

First-order Proofs Without Syntax: Summary of Work in Progress

Dominic J. D. Hughes

Stanford University*

25 May 2014

“...mathematicians care no more for logic than logicians for mathematics.”

Augustus de Morgan, 1868

Formal proofs are traditionally syntactic, build inductively from symbolic inference rules. This paper reformulates classical first-order logic with proofs which are combinatorial rather than syntactic. A *combinatorial proof* of a formula F is a graph on a variant formula F' which satisfies geometric properties. For example, an edge between the two P s in $\forall x P x \Rightarrow \forall y P y$ constitutes a combinatorial proof of $\exists x (P x \Rightarrow \forall y P y)$.

Combinatorial proofs provide a canonical abstraction of Gentzen’s sequent calculus LK: the correctness of a combinatorial proof can be verified in polynomial time (conjectured linear), and there is a function from LK proofs to combinatorial proofs which identifies many LK proofs.

This paper summarizes ongoing work on a sequel to *Proofs Without Syntax* [Annals of Mathematics, 2006], which introduced propositional combinatorial proofs.

1 Introduction

The dry syntactic manipulations of formal logic can be distasteful to mathematicians who delight in beautiful symmetries, geometries, and rich layers of structure. This paper reformulates classical first-order logic with proofs which are combinatorial rather than syntactic. A *combinatorial proof* of a formula F is a graph on the predicate symbols of a variant formula F' .¹ For example, here is a combinatorial proof of $\exists x (P x \Rightarrow \forall y P y)$:²

$$\overbrace{\forall x P x \Rightarrow \forall y P y}$$

Four more illustrative examples follow. A combinatorial proof of $\exists x (P a \vee P b \Rightarrow P x)$:

$$\overbrace{\forall x (P a \vee P b) \Rightarrow \exists x P x \vee \exists x P x}$$

A combinatorial proof of $\forall x (P f x \Rightarrow P x) \Rightarrow \forall y (P f f y \Rightarrow P y)$:

$$\overbrace{\forall x (P f x \Rightarrow P x) \wedge \forall x (P f x \Rightarrow P x) \Rightarrow \forall y (P f f y \Rightarrow P y)}$$

Combinatorial proofs of *Peirce’s Law* $((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$ and $\forall x (\top \vee \perp)$:

$$\overbrace{P \vee P \Rightarrow P \vee P} \qquad \forall x \top$$

*Visiting Scholar, Mathematics Department. I am very grateful to my host, Sol Feferman. This research began whilst a Visiting Scholar in the Computer Science Department; thanks to Vaughan Pratt for hosting me there. Thanks to Dale Miller, Lutz Strassburger, Assia Mahboubi, Stéphane Graham-Lengrand and Kaustuv Chaudhuri for inviting me to present an earlier version of this work in progress at the LIX Colloquium 2013, which led to valuable feedback. Also to Laurent Regnier and Emmanuel Beffara for hosting me at IML Marseille and Lyon.

¹We work with formulas modulo associativity of $\wedge$ and $\vee$ and commutativity of $\vee$. We assume $\forall x$ and $\exists x$ bind more strongly than $\wedge$ and $\vee$, which bind more strongly than $\Rightarrow$. For example, $P a \vee P b \Rightarrow \exists x P x \vee \exists x P x$ abbreviates $(P a \vee P b) \Rightarrow ((\exists x P x) \vee (\exists x P x))$. We write $\top$ and $\perp$ for the logical constants *true* and *false*.

²This formula, often known as the *drinker paradox*, was popularized by Smullyan [Smu78]: *every pub has a person x such that, if x is drinking, everyone is drinking.*

A combinatorial proof of F is defined by three main conditions: (1) the variant formula F' pulls predicate symbols (and $\top$ s) from F along a particular kind of graph homomorphism, (2) edges induce consistent variable assignments, e.g.

$$\frac{\frac{\frac{x = fy}{\forall x(Pfx \Rightarrow Px) \wedge \forall x'(Pfx' \Rightarrow Px')} \Rightarrow \forall y(Pffy \Rightarrow Py)}{x' = fx} \quad \frac{}{x' = y}}{\quad}$$

and (3) derived graphs, called *switchings*, are trees (related to proof nets [Gir87, DR89, Gir96] for linear logic).

