamma} \vee \quad \frac{}{\vdash B, C, D, \Gamma} \vee \\ \frac{}{\vdash A \vee B, C, D, \Gamma} \vee \quad \frac{}{\vdash A, B, C \vee D, \Gamma} \vee \quad \frac{}{\vdash A \wedge B, C, D, \Gamma} \wedge \quad \frac{}{\vdash A, C \vee D, \Gamma} \vee \quad \frac{}{\vdash B, C \vee D, \Gamma} \vee \\ \frac{}{\vdash A \vee B, C \vee D, \Gamma} \vee \quad \frac{}{\vdash A, C, \Gamma \vdash A, D, \Gamma} \wedge \quad \frac{}{\vdash B, C, \Gamma \vdash B, D, \Gamma} \wedge \quad \frac{}{\vdash A, C, \Gamma \vdash B, C, \Gamma} \wedge \quad \frac{}{\vdash A, D, \Gamma \vdash B, D, \Gamma} \wedge \\ \frac{}{\vdash A \wedge B, C \wedge D, \Gamma} \wedge \quad \frac{}{\vdash A \wedge B, C \wedge D, \Gamma} \wedge \quad \frac{}{\vdash A \wedge B, C, \Gamma} \wedge \quad \frac{}{\vdash A \wedge B, D, \Gamma} \wedge$$

Fig. 5. G3p proof equivalence

Proof This follows from the result on LK in [11] and the observation that any G3p derivation can be simulated in LK by making heavy use of C and W, but without changing the AX-instances in the proof. \square

Remark 3.3 Observe that the relation between B -edges in $\phi_{d(F)}$ and AX-instances in $d(F)$ is not a bijection, as can be seen by the example (due to Hughes) shown in Fig. 4 where the naive translation is not a combinatorial proof because the induced mapping is not a skew fibration. In the correct combinatorial proof the B -edge coming from the AX-instance on b, \bar{b} is deleted.

In sequent calculus, the standard notion of proof identity is defined via rule permutations. The generating permutation we use for G3p are shown in Fig. 5.

Theorem 3.4 *If $d(\Gamma)$ and $d'(\Gamma)$ are two G3p derivations that are equivalent modulo the rule permutations in Figure 5, then $\phi_{d(\Gamma)} = \phi_{d'(\Gamma)}$.*

To prove this theorem, we will prove a stronger result for analytic tableaux in the next section and then reflect it back to the sequent calculus. For LK, such a statement is less trivial, since due to the presence of weakening and contraction, there are permutations that delete or duplicate subproofs, and such operations are not preserved by combinatorial proofs (see Remark 3.3 above).

4 Analytic Tableaux

Analytic tableaux are a formalism for refutations based on the decomposition of the negation \bar{F} of a formula in order to find contradictions between its sub-

formulas and conclude, by completeness, the provability of F . This is done by expanding a formula \bar{F} over a tree of its subformulas via *expansion rules* until all branches contain a formula and its negation. The resulting tree with root \bar{F} is related to the disjunctive normal form $\text{DNF}(\bar{F})$ as follows: each branch represents the conjunction of the formulas appearing in its nodes and the tree represents the disjunction of its branches.

We work here with a non-cumulative formulation of the tableaux formalism.

Definition 4.1 (Tableau) A *tableau* is a rooted binary tree with nodes labeled by sets of occurrences of formulas according with the following conditions:

- The tree consisting of a single node with formula set $\{\bar{F}\}$ is a tableau of F ;
- If T_F is a tableau of F , then the tree obtained by the application of one of the following *tableau expansion rules* is a tableau of F :
 - If ℓ is a leaf of T_F with formula set L containing a conjunction $A \wedge B$, then the tree obtained extending T_F with a leaf ℓ_1 attached to ℓ with formula set $L \cup \{A, B\} \setminus \{A \wedge B\}$ is a tableau of F ;
 - If ℓ is a leaf of T_F with formula set L containing a disjunction $A \vee B$, then the tree obtained by extending T_F with two leaves ℓ_1 and ℓ_2 attached to ℓ with respective formula sets $L \cup \{A\} \setminus \{A \vee B\}$ and $L \cup \{B\} \setminus \{A \vee B\}$ is a tableau of F .

A branch of a tableau is *closed* if its leaf contains a formula and its negation, otherwise it is *open*. A tableau is *closed* if all its branches are. A branch is *atomic closed* if the closing formulas are atoms. A tableau is *full* if no expansion rule can be applied to its open branches, *non-expandable* if no expansion rule can be applied to any branch.

If T_G is a tableau of G , we denote T_G, A the tableau obtained by adding to T_G the formula A to each node. A *redundant* tableau T_F is a full tableau of F such that there is a closed tableau T_G such that T_F has two closed branches T_G, A and T_G, \bar{B} . Figures 8 and 9 show some examples.