The main theorem is soundness and completeness for classical first-order logic:

Theorem. *A formula is valid if and only if it has a combinatorial proof.*

The correctness of a combinatorial proof can be checked in polynomial time.

Conjecture. *The correctness of a combinatorial proof can be verified in linear time.*

Every syntactic proof of Gentzen's first-order classical sequent calculus LK [Gen34] translates to a combinatorial proof.³ For example, the following LK proof translates to the combinatorial proof $\forall x \overline{Px} \Rightarrow \forall y \overline{Py}$ of $\exists x(Px \Rightarrow \forall y Py)$:

$$\frac{\frac{\frac{\frac{\frac{\frac{}{P(y) \vdash P(y)}{\text{axiom}}}{P(y) \vdash P(y), \forall y P(y)}{\text{weaken}}}{\vdash P(y), P(y) \Rightarrow \forall y P(y)}{\text{implies}}}{\vdash P(y), \exists x (P(x) \Rightarrow \forall y P(y))}{\text{there exists}}}{\vdash \forall y P(y), \exists x (P(x) \Rightarrow \forall y P(y))}{\text{for all}}}{\text{weaken}} \frac{\frac{\frac{\frac{\frac{\frac{\frac{}{P(x) \vdash \forall y P(y), \exists x (P(x) \Rightarrow \forall y P(y))}{\text{implies}}}{\vdash P(x) \Rightarrow \forall y P(y), \exists x (P(x) \Rightarrow \forall y P(y))}{\text{there exists}}}{\vdash \exists x (P(x) \Rightarrow \forall y P(y)), \exists x (P(x) \Rightarrow \forall y P(y))}{\text{contract}}}{\vdash \exists x (P(x) \Rightarrow \forall y P(y))}$$

The translation identifies LK proofs modulo inessential rule transpositions. For example, if we modify the above proof by transposing the second *weaken* rule up over the *for all* rule, it still translates to the same combinatorial proof. The identification of LK proofs, together with polynomial time complexity (conjectured linear), suggests that combinatorial proofs provide a canonical abstraction of sequent calculus proofs.

Related Work. Propositional combinatorial proofs were introduced in [Hug06a] and studied as invariants of syntactic proofs in [Hug06b]. Combinatorial proofs build on a long history of work on graphical invariants of proofs.

The idea of linking dual occurrences (such as the P s in $\forall x \overline{Px} \Rightarrow \forall y \overline{Py}$) is widespread in the study of various forms of syntax, going back many decades in many disciplines, including category theory (compatibility graphs) [EK66, KM71], automated theorem proving (matings and connections) [Dav71, Bib74, And76, And81] and linear logic (axiom links) [Gir87].

A key advantage of combinatorial proofs over the matrix/connection method is their linear time verifiability: a matrix/connection requires exponential time. In addition, combinatorial proofs are not restricted to disjunctive normal forms. Lamarche and Straßburger [LS05] study

³By adjusting the definition of combinatorial proofs slightly (dropping associativity of $\wedge$), the translation can be made surjective. The translation encodes an LK cut between A and $\overline{A}$ as a subformula $A \wedge \overline{A}$.

propositional matrix/connection-like structures from a categorical perspective. (Correctness remains exponential.) Buss analyses the flow of atoms down propositional proofs [Bus89] (without a correctness criterion); a relationship between flow graphs and propositional combinatorial proofs is studied in [Car10].

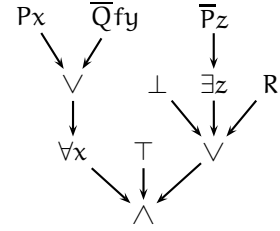
The switchings employed in the definition of a combinatorial proof are an abstraction of those in first-order multiplicative linear proof nets [Gir95]; combinatorial proofs are simpler by virtue of leaving existential witness explicit via unification. Propositional proof nets with weakening are presented in [Bel03], and propositional proof nets for classical logic are in [Rob03, McK13]. Heijltjes and McKinley have studied graphical representations of first-order classical proofs [Hei10, McK10].

Future work includes studying the relationship between combinatorial proofs and the matrix/connection method in more detail, and the relationship with expansion proofs and multi-focussed sequent proofs [Mil84, CHM14].

2 Notation and terminology

We work with classical first-order logic (predicate calculus) without equality, as per any introductory undergraduate text, e.g. [Joh87], and assume the reader is familiar with first-order logical formulas, free and bound variables, and validity.

We shall formalize formulas as labelled trees, such as the one shown right, providing a representation which is canonical modulo commutativity of $\vee$ and associativity of $\wedge$ and $\vee$: the children (inbound edges) of $\wedge$ nodes are ordered⁴ (but those of $\vee$ nodes are not), and every $\vee$ (resp. $\wedge$) node has at least two children, none of which is a $\vee$ (resp. $\wedge$).



We adopt the following conventions throughout the paper:

$x \ y \ z$	(term) variables	$s \ t \ u$	terms
$f \ g \ h$	n -ary function symbols ($n \geq 1$)	$P \ Q \ R$	predicate symbols
$a \ b \ c$	0-ary function symbols	$A \ B \ C$	formulas

Fix an arity-preserving **negation** function ($\bar{}$) on predicate symbols such that $\bar{\bar{P}} = P$ for all P . A **literal** is an expression $Pt_1 \dots t_n$ for any n -ary predicate symbol P and terms t_i . Parentheses and commas may be inserted to increase readability, e.g., $Paffy = P(a, ffy) = P(a, f(f(y)))$. An **atom** is a literal or a logical constant **true** $\top$ or **false** $\perp$.

A **label** is any of: (a) an atom, (b) a connective $\wedge$ or $\vee$, (c) for any variable x , a quantifier $\forall x$ or $\exists x$. A **formula** is a finite directed tree labelled with atoms on leaves and connectives/quantifiers on internal nodes, with edges directed towards the root, such that (1) the inbound edges of each $\wedge$ node are strictly linearly ordered⁵, (2) every $\vee$ (resp. $\wedge$) node has at least two inbound edges, none from a $\vee$ (resp. $\wedge$) node, (3) every quantifier node has just one inbound edge. Variable binding is defined in the standard way (viewing our formulas as parse trees of more standard syntactic formulas [Joh87]). We identify formulas modulo renaming of bound variables.

We refer to a conventional syntactic formula (generated from atoms by binary $\wedge$ and $\vee$ and unary $\exists x$ and $\forall x$ for each variable x) as a **formula expression**. Define **negation** on formula expressions by: $\overline{Pt_1 \dots t_n} = \bar{P}t_1 \dots t_n$, $\bar{\top} = \perp$, $\bar{\perp} = \top$, $\bar{\wedge} = \vee$, $\bar{\vee} = \wedge$, $\bar{\exists x} = \forall x$ and $\bar{\forall x} = \exists x$. We use implication $A \Rightarrow B$ as an abbreviation for the formula expression $\bar{A} \vee B$.

Every formula expression defines a formula (tree) in the obvious way. For example, each of the following formula expressions denotes the formula whose tree was depicted above:

$$(\forall x (Px \vee \bar{Q}fy)) \wedge \top \wedge (\perp \vee (\exists z \bar{P}z) \vee R)$$

$$(\forall x (Qfy \Rightarrow Px)) \wedge \top \wedge ((\exists z \bar{P}z) \vee R \vee \perp)$$

⁴The conjecture that a combinatorial proof can be verified in linear time is a direct result of ordered $\wedge$ nodes. It is perfectly possible to define combinatorial proofs modulo commutativity of $\wedge$, with unordered $\wedge$ nodes (as in the original propositional version [Hug06a]), but correctness, while obviously cubic, does not appear to be linear.