There is a one-to-one correspondence between atoms in the leaves of a full tableau of with root labeled by a formula \bar{F} and occurrences of atoms in its disjunctive normal form (denoted DNF) clauses due to the correspondence between tableau expansion rules and dual clause form algorithm [7].

Proposition 4.2 *If T_F is a full tableau of F with a non-closed branch, then*

$$\text{DNF}(\bar{F}) = \bigvee_{L_i \text{ leaves of } T_F} \left(\bigwedge_{A_{ij} \text{ formulas in } L_i} A_{ij} \right).$$

There is a close correspondence between G3p proofs and tableaux, which has been established in [7], and which allows to define a *flipping translation* (here denoted as the function Flip_{G3p}), which associates to any full tableau T_F a G3p derivation tree $d_T(F)$ “by tuning it upside-down and negating everything” and viceversa. In particular, we associate an axiom-rule $\text{AX}_{L_i}^T$ with conclusions $\vdash A_1, \dots, A_n$ whenever there is a non-closed leaf L_i (with formulas $\bar{A}_1, \dots, \bar{A}_n$); we define the *theory of T_F* (denoted \mathcal{T}_{T_F}) to be the set of such axioms. An example

$$\begin{array}{ccc}
 \begin{array}{c}
 (a \vee b) \wedge (\bar{a} \vee c) \\
 (a \vee b), (\bar{a} \vee c) \\
 (a \vee b), \bar{a}, c \\
 \swarrow \quad \searrow \\
 \boxed{a}, \boxed{\bar{a}}, c \quad b, \bar{a}, c
 \end{array}
 & \xrightarrow{\text{Flip}_{\text{G3p}}} &
 \begin{array}{c}
 \frac{}{\vdash a, \bar{a}, \bar{c}} \text{AX} \quad \frac{}{\vdash b, a, \bar{c}} \text{AX}^T \\
 \wedge \\
 \frac{}{\vdash \bar{a} \wedge \bar{b}, \bar{c}, a} \\
 \vee \\
 \frac{}{\vdash \bar{a} \wedge \bar{b}, \bar{c} \vee a} \\
 \vee \\
 \frac{}{\vdash (\bar{a} \wedge \bar{b}) \vee (\bar{c} \vee a)}
 \end{array}
 \end{array}$$

Fig. 6. A tableau of $(\bar{a} \wedge \bar{b}) \vee (\bar{c} \vee a)$ and its associate G3p derivation

is shown in Fig. 6. This translation suggests the definition of an equivalence relation, shown in Fig. 7, over tableaux derived from the G3p proof equivalence (Fig. 5). The last equation does not occur in the G3p equivalence because it cannot be written as a rule permutation. Its interpretation is the following: if $T_{\bar{F}}$ is closed, then so is any tableau with root $\Gamma, A \vee B$. Hence, any full tableau $T_{\bar{F}, \bar{B}} \neq T_{\bar{F}}, B$ is closed, and we consider the contribution given by the formula \bar{B} to be superfluous to the closure of $T_{\bar{F}}, A \vee B$. However, if B is not a closing formula, we can not discard the corresponding tableau branch because otherwise we lose the information about the branch leaves (see Fig. 8).

In order to keep track of the information about branching and avoid mismatching in translation, we consider different occurrences a_1, \dots, a_n of the same atom a in F as different atoms for the following definitions.

Definition 4.3 (Tableaux oversaturation, Sprout) If T_F is a tableau of F , its *oversaturation* is a tree $T_{\bar{F}}^*$ obtained by updating the formula sets of each vertex of T_F inductively from the leaves to the root as follows:

- If a leaf is closed, the formulas of this vertex in $T_{\bar{F}}^*$ are only the two closing formulas;
- If a vertex of T_F has two children then its formulas are $A \vee B, F_1, \dots, F_n$. If both its children have been updated then:
 - If one child contains A and the other contains B then they contain two skews F'_i and F''_i (it can be $F'_i = F''_i$) of F_i , then we replace F_i by $F'_i \wedge F''_i$ in $T_{\bar{F}}^*$. If only one of the children contains a skew F'_i of F_i , then we replace F_i by F'_i ;
 - If no child contains A (similary if no child contains B), we replace the vertex and the subtree having this vertex as root with the child containing neither A nor B and its corresponding subtree;
- If a vertex of T_F has one child then its formulas are $F_1 \wedge F_2, F_3, \dots, F_n$. If the child contains a skew F'_i of F_i , then we replace F_i by F'_i in $T_{\bar{F}}^*$.

The root formula of T^* is called the *sprout* of T_F , denoted $\text{spr}_T(F)$.