⁵ $\vee$ nodes remain unordered.

To reduce parentheses in formula expressions we assume quantifiers minimize scope, for example, $\forall x P_x \vee (\exists y \bar{P}_y \wedge Q) = (\forall x P_x) \vee ((\exists y \bar{P}_y) \wedge Q)$.

The **meet** of two vertices is the vertex at which their paths to the root meet. For example, the meet of $\perp$ and $\bar{P}z$ in the formula displayed earlier is the right-most $\vee$ and the meet of $\bar{Q}fy$ and $\exists z$ is the $\wedge$. Vertices v and w are **conjunctively related**, denoted $v \wedge w$, if their meet is a $\wedge$, and v comes before w under the hereditary child-orderings at $\wedge$ nodes. For example, in the formula tree depicted earlier $\bar{Q}fy \wedge \exists z$, $P_x \wedge \bar{P}z$, and $v \wedge w$ for v/w the left/right $\vee$ -vertices. To emphasize the ambient formula B , we write $\wedge_B$. Vertices are **disjunctively related** if their meet is a $\vee$.

A formula is **plain** if no quantified variable occurs free, and two quantifiers have the same variable only if they are disjunctively related. For example, $\forall x P_x \vee \forall y Q_{yz}$, $\forall x P_x \wedge \forall y Q_{yz}$ and $\forall x P_x \vee \forall x Q_{xz}$ are plain, but $P_x \vee \forall x Q_x$, $\forall x P_x \wedge \forall x Q_{xz}$, $\forall x \forall x R_x$ and $\forall x \forall x \top$ are not. For technical convenience, *henceforth we assume all formulas are plain* (renaming bound variables if necessary). A formula is **clean** if all quantifiers have distinct variables, e.g. $\exists x P_x \vee \exists y Q_y$ but not $\exists x P_x \vee \exists x Q_x$.

The **line** of a leaf v is the sequence of quantifiers along the path from the root to v , followed by the atom of v . For example, in $\exists x (\bar{P}_x \vee \forall y P_y)$ the line of P_y is $\exists x \forall y P_y$ and that of $\bar{P}_x$ is $\exists x \bar{P}_x$. A quantifier is **vacuous** if it binds no variable, e.g., $\exists x$ in $\exists x \forall y (P_y \vee \perp)$. A line **extends** another if it results from adding zero or more vacuous quantifiers, e.g., $\exists x \forall y \forall z P_y$ extends $\forall y P_y$.

3 Morphisms

Let B be a formula. Without loss of generality, assume B is clean (renaming bound variables if necessary). A **morphism** $\theta : A \rightarrow B$ from a formula A to B is a function θ from the leaves of A to the leaves of B such that:

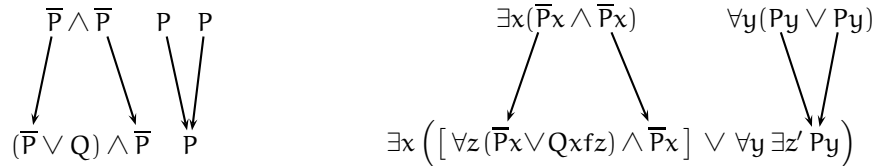
(LINE) θ extends lines: the line of $\theta(v)$ extends the line of v , for every leaf v .

(CONJ) θ preserves conjunctive relationships: if $v \wedge_A w$ then $\theta(v) \wedge_B \theta(w)$.

(SKEW) θ is a **skew fibration**: if $\theta(v) \wedge_B w$ there exists w' with $v \wedge_A w'$ and $w \not\wedge_B \theta(w')$.