Lemma 4.4 *If T_F is a tableau of F , then $\text{spr}_T(F)$ is a skew of \bar{F} .*

Proof By induction over the structure of T_F . If T_F has no branching then \bar{F} is either a conjunction of formulas or $\bar{F} = (A \wedge \bar{A}) \wedge \bar{F}'$; then \bar{F} is respectively $\text{spr}_T^*(F)$ or $\bar{F} \models_{w\uparrow} (A \vee \bar{A})$. If T_F has branchings we can assume without loosing generalities T_F expandable with root $\bar{F} = (A \vee B) \wedge C$ and leaves $\{A, C\}$ and $\{B, C\}$. Then $\text{spr}_T(F) = (A \wedge B) \vee (C \wedge C)$, and therefore $\bar{F} \models_{c\uparrow} \text{spr}_T(F)$. We conclude by composition that $\bar{F} \models_{c\uparrow, w\uparrow} \text{spr}_T(F)$. \square

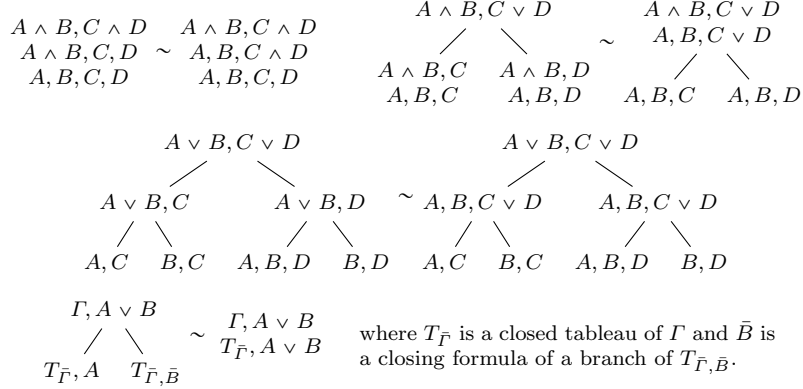


Fig. 7. Expansion rules permutation generating tableaux standard equivalence

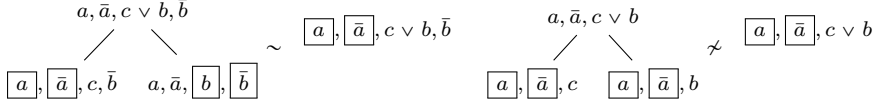


Fig. 8. Tableaux equivalences in case of redundant tableaux

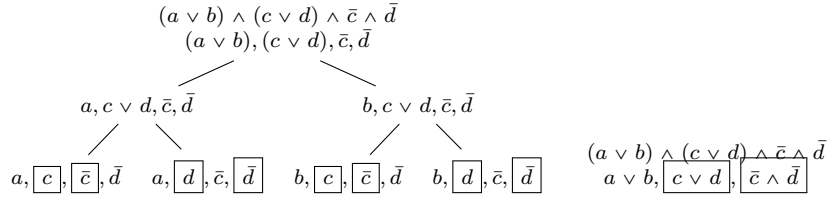


Fig. 9. A redundant tableau and a non-redundant one of the same formula

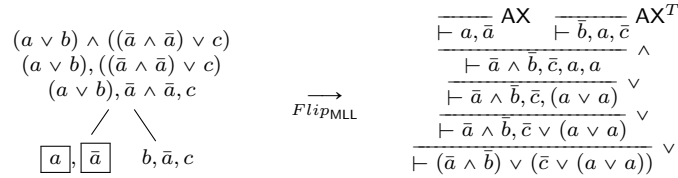


Fig. 10. The oversaturation of tableau in Figure 6 and the associated MLL derivation.

Definition 4.5 (Harvest of T) We define the *harvest* of a tableau T_F , denoted $H_T(F)$, as as the conjunction over the non-closed leaves of T_F^* of the disjunction of the formulas in each of these leaves:

$$H_T(F) = \bigvee_{L_i \text{ non-closed leaves of } T_F^*} \left(\bigwedge_{A_{ij} \text{ formulas in } L_i} A_{ij} \right) .$$

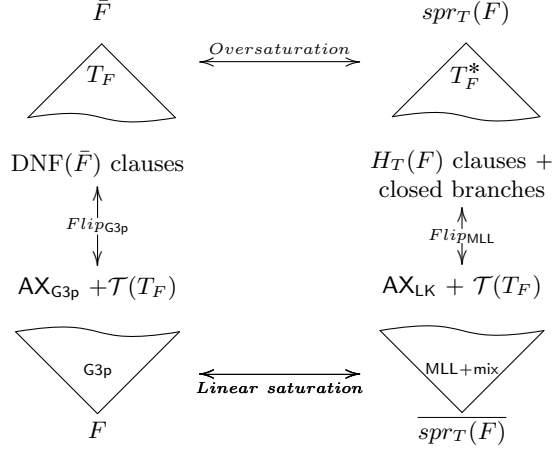


Fig. 11. Rosetta stone of tableaux translation

If T_F is a tableau and we define the *theory* of T_F (denoted $\mathcal{T}(T_F)$) as a set of additional axiom-rules $\text{AX}_{L_i}^T$ with conclusions $\vdash A_1, \dots, A_n$ where $\bar{A}_1, \dots, \bar{A}_n$ are the formulas in the non-closed leaf L_i , we have the following:

Lemma 4.6 *For every tableau T_F , the formula $\overline{\text{spr}_T(F)}$ is derivable in $\text{MLL} + \mathcal{T}(T_F)$. Furthermore, if T_F is closed then $\text{spr}_T(F)$ is provable in $\text{MLL} + \text{mix}$, and finally, if T_F is non-redundant and closed then $\text{spr}_T(F)$ is provable in MLL .*

Proof By the Flip_{G3p} operation we have a G3p proof that we translate inductively via a procedure that we call *linear saturation* into an proof in $\text{MLL} + \text{mix} + \mathcal{T}(T_F)$ of $\overline{\text{spr}_T(F)}$:

- a G3p axiom with conclusion $\vdash A, \bar{A}, \Gamma$ is translated into an axiom $\vdash A, \bar{A}$;
- a $\mathcal{T}(T_F)$ -axiom in G3p remains the same $\mathcal{T}(T_F)$ -axiom (in LK);
- a \vee_{G3p} instance is translated into a \vee_{LK} and formulas are replaced with their relative skews if both principal formulas occur. If one or both principal formulas are missed, this inference disappears during the translation.
- a \wedge_{G3p} instance is translated into a \wedge_{LK} instance (respectively mix instance) on the relative formulas skews if both principal formulas occur (if none of its principal formulas occur) followed by \vee_{LK} instances on whenever two skews of a same formula F_i appear in both premises. If one of the two principal formula is missed, we consider the \wedge_{G3p} inference premises in which this should belong, we keep this branch in our derivation, we discard the other one and the inference disappears.

The other statements follow immediately by case inspection. \square

Figure 10 shows an example for Lemma 4.6. Its proof suggests that analogously to the operation Flip_{G3p} that relates tableaux and G3p derivations, we can define an operation Flip_{MLL} that relates oversaturated tableaux and derivations $\text{MLL} + \text{mix} + \mathcal{T}(T_F)$ “by flipping it upside-down and negating everything”.

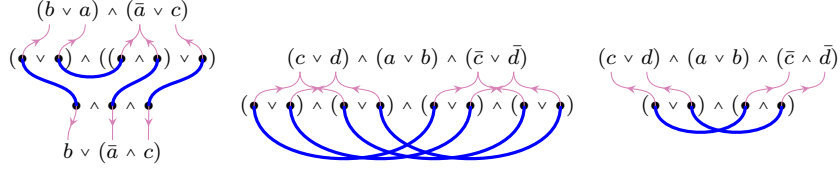


Fig. 12. The combinatorial proofs associate to the tableaux in Figures 6 and 9

The interactions between the two flipping translation, the oversaturation and the sprouting procedure can be summarized by the diagram in Figure 11.

We can now give a polynomial translation from tableaux into combinatorial proofs.

Theorem 4.7 *Let T_F be a full tableau for F . If T_F is closed then there is a combinatorial proof $\phi_{T_F} : \bar{F} \vdash \circ$. Otherwise, $\phi_{T_F} : \bar{F} \vdash H_T(F)$. In either case, if T_F is non-redundant the atoms pairs in the formula pairs that close the branches are mapped to the B-edges in ϕ_{T_F} . If T_F is closed, then this is a bijection.*

Proof We define $\phi_{T_F} : \bar{F} \vdash H_T(F)$ as follows:

- The R&B-cograph \mathfrak{C}_{T_F} of ϕ_{T_F} is given by the cograph $\mathfrak{C}_{T_F}^\downarrow = \overline{\mathfrak{G}(\text{spr}_T(F))} \vee \mathfrak{G}(H_T(F))$ enriched with a matching B_{T_F} constructed as follows:
 - For each closed branch of T_F we consider its closing pair of formulas (G, \bar{G}) in its leaf label. For each atom a_i in G and \bar{a}_i in \bar{G} , we define an edge between their corresponding vertices in $\mathfrak{G}(\text{spr}(T))$;
 - For each non-closed branch of T_F , the associate clause of $\text{spr}_T(F)$ occurs in its harvest $H_T(F)$. We define an edge between the vertices corresponding the the associated atoms in $\mathfrak{G}(H_T(F))$ and $\mathfrak{G}(\text{spr}(T))$.
- The skew fibration $f : \mathfrak{C}^\downarrow \rightarrow \mathfrak{G}_{T_F}$ where $\mathfrak{G}_{T_F} = \overline{\mathfrak{C}(\bar{F})} \vee \mathfrak{C}(H_T(F))$ is given by the disjunction $f = f^\dagger \vee 1_{H_T(F)}$ of the identity skew fibration $1_{H_T(F)}$ over $\mathfrak{G}(H_T(F))$ and the skew fibration f^\dagger defined by the sprouting derivation $\bar{F} \vDash \text{spr}_T(F)$.