As a special case (ignoring quantifiers), LINE implies that θ preserves the atomic labels of leaves (i.e., if the atom on v is $P\bar{t}$ then the atom on $\theta(v)$ is $P\bar{t}$). Viewing $\wedge_A$ and $\wedge_B$ as simple directed graphs, CONJ requires θ to be a directed graph homomorphism, and SKEW requires θ to be a lax variant of a graph fibration; see [Hug06a, Hug06b].⁶ Condition SKEW is equivalent to the preservation of maximal (directed) cliques [Hug06b], or, since $\wedge$ is a strict order, preservation of maximal chains.

A morphism $(\bar{P} \wedge \bar{P}) \vee P \vee P \rightarrow ((\bar{P} \vee Q) \wedge \bar{P}) \vee P$ is below-left.⁷ To reduce clutter we render outer $\vee$ as disjoint union.



Arrows depict the leaf function, e.g., the the third and fourth leaves P in $(\bar{P} \wedge \bar{P}) \vee P \vee P$ map to the last leaf P in $((\bar{P} \vee Q) \wedge \bar{P}) \vee P$. A morphism

$$\exists x (\bar{P}_x \wedge \bar{P}_x) \vee \forall y (P_y \vee P_y) \rightarrow \exists x ([\forall z (\bar{P}_x \vee Q_x f z) \wedge \bar{P}_x] \vee \forall y \exists z' P_y)$$

is above-right (a first-order embellishment of the previous example). Illustrating LINE, the line $\exists x \forall y \exists z' P_y$ of the output leaf P_y extends the line $\forall y P_y$ of each of the two input leaves P_y .

⁶The only difference here is that we take the edge relation $\wedge$ to be directed; this yields the conjecture that correctness can be checked in linear time.

⁷The target $((\bar{P} \vee Q) \wedge \bar{P}) \vee P$ is *Peirce's Law* $((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$.

4 Linkings

Two literals are **dual** if their predicate symbols are dual (e.g. Pxy and $\bar{P}yfa$) and two leaves are dual if they have dual literals. A **link** is a pair $\{v, w\}$ of dual leaves. A **linking** on a formula is a set of disjoint links whose union contains every literal-labelled leaf. We draw a link $\{v, w\}$ as an edge between v and w , e.g.

$$\begin{array}{ccccc} & \overline{\hspace{1.5cm}} & & \overline{\hspace{1.5cm}} & \\ \forall y \exists x \bar{P}(gy, fx, a) & & \exists x' \forall y' P(x', fy', a) \wedge Q(hza) & & \exists z' \bar{Q}z' \wedge \top \end{array}$$

Let λ be a linking on B . Without loss of generality, assume B is clean (renaming bound variables if necessary). A **unifier** for λ is an assignment of terms to existential variables which equalizes the term sequences at either end of every link. For example, $U = [x \mapsto y', x' \mapsto gy, z' \mapsto hza]$ is a unifier for the linking displayed above: upon substituting according to U the first link has the three-term sequence (gy, fy', a) at either end, and the second has the one-term sequence (hza) :

$$\begin{array}{ccccc} & \overline{\hspace{1.5cm}} & & \overline{\hspace{1.5cm}} & \\ \bar{P}(gy, fy', a) & & P(gy, fy', a) & & Q(hza) & & \bar{Q}(hza) \wedge \top \end{array}$$

The formal unification problem is as follows. A link between $P(s_1, \dots, s_n)$ and $\bar{P}(t_1, \dots, t_n)$ determines n equations $s_i = t_i$. Taking the union across all links, we obtain a set of equations E . Solve E for the existential variables (treating free and universal variables as constants). For example, the link $\bar{P}(gy, fx, a) \text{ --- } P(x', fy', a)$ of the linking at the beginning of the section determines three equations $gy = x'$, $fx = fy'$ and $a = a$, the link $\bar{Q}(hza) \text{ --- } \bar{Q}(z')$ yields $hza = z'$, so $E = \{gy = x', fx = fy', a = a, hza = z'\}$. We solve E for the existential variables x, x' and z' (treating the universal y and y' as constants): $[x \mapsto y', x' \mapsto gy, z' \mapsto hza]$.