Similarly if T_F is closed, then the R&B-cograph \mathfrak{C}_{T_F} of ϕ_{T_F} is the cograph $\mathfrak{C}_{T_F}^\downarrow = \overline{\mathfrak{G}(\text{spr}_T(F))}$ enriched with the corresponding matching B_{T_F} while the skew fibration $f : \mathfrak{C}^\downarrow \rightarrow \mathfrak{G}_{T_F}$ where $\mathfrak{G}_{T_F} = \overline{\mathfrak{C}(\bar{F})}$ is given by the sprouting derivation from \bar{F} to $\text{spr}_T(F)$. \square

Examples for this construction are shown in Figure 12.

Theorem 4.8 *If T_F and T'_F are two equivalent tableaux of F , $\phi_{T_F} = \phi_{T'_F}$.*

Proof The two tableaux T_F and T'_F are equivalent if and only if their corresponding derivations $d_T^*(F)$ and $d_{T'}^*(F)$ in $\text{MLL} + \text{mix} + \mathcal{T}(T_F)$ are. By the canonicity of MLL proof nets [1, 14] we conclude that ϕ_{T_F} and $\phi_{T'_F}$ have the same R&B-cograph. Furthermore, from the work in [16], [5], and [11], we know that skew fibrations are canonical. Hence, $\phi_{T_F} = \phi_{T'_F}$. \square

Via the flipping operation $Flip_{G3p}$ we can now immediately obtain the proof of Theorem 3.4.

In order to translate a combinatorial proofs back into a tableau, we associate to a CP $\phi : F \vdash \circ$ with skew fibration $f : \mathfrak{C}^\downarrow \mapsto \mathfrak{G}_{T_F}$ the MLL derivation $d_{T^*}(F')$ represented by the cograph \mathfrak{C}^\downarrow where $F' = \overline{spr_T(F)}$ is obtained by labeling the vertices of \mathfrak{C}^\downarrow with their images under f . Then, by $Flip_{MLL}$ we have an oversaturated tableau T_F^* . If mix is absent, T_F^* contains all the information to invert the oversaturation and reconstruct T_F . However, in the presence of mix , even if we can translate any instance of rule mix-rule into a \wedge -expansion, we can not recover the structure of T_F in general.

5 Resolution

Resolution is a refutation system related to conjunctive normal forms. A resolution proof consist of applying *resolution rule* on clauses of conjunctive normal form of a formula F in order to produce an empty clause. However, the resolution technique does not require the full conversion of a formula to its conjunctive normal form, in the same way tableaux can be closed before a complete expansion. A general resolution proof consist of a sequence of *expansion rules* intercutted by resolution rules terminating with the production of an empty clause or a clause form formula.

In resolution we also denote formulas in *Davis-Putnam's block notation* used in [7, 2, 9] which interprets lists (X_1, \dots, X_n) and $[X_1, \dots, X_n]$ as respectively the conjunction $X_1 \wedge \dots \wedge X_n$ and the disjunction $X_1 \vee \dots \vee X_n$ of their elements.

Definition 5.1 If $F = (C_1, \dots, C_n, C)$ is a formula with $C = [X_1, \dots, X_n]$, we define the following (*resolution expansion rules*):

- if $X_i = A \vee B$, then $F \rightarrow_{\vee}^{A \vee B} (C_1, \dots, C_n, C')$ and we say that the clause C' is generated by the clause C where $C' = [X_1, \dots, X_{i-1}, A, B, X_{i+1}, \dots, X_n]$;
- if $X_i = A \wedge B$ then $F \rightarrow_{\wedge}^{A \wedge B} (C_1, \dots, C_n, C', C'')$ and we say that the clauses C' and C'' are generated by the clause C where C' and C'' are respectively $[X_1, \dots, X_{i-1}, A, X_{i+1}, \dots, X_n]$ and $[X_1, \dots, X_{i-1}, B, X_{i+1}, \dots, X_n]$;

An *expansion of F* is the last formula F' produced by a sequence of application of expansion rules starting from F .

Definition 5.2 (Resolution Rule) If $F = (C_1, C_2, \Sigma)$, $C_1 = [I, X_1, \dots, X_n]$ and $C_2 = [\Delta, \bar{X}_1, \dots, \bar{X}_m]$ where X_i and \bar{X}_i are respectively occurrences of X and \bar{X} , we say $F' = ([I, \Delta], \Sigma)$ is the result of resolving C_1 and C_2 on the *resolving formula X* (denoted $F \rightarrow_X^R F'$) and that the clause $[I, \Delta]$ is generated by the clauses C_1 and C_2 .

Definition 5.3 (Resolution Proof) A *resolution expansion R_F* of a formula F is a sequence of formulas $F = F_0 \rightarrow \dots \rightarrow F_n$ such that F_{i+1} is obtained by F_i by applying an expansion rule or a resolution rule. We call F_n the *result of the resolution expansion R_F* .

$$\begin{array}{c}
\frac{[(a \vee b) \wedge (\bar{a} \vee c)]}{[a \vee b][\bar{a} \vee c]} \wedge \\
\frac{[a, b][\bar{a} \vee c]}{[a, b][\bar{a}, c]} \vee \\
\frac{[a, b][\bar{a}, c]}{[b, c]} \text{Res}^a
\end{array}
\wedge
\frac{[\bar{a} \wedge (a \vee d) \wedge (\bar{d} \vee (b \vee c))]}{[\bar{a} \wedge (a \vee d)][\bar{d} \vee (b \wedge c)]} \wedge \\
\frac{[\bar{a}][a \vee d][\bar{d} \vee (b \wedge c)]}{[\bar{a}][a \vee d][\bar{d}, b \wedge c]} \vee \\
\frac{[\bar{a}][a, d][\bar{d}, b \wedge c]}{[\bar{a}][a, b \wedge c]} \text{Res}^d \\
\frac{[\bar{a}][a, b \wedge c]}{[\bar{a}][a, b][a, c]} \wedge \\
\frac{[\bar{a}][a, b][a, c]}{[b][a, c]} \text{Res}^a
\end{array}
\wedge
\frac{[(a \vee b) \wedge (c \vee d) \wedge \bar{c} \wedge \bar{d}]}{[a \vee b][(c \vee d) \wedge \bar{c} \wedge \bar{d}]} \wedge \\
\frac{[a \vee b][c \vee d][\bar{c} \wedge \bar{d}]}{[a \vee b][c \vee d]} \text{Res}^{c \vee d}$$

Fig. 13. A resolution expansion of $(a \vee b) \wedge (\bar{a} \vee c)$, one of $\bar{a} \wedge (a \vee b) \wedge (\bar{d} \vee (b \vee c))$ and a resolution proof of $(\bar{a} \wedge \bar{b}) \vee (\bar{c} \wedge \bar{d}) \vee c \vee d$.

A resolution expansion is *closed* if F_n contains an empty clause $[\]$, *non-expandable* if no expansion rules can be applied to F_n and *full* if it is non-expandable and no resolution rule can be applied to F_n .

A resolution proof of \bar{F} is a closed resolution expansion of F . Figure 13 shows some examples. If a clause C generates after a certain number of expansions and resolution a clause C' we say that C' is *derived* by C . As for the other proof and refutation systems in this paper, we want to define a notion of equivalence on resolution expansions.

Definition 5.4 We define the *resolution derivation equivalence* to be the smallest equivalence relation over resolution expansions generated by the following relations for any formulas F and G :

- Resolution rule inferences commute with resolution rule and \vee -expansions inferences;
- \vee -expansions inferences commute with both resolution rule and \vee -expansions inferences;
- If F and G do not belong to the same clause then $\rightarrow_F^{\vee} \rightarrow_G^{\wedge} = \rightarrow_G^{\wedge} \rightarrow_F^{\vee}$;
- If F and G do not belong to the same clause then $\rightarrow_F^{\vee} \rightarrow_G^R = \rightarrow_G^R \rightarrow_F^{\vee}$;
- If G does not belong to the same clause of the resolving formula F or its corresponding \bar{F} then $\rightarrow_F^R \rightarrow_G^{\wedge} = \rightarrow_G^{\wedge} \rightarrow_F^R$;
- If F and G belong to the same clause then $\rightarrow_F^{\vee} \rightarrow_G^{\wedge} = \rightarrow_G^{\wedge} \rightarrow_{F_1}^{\vee} \rightarrow_{F_2}^{\vee}$ where F_1 and F_2 are the two copies of F belonging in the two clauses produced by the \wedge -expansion of G .

Moreover, we ask the following condition: if $F_0 \rightarrow \dots \rightarrow F_{k-1} \rightarrow F_k \rightarrow F_{k+1} \dots \rightarrow F_n$ is a resolution expansion such that $F_{k-1} \rightarrow F_k$ is a resolution inference which generate an empty clause, then

$$F_0 \rightarrow \dots \rightarrow F_{k-1} \rightarrow F_k \rightarrow F_{k+1} \dots \rightarrow F_n = F_0 \rightarrow \dots \rightarrow F_{k-1} \rightarrow F_k.$$

Lemma 5.5 *If a formula F' is obtained by applying an expansion rule to a formula F , then F' is a skew of F .*

Proof The first transformation given in Definition 5.1 corresponds to the associativity of \vee , and the second transformation corresponds to $(A \wedge B) \vee \Delta \xrightarrow{c\uparrow}$

$$\begin{aligned} & (A \wedge B) \vee (\Delta \wedge \Delta) \xrightarrow{c\uparrow} ((A \wedge B) \vee (\Delta \wedge \Delta)) \wedge ((A \wedge B) \vee (\Delta \wedge \Delta)) \xrightarrow{4\cdot w\uparrow} \\ & (A \vee \Delta) \wedge (B \vee \Delta) \text{ where } A \wedge B = X_i \text{ and } \Delta = \bigwedge_{j \neq i} X_j. \quad \square \end{aligned}$$

Definition 5.6 (Pseudo-resolution expansion) If $F = (\Gamma, [\Delta])$ is a formula, we define the following (*resolution*) *pseudo-expansion rules*:

- *clause duplication*: $F \rightarrow_{[\Delta]}^{\delta} (\Gamma, [\Delta], [\Delta])$;
- *clause erasing*: $F \rightarrow_{[\Delta]}^{\epsilon} (\Gamma)$.