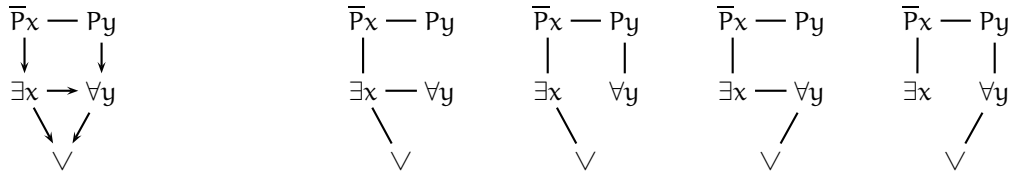
Unifiability can be determined in linear time, and the **most general unifier** or **mgu** returned [MM76]. The mgu yields every other unifier by substitution. For example, the mgu of $\exists x \bar{P}x \vee \forall y \bar{P}y$ is $U = [x \mapsto z, y \mapsto z]$: every unifier is $U_t = [x \mapsto t, y \mapsto t]$ for some term t , and U yields U_t by substituting t for z . The mgu is defined up to free variable renaming [LMM88]: $[x \mapsto z', y \mapsto z']$ also represents the mgu.

Let λ be a unifiable linking on a formula B . Without loss of generality, assume B is clean. A **dependency** $x \curvearrowright y$ is an existential variable x and a universal variable y such that the mgu of λ assigns to x a term containing y .⁸ For example, the dependencies of the linking drawn at the start of the section are $x \curvearrowright y'$ and $x' \curvearrowright y$. All dependencies can be determined in linear time: make a single pass of the mgu.

The **graph** $G(\lambda)$ of λ is the directed tree B together with

- for every link $\{v, w\}$ in λ , an undirected edge $v \text{ --- } w$;
- for every dependency $x \curvearrowright y$, a directed edge $\exists x \rightarrow \forall y$.

For example, the graph of $\exists x \bar{P}x \vee \forall y \bar{P}y$, whose unique unifier is $[x \mapsto y]$, is below-left.



A **switching** of λ is any derivative of $G(\lambda)$ obtained by deleting all but one edge into each $\vee$ and $\forall$ vertex and undirecting all remaining edges. The four switchings of $\exists x \bar{P}x \vee \forall y \bar{P}y$ are above-right. A linking is **correct** if all switchings are trees (acyclic and connected). The linking $\exists x \bar{P}x \vee \forall y \bar{P}y$ is correct: all four switchings above-right are trees.

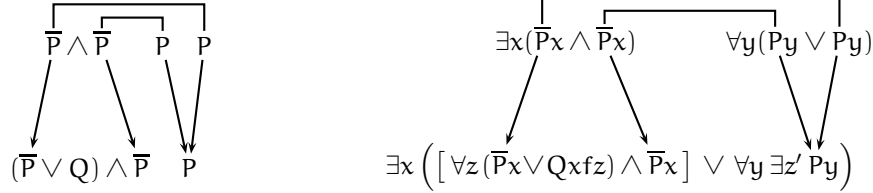
⁸Equivalently, every unifier of λ assigns to x a term containing y . The mgu-based definition is well-defined modulo renaming of free variables in the mgu, since free variables are distinct from quantified variables.

5 Combinatorial proofs

Let B be a formula. Without loss of generality assume B is clean (renaming bound variables if necessary). A **combinatorial proof** of B comprises:

- a morphism $A \rightarrow B$ from a $\perp$ -free formula A ;
- a correct linking on A .

We call A the **core**. A combinatorial proof of $((\bar{P} \vee Q) \wedge \bar{P}) \vee P$ is below-left, with outer $\vee$ drawn as disjoint union.



The four directed edges define a morphism $(\bar{P} \wedge \bar{P}) \vee P \vee P \rightarrow ((\bar{P} \vee Q) \wedge \bar{P}) \vee P$ and the two undirected edges define a correct linking on $(\bar{P} \wedge \bar{P}) \vee P \vee P$. A combinatorial proof of $\exists x ([\forall z (\bar{P}x \vee Qx f z) \wedge \bar{P}x] \vee \forall y \exists z' P y)$ is above-right.