A *pseudo-resolution expansion* of F is a sequence of formulas $F = F_0 \rightarrow \dots \rightarrow F_n$ such that F_{i+1} is obtained by F_i by applying an expansion rule, a resolution rule or a pseudo-expansion rule.

As for tableaux, we associate to any resolution expansion R_F a formula spr_R which admits a linear derivation from spr_R to the result of the resolution R_F . We introduce an oversaturation procedure to give a pseudo-resolution expansion in which all the resolution rule inference are applied after the expansions and no superfluous information such as non-empty clauses in a closed resolution is kept. We observe that during the oversaturation some clauses may be duplicated if some \wedge -expansion and resolution inferences are permuted.

Definition 5.7 (Oversaturation of R_F) We define the *oversaturation* of R_F as a pseudo-resolution R_F^* obtained by the following procedure:

- $R_F^* = R_F$ and we say that all its resolution rule inference are *active*;
- We proceed by induction over the number of active resolution rule inferences in R_F^* . We start from the last resolution rule inferences $F_k \xrightarrow{X} F_{k+1}$ in the pseudo-expansion R_F^* and we deactivate it as follows:
 - if it generates the empty clause, then we apply to any clause in F_k which do not contain the resolving formulas a clause erasing rule;
 - if no rule inference is applied to the clause generated by an active resolution inference, then move the application of this inference at the end the pseudo-expansion and we deactivate it;
 - if a \vee -expansion is applied to the clause $[X_1, \dots, A \vee B, \dots X_n]$ of F_{k+1} generated by an active resolution inference resolving on a formula Y , then we permute them: we apply a \vee -expansion to the unique clause C_1 in F_k containing $A \vee B$ and then resolve on Y ;
 - if a \wedge -expansion is applied to the clause $[X_1, \dots, A \wedge B, \dots X_n]$ of F_{k+1} generated by an active resolution inference, then we permute them: we apply the \wedge -expansion to the unique clause C_1 in F_k containing $A \wedge B$ and the resolving formula X , we apply a clause duplication on the clause C_2 containing the corresponding resolving formula \bar{X} and then we apply two resolution inferences to the corresponding pairs of clause. These two resolution rule inferences are active.

Figure 14 shows two examples.

Lemma 5.8 *The oversaturation procedure terminates.*

Proof We define the *weight* of a resolution rule generating a clause C in a resolution expansion R_F as the number of all the \vee - and \wedge -expansion inferences applied to any clause derived by C . The weight of a resolution expansion is the sum of the weight of its resolution rules. This decreases at each step. \square

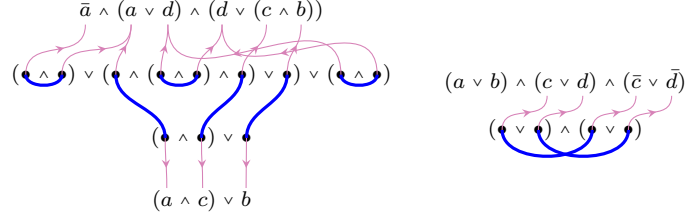


Fig. 16. Combinatorial proofs for the oversaturated resolutions in Figure 14

- We define an edge between each vertex in $\overline{\mathfrak{G}(\text{spr}_R(F))}$ and the corresponding vertex in $\mathfrak{G}(R_F(F))$;
- The skew fibration $f: \mathfrak{C}^\downarrow \rightsquigarrow \mathfrak{G}_{R_F}$ where $\mathfrak{G}_{T_F} = \overline{\mathfrak{C}(F)} \vee \mathfrak{C}(R_F(F))$ is given by the disjunction $f = f^\uparrow \vee 1_{\mathfrak{G}(R_F(F))}$ where f^\uparrow is the sprouting fibration of $\text{spr}_R(F)$ and $1_{\mathfrak{G}(R_F(F))}$ is the identity over $\mathfrak{G}(R_F(F))$.