Theorem 1 (Soundness & Completeness).

A formula is valid if and only if it has a combinatorial proof.

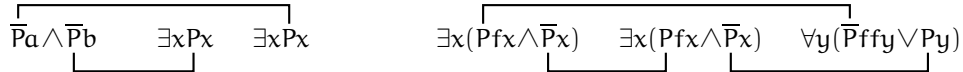
Conjecture 1 (Linear complexity).

The correctness of a combinatorial proof can be checked in linear time.

Concise inline presentation. When B has at most one occurrence of each atom, we can compress the presentation of a combinatorial proof of B by omitting the leaf function, since it is fully determined by the preservation of atomic labels (implied by the line-extension condition **LINE**). For example, the combinatorial proof of $\exists x (\bar{P}x \vee \forall y P y)$ below-left

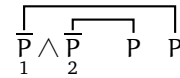


can be abbreviated to the linking above-right: the morphism from $A = \exists x \bar{P}x \vee \forall y P y$ to $B = \exists x (\bar{P}x \vee \forall y P y)$ is implicit since line-extension **LINE** forces the leaf function to take $\bar{P}x$ (resp. $P y$) in A to $\bar{P}x$ (resp. $P y$) in B . Also with formula morphisms implicit, here are combinatorial proofs of $\exists x ((\bar{P}a \wedge \bar{P}b) \vee P x)$ and $\exists x (P f x \wedge \bar{P}x) \vee \forall y (\bar{P} f y \vee P y)$:



These three examples are exactly the first three on page 1, via $A \Rightarrow B = \bar{A} \vee B$.

With multiple copies of an atom in the target, we can still present a combinatorial proof concisely inline, with tags. For example, on the right is a combinatorial proof of $((\bar{P} \vee Q) \wedge \bar{P}) \vee P$, where the underset 1 or 2 indicate that the morphism takes the $\bar{P}$ to the 1st or 2nd occurrence of $\bar{P}$ in $((\bar{P} \vee Q) \wedge \bar{P}) \vee P$.⁹ In fact, in this particular case we don't actually require the tags: condition **CONJ** forces the order of the $\bar{P}$ s either side of the $\wedge$ to be preserved. This is the same combinatorial proof as the one drawn at the top of this page with explicit morphism arrows. It was also depicted in the introduction on page 1, via $A \Rightarrow B = \bar{A} \vee B$.¹⁰



⁹Since we defined a formula in canonical form modulo commutativity of $\vee$ and associativity of $\vee$ and $\wedge$, when rendering a combinatorial proof this way we must assume a particular ordering of leaves in the proved formula, for example, as occurs when the proved formula is presented as a formula expression.

¹⁰Since every $\wedge$ is ordered, every $\vee$ on the left of an implication $\Rightarrow$ is ordered.

6 Canonical abstraction of sequent calculus

Every syntactic proof of Gentzen’s first-order classical sequent calculus LK [Gen34] translates to a combinatorial proof.¹¹ An example was shown on page 2. The translation identifies LK proofs modulo inessential rule transpositions, such as reorderings of contraction and weakening. The identification of LK proofs, together with polynomial time complexity (conjectured linear), suggests that combinatorial proofs provide a canonical abstraction of sequent calculus proofs.

By adjusting the definition of combinatorial proofs slightly (dropping the associativity of $\wedge$), the translation can be made surjective.