Similarly, if R_F is closed, then the R&B-cograph \mathfrak{C}_{R_F} of ϕ_{R_F} is the cograph $\mathfrak{C}_{R_F}^\downarrow = \overline{\mathfrak{G}(\text{spr}(T))}$ enriched with the corresponding matching B_{T_F} and the skew fibration $f: \mathfrak{C}^\downarrow \rightsquigarrow \mathfrak{G}_{T_F}$ where $\mathfrak{G}_{T_F} = \overline{\mathfrak{C}(F)}$ is given by the sprouting fibration from F to $\text{spr}(T)$. \square

Remark 5.13 The number of B -edges in ϕ_{R_F} is equal to the sum of the number of atoms occurring in resolved formulas, each of which is counted as many times as the product of positive and negative occurrences of the resolved formula to which the atom belongs. This size explosion can be avoided in the special case where resolution inferences resolve the same number of formulas occurrences. Then we can adapt the translation suggested by Das in [4], by replacing the interpretation of resolution rule in Figure 15 with the following:

$$\left[\Gamma, \frac{X^{\vee n} [\bar{X}^{\wedge n}]}{(\bullet \wedge \bullet)} \text{Res}^X, \Delta \right]$$

Theorem 5.14 *If R_F and R'_F are equivalent, then $\phi_{R_F} = \phi_{R'_F}$.*

Proof The permutation of expansion rules does not change the number of formula occurrences in the resolution rule instances. Hence, we can conclude by the same reasoning as in the proof of Theorem 4.8. \square

It is possible to associate to $\phi: F \vdash \circ$ with skew fibration $f: \mathfrak{C}^\downarrow \rightsquigarrow \mathfrak{G}_F$ a resolution expansion as follows. If F' is the formula associated to \mathfrak{C}^\downarrow by labeling its vertices according to f , then we can transform F' to its conjunctive normal form by applying $c\uparrow$ and $w\uparrow$. By Lemma 5.5 the composition of this expansion with f represents the expansion part of the pseudo-resolution, while the perfect matching of the R&B-cograph \mathfrak{C}^\downarrow takes track of the resolution rule instances. If multiple matchings connect atoms in the same clause this correspond to a unique resolution rule inference, otherwise a clause duplication has been performed during the resolution expansion oversaturation, which means that the some \wedge -expansions have been performed after the corresponding resolution rule.

6 Conclusions

In this paper we tried to make a case that *combinatorial proofs* can serve as canonical representation for proofs in classical propositional logic, by showing how natural notions of proof identity in various syntactic formalisms are reflected by combinatorial proofs. We extended the investigation by Hughes [11] from sequent calculus to other formalisms that are employed in automatic reasoning.

There is ongoing research investigating the possible structure for combinatorial proofs for intuitionistic logic, and in future research we plan to investigate combinatorial proofs for first-order logic, based on Hughes' *unification nets* [12], and for modal logics, based on the recent development on *nested sequents* [13].

References

1. Bellin, G., van de Wiele, J.: Subnets of proof-nets in MLL^- . In: *Advances in Linear Logic*, pp. 249–270. Cambridge University Press (1995)
2. Brünnler, K., Tiu, A.F.: A local system for classical logic. In: Nieuwenhuis, R., Voronkov, A. (eds.) *LPAR 2001*. LNAI, vol. 2250, pp. 347–361. Springer (2001)
3. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. *J. of Symb. Logic* 44(1), 36–50 (1979)
4. Das, A.: On the relative proof complexity of deep inference via atomic flows. *Logical Methods in Computer Science* 11(1) (2015)
5. Das, A., Straßburger, L.: On linear rewriting systems for boolean logic and some applications to proof theory. *Logical Methods in Computer Science* 12(4) (2016)
6. Duffin, R.: Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications* 10(2), 303 – 318 (1965)
7. Fitting, M.: *First-order logic and automated theorem proving*. Springer Science & Business Media (2012)
8. Gentzen, G.: Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift* 39, 176–210 (1935)
9. Guglielmi, A., Straßburger, L.: Non-commutativity and MELL in the calculus of structures. In: *CSL 2001*. LNCS, vol. 2142, pp. 54–68. Springer (2001)
10. Hughes, D.: Proofs Without Syntax. *Annals of Math.* 164(3), 1065–1076 (2006)
11. Hughes, D.: Towards Hilbert's 24th problem: Combinatorial proof invariants: (preliminary version). *Electr. Notes Theor. Comput. Sci.* 165, 37–63 (2006)
12. Hughes, D.J.: Unification nets: canonical proof net quantifiers. In: *LICS 2018* (2018)
13. Marin, S., Straßburger, L.: Label-free Modular Systems for Classical and Intuitionistic Modal Logics. In: *Advances in Modal Logic* 10 (2014)
14. Retoré, C.: Handsome proof-nets: perfect matchings and cographs. *Theor. Comput. Sci.* 294(3), 473–488 (2003)
15. Smullyan, R.M.: *First-order logic*. Courier Corporation (1995)
16. Straßburger, L.: A characterization of medial as rewriting rule. In: *RTA'07*. vol. 4533, pp. 344–358. Springer (2007)
17. Straßburger, L.: *Combinatorial Flows and Proof Compression*. Research Report RR-9048, Inria Saclay (2017), <https://hal.inria.fr/hal-01498468>
18. Straßburger, L.: Combinatorial flows and their normalisation. In: Miller, D. (ed.) *FSCD'17. LIPIcs*, vol. 84, pp. 31:1–31:17. Schloss Dagstuhl (2017)
19. Troelstra, A.S., Schwichtenberg, H.: *Basic proof theory*, vol. 43. Cambridge University Press (2000)