References

- [And76] Peter B. Andrews. Refutations by matings. *IEEE Transactions on Computers*, 25(8), 1976.
- [And81] Peter B. Andrews. Theorem proving via general matings. *J. ACM*, 28(2), 1981.
- [Bel03] Gianluigi Bellin. Two paradigms of logical computation in affine logic? de Queiroz, Ruy J. G. B. (ed.), *Logic for concurrency and synchronisation*. Trends Log. Stud. Log. Libr. 18, 2003.
- [Bib74] W. Bibel. An approach to a systematic theorem proving procedure in first-order logic. *Computing*, 12(1), 1974.
- [Bus89] Samuel R. Buss. The undecidability of k-provability. *Annals of Pure and Applied Logic*, 53, 1989.
- [Car10] Alessandra Carbone. A new mapping between combinatorial proofs and sequent calculus proofs read out from logical flow graphs. *Inf. Comput.*, 208(5):500–509, 2010.
- [CHM14] Kaustuv Chaudhuri, Stefan Hetzl, and Dale Miller. The isomorphism between expansion proofs and multi-focused sequent proofs. *J. Logic & Computation*, 2014. to appear.
- [Dav71] G.V. Davydov. The synthesis of the resolution method with the inverse method. *Zapiski Nauchnykh Seminarov Lomi 20 1971 24–35*, 20:24–35, 1971. Translation in *J. Sov. Math.* 1(1) 1973 12–18.
- [DR89] V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.
- [EK66] S. Eilenberg and G. M. Kelly. A generalization of the functorial calculus. *J. Algebra*, 3:366–375, 1966.
- [Gen34] G. Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39:176–210, 1934.
- [Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
- [Gir95] J.-Y. Girard. Geometry of interaction III: Accommodating the additives. In *Advances in Linear Logic*, volume 222 of *London Math. Soc. LNS*, pages 329–389. Cambridge University Press, 1995.
- [Gir96] J.-Y. Girard. Proof-nets: the parallel syntax for proof theory. In *Logic and Algebra*, volume 180 of *Lecture Notes In Pure and Appl. Math.* Marcel Dekker, New York, 1996.
- [Hei10] Willem Heijltjes. Classical proof forestry. *Annals of Pure and Applied Logic*, 161(11):1346–1366, 2010.
- [Hug06a] D. J. D. Hughes. Proofs Without Syntax. *Annals of Mathematics*, 143:1065–1076, 2006.
- [Hug06b] D. J. D. Hughes. Towards Hilbert’s 24th Problem: Combinatorial Proof Invariants. In *Proc. WOL-LiC’06*, volume 165 of *Lec. Notes in Comp. Sci.*, 2006.
- [Joh87] P. T. Johnstone. *Note on Logic and Set Theory*. Cambridge University Press, 1987.
- [KM71] G. M. Kelly and S. Mac Lane. Coherence in closed categories. *J. Pure Appl. Algebra*, 1:97–140, 1971.
- [LMM88] J.-L. Lassez, M. J. Maher, and K. Marriott. Unification revisited. In J. Minker, editor, *Foundations of deductive databases and logic programming*, pages 587–625. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- [LS05] F. Lamarche and L. Straßburger. Naming proofs in classical propositional logic. In *Proc. Typed Lambda Calculus ’05*, *Lec. Notes in Comp. Sci.*, 2005. To appear.
- [McK10] Richard McKinley. Expansion nets: Proof nets for for propositional classical logic. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 17)*, volume 6397 of *Lecture Notes in Computer Science*, pages 535–549. Springer, 2010.

¹¹The translation encodes an LK cut between A and $\overline{A}$ as a subformula $A \wedge \overline{A}$.

- [McK13] Richard McKinley. Canonical proof nets for classical logic. *Annals of Pure and Applied Logic*, 164(6):702–732, 2013.
- [Mil84] Dale A. Miller. Expansion tree proofs and their conversion to natural deduction proofs. *Lec. Notes in Comp. Sci.*, 170:375–393, 1984.
- [MM76] A. Martelli and U. Montanari. Unification in linear time and space: a structured presentation. Technical Report B76-16, University of Pisa, 1976.
- [Rob03] E. Robinson. Proof nets for classical logic. *J. Logic and Computation*, 13(5):777–797, 2003.
- [Smu78] R. M. Smullyan. *What is the Name of this Book? The Riddle of Dracula and Other Logical Puzzles*. Prentice Hall, 1978